Aurora Martina
Angelo Raffaele Meo
Clotilde Moro
Mario Scovazzi

Passo dopo passo impariamo a programmare con PYTHON

RINGRAZIAMENTI

Molte sono le persone che durante la stesura di questo manuale ci hanno assistito con consigli e suggerimenti sempre molto preziosi. A tutti, indistintamente, i nostri ringraziamenti.

In particolare ricordiamo:

* i "ragazzi" del Dipartimento di Informatica e Automatica del Politecnico di Torino che ci hanno aiutato con la loro indiscussa competenza e con i loro consigli
* le professoresse Sandra Sonego e Gisella Picollo che hanno "testato" il nostro lavoro con spirito pionieristico
* gli alunni della scuola media Peyron – Fermi di Torino, corsi E, H che hanno fatto allegramente da cavie, tuffandosi per la prima volta nel profondo mare della programmazione.
  P. S.: nessuno è annegato

PREMESSA

Abbiamo realizzato questo manuale **con l'intento di insegnare le basi della programmazione informatica ai ragazzi di età compresa tra i 10 e i 14 anni circa.**

Speriamo che possa risultare utile anche a quei docenti, educatori, genitori che condividono con noi l'idea che oggi insegnare l'informatica ai ragazzi sia non solo importante ma indispensabile al fine di contribuire a creare in loro maggiore autonomia, sicurezza e capacità di gestire il proprio futuro.

Ci auguriamo che possa essere gradito soprattutto ai ragazzi, spesso avviati dal sistema scolastico a una conoscenza troppo "*applicativa*" dell'informatica.

Convinti che le basi della programmazione informatica, opportunamente mediate, possano e debbano, far parte del bagaglio culturale dei ragazzi, abbiamo ideato una serie di schede che, passo dopo passo, consentiranno agli alunni di avviare un percorso di conoscenza e di esperienza che li porterà, un po' più avanti nel tempo, ad acquisire le capacità necessarie a padroneggiare *in maniera consapevole* il mezzo informatico.

Nella **strutturazione del percorso** di conoscenza, così come nell'organizzazione dei contenuti e nella scelta della veste grafica, sono state adottate soluzioni utili a consentire un uso del manuale quanto più autonomo da parte:

- *degli alunni,* che potranno scaricare liberamente i materiali dalla rete
- *dei docenti* che potranno liberamente disporne per le proprie lezioni

Il manuale è stato creato anche per essere fruito on line sulle piattaforme per la didattica oggi più diffuse.

La scelta del linguaggio di programmazione Python non è casuale; esso risponde infatti a molti dei requisiti da noi ritenuti fondamentali. Tra questi:

* libero
* didatticamente adeguato
* attuale
* possibilità di utilizzo e di approfondimento nel tempo
* diffusione nella pratica informatica

Quella che presentiamo è una prima stesura e pertanto ci scusiamo per eventuali refusi e omissioni,

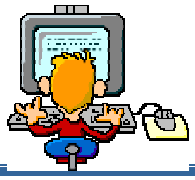
Gli Autori

Qualche indicazione per usare al meglio il manuale

Il corso si compone di dodici capitoli (Step 1, Step 2,…) nei quali, passo dopo passo, vengono spiegate le basi di un linguaggio di programmazione chiamato Python.
In ogni capitolo la parte teorica precede quella dedicata agli esercizi (tutte le soluzioni sono al fondo del manuale). Ogni tre passi c'è uno stop dedicato alla valutazione.
Le pagine sono generalmente impostate come una scheda: da una parte il testo con le spiegazioni e dall'altra simpatiche *clip art* utili non solo ad abbellire il testo e a rendere più piacevole lo studio, ma anche a richiamare l'attenzione sul tipo di attività da svolgere. Qualche esempio:

	La nostra **classe virtuale**, vivace, molto curiosa e desiderosa di apprendere
	**Super Teacher**, tutor del corso, compare nelle schede per guidare il lavoro e per dare consigli.
	**A spiegare i contenuti importanti trovi il professor Angelo.**
	**Martina** è l'assistente del professor Angelo. Lei e Mario affiancano il professore nelle spiegazioni e nelle esercitazioni guidate.
	Ecco **Mario**, il tecnico di laboratorio.

Il **cappello pensante** va indossato quando è importante concentrarsi e capire un nuovo concetto o risolvere qualche quesito. Quando hai trovato la soluzione….alza la mano!



Questo sei tu quando devi "smanettare".

Quando invece arriva l'ora di esercitarti a ricordartelo c'è **RamRam**, un simpatico topolino, oppure…. ***Gigi*** l'allenatore o **Fabio** l'atleta.

Quando compaiono questi simboli bisogna leggere con molta attenzione, a fianco troverai una spiegazione, una definizione, un approfondimento….o semplicemente qualcosa da memorizzare.

E poi c'è **Moon**, che ha per cuccia una scatola. Leggendo le prossime pagine capirai perché. E' il nostro cucciolo, la mascotte della classe. Ha tanti amici che ti accompagneranno …passo dopo passo!

E naturalmente c'è **PyPy,** il nostro pitone. Insomma, ci sono tutti gli ingredienti per lavorare con serietà ma anche con allegria.

Adesso verifica che sul tuo PC, o su quello della scuola, sia installato Python.
Se ti occorre puoi scaricare liberamente il programma dalla rete (http://www.python.org/download/); ricordati naturalmente di scegliere la versione adatta al sistema operativo installato sul tuo PC.

Adesso siamo "quasi" pronti a cominciare. Prima però…

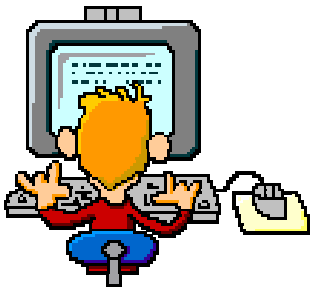
PRIMA DI PARTIRE…

VERIFICHIAMO DI POSSEDERE I REQUISITI

Prima di tutto è bene verificare di possedere i requisiti necessari per partecipare al corso. **Cosa devi fare?** Niente di speciale ma **prova a chiederti se:**

1. **sai utilizzare il computer?** ovvero:
   scrivere, creare file, salvarli, stampare e, naturalmente, fare cose come navigare in Internet e utilizzare i motori di ricerca per trovare le informazioni che vi servono.
2. **ti piace la matematica?**
3. **sei ordinato?**

Bene, tutto questo non è proprio indispensabile ma…aiuta! Se così non fosse, nessuna preoccupazione, a tutto c'è rimedio.
Un suggerimento?



Sarebbe utile conoscere il sistema di numerazione binario e il codice ASCII, capire il concetto di variabile, sapere che esistono gli algoritmi, sapere cos'è un diagramma di flusso e come si costruisce, avere un'idea di come è fatto e di come funziona un computer…

Puoi trovare tutto questo sui tuoi libri scolastici e concordare con i tuoi insegnanti un momento di ripasso prima di iniziare. Puoi anche creare un piccolo gruppo di discussione (in classe oppure online) e confrontarti su questi argomenti con i tuoi compagni.
Molto materiale è disponibile anche in rete.

Per quanto riguarda l'ordine dovrai fare molta attenzione a gestire l'archivio dei tuoi materiali, creandoti delle cartelle e salvando con cura i documenti. Puoi stampare questo manuale scaricandolo dalla rete e crearti un dossier cartaceo dove raccogliere i materiali che ritieni più utili e interessanti.

Nelle pagine che seguono ti viene fornito un breve, sintetico ripasso di alcuni argomenti. Se lo ritieni utile, consulta il nostro ABC prima di passare allo step 1.
Se invece sei già ferratissimo in materia…passa direttamente allo step 1.

Buon lavoro!

Prima di partire, ripassiamo l'ABC

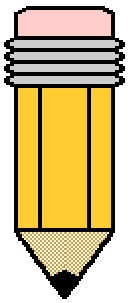
A: Il sistema di numerazione binario

L'uomo usa normalmente il sistema di numerazione decimale (probabilmente perché ha dieci dita), ma questo non è l'unico sistema ad essere utilizzato (il contadino conta le uova a dozzine e se tu devi misurare il tempo procedi di sessanta in sessanta…).
Il sistema di numerazione decimale si dice anche in base dieci e usa i simboli (o cifre): 0,1,2,3,4,5,6,7,8,9 con cui si scrivono tutti i numeri.
La base di un sistema di numerazione corrisponde al numero di simboli usati per scrivere tutti i numeri.
Un sistema di numerazione molto importante per il calcolatore è il **sistema binario o sistema di numerazione in base due, con le cifre 0 e 1.**
In pratica il calcolatore usa solo due dita!
Con i due simboli 0 e 1 si possono rappresentare tutti i numeri.
Per passare da un numero decimale al corrispondente binario si devono eseguire delle divisioni in successione.
Proviamo a trasformare il numero 25 nel suo corrispondente binario:

25: 2= 12 12: 2= 6 6: 2= 3 3: 2= 1 1: 2= 0	resto 1 resto 0 resto 0 resto 1 resto 1		**I resti delle divisioni, scritti in ordine inverso, corrispondono al numero in base 2**

A 25 corrisponde pertanto il numero binario 11001.

Ma perché scrivere con il sistema binario?
È più veloce scrivere 25 che 11001.



O 1

Tieni presente che nel sistema binario ogni numero può essere scritto con due soli segni 0 e 1 facilmente traducibili dal calcolatore in segnali elettrici alti o bassi (potrai sentir dire semplicemente "acceso o spento").
Ciò aiuta il computer ad eseguire velocemente delle operazioni anche molto lunghe : il più semplice dei computer può infatti calcolare milioni di operazioni al secondo del tipo:
390387626438 X 298436476488.

Ma non ti preoccupare, per fortuna l'uso del sistema binario riguarda soltanto i suoi circuiti interni e sarà lo stesso calcolatore, come vedremo più avanti, a tradurre automaticamente per noi i numeri.

B: Codici e linguaggi

Se il semaforo è rosso ti fermi, se è verde attraversi la strada. Al mare se sventola la bandiera gialla devi fare attenzione ma se c'è bandiera rossa è proibito fare il bagno perché il mare è troppo agitato. In tutti e due gli esempi descritti si è utilizzato un codice.
Di codici ne esistono tanti: c'è il codice Morse, quello postale, quello fiscale…….
Insomma, **gli uomini stabiliscono dei segni convenzionali che hanno per tutti un preciso significato e che chiamano codici.**
Anche noi usiamo dei codici per comunicare con il calcolatore. Sappiamo che il calcolatore preferisce usare i numeri binari perché gli è facile trasformare i segni 1 e 0 in segnali elettrici alti o bassi.
Ma se noi facciamo corrispondere a un numero binario ogni carattere della tastiera e anche i comandi fondamentali, siamo in grado di comunicare al computer numeri, lettere e ogni altra informazione necessaria per farlo funzionare. Con otto cifre binarie a disposizione, possiamo codificare 256 tra lettere, numeri e anche comandi del calcolatore.

Definizione

Bit: è la più piccola unità di informazione che possiamo trasmettere al calcolatore e corrisponde a una cifra binaria 0 oppure 1.
Byte: è un insieme di otto bit, ovvero di otto cifre binarie, generalmente utilizzato per indicare un elemento di informazione.
Per misurare la capacità di memorizzazione (ad esempio della RAM) del computer, si usano i multipli del byte:
kilobyte (kB) = 1.000 byte
Megabyte (MB) = 1.000.000 byte
Gigabyte (GB) = 1.000.000.000 byte

Uno dei codici più utilizzati è il codice ASCII(si pronuncia aschi) e significa "Codice Standard Americano per lo Scambio dell'Informazione". In una apposita tabella sono riportati i vari caratteri e comandi in codice ASCII corrispondenti ai singoli numeri del sistema di numerazione decimale, binario ed esadecimale.
È anche quello che utilizzerai tu sul tuo computer.
Un esempio: Meo in codice ASCII si scrive : 010011010110010101101111
Collegati al sito: http://www.asciitable.it/ascii.asp. Approfondisci l'argomento e scarica la tabella estesa.
Adesso divertiti a rispondere alla domanda di cane Moon!

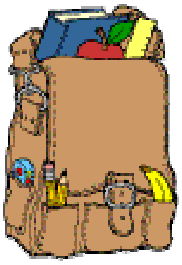
Sappiamo bene che il computer può fare molte cose: scrivere, calcolare, disegnare, etc. ma per fare tutto ciò ha bisogno del software, ovvero dei programmi, che sono un insieme di istruzioni che un programmatore ha compilato utilizzando uno specifico linguaggio detto **linguaggio di programmazione.** Esistono tanti tipi di linguaggi, ognuno adatto a particolari tipi di problemi: **Logo, Basic, Visual Basic, C, Java…**
Sostanzialmente i linguaggi si dividono in: **linguaggi compilati e linguaggi interpretati.**
Quando si usa un linguaggio bisogna rispettare attentamente le sue regole, ovvero la sintassi e il lessico.
Ricordatelo quando inizierai a programmare…!

C: Algoritmi e diagrammi di flusso

Ti sei mai soffermato a pensare a quanti problemi risolvi in una giornata, a quante decisioni prendi?
Cosa mangio a colazione? come mi vesto? quali amici invito alla festa? cosa guardo in tv? Forse non lo sai ma inconsapevolmente, ogni giorno, costruisci tantissimi algoritmi (procedimento che risolve un problema).

definizione

« Algoritmo: sequenza logica di istruzioni elementari (univocamente interpretabili) che, eseguite in un ordine stabilito, permettono la soluzione di un problema in un numero finito di passi **»**

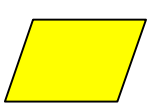
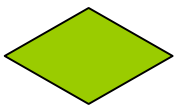
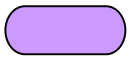


Quando ti si presenta un problema e vuoi creare un algoritmo, devi innanzitutto riflettere, individuare gli elementi noti, ovvero i DATI, dai quali devi partire per arrivare a trovare gli altri elementi del problema, le INCOGNITE. Solo dopo aver chiarito bene quali sono i dati e le incognite puoi passare alla fase risolutiva del problema. Un esempio? Risolvi il seguente problema: Oggi devi andare a scuola, hai lezione di italiano, francese e ginnastica.
Come fai?

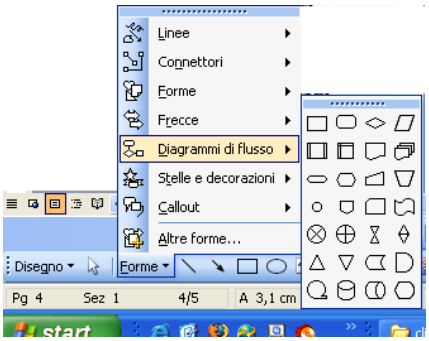
1. **Identifichi il problema:** devo andare a scuola e devo mettere in cartella tutto il necessario
2. **isoli i dati:** italiano,francese, ginnastica
3. **le incognite:** cosa metto nello zaino?

cerchi la soluzione migliore: metto nello zaino tuta e scarpe da ginnastica, libro e quaderno di Italiano e francese, diario, portapenne e merenda. Lascio a casa il vocabolario perché è troppo pesante e posso usare quello della biblioteca.

I diagrammi di flusso (in inglese "flowchart") servono a rappresentare graficamente,in modo chiaro e preciso la successione delle operazioni necessarie per risolvere un problema, (ovvero gli algoritmi).
Per realizzarli si usano dei "blocchi" in cui inserire dati, informazioni, comandi:
Blocchi di ingresso o uscita: i dati iniziali e finali vanno inseriti in *parallelogrammi*
Blocchi di istruzioni: le istruzioni e le operazioni vanno inserite in *rettangoli*
Blocchi di decisione: quando ci sono momenti in cui si può scegliere tra due possibilità alle quali si può rispondere con un si o con un no i dati relativi si inseriscono in *rombi*
Blocchi di inizio e fine: l'inizio e la fine del diagramma si indica con un ovale.

Blocchi di ingresso/uscita	Blocchi di istruzioni	Blocchi di decisione	Blocchi di inizio/fine
			
Parallelogrammo	**Rettangolo**	**Rombo**	**Ovale**
Le frecce indicano la direzione da seguire			

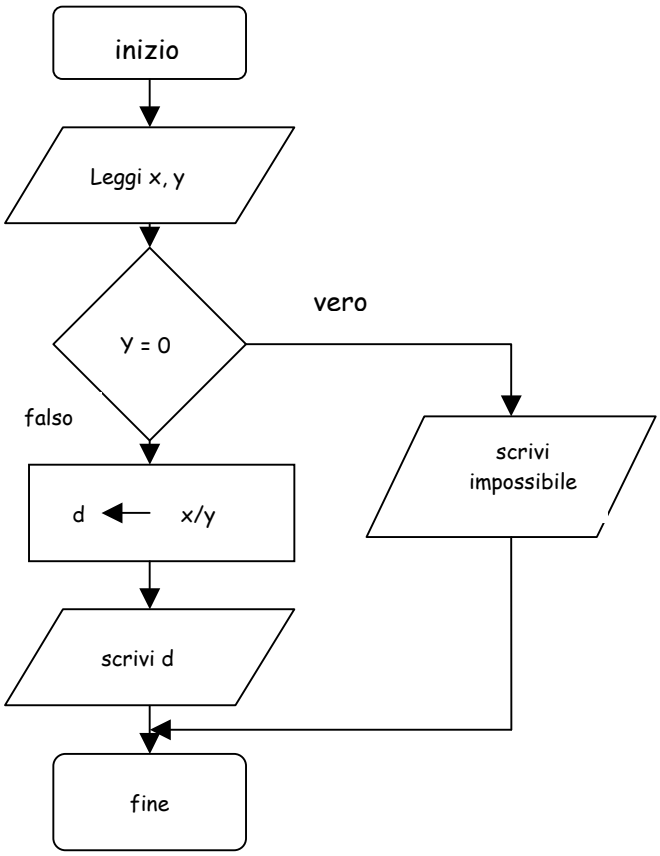
Per disegnare un diagramma di flusso puoi usare gli strumenti grafici di Word o di Writer :



Proviamo insieme a creare l'algoritmo di un semplice problema: << qual è il risultato della divisione 22 :11?>> e a disegnarne poi il diagramma di flusso.
Iniziamo creando un algoritmo di tipo generico capace di risolvere una divisione:

1. Inizio algoritmo;
2. acquisire i valori **x** e **y**;
3. se **y**=0 la divisione è impossibile andare al punto 6;
4. calcolare **x** diviso **y** e mettere il risultato in **z**;
5. comunicare il risultato **z**;
6. fine algoritmo.

Proseguiamo disegnando il diagramma di flusso:



Puoi utilizzare questo algoritmo tutte le volte che devi fare una divisione, è sufficiente seguire le istruzioni nell'ordine e attribuire i valori a **x** e **y** (esempio: x= 22; y= 11).
Il computer però non può usare questo algoritmo perché non capisce il nostro linguaggio, occorre tradurre l'algoritmo utilizzando un linguaggio conosciuto dal computer, come ad esempio Python, ovvero **creare un programma**.
Un programma è la descrizione di un algoritmo in forma comprensibile, e quindi eseguibile, dal computer.
L'elaborazione è la fase in cui il programma viene eseguito dal computer.

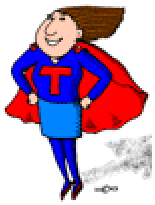
Ecco una piccola anticipazione. Questo è il programma che scriverai per istruire il tuo computer a fare le divisioni. Ricordati che stiamo usando Python come linguaggio.

Questo è il programma:

```
x=input("scrivi la x ")
y=input("scrivi la y ")
if y == 0:
    print "la divisione è impossibile"
else:
    z=x/y
    print "x diviso y è uguale a ", z
```

E questo è quello che otterrai come risultato:

```
scrivi la x 6
scrivi la y 0
la divisione è impossibile
```



Riassumendo:
identifico un problema, isolo i dati, le incognite, cerco la soluzione migliore. Creo l'algoritmo e lo disegno usando il diagramma a blocchi (o di flusso). Scelgo un linguaggio di programmazione e traduco il mio algoritmo. E poi?

Bene, è arrivato il momento di iniziare a esplorare il mondo della programmazione.

Siamo pronti a incominciare!

Passo dopo passo……impariamo a programmare!
Buon lavoro!

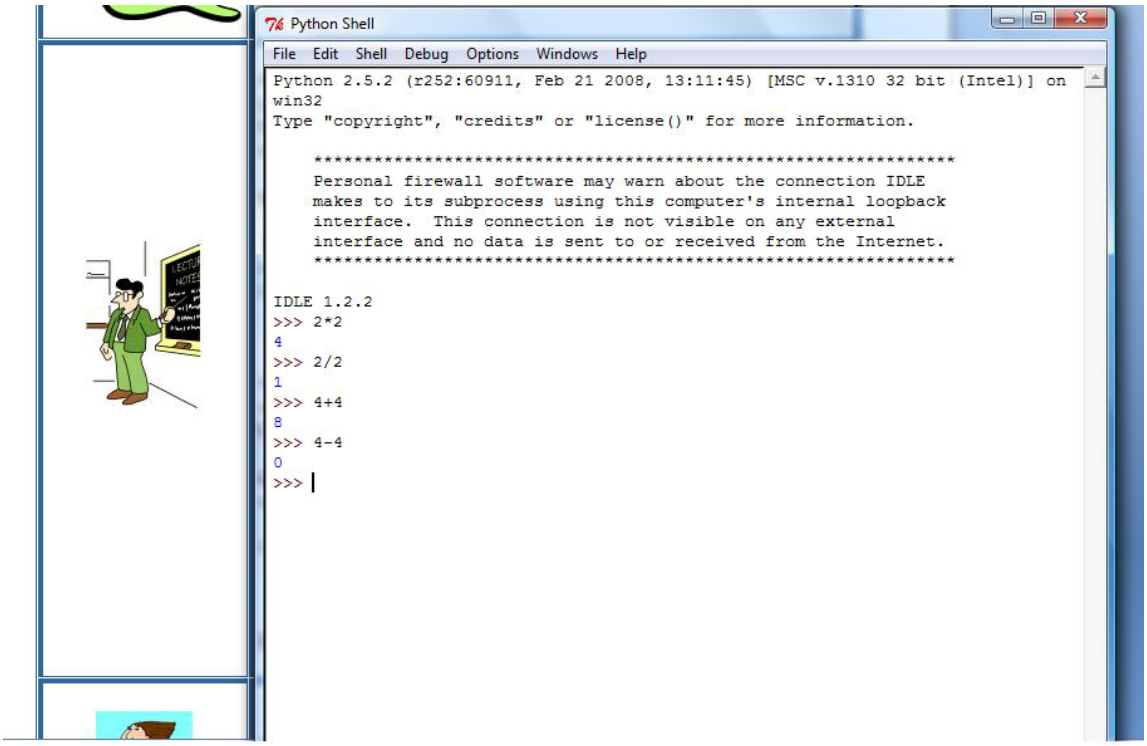
STEP 1
IL PROGRAMMA

In questo step vedremo:

* perchè usare Python
* come usare Python come calcolatrice:

  >>> 6*12/4.-8

  10.0



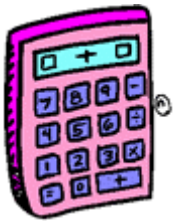
STEP 1 Ovvero, un passo dopo l'altro impariamo a programmare	**Il programma**
	Il corso utilizza un linguaggio di programmazione che si chiama **Python*.** Perché?
	Perché è adatto ai "principianti", è facile da imparare (e da insegnare) e consente di affrontare quasi tutti i temi legati alla programmazione. Non è usato solo dagli studenti ma anche dai programmatori professionisti e molti "sviluppatori" se ne occupano attivamente. **E' open-source** e può essere utilizzato liberamente.** Troverai il programma (Python) installato sui computer del laboratorio a scuola ma, se così non fosse o se desideri esercitarti a casa puoi scaricare il programma dal sito: http://www.python.org/download/ facendo attenzione a scegliere la versione adatta al sistema operativo installato sul tuo calcolatore (Windows, Linux, Mac OS X, OS2, Amiga). Es.: Python 2.5 Windows installer. **Quando avvii il programma sul video del tuo computer** compariranno delle scritte che riguardano anche informazioni sul copyright. Leggile per tua informazione e poi posizionati direttamente dopo il simbolo che indica il prompt: **>>>** **Ogni volta che il computer visualizza questo simbolo significa che Python attende le istruzioni ed è pronto a lavorare.**
	Rifletti *Sai perchè questo linguaggio si chiama così? **Ti sei chiesto cosa significa open-source? Più avanti ti accorgerai che definiremo Python "programma interprete", prova a pensare perché.

Usare Python come calcolatrice

Sul tuo computer clicca su: start, programmi, Python, IDLE e avvia il programma. **Inizialmente useremo Python Shell.**
Avviato il programma sei pronto a dare le prime istruzioni!

Un'istruzione è un'operazione che l'interprete di Python è in grado di eseguire.

Proviamo ad usare Python come fosse una calcolatrice, ovvero, impariamo a dare istruzioni al computer perché compia alcune operazioni.
Già, ma quali simboli dobbiamo usare per indicare le operazioni che vogliamo fare?



*** per la moltiplicazione 2*2**
/ per la divisione 2/2
+ per l'addizione 2+2
- per la sottrazione 2-2
l'elevamento a potenza si indica con **
esempio: 22 significa due elevato alla seconda**

Adesso prova a fare cinque semplici esercizi: scrivi l'operazione che vuoi eseguire usando i simboli e dopo ciascuna dai invio, come nell'esempio sotto.

>>>2*2 4	>>>2/2 1	>>>2+2 4	>>>2-2 0	>>>2**2 4

Definizione

I valori che usiamo nei calcoli si chiamano **operandi,** mentre gli **operatori** sono i simboli delle operazioni.

Addizione, sottrazione, moltiplicazione ed elevamento a potenza si comportano come tu ti aspetti da loro, ma per la divisione non è così.

Prova a scrivere:
>>>5/2 (e poi dai invio) otterrai:
2
Verifica ancora:
>>>15/4 (e poi dai invio) otterrai:
3

Come mai? Sai trovare la soluzione?
Se si, bene, se no troverai la risposta nella pagina seguente, ma sforzati di riflettere e di ragionare per risolvere da solo il quesito.

Soluzione al quesito:
Per Python la divisione tra due numeri interi è sempre un numero intero approssimato per difetto.

Come fare allora?
Basta usare la divisione in virgola mobile!

Ovvero: scriviamo il numero seguito dal punto (non la virgola) come nell'esempio:

>>>5.0/2 (invio)
2.5

Perché il punto e non la virgola?
Perchè gli americani fanno così!

Proviamo adesso ad impostare delle semplici espressioni, ma ….. **potrai fare uso solo delle parentesi tonde.**

In Python quando lavoriamo con i numeri non si possono usare le parentesi quadre e quelle graffe e si deve fare molta attenzione ad usare le parentesi tonde nel modo giusto, **iniziando a fare le operazioni contenute nelle parentesi più interne.** Proviamo a scrivere:
>>>(3+2)*5 (invio)
25
E ancora:
>>>((3+2)*5)+3 (invio)
28

Python segue le stesse regole della matematica per quanto riguarda l'ordine di esecuzione delle operazioni:
prima le parentesi (partendo da quelle più interne), poi l'elevamento a potenza, poi moltiplicazione e divisione e infine somma e addizione. Quando due operatori hanno la stessa priorità si procede da sinistra verso destra.
Prova a scrivere e verifica:

>>>2*(5-2) **6**	>>>(1+1)**(7-4) **8**	>>>2**1+1 **3**
>>>2*5-2 **8**	>>>2/3 **0**	>>>2/3-1 **-1**

Se in una divisione ti interessa solo sapere il resto usa il simbolo %:
>>> 15%12
3
Se invece vuoi calcolare la $\sqrt{\ }^2$… è un po' più difficile, così te lo spiegherò fra un po' .

Esercitiamoci un po'

puoi usare Python Shell per fare i calcoli anche quando fai i compiti...

1. Prova a calcolare 153/4 e 73.0/8: che differenza c'è tra queste due operazioni?

2. a) 5*6+(4,5*6)
   b) 4+4.5-(6*4/2)
   quale di queste due espressioni contiene un errore?

3. Scrivi l'espressione per calcolare "quanti mesi hai".

4. Inventa un'espressione che dia come risultato 48 ed una che dia come risultato 11.

5. a) Per andare da casa di Sandrone a casa di Giulia ci sono 3 km
   b)per andare da casa di Giulia a casa di Clotilde ci sono 4 km
   scrivi un'espressione che calcoli quanti km deve fare Sandrone per andare a trovare Giulia e Clotilde e tornare a casa ripassando da casa di Giulia.

6. Ora calcola quanti km ci vogliono per andare a trovare i tuoi 4 migliori amici e poi tornare a casa.

7. Calcola la lunghezza della tua "spanna"(la distanza tra il pollice e il mignolo in estensione) ed ora misura con la tua spanna il banco di scuola. Trova l'area del ripiano del banco.

8. Prendi il numero di telefono della mamma o del papà e prendi il numero di telefono della scuola:
   - Moltiplica i due numeri
   - Dividi i due numeri
   - Sottrai i due numeri
   - Eleva al quadrato i due numeri

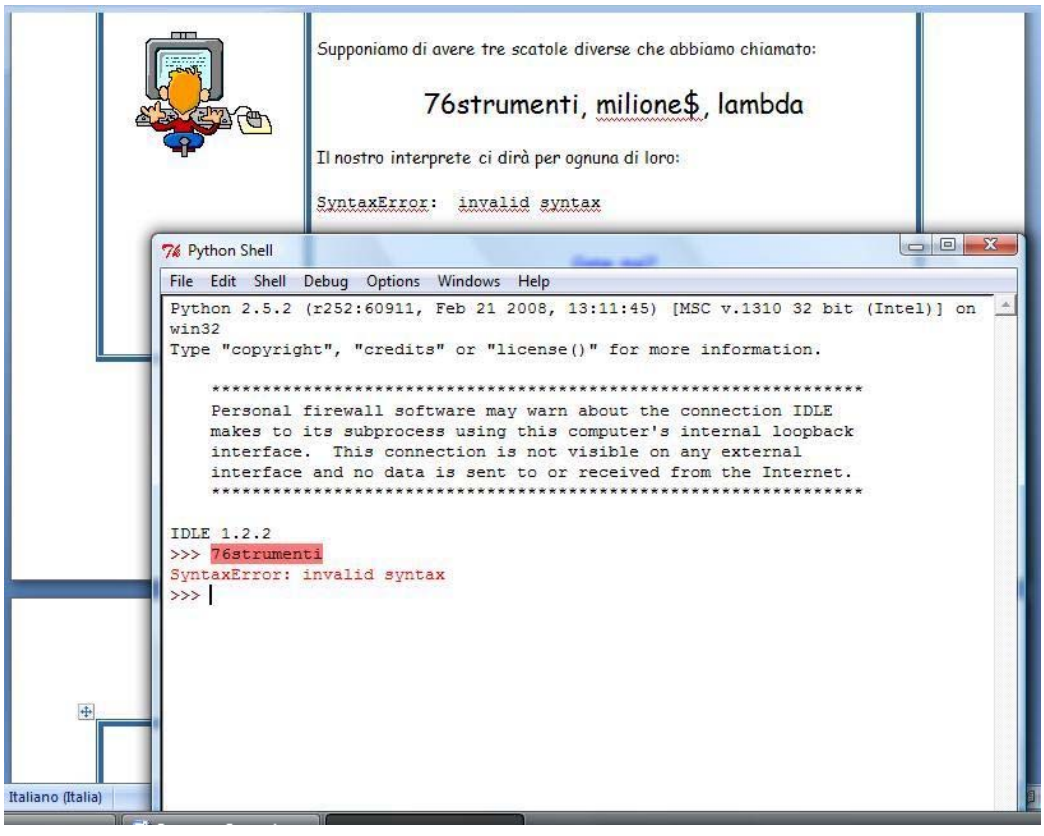
9. Calcola l'area della cucina di casa tua.

10. Calcola il diametro, la circonferenza, l'area di un cerchio di raggio 2.5 cm.

STEP 2
LE SCATOLE

In questo step impareremo:

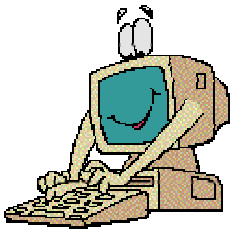
* L'uso delle scatole
* I nomi delle scatole che si possono usare e quelli "illegali"



STEP 2

Ovvero passo dopo passo impariamo a programmare

Le Scatole

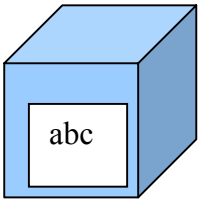


Istruire un calcolatore significa essenzialmente creare e usare degli oggetti.

Tra questi oggetti quelli di uso più comune sono quelli che chiameremo **scatole.**

Le scatole vengono usate per contenere numeri, caratteri, parole o frasi.

Immaginate la parete di una cantina suddivisa in tante scatole piccole e grandi per contenere gli svariati oggetti che si ammassano in cantina.
Queste scatole contengono tanti tipi di oggetti e hanno davanti un'etichetta che ci permette di individuare immediatamente il loro contenuto.
Senza queste etichette le scatole sulla parete sarebbero assolutamente inutili.



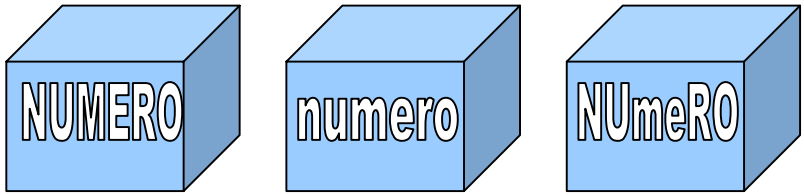
Le nostre scatole sono del tutto simili a questa.

Ogni scatola che noi creiamo deve avere un nome. Il nome che assegniamo alla scatola e' l'equivalente dell'etichetta sulla scatola della cantina.
Ovviamente dovremo scegliere dei nomi significativi per le nostre scatole per documentare così a cosa servono. Ad esempio: SCATOLA1, SCAT1, SAL1, SAL2, SALAME, PIPPO, PIPPO2A, PIPPO4C, SCARPEVECCHIE, VINO.
Sono validi anche nomi molto corti come: A, B, C, A1, B3 o lunghi come:

ILNOMEPIULUNGOCHEMIVIENEINMENTEPERILMIOCANE

I nomi delle scatole possono essere lunghi quanto si desidera e possono contenere sia lettere che numeri, ma devono sempre iniziare con una lettera oppure con il carattere "_".
È legale usare sia lettere maiuscole che minuscole.
Ricordatevi comunque che il nostro computer interpreta in modo diverso i caratteri minuscoli dai caratteri maiuscoli.

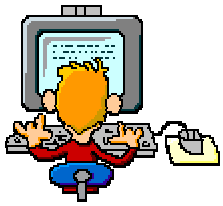
Quindi le scatole che si chiamano:



sono per il calcolatore tre scatole diverse.

Il carattere di sottolineatura (_) può far parte di un nome ed è spesso usato in nomi di scatole composti da più parole. Ad esempio: il_mio_nome, il_prezzo_del_pane, _il_mio_cane
In alternativa i nomi possono essere composti usando l'iniziale maiuscola per ciascuna di esse, con il resto dei caratteri lasciati in minuscolo come in IlMioNome, IlPrezzoDelPane, Le_scarpe_del_bimbo
Sembra che tra i due metodi quest'ultimo sia il più diffuso.

Non e' possibile invece "battezzare" la nostra scatola con nomi come 1A, 3P, 4SAL perchè il suo nome non può iniziare con un numero, ma solo con una lettera. **Assegnando un nome di questo tipo ad una scatola otterremo un messaggio di errore di sintassi. L'interprete ci dirà che il nome che abbiamo usato e' illegale ma non dirà perchè è illegale, dovremo scoprirlo noi.**



Vediamo cosa dice il calcolatore quando sbagliamo il nome di una scatola.

Supponiamo di avere tre scatole diverse che abbiamo chiamato:

76strumenti, milione$, lambda

Il nostro interprete ci dirà per ognuna di loro:

```
SyntaxError:  invalid syntax
```

Come mai?

Prova a trovare l'errore nei nomi delle scatole. Se non ci riesci leggi la risposta nella sessione successiva.

Soluzione al quesito:

Il nome 76strumenti e' illegale perchè non inizia con una lettera, m**ilione$** e' illegale perchè contiene un carattere proibito, il $,
lambda e' illegale perchè è una delle parole riservate di Python.

Tutti i linguaggi di programmazione hanno alcune parole riservate che ne definiscono le regole e non possono essere usate come nomi delle scatole. Le parole riservate di Python sono 28 e sono le seguenti:

and	**continue**	**else**	**for**
import	**not**	**raise**	**assert**
def	**except**	**from**	**in**
or	**return**	**break**	**del**
exec	**global**	**is**	**pass**
try	**class**	**elif**	**finally**
if	**lambda**	**print**	**while**

Esercitiamoci un po' utilizzando Python Shell

1. Puoi dare a una scatola il nome 10ART?

2. Quale di questi nomi è sbagliato?
Cane_M_4 CaneM4 4cane_M CANE_M4

3. Puoi dare a due scatole diverse i nomi SCATOLA_1 e Scatola_1?

4. Puoi chiamare una scatola Print?

5. Se chiami una scatola Zio Pippo, cosa succede?

Adesso prova ad inventare tu qualche nuovo esercizio.

Continua ad esercitarti utilizzando Python Shell.

Il contenuto delle scatole

Le nostre scatole sono state create per avere dei contenuti.
Dobbiamo fare attenzione a non confondere il nome della scatola con il suo contenuto.
Il nome di una scatola non cambia mai mentre il suo contenuto cambierà spesso.
Ad esempio la scatola PIPPO potrà contenere, in un certo momento, il numero 8, poi il numero 999 e quindi il numero 2.5.

Un programmatore conosce sempre il nome della scatola perché è stato lui a "battezzarla", ma di solito non ne conosce il contenuto, perchè può averlo dimenticato oppure perchè la scatola è stata utilizzata per calcoli complicati.

Tuttavia il programmatore in qualsiasi momento può aprire la scatola e controllarne il contenuto.

Per ricordare meglio la differenza fra nome e contenuto di una scatola, immaginiamo di aver scritto il nome della scatola con un pennarello indelebile sulla scatola stessa e che il contenuto sia stato scritto su un foglietto che in qualunque momento può essere tolto dalla scatola e sostituito con un altro foglietto.

Il numero che noi inseriamo nella scatola e' il suo valore numerico.

Un valore e' una delle cose fondamentali manipolate da un programmatore, come le lettere dell'alfabeto nella scrittura. I valori possono essere di tipo diverso, numeri e caratteri.

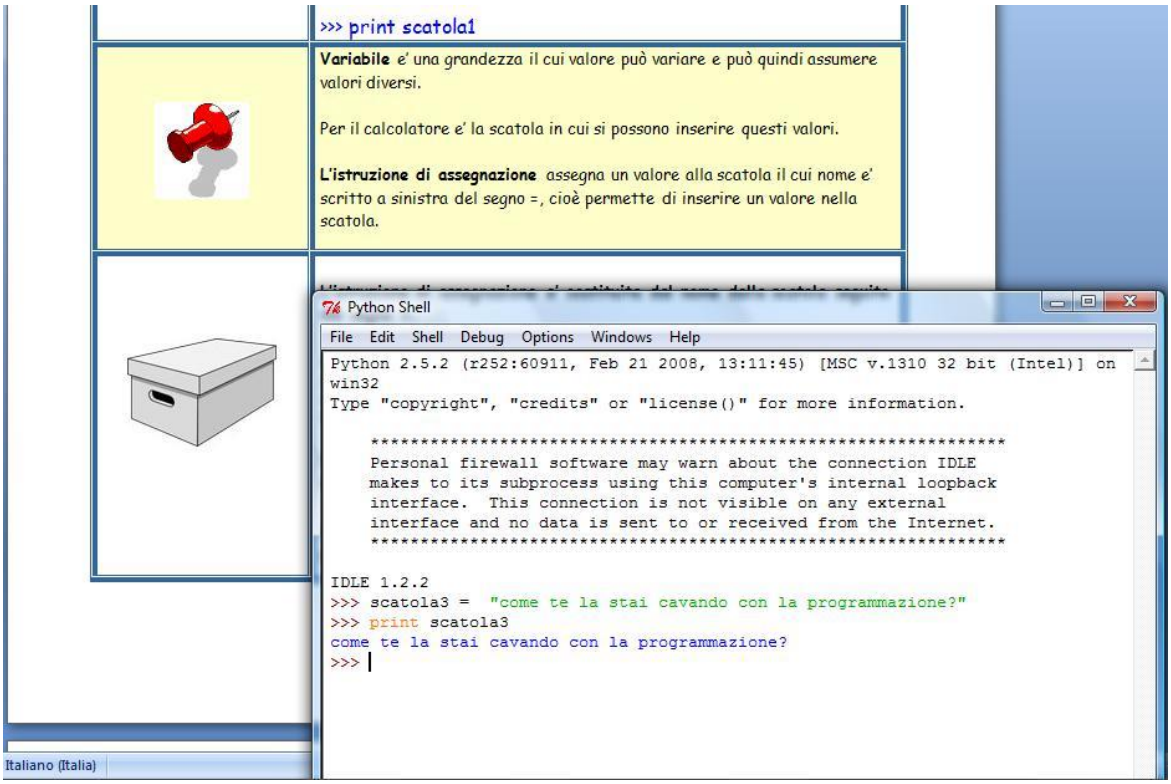
Come faccio ad aprire la scatola e a controllare il contenuto?

Prova a riflettere: come si può fare ad inserire un valore in una scatola e poi a controllarne il contenuto?
Se non riesci a trovare la soluzione, la spiegazione e' nella scheda STEP 3.

STEP 3
LE VARIABILI
L'istruzione di assegnazione e l'istruzione di stampa

In questo step impareremo:

* come assegnare dei valori alle scatole:
  >>> scatola1=3
* a stampare il contenuto delle scatole:
  >>> print scatola1
  3



STEP 3
Ovvero,un passo dopo l'altro impariamo a programmare

Le variabili, l'istruzione di assegnazione e l'istruzione di stampa

Il modo più semplice per farsi consegnare una scatola dal calcolatore, darle un nome e riempirla con un numero e' quello di scrivere una frase come

>>> SCATOLA1=7

La scatola rappresenta quella che matematicamente viene definita **variabile**. Ad essa e' possibile assegnare un valore con l'istruzione di assegnazione.
Nella parete piena di scatole in cantina le variabili sono le etichette sulle scatole.
Il valore di una variabile può essere di tipo diverso: numero, carattere o serie di caratteri.

L'istruzione di assegnazione contiene un ordine per il calcolatore e costituisce un primo esempio di "istruzione", ovvero un'operazione che Python capisce ed e' in grado di eseguire.

L'istruzione di assegnazione non produce risultati visibili.

Per ordinare al calcolatore di mostrare il contenuto di una scatola dobbiamo usare l'istruzione di stampa, scrivendo :

>>> print scatola1

Variabile e' una grandezza il cui valore può variare e può quindi assumere valori diversi.

Per il calcolatore e' la scatola in cui si possono inserire questi valori.

L'istruzione di assegnazione assegna un valore alla scatola il cui nome e' scritto a sinistra del segno =, cioè permette di inserire un valore nella scatola.

L'istruzione di assegnazione e' costituita dal nome della scatola seguito dal segno =.

Ad esempio:

```
SCATOLA1 = 7
SCATOLA2 = 3.14
SCATOLA3 = "come va?"
```

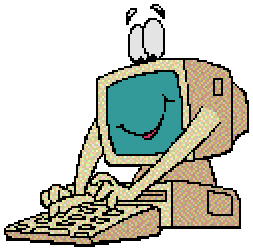
Si noti che il simbolo = non ha esattamente il significato matematico che conosciamo. Nell'istruzione di assegnazione

SCATOLA1 = 7

il simbolo = significa:
nella scatola di nome SCATOLA1 metti il numero 7.

In Python le variabili assumono il tipo del valore che viene loro assegnato: numero, carattere o serie di caratteri, e lo mantengono fino a quando non viene loro assegnato un valore di tipo diverso.

L'istruzione di stampa – print - permette di visualizzare il contenuto delle variabili sullo schermo del calcolatore.
Con questa istruzione ordiniamo al calcolatore di comunicare il contenuto di una scatola.



Ecco come fa il calcolatore a riempire una scatola con un numero e farci vedere il contenuto, se la scatola di nome SCATOLA1 non e' mai stata usata.

1. mette a disposizione una scatola vuota
2. battezza la scatola scrivendo in modo indelebile sul coperchio il suo nome (che in questo caso e' SCATOLA1)
3. mette dentro la scatola un foglietto sul quale e' scritto il numero 7

Quando e' presente l'istruzione print il calcolatore:

1. cerca la scatola indicata (in questo caso SCATOLA1),
2. la apre e ne legge il contenuto,
3. stampa sullo schermo il contenuto (in questo caso 7).

Facciamo un esempio:

```
>>> scatola1 = 8+4
>>> print scatola1
12
```

Sul video comparirà il risultato dell'addizione: 12.

Il calcolatore ha messo a disposizione la scatola di nome SCATOLA1 e dopo aver svolto i calcoli a destra del segno = ha posto nella scatola un foglietto contenente il risultato del calcolo. Quindi con l'istruzione print abbiamo ordinato al calcolatore di comunicare il contenuto della scatola.

Ricordati di scrivere print sempre minuscolo!

Cosa succede se SCATOLA1 e' invece già stata usata?
Utilizzando l'interprete di Python, prova a scrivere la sequenza di operazioni e le istruzioni per il calcolatore e poi controlla di seguito la soluzione.

Il calcolatore fa quanto segue:

1. cerca la scatola che ha nome SCATOLA1,
2. apre la scatola e toglie il contenuto lasciandola vuota, ovvero
3. toglie il foglietto che era stato introdotto prima,
4. mette nella scatola il nuovo contenuto che e' il foglietto con scritto il numero 7.

Quindi ripete le operazioni da 1 a 3 per il print.

Le istruzioni per il calcolatore saranno:

```
>>> scatola1 = 7
>>> print scatola1
7
```

Esercitiamoci un po'.

1. Scrivi in Python la seguente istruzione per il calcolatore: dammi la scatola GAIA3 e mettici dentro il numero 10.
2. Illustra i passaggi attraverso i quali il calcolatore inserisce un numero in una scatola che era già stata usata in precedenza.
3. Qual e' il significato dell'istruzione print?
4. Se hai la scatola MELA2 che contiene il numero 7, quale istruzione devi scrivere per visualizzare il contenuto?
5. Cosa visualizza il calcolatore se scrivo:

```
SCATOLA2 = 10-3
 print SCATOLA2

SCATOLA3 = 7 * 4
 print SCATOLA3

SCATOLA4 = 24/8
 print SCATOLA4
```

6. Se adesso scrivo:
```
SCATOLA3 = 20/2 * 5
Print SCATOLA3
```
Cosa contiene SCATOLA3? E se scrivo PRINT SCATOLA3 ?

E' arrivato il momento di…	**CONSOLIDARE LE CONOSCENZE**
	Fatti i primi tre passi! Facciamo un attimo di pausa e concediamoci un momento di riflessione. Prova a domandarti se: 1. Hai seguito con attenzione? 2. Capito bene tutto quello che hai letto e (spero) studiato? 3. Hai completato gli esercizi assegnati? 4. Provato a inventare nuovi esercizi? Pensi di si? Molto bene. Prima però di proseguire nel nostro percorso di studio, dobbiamo essere sicuri non solo di avere compreso tutto quello che hai letto nelle schede, ma anche di averlo ben assimilato e memorizzato. *Insomma, dobbiamo fare un po' di allenamento ed esercitare la nostra mente.* La professoressa Martina si è divertita a inventare un po' di esercizi da farti fare e adesso tocca a te risolvere i semplici problemi che troverai nella pagina seguente. Ma prima, leggi il post-it.
	All'inizio dello step 1 ti ho posto tre domande, hai trovato le risposte? No? Allora incomincio a rispondere alla prima (perché questo linguaggio si chiama Python?) dandoti qualche informazione in più. Python è "*un linguaggio di programmazione interpretato, interattivo e orientato a oggetti*". Il suo inventore è un geniale signore olandese: Guido Van Rossum. Nel Natale del 1989 decise di passare le vacanze scrivendo un linguaggio che correggesse i difetti che, secondo lui, erano presenti in altri linguaggi. Dopo di lui moltissimi altri sviluppatori hanno proseguito il suo lavoro. A Guido piaceva tantissimo un gruppo di comici inglesi famosi negli anni sessanta: i Monty Python. A loro e alla loro comicità un po' demenziale ha dedicato il suo lavoro. Evidentemente dovevano piacere anche a molti altri nell'ambiente, perché non è l'unica volta che questi comici hanno dato il nome a qualcosa di informatico. Il termine ***spam,*** che viene utilizzato per indicare la posta elettronica indesiderata, deriva da un loro sketch, in cui compariva un ristorante nel cui menù erano inseriti tutti piatti ricoperti di spam, un tipo di carne macinata in scatola, particolarmente disgustosa. Guido, dopo tanta fatica, ha deciso di donare a tutti il suo lavoro, così oggi noi possiamo disporre liberamente e gratuitamente di questo linguaggio. Nel mondo dell'informatica queste cose succedono.. ne riparleremo. Per adesso grazie a Guido e a tutti coloro che hanno continuato la sua opera.

E' arrivato il momento di…	ESERCITARCI CON PYTHON
	Prova a svolgere questi esercizi. Quando hai finito, vai al fondo del libro dove trovi le soluzioni, confrontale con quello che hai scritto e assegnati 2 punti per ogni esercizio eseguito correttamente, nessun punto se l'esercizio è incompleto o errato. Quando hai finito vai alla pagina dell' autovalutazione. Buon Lavoro.

Esercizio	Punti
Esercizio n°1: Il mago Silvan fa tanti giochi di magia: dal suo cappello a cilindro escono tre conigli bianchi e due neri. Quanti conigli sono nascosti nel cappello?	……
Esercizio n°2: Al mago Berri invece piace fare le magie con le maxi-carte: sono così grandi che quasi non stanno sul tavolo! Se ciascuna carta è lunga cm. 45 e larga cm. 30, quanto è grande la superficie di ciascuna carta?	……
Esercizio n°3: Quale superficie del tavolo occupano i quattro assi usati dal mago Berri per i suoi giochi di magia, affiancati per il lato più lungo?	……
Esercizio n°4: Il mago Gian ha un bellissimo mantello di seta nera ricamato con tante stelle argentate. Per farlo il sarto ha utilizzato ben 5 metri di stoffa. Se la stoffa costava 13 € al metro, quanto ha speso per comprarla?	……
Esercizio n°5: Se un mantello costa 80 €, un cappello a cilindro 45 €, una bacchetta magica 20 €, un mazzo di maxi-carte 13 €, quanto costa l'attrezzatura per fare il mago?	……
Esercizio n°6: Se ho 5 € di paghetta settimanale, a quanto ammonta la mia paghetta mensile? Quanti € posso spendere in un anno? Se un CD musicale costa 15 €, quanti ne posso comprare con la paghetta del mese di aprile?	……
Esercizio n°7: Se il cortile della scuola misura m. 75 di lunghezza e m. 50 di larghezza, qual è la sua superficie? Sapendo che un campo di calcetto misura m. 40 di lunghezza e m. 20 di larghezza, quanti campi di calcetto potrai ricavare dal cortile della scuola?	……
Esercizio n°8: Nella tua classe ci sono 8 maschi e 10 femmine. Se Mario è alto m. 1.55, Fabio, Matteo e Luca sono alti m. 1.60, Andrea, Aldo, Giovanni e Giacomo m. 1.50, qual è l'altezza media dei maschi della classe? Se Marta, Giovanna, Elisabetta e Francesca sono alte come Mario, Stefania, Chiara e Simonetta sono alte m. 1.50, Daria e Domitilla sono 5 cm più piccole di Arianna che è alta m. 1,68, qual è l'altezza media delle femmine della classe? Qual è l'altezza media della classe?	……
Totale parziale punti	…/16

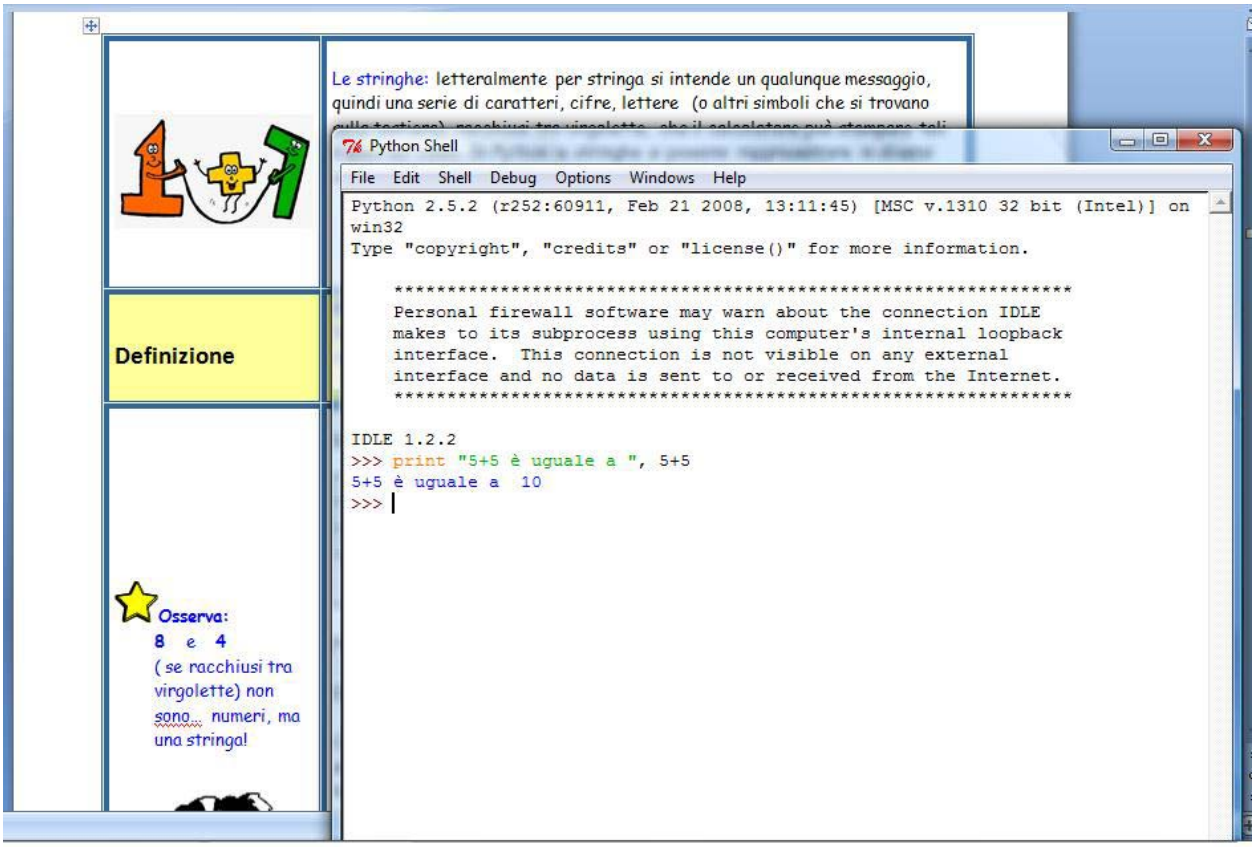
E' ARRIVATO IL MOMENTO DI …	**AUTOVALUTARCI**
	Adesso proviamo a riflettere sull'attività fin qui svolta. Per verificare se hai fatto un buon lavoro poniti le seguenti domande e rispondi (sinceramente!):
 Vediamo un po'…….	1. Ho letto con attenzione tutte le schede? 2. Le ho capite e studiate? 3. Ho completato gli esercizi assegnati? 4. Ho provato a inventare nuovi esercizi? 5. Ho rispettato i tempi assegnati? Per ogni risposta affermativa hai guadagnato 2 punti. Segna qui a fianco il punteggio ottenuto ……./**10**
Tiriamo le somme	Adesso somma il punteggio ottenuto rispondendo alle domande sopra con quello degli esercizi svolti. Esercizi ……/16 + Domande ……/10 = Totale ……/26 Se hai totalizzato meno di 25 punti ….. segui i consigli di Martina : studia di più, esercitati di più, chiedi aiuto per capire meglio, prima di proseguire l'attività.
 	Un tempo gli alunni erano soliti portare una mela ai propri insegnanti. Era un modo semplice e spontaneo per dimostrare il proprio affetto ed apprezzamento per il lavoro svolto. Se vuoi farlo anche tu, disegna qui sotto quante mele vuoi, ma che siano tutte….belle rosse come le meline di campagna! Più mele disegni, ovviamente, più ti è piaciuto quello che abbiamo fatto. ----------

STEP 4
DATI E TIPI DI DATI

In questo step impareremo:

* i **dati** e i tipi di dato: numeri, caratteri, stringhe
* che cos'è una **stringa**
* ad eseguire delle operazioni con le stringhe

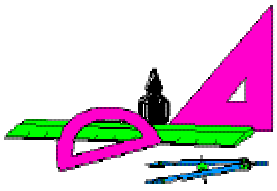
```
>>> print "ciao "*4
ciao ciao ciao ciao
```



STEP 4

Ovvero, un passo dopo l'altro impariamo a programmare

Dati e tipi di dati (numeri, caratteri, stringhe)



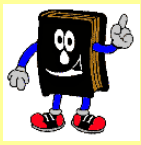
Per realizzare qualsiasi lavoro, come anche la programmazione, abbiamo bisogno di almeno tre ingredienti:

1. attrezzi
2. materiali
3. tecniche di lavoro

Ad esempio: per fare un disegno ho bisogno di:
- attrezzi: squadre, compasso, pennelli, matite,…
- materiali: acqua, fogli, cartoncini…
- tecniche: acquerello, tempera, chiaroscuro…

Gli ingredienti necessari per scrivere un programma sono:
- attrezzi: il calcolatore su cui lavorare e il software (Python)
- materiali: i dati che dobbiamo manipolare
- tecniche: le regole del linguaggio che stiamo usando.

Cerchiamo ora di scoprire cosa sono i dati o i valori, che sono una delle cose fondamentali manipolate da un programmatore.



Che cosa significa realmente la parola "**dato**"? Il dizionario dice:
"fatti o figure da cui si può trarre informazione".

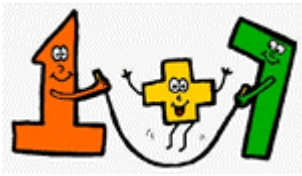
In effetti per il programmatore i dati sono la materia prima che verrà elaborata, senza i dati il nostro programma non può fare nulla.

I dati (valori) sono di tanti tipi diversi e dal tipo del dato dipendono le operazioni che possiamo fare con essi.

Esistono tipi di dati:

- semplici che includono lettere, numeri e serie di lettere
- complessi che non sono altro che una combinazione dei primi

Dei dati complessi parleremo molto più avanti, ora analizziamo insieme quelli semplici:
I numeri : sappiamo che possono essere di tipo intero, reale o "floating point" (virgola mobile o, più semplicemente, numeri decimali con virgola). Nello step 3 i nostri dati (valori) erano perlopiù numeri e abbiamo visto, ad esempio, come possiamo sommarli. L'addizione e' un'operazione che possiamo fare sui dati di tipo "numero".
I caratteri: un carattere è un simbolo come, ad esempio, la lettera "a" minuscola, la lettera "Q" maiuscola, la cifra "7", il simbolo "+" e qualunque altro simbolo che compaia sulla tastiera.



Le stringhe: letteralmente per stringa si intende un qualunque messaggio, quindi una serie di caratteri, cifre, lettere (o altri simboli che si trovano sulla tastiera), racchiusi tra virgolette, che il calcolatore può stampare tali e quali sul video. In Python le stringhe si possono rappresentare in diversi modi:

- tra virgolette semplici: 'ciao'
- tra virgolette doppie: "ciao"
- tra virgolette doppie ripetute tre volte """ciao"""

Definizione

STRINGA e' una serie di caratteri, lettere, numeri o altri simboli, racchiusi tra virgolette, che viene memorizzata dal calcolatore come un unico elemento.

Osserva:
8 e **4**
(se racchiusi tra virgolette) non sono… numeri, ma una stringa!

Ecco alcuni esempi di stringhe composte da:

una serie di lettere	"ciao"
una serie di lettere che formano un messaggio	"viva la Juve"
Una serie di simboli @#$%	"@#$%"
Una serie di simboli: anche i numeri tra virgolette vengono letti come simboli e stampati tali e quali	"8-5=3"

Prova a scrivere utilizzando Python Shell:
print "Ciao"
Il calcolatore visualizzerà tutti i caratteri compresi tra le virgolette:
Ciao
Ora scrivi: print "Viva la Juve"
Anche in questo caso il calcolatore visualizzerà tutti i caratteri racchiusi tra virgolette: Viva la Juve

Se scriviamo l'istruzione: print "8+4"
Il calcolatore visualizzerà: 8+4
senza fare nessun calcolo. Perché?
Perché hai dato un ordine al computer: *visualizza la stringa costituita da tre caratteri: 8 + 4.*

I numeri 8 e 4 sembrano effettivamente dei numeri ma essendo chiusi tra virgolette vengono interpretati come i caratteri di un messaggio qualunque e quindi come…"una stringa"!

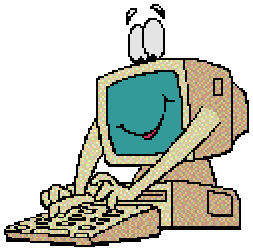
Se poi scrivi l'istruzione: print "8+4=13" il calcolatore scriverà: 8+4=13 senza accorgersi dell'errore di calcolo, perché l'ordine che gli hai dato è solo quello di scrivere la stringa composta dai caratteri: 8+4=13

Ricordati: le virgolette non fanno parte della stringa, servono solo a indicarne l'inizio e la fine.
Fanno invece parte della stringa gli spazi vuoti fra le parole del messaggio.

Ovviamente possiamo mettere la stringa in una scatola, come abbiamo fatto con i numeri, usando l'istruzione di assegnazione. **Ad esempio:**

```
>>>SCATOLA1 = "scuola media"
>>>print SCATOLA1
```

Il risultato sarà: `scuola media`



I seguenti comandi:

```
>>>print "MATEMATICA"
>>>print "Via Roma 100"
>>>print "Area"
>>>print "++++++++++++"
```

visualizzeranno rispettivamente:

```
MATEMATICA
Via Roma 100
Area
   ++++++++++++
```

Esaminiamo ora le due istruzioni:

print "8+4"	print 8+4

Il calcolatore visualizzerà:

8+4	12
In questo caso ci sono le virgolette e il calcolatore riceve l'istruzione di visualizzare i caratteri contenuti tra le " "	Visto che non ci sono le virgolette il calcolatore interpreta 8+4 come un'espressione aritmetica e la esegue visualizzandone il risultato

Proviamo ora a collegare le due istruzioni e scriviamo:

print "8+4 =", 8+4
Il calcolatore eseguirà prima l'ordine di visualizzare la stringa"8+4= " e poi quello di visualizzare il risultato dell'operazione 8+4 e quindi il risultato sarà: 8+4= 12

Facciamo ancora due esempi :

1- print "3+5= ", 3+5
   Avrà come risultato: 3+5= 8

2- print "4*4= ", 4*4
   Avrà come risultato: 4*4= 16

Sono stringhe	Sono numeri
"3+5= "	3+5
"4*4= "	4*4

Nell'esempio successivo cambia solo l'ordine con cui devono essere eseguiti gli ordini, in questo caso prima l'operazione e poi la stringa:

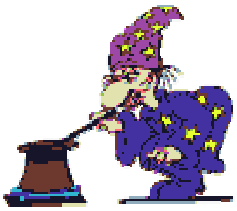
```
>>>print 100-10, " = 100 -10"
```

Il risultato sarà:

90= 100-10

" …L'operazione matematica viene inserita nella variabile con l'istruzione di assegnazione e poi…. viene dato l'ordine di visualizzare la stringa e il contenuto della variabile ….bididibodidibu' "

L'istruzione dagliela tu …bididibodidibu' !



Magico..Magogian!

Facciamo qualche esempio utilizzando le variabili…
…cercando di interpretare il difficile messaggio di Magogian!

L'operazione matematica: 14+8
viene inserita nella variabile: scatola2
con l'istruzione di assegnazione: scatola2 = 14+8
e poi viene dato l'ordine di visualizzare
la stringa e il contenuto della variabile: print "14+8= ", scatola2
il risultato sarà: 14+8= 22

Ancora un esempio: 5*3
scatola3=5*3
print "5*3= ", scatola3
il risultato sarà: 5*3= 15

attenzione!

1. La parte a sinistra dell'uguale è una stringa anche se nel messaggio in uscita sembrano tutti numeri. E pertanto 14, 8, 5, 3 non sono numeri ma caratteri.
2. Hai notato che è stata utilizzata una virgola per separare le due parti del messaggio (print "5*3= ", scatola3)? Sai perché?
L'istruzione print dice a Python: "devi visualizzare le seguenti cose".
Se vuoi fare visualizzare più oggetti sulla stessa riga devi separarli con una virgola. Nel tuo caso hai detto a Python:
visualizzami la stringa "5*3= " così com'è
e subito dopo hai aggiunto:
visualizzami il risultato della moltiplicazione tra i numeri 5 e 3.

Prova a rispondere ai seguenti quesiti:

1. In quale di questi due comandi i numeri sono numeri e non simboli?
`print "6 + 3"` e `print 6+3`

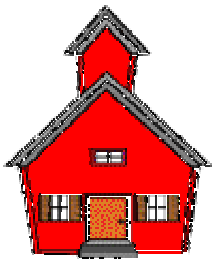
2. Quale sarà il risultato dei due comandi?

3. Quale sarà il risultato del comando:
`print "9 * 5 = ", 9 * 5`

4. Quale risultato dara` l'istruzione seguente?
`print "Ciao a " + "tutti"`

Soluzioni:

Nel secondo.
Print "6+3" visualizzerà 6+3 ; print 6+3 visualizzerà 9 ; 9*5 =45 ; Ciao a tutti
(Per capire meglio l'esercizio 4 leggi la spiegazione che il professore ti farà nella pagina seguente)



Dati e tipi di dati (operazioni con le stringhe)

Il quesito 4 della pagina precedente anticipa le operazioni sulle stringhe.
Interi e decimali possono essere utilizzati per operazioni matematiche, le stringhe no, anche se il loro contenuto sembra un numero.
Scrivere:
"ciao"/12 oppure "18"+5
e' sbagliato e genera un `SYNTAX ERROR`

Tuttavia gli operatori matematici **+** e *****, e soltanto questi, possono essere utilizzati anche con le stringhe ma con una funzione diversa.
Se ad una stringa viene sommata un'altra stringa l'effetto che si ottiene e' il **concatenamento**, cioè la seconda stringa si aggiunge al fondo della prima.
Ad esempio:
"casa"+" dolce "+"casa" genera a video
`casa dolce casa`
Gli spazi prima e dopo la parola "dolce" fanno parte della stringa e sono necessari, altrimenti a video le stringhe concatenate sarebbero attaccate come un'unica parola.
Se invece vogliamo ripetere tante volte la stessa stringa (**ripetizione**) possiamo moltiplicarla per un numero intero usando l'operatore matematico *.
Ad esempio:
"ciao"*3 diventa `ciaociaociao`
"ciao "*3 diventa `ciao ciao ciao`
"Ha, "*5 diventa `Ha, Ha, Ha, Ha, Ha,`
(Nota bene lo spazio inserito nel secondo e nel terzo esempio)

Possiamo anche mettere una stringa in una scatola, ossia assegnare una stringa ad una variabile e poi applicare le operazioni possibili sulle stringhe.

Ad esempio:
```
>>>SCATOLA1 = "casa "
>>>SCATOLA2 = "dolce casa"
>>>print SCATOLA1  + SCATOLA2
```
Il risultato sarà:
`casa dolce casa`

Le operazioni possono anche essere combinate tra loro nella stessa istruzione.
Ad esempio:
```
>>>print "ciao " + (" ciao " * 3)
```
darà come risultato
`ciao ciao ciao ciao`
Questo risultato si può ottenere in altri modi, ad esempio:
```
>>>print "ciao "*4
>>>print "ciao "+" ciao "+" ciao"+" ciao "
```

Esercitiamoci un po'

Ricordati di usare Python Shell per fare gli esercizi

1. Cinque per tre e' uguale a 15.
   Puoi ottenere questo risultato con o senza scatole, ossia le variabili.
   Trova le due soluzioni.

2. Scrivi tutte le sequenze di istruzioni possibili per visualizzare il messaggio "Buon Compleanno" cinque volte.

3. Scrivi la sequenza di istruzioni per visualizzare il tuo nome e cognome in due stringhe separate.

4. Scrivi la sequenza di istruzioni per ottenere il messaggio seguente utilizzando le variabili: l'area del rettangolo e' uguale a 50.

5. Scrivi la sequenza di istruzioni per ottenere il perimetro e l'area di un rettangolo.

6. Scrivi le istruzioni per un programma che concateni due variabili stringa (attenti agli spazi) e moltiplichi due variabili numeriche (intere). Infine visualizza il risultato sullo schermo.

7. Scrivi un programma che faccia un disegno con i caratteri; per esempio prova a copiare il programma:

```
print ' O  O '
print '   | '
print '  \_/ '
```

   e poi scrivine uno tu.

8. Trova l'errore:
   a. print ciao+4
   b. print "ciao+4"

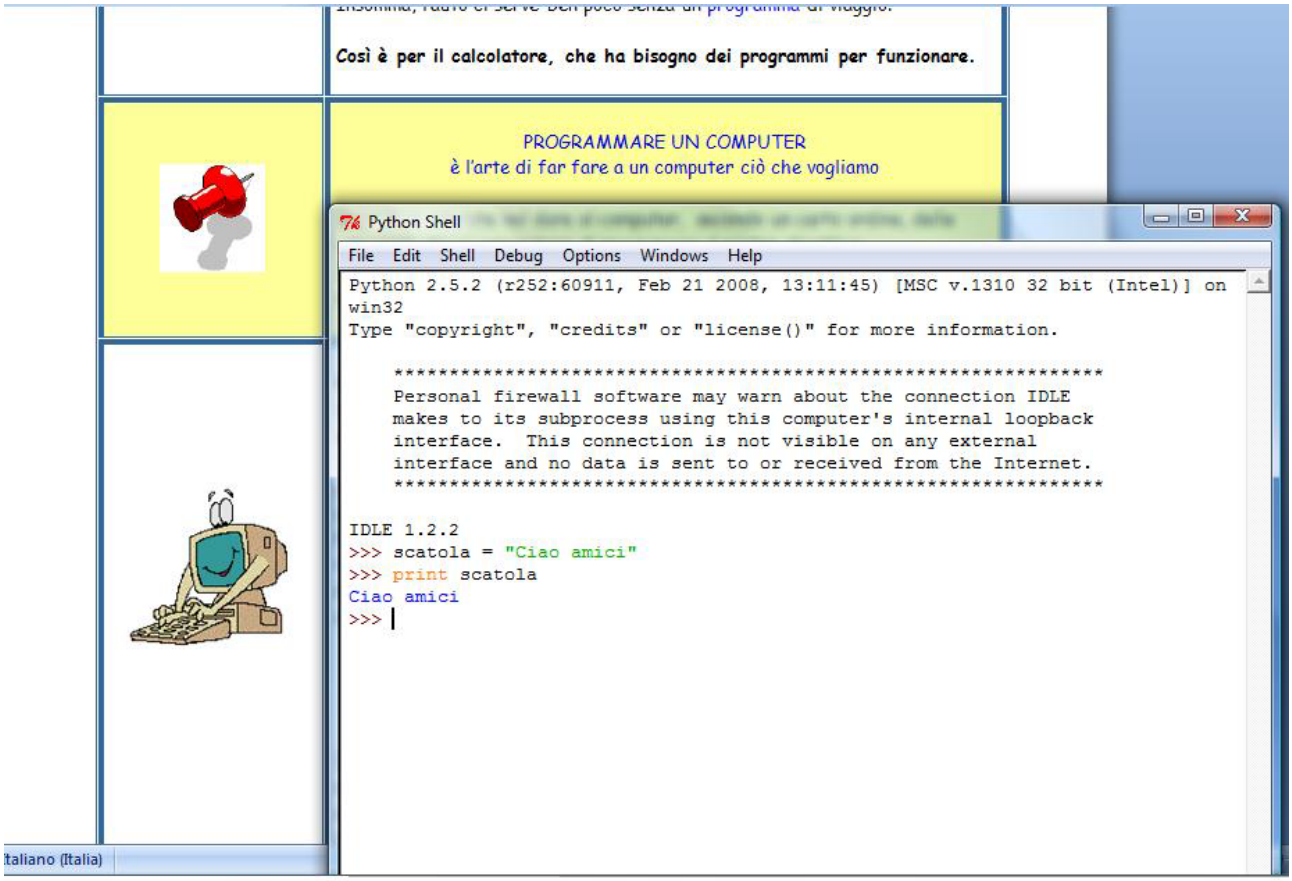
9. Trova l'errore:
   a. 6scatola = "che bello il telefonino"
      print 6scatola
   b. farfalla = "cane"
      print "è bello il mio", farfalla, "Joe!"

10. Trova l'errore:
    a. scatola = "viva il calciobalilla"*3
       print scatola
    b. farfalla = "cane"/5
       print "è bello il mio", farfalla, "Joe!"

STEP 5
IL PROGRAMMA

In questo step impareremo:

* cosa significa **programmare** e che cos'è un **programma**
* cosa è un **interprete** e un **compilatore**:
  (Python è un linguaggio interpretato)



STEP 5

Ovvero, un passo dopo l'altro impariamo a programmare

Il Programma

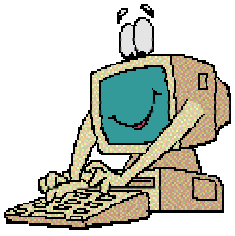
Approfondiamo un po'
Parlando di codici, linguaggi e interpreti

Se dobbiamo fare un viaggio e abbiamo deciso di usare l'auto, che cosa ci serve? Fondamentalmente due cose: l'auto e un programma di viaggio.
Per prima cosa dobbiamo avere un'auto e saperla guidare, ma è altrettanto importante l'organizzazione del viaggio: decidere il percorso da seguire, raccogliere le informazioni necessarie, stabilire le tappe da fare, etc.
Insomma, l'auto ci serve ben poco senza un programma di viaggio.

Così è per il calcolatore, che ha bisogno dei programmi per funzionare.

PROGRAMMARE UN COMPUTER
è l'arte di far fare a un computer ciò che vogliamo

In pratica consiste nel dare al computer, secondo un certo ordine, delle istruzioni che ci consentano di raggiungere il nostro obiettivo.
Il programma è semplicemente la sequenza di istruzioni che dicono al computer come fare un certo lavoro.



Per prima cosa dobbiamo saper accendere il computer e "dialogare" con lui, in modo da riuscire a comunicargli alcune istruzioni e a fargliele eseguire correttamente (come ti abbiamo spiegato nei primi step di questo manuale).

Successivamente dovremo usare una "lingua" per comunicare con il computer, proprio come facciamo per parlare con un amico.
Purtroppo l'unica lingua che il computer capisce è chiamata "codice binario". Questo codice, molto difficile da leggere e da scrivere per gli esseri umani, è invece molto facile per i circuiti elettronici del calcolatore che devono distinguere solo due simboli:

0 1

I programmatori dei primi computer dovevano scrivere il programma utilizzando un complicato linguaggio macchina di tipo binario, operazione che ancora oggi viene chiamata programmazione in linguaggio macchina e che è incredibilmente difficoltosa.

Per semplificare le cose sono stati creati dei linguaggi intermedi che stanno a metà strada tra il linguaggio macchina e il linguaggio degli uomini, detti linguaggi di programmazione di alto livello.
Sono più facili da imparare e da usare, ma sono estremamente precisi, com'è necessario che sia, affinché il computer li possa interpretare in modo corretto.

Insieme ai linguaggi di programmazione intermedi sono stati sviluppati programmi per la traduzione dal linguaggio di alto livello al linguaggio macchina. Questi ultimi convertono semplicemente parole inglesi equivalenti al codice binario in binario, in modo che invece di dover ricordare complicate sequenze di O e di 1, i programmatori possano scrivere le istruzioni usando parole della lingua inglese.

Per programmare con il computer è necessario usare un linguaggio intermedio che viene poi tradotto in un programma di basso livello (cioè in binario) da un opportuno "programma traduttore" in modo che diventi comprensibile al computer.
I traduttori si dividono in due categorie: i compilatori e gli interpreti.
Il compilatore: legge il programma di alto livello e lo traduce tutto in linguaggio macchina, prima che il programma possa essere eseguito. In questo caso il programma di alto livello viene chiamato codice sorgente ed il programma tradotto codice oggetto.
L'interprete: legge il programma di alto livello e lo esegue trasformando ogni riga di istruzione in un'azione. L'interprete elabora il programma un po' alla volta, alternando la lettura delle istruzioni all'esecuzione dei comandi che le istruzioni descrivono.
E' un po' come quando il Presidente degli Stati Uniti e il Presidente della Russia si parlano durante un incontro internazionale: il primo parla in inglese e un interprete ripete in russo tutto quello che ha detto. Il secondo risponde in russo e l'interprete nuovamente ripete in inglese quello che ha detto.
Il programma che traduce il linguaggio intermedio (di alto livello) in binario si chiama proprio **INTERPRETE**.

Con il tempo gli esperti di informatica hanno sviluppato linguaggi per il computer di livello sempre più alto, più vicini al modo di parlare e ragionare degli esseri umani che non a quello delle macchine.
Python è il nome del linguaggio intermedio che abbiamo scelto per parlare con il computer e che ci aiuterà a impartire ordini e istruzioni per fargli svolgere i compiti che noi gli assegneremo.
Python è considerato un linguaggio interpretato perché i programmi scritti in Python sono eseguiti da un interprete.
Comunque, dal nostro punto di vista, questo non fa alcuna differenza:
noi scriviamo il codice sorgente ed utilizziamo uno strumento che si chiama Python per consentire al computer di leggerlo ed eseguirlo.

Non lasciarti intimidire da quanto detto fin qui.
Un programma per computer è semplicemente un insieme di istruzioni che indicano al computer come eseguire un particolare compito. Le istruzioni che abbiamo scritto negli step precedenti possono già essere considerate dei primi semplici programmi per computer.
Il programma più semplice che viene preso come esempio è:

```
print "Ciao amici"
```

Quando viene eseguito dal computer vi saluta con un:

```
Ciao amici
```

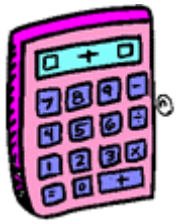
Possiamo paragonare il programma ad una ricetta di cucina: un insieme di istruzioni per indicare a un cuoco come cucinare un piatto.
Una ricetta descrive gli ingredienti (i dati) e la sequenza di istruzioni (il procedimento) necessari, ad esempio, per fare una frittata:

Ingredienti	*Frittata*
Una noce di burro Un uovo Sale	Metti il burro nella padella, quando il burro frigge aggiungi l'uovo sbattuto, cuoci a fuoco lento per cinque minuti circa e poi gira la frittata, cuoci altri cinque minuti aggiungi un pizzico di sale.

I programmi sono concettualmente assai simili alle ricette.
Per prima cosa bisogna avere le idee chiare su cosa si vuol fare (ad esempio…una frittata), poi si scompongono tutte le operazioni necessarie per fare quella cosa in tanti piccoli passi logici ovvero in una sequenza di istruzioni, in modo che il computer le possa interpretare senza sbagliare.
Ognuna di queste sequenze è un programma.

Definizione

Programma:
la sequenza delle istruzioni, ossia degli ordini che dicono al calcolatore come deve operare



Il programma è l'elemento importante, fondamentale che distingue il computer dalla calcolatrice.
Nella calcolatrice l'operatore introduce i dati sui quali deve essere eseguita un'operazione, poi ne ordina l'esecuzione, infine legge il risultato e se deve ricordare più dati è costretto ad usare la sua memoria. Inoltre l'intervento dell'operatore fra un'operazione e l'altra limita la velocità complessiva del calcolo.
Nel computer il programma rende automatico il procedimento di risoluzione del problema. E' sufficiente dare al computer la giusta sequenza di istruzioni, IL PROGRAMMA, e poi sarà lui ad eseguire fedelmente le nostre istruzioni.

<table>
<tr><td>  </td><td colspan="3">Esercitiamoci un po'</td></tr>
<tr><td>Esempio</td><td>Scrivi l'elenco dei dati e delle operazioni necessarie per preparare la tavola</td><td>Dati:
4 bicchieri
4 piatti
4 forchette
4 cucchiai
4 coltelli
4 tovaglioli
1 tovaglia</td><td>Operazioni:
dispongo la tovaglia
posiziono i 4 piatti
posiziono le 4 forchette
posiziono i 4 coltelli
posiziono i 4 cucchiai
posiziono i 4 cucchiai
posiziono i 4 tovaglioli</td></tr>
<tr><td>Esercizio
n. 1</td><td>Scrivi l'elenco dei dati e delle operazioni necessarie per preparare la cartella per andare a scuola.</td><td>Dati:</td><td>Operazioni:</td></tr>
<tr><td>Esercizio
n. 2</td><td>Scrivi l'elenco dei dati e le operazioni necessarie per acquistare un quaderno in un centro commerciale.</td><td>Dati:</td><td>Operazioni:</td></tr>
<tr><td>Esercizio
n. 3</td><td>Scrivi l'elenco dei dati e le operazioni necessarie per sommare due numeri.</td><td>Dati:</td><td>Operazioni:</td></tr>
<tr><td>Esercizio
n. 4</td><td>Scrivi l'elenco dei dati e le operazioni necessarie per trovare il m.c.m. tra due numeri.</td><td>Dati:</td><td>Operazioni:</td></tr>
<tr><td>Esercizio
n. 5</td><td>Scrivi l'elenco dei dati e le operazioni necessarie per trovare due numeri la cui somma è 40 e la cui differenza è 8.</td><td>Dati:</td><td>Operazioni:</td></tr>
<tr><td>Esercizio
n. 6</td><td>Scrivi l'elenco dei dati e le operazioni necessarie per trovare la lunghezza di due segmenti sapendo che uno è il triplo dell'altro e che la loro differenza è 24 cm.</td><td>Dati:</td><td>Operazioni:</td></tr>
<tr><td>Esercizio
n. 7</td><td>Scrivi l'elenco dei dati e le operazioni necessarie per…….?
Adesso tocca a te, inventa questo esercizio….e se vuoi…
tanti altri!!!</td><td>Dati:</td><td>Operazioni:</td></tr>
<tr><td>Riepilogando……..</td><td colspan="3">8. Scrivi tutte le sequenze di istruzioni possibili per visualizzare il messaggio "Ho conosciuto una principessa" cinque volte.

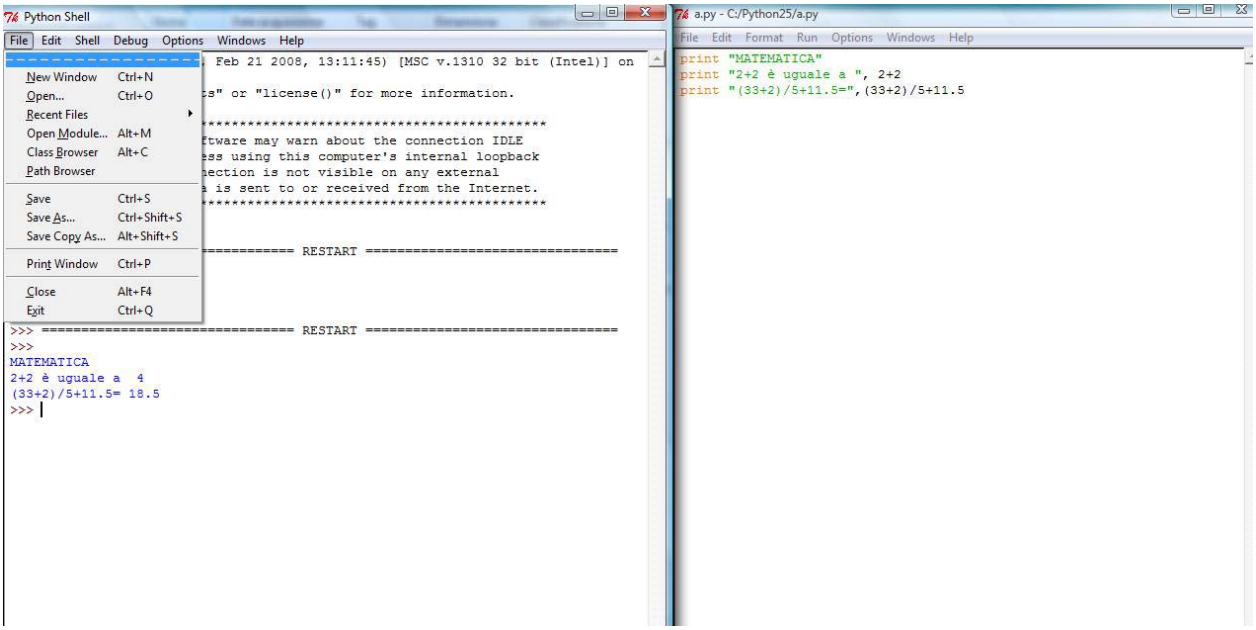
9. Scrivi l'elenco dei dati, le operazioni e le istruzioni per ottenere che nella scatola1 ci sia 30, nella scatola2 ci sia anni e stampi hai 30 anni domani .

10. Scrivi la sequenza di istruzioni per ottenere il messaggio seguente utilizzando le variabili: l'area del rettangolo e' uguale a 20 e l'area del triangolo rettangolo è uguale a 20</td></tr>
</table>

STEP 6
PROGRAMMARE IN PYTHON

In questo step impareremo:

* a **scrivere** i programmi utilizzando IDLE
* a **salvare** i programmi che abbiamo scritto
* a **eseguire** i programmi che abbiamo salvato



STEP 6

Ovvero, un passo dopo l'altro impariamo a programmare

Programmare in Python

Adesso che abbiamo iniziato a comprendere cosa sono i dati e come possiamo usarli ed abbiamo installato sul computer Python, proviamo ad usarlo sia per dare al calcolatore semplici comandi che vengono eseguiti uno per volta, sia per dare sequenze di istruzioni più o meno lunghe, che verranno eseguite tutte insieme, secondo l'ordine di sequenza.

Chiameremo programmi le sequenze di istruzioni che realizzeremo.

In Python ci sono diversi modi di lavorare, tra questi puoi:

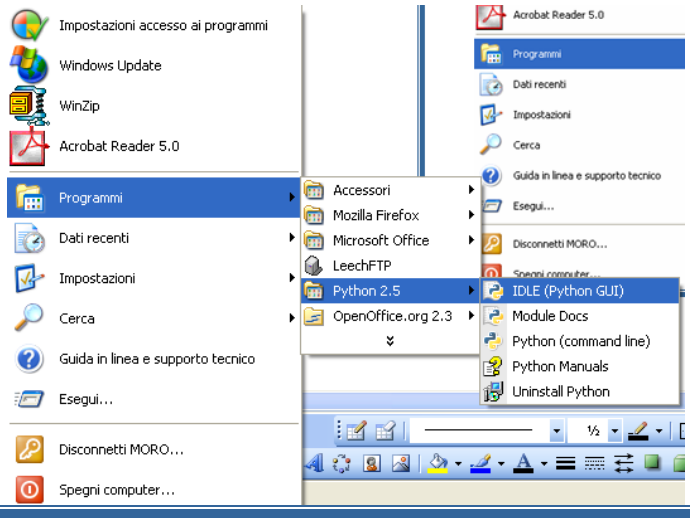
- **Utilizzare IDLE** (Integrated DeveLopment Environment) che significa "ambiente di sviluppo integrato", ossia insieme di strumenti collegati tra loro per la scrittura di programmi
- **Scrivere linee di comando di Python (command line)**

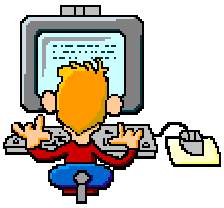
La differenza tra i due è questa:

- **in IDLE** in una sola finestra troviamo sia l'editor per scrivere il programma sia l'interprete per farlo eseguire. (Le versioni di IDLE per Windows e per Linux sono sostanzialmente uguali).
- **scrivere linee di comando** significa invece fare le seguenti cose:

a. scrivere il programma con un editor qualsiasi, ad esempio Wordpad
b. dare al programma scritto un nome (ad esempio PROGRAMMA1)
c. SCRIVERE IL COMANDO Python seguito dal nome del programma per eseguire il programma stesso
(ad esempio: Python PROGRAMMA1)

In base alla diversa versione del sistema operativo e di Python installati, se dal desktop selezioniamo **START**, poi **PROGRAMMI**, **PYTHON** e infine **IDLE** comparirà una videata simile a questa:

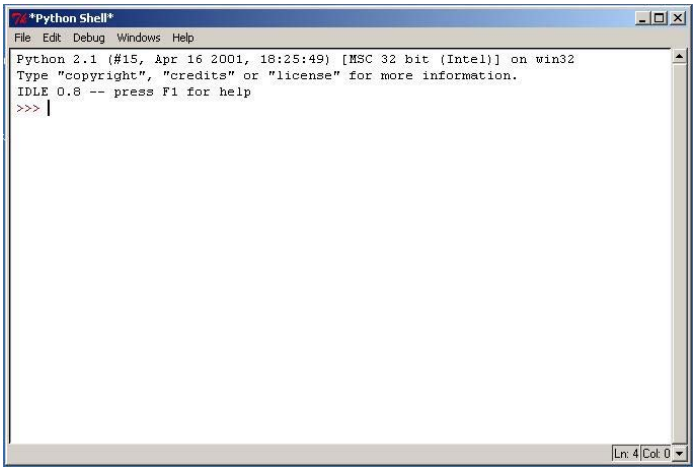




A questo punto si apre una nuova finestra, simile sia in Windows sia in Linux, che si chiama "Python Shell":

è la finestra principale di IDLE, la finestra "INTERPRETE"

***



L'interprete ci permette di inserire i comandi direttamente in Python: non appena immettiamo un comando, Python lo esegue e ne restituisce immediatamente il risultato. Ciò è molto comodo e funzionale al nostro lavoro.

Prova ad analizzare la scritta (vedi sopra ***):

```
Python 1.5.2 (#0, Apr 13 1999, 10:51:12) [MSC 32 bit (Intel)]
on win32
Copyright 1991-1995 Stichting Mathematisch Centrum, Amsterdam
>>> print 1 + 1
2
```

>>>:

La prima linea di questo esempio è il comando che fa partire l'interprete Python <u>in ambiente Linux</u>; la seconda linea fornisce informazioni di copyright mentre la terza inizia con **>>>:** questa è l'indicazione di "prompt" che l'interprete usa per indicare che è pronto a ricevere istruzioni.

Noi abbiamo inserito l'istruzione print 1+1 e l'interprete risponderà con 2.

Non ti preoccupare se ti senti un po' disorientato e non ti senti sicuro ed esperto nella programmazione del computer.
Ricordati che quello che noi vogliamo fare ora è semplicemente sperimentare l'uso di Python, realizzare brevi programmi digitando le necessarie istruzioni.
Il percorso per imparare a programmare è ancora lungo ma sicuramente ricco di soddisfazioni. Ricordati che l'esercizio e la pratica sono fondamentali.

Intanto hai già imparato molte cose, tutte importanti e utili per programmare.
Prova allora a usare la fantasia e a inventare qualche semplice programma, tanto per prendere confidenza con l'interprete.
Qualche idea per cominciare? Prova a scrivere:

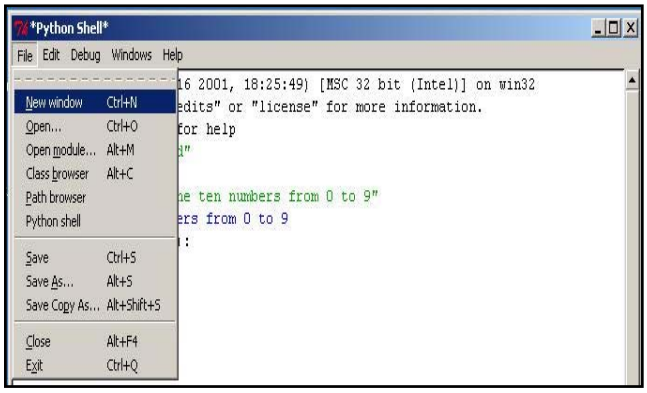
```
>>>print "MATEMATICA"
>>>print "2+2 è uguale a  ",2+2
>>>print "(33+2)/5+11.5 = ", (33+2)/5+11.5
```

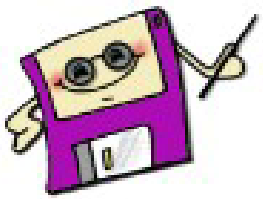
Buon lavoro!

Programmare in Python – SAVE

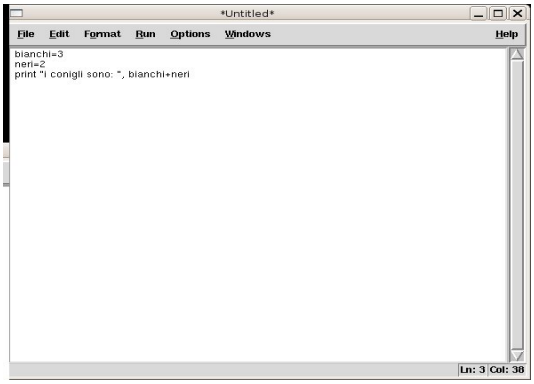
Ho notato che la finestra "Interprete" comprende sia i nostri comandi sia le risposte del sistema.
Se chiudo Python e lo riapro, come fa il computer a ricordarsi quello che ho scritto? Come si ricorda il computer dei nostri programmi?
Come fa Python a capire che il programma è composto da più istruzioni?

Per prima cosa dobbiamo imparare a salvare il nostro lavoro.
Prima di tutto avvia l'interprete di Python e apri una nuova finestra di lavoro.
Ora quello che vogliamo ottenere è un file con solo le nostre istruzioni e la possibilità di salvarlo, in modo da poterlo riaprire senza dover riscrivere ogni volta tutte le istruzioni.
In ambiente Python, nella finestra Interprete (Python Shell) selezioniamo i menu **File** e poi **New Window**.

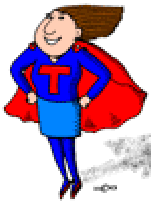




La nuova finestra si chiama **Untitled** : qui possiamo scrivere le nostre istruzioni e Python non interferirà con le sue risposte mentre scriveremo il programma. Da ora in poi la chiameremo Finestra Programma.



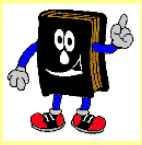
Dopo aver scritto le istruzioni dell'esercizio, cliccando su **File** e poi **Save as** siamo in grado di salvare il nostro lavoro (oppure *salva come*). Ricorda di dare al nome del tuo file l'estensione **.py** come stabilito da convenzione.
(Hai notato che non compare il prompt ">>>" ? esso infatti non fa parte del nostro programma).



Adesso che hai imparato come fare a salvare il tuo programma in Python, devi imparare a:

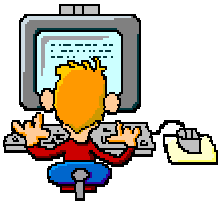
1. Farlo eseguire
2. recuperarlo per poterlo modificare o per utilizzarlo tutte le volte che vuoi

Prova a studiare i menu della finestra di Python e a scoprire da solo come si fa…non è difficile!
Un suggerimento per iniziare? …come si dice"corri" in inglese?



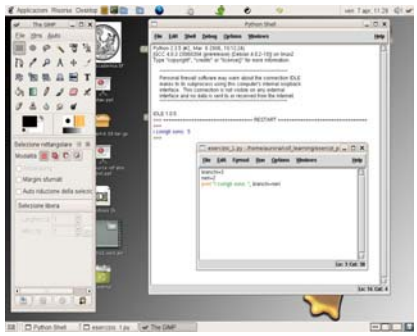
RUN: in italiano significa "corri" o meglio, nel nostro caso, "lancia"

RUN MODULE ***(nelle edizioni di Python più vecchie puoi trovare Run script):*** lancia il modulo (ovvero il programma)



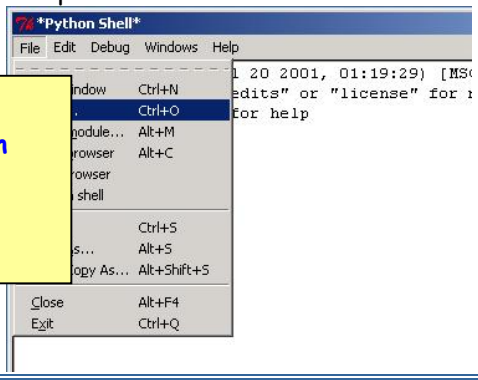
1. Selezionando RUN MODULE dalla finestra programma, Python eseguirà il nostro programma nella finestra interprete. Avremo così, aperte contemporaneamente, una finestra con il testo del programma ed un'altra con il risultato. Questa è una gran comodità! È infatti molto utile vedere contemporaneamente il codice del programma e il suo risultato.

RUN MODULE per eseguire il programma

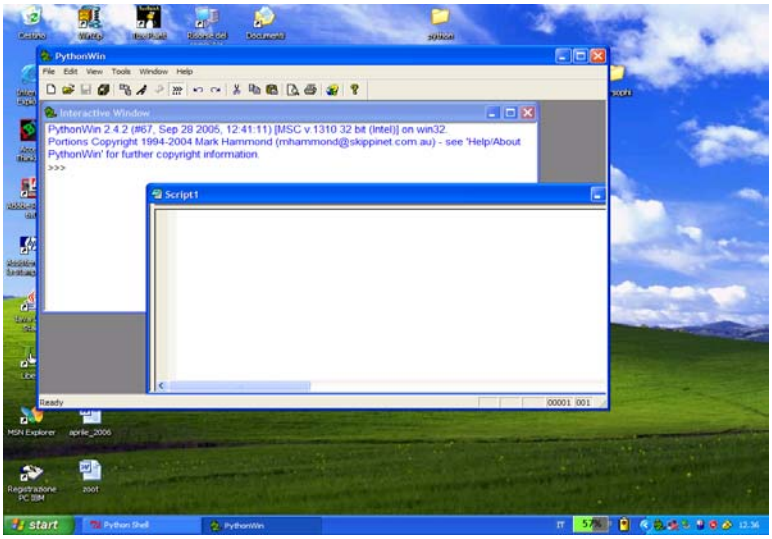


2. Chiudiamo tutte le finestre di Python aperte e rilanciamo IDLE: nel menù FILE della finestra interprete troveremo il comando *OPEN* (apri).

OPEN Per richiamare un programma salvato in precedenza



La stessa operazione puoi farla dalla finestra PROGRAMMA: dal menu FILE della finestra interprete seleziona NEW WINDOW e poi dal menù FILE seleziona il comando OPEN. Questa seconda possibilità è utile quando hai un programma già aperto e vuoi aprirne un altro, magari per copiarne un pezzo. Ti consiglio comunque di organizzare molto bene le cartelle dove salvare i tuoi programmi, creandoti un archivio di facile consultazione. Utilizzando Python regolarmente troverai il tuo modo personale di muoverti tra le finestre "Interprete" e "Programma": devi considerarle come un laboratorio per sperimentare nuovi programmi.

E' arrivato il momento di…	**CONSOLIDARE LE CONOSCENZE**
	Dobbiamo chiarire ancora una questione: Martina prima ti ha fatto un esempio realizzato in ambiente Linux. Come fare allora se ti trovi in ambiente Windows e usi PythonWin? Puoi eseguire tranquillamente le operazioni descritte nello step 6 ed ottenere gli stessi risultati. Come ti ha precedentemente spiegato Martina, le differenze tra una versione e l'altra del programma sono minime. Devi allora aprire PythonWin, cliccare su file e poi su new: apparirà una finestra che ti chiederà di aprire Python Script, rispondi Ok e otterrai :  Esaurita quest'ultima questione concediamoci un attimo di pausa e di riflessione visto che, piano piano, abbiamo fatto altri tre passi! Allora: 1. Hai seguito con attenzione? 2. Capito bene tutto quello che hai letto e (spero) studiato? 3. Hai completato gli esercizi assegnati? 4. Provato a inventare nuovi esercizi? Pensi di si? Molto bene. Prima di proseguire nel percorso di studio, devi essere sicuro non solo di avere compreso tutto quello che hai letto nelle schede, ma anche di averlo ben assimilato e memorizzato. *Insomma, devi fare un po' di allenamento ed esercitare la tua mente.* La professoressa Martina si è divertita a inventare un po' di esercizi da farti fare e adesso tocca a te risolvere i semplici problemi che troverai nella pagina seguente.
	All'inizio dello step1 ti ho fatto tre domande, una di queste era: *Più avanti ti accorgerai che definiremo Python "programma interprete", prova a pensare perché.* Adesso sapresti dare una risposta corretta? Ricordati che *Python è un linguaggio interpretato perché permette di eseguire direttamente il codice sorgente che hai scritto. Puoi anche scrivere le istruzioni dal prompt dei comandi senza la necessità di creare un file sorgente(per questo Python è detto anche "interattivo").*

E' arrivato il momento di…	ESERCITARCI CON PYTHON
	Prova a svolgere questi gli esercizi. Quando hai finito, vai al fondo del libro dove troverai le soluzioni, confrontale con quello che hai scritto e assegnati 2 punti per ogni esercizio eseguito correttamente, nessun punto se l'esercizio è incompleto o errato. Quando hai finito vai alla pagina dell' autovalutazione. Buon Lavoro.

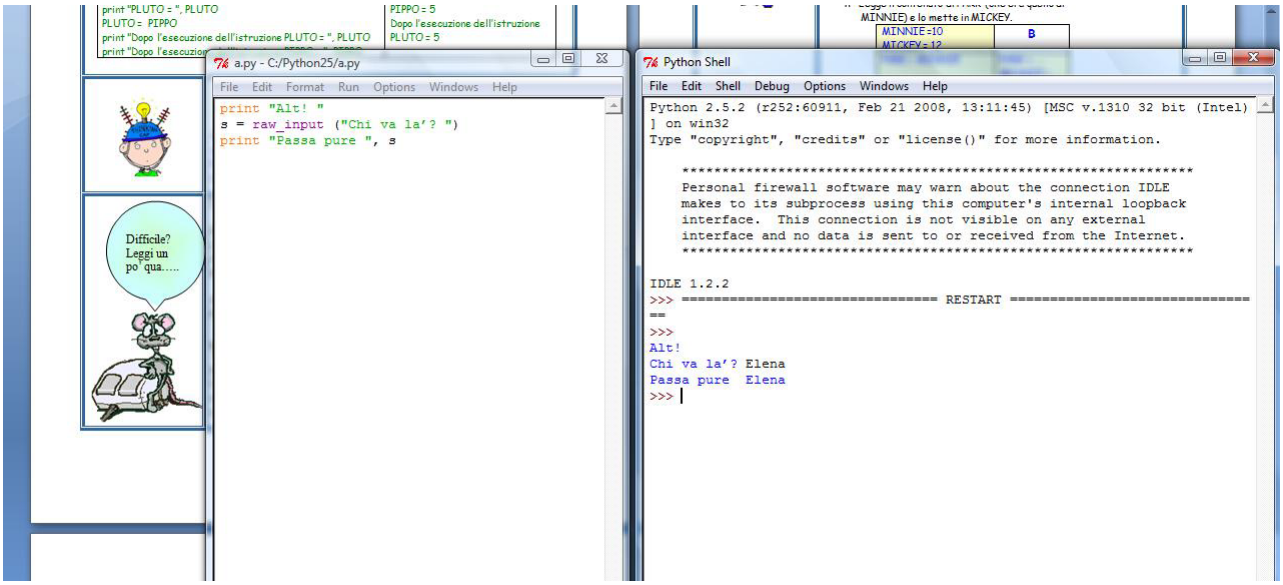
Esercizio	Punti
Esercizio n. 1: scrivi un programma per sommare i primi dieci numeri pari.	……
Esercizio n. 2: scrivi un programma che visualizzi cinque stati e le corrispondenti capitali in colonna.	……
Esercizio n. 3: scrivi il programma che calcola e visualizza il quadrato e il cubo di 3.	……
Esercizio n. 4: scrivi il programma per calcolare l'area di un quadrato avente il lato uguale a 5 cm.	……
Esercizio n. 5: osserva il seguente programma che visualizza un rettangolo: print "*******" print "*******" print "*******" print "*******" print "*******" Prova a mandarlo in esecuzione. Scrivi poi un programma che visualizzi un triangolo rettangolo formato da asterischi.	……
Esercizio n. 6: scrivi un programma che visualizzi un trapezio formato da Asterischi.	……
Esercizio n. 7: crea tre scatole: Qui= 6 Quo= 12 Qua= 8 Scrivi un programma che visualizzi la somma del contenuto delle scatole e inserisci il risultato in un'altra scatola.	……
Esercizio n. 8: scrivi un programma che visualizzi le potenze del 2. (*Quest' ultimo esercizio ti sembrerà un po' difficile, ma prova lo stesso a risolverlo, tuffati senza paura nel mare della programmazione!)*	……
Totale punti	…/16

E' ARRIVATO IL MOMENTO DI……	**AUTOVALUTARCI**
	Adesso proviamo a riflettere sull'attività fin qui svolta. Per verificare se hai fatto un buon lavoro rispondi alle seguenti domande (sinceramente!):
 Vediamo un po'…….	1. Che cos'è una stringa? Da cosa può essere composta? 2. Che differenza c'è tra un programma compilato ed uno interpretato? 3. Python è un programma interprete? Perché? Per ogni risposta corretta hai guadagnato 2 punti. Segna qui a fianco il punteggio ottenuto …/6
Tiriamo le somme	Adesso somma il punteggio ottenuto rispondendo alle domande sopra con quello degli esercizi svolti nella pagina precedente: Esercizi ……/16 + Domande ……/6 = ---------- Totale ……/22 punti Se hai totalizzato meno di 12 punti ….. segui i consigli di Super Teacher : studia di più, esercitati di più, chiedi aiuto al tuo insegnante per capire meglio prima di proseguire l'attività. Se hai ottenuto 12 punti o più puoi passare direttamente allo step 7.
	Qualcosa ti è sembrato difficile? Non hai capito qualche argomento? Annotalo qui sotto e segnalalo al tuo insegnante: ---------- ---------- ---------- Se invece il lavoro che abbiamo fatto per te ti è piaciuto disegna tante meline rosse (da 1 a 10) ----------

STEP 7
LE PRIME ISTRUZIONI

In questo step impareremo:

- come inserire un numero o una stringa in una scatola, ovvero un dato in una variabile utilizzando le istruzioni di assegnazione del tipo:
  >>> SCATOLA1=37,5
- come spostare i dati da una scatola all'altra
- come utilizzare le istruzioni: **input e raw_input**



STEP 7

Ovvero, un passo dopo l'altro impariamo a programmare

Da una scatola all'altra

COPIATURA

Supponiamo di avere a disposizione due scatole di nome **pippo** e **pluto**, con **pippo** = 5 e **pluto** = 15.
Vediamo che cosa succede quando diamo al computer un'istruzione come questa:

pippo = pluto

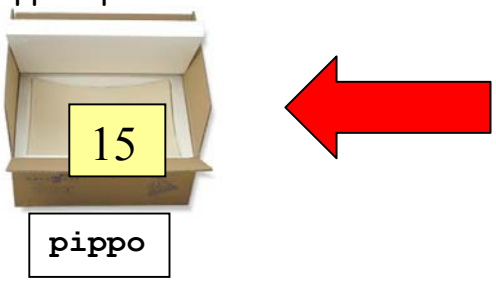
Il computer fa le seguenti operazioni:

1. mette a disposizione la scatola di nome **pippo**
2. mette a disposizione la scatola di nome **pluto**
3. legge il contenuto (vi ricordate il foglietto nella scatola?) di **pluto** e lo mette nella scatola **pippo**.

Al termine di queste operazioni le due scatole **pippo** e **pluto** contengono lo stesso numero 15.
(Prima dell'esecuzione dell'istruzione):

(Dopo l'esecuzione dell'istruzione)

pippo = pluto



Osserva che il contenuto di **pluto** (la scatola che si trova a destra del segno uguale) viene inserito in **pippo** (nella scatola a sinistra dell'uguale) e non viceversa.
Non si tratta di un trasferimento ma di una copiatura, infatti pluto non perde il suo contenuto.

Cosa succede invece se scriviamo: **pluto** = **pippo**, quando **pippo** = 5 e **pluto** = 15? Che le due scatole conterranno il numero 5. Vediamo in dettaglio i due programmi corrispondenti a queste due istruzioni:

Programma 1 - pippo = pluto	**Il risultato sarà:**
pippo=5 pluto=15 print "pippo = ", pippo print "pluto = ", pluto pippo = pluto print "Dopo l'esecuzione dell'istruzione pippo = ", pippo print "Dopo l'esecuzione dell'istruzione pluto = ", pluto	pippo = 5 pluto = 15 Dopo l'esecuzione dell'istruzione pippo = 15 Dopo l'esecuzione dell'istruzione pluto = 15

Programma 2 - pluto = pippo	**Il risultato sarà:**
pippo=5 pluto=15 print "pippo = ", pippo print "pluto = ", pluto pluto = pippo print "Dopo l'esecuzione dell'istruzione pluto = ", pluto print "Dopo l'esecuzione dell'istruzione pippo = ", pippo	pippo = 5 pluto = 15 Dopo l'esecuzione dell'istruzione pluto = 5 Dopo l'esecuzione dell'istruzione pippo = 5

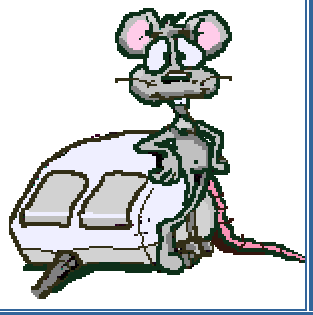
INCREMENTO

Adesso prova a immaginare cosa fa il computer per eseguire la seguente istruzione:

pippo = **pluto** + 3

Utilizzando Python, scrivi un programma che esegua questa istruzione.

Difficile?....



Il computer farà così:

Mette a disposizione una scatola di nome **pippo** pippo = 5
Cerca **pluto** e ne legge il contenuto pluto = 15
Aggiunge 3 al contenuto di **pluto** e pippo = pluto + 3
mette in **pippo** il risultato dell'operazione a destra dell'uguale.

A questo punto, che cosa conterrà **pippo** e che cosa conterrà **pluto**?
pippo conterrà il numero 18 e **pluto** conterrà sempre il numero 15.

In pratica il computer inizia a lavorare sull'operazione a destra dell'uguale e il risultato viene messo nella scatola a sinistra.
Osserva ancora che il nome della scatola a destra dell'uguale indica il suo contenuto, mentre il nome della scatola a sinistra precisa la scatola che conterrà il risultato.

SCAMBIO

Un po' più complessa è l'operazione di scambio del contenuto di due scatole.
Ad esempio se **minni** = 10 e **mickey** = 12 come posso scambiare il contenuto di **minni** e **mickey**, cioè inserire 12 in **minni** e 10 in **mickey**?
Prova a riflettere: è come scambiare il contenuto di due bicchieri uno pieno di coca cola e l'altro pieno di aranciata.

Ci serve una terza scatola che possiamo chiamare PARK, nella quale accantoniamo il contenuto di una delle due scatole.

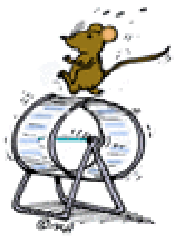
Cosa fa il computer?

1. Mette a disposizione una scatola di nome **minni** e una di nome **mickey**.
2. Mette a disposizione una scatola di nome PARK e ci inserisce il contenuto di **minni**.
3. Legge il contenuto di **mickey** e lo mette in **minni**.
4. Legge il contenuto di PARK (che era quello di **minni**) e lo mette in **mickey**.

minni =10 **mickey** = 12	
PARK = **minni**	PARK =..... **minni** =.....
minni = **mickey**	**minni** =..... **mickey** =.....
mickey = PARK	**mickey** =..... PARK =.....

Cosa contengono le scatole della colonna arancione?

Prima di proseguire..
..esercitiamoci un po'



Esercizio n. 1

Se click1 = 24 e slam1 = 32 come faccio per copiare il contenuto di click1 in slam1? E quando l'ho copiato come faccio per rimettere nelle due scatole il contenuto originale? Prova a illustrare i vari passaggi attraverso i quali il calcolatore copia il contenuto di una scatola in un'altra.

Esercizio n. 2

Scrivi un programma per scambiare il contenuto delle due scatole seguenti:
pluto = "America" pippo = "Asia"

Esercizio n. 3

La scatola star contiene il numero 8. Come posso ordinare al computer di svuotarla e di mettere 15 al posto di 8?

Esercizio n. 4

La scatola blam contiene il numero 2. Scrivi il programma che calcola il cubo del contenuto e lo mette nella scatola blam3.

input e raw_input

Finora abbiamo visto come inserire un numero o una stringa in una scatola, cioè un dato in una variabile utilizzando le istruzioni di assegnazione del tipo:

scatola1 = 37,5

oppure

scatola1 = "Viva la Juve"

Oltre a questo, esiste un altro modo, molto importante, rappresentato dalle istruzioni input e raw_input.

input, che significa letteralmente "inserisci", si usa nel modo seguente:

scatola = input(prompt)

Dove:

scatola è il nome della scatola che intendiamo utilizzare,

input è il comando che diamo al computer e che serve a inserire un numero nella scatola,

prompt è un messaggio che diamo all'utilizzatore perchè sappia che cosa deve inserire.

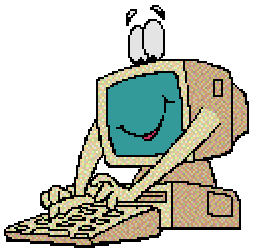
Ad esempio con:

Nota lo spazio dopo il punto interrogativo, a cosa serve?

pluto = input ("Quanti anni hai? ")

chiediamo all'utilizzatore di indicare i propri anni, il computer leggerà il numero e lo inserirà nella scatola di nome pippo.

Si lascia uno spazio perchè cosi il numero indicato non risulterà attaccato al punto interrogativo, ma sarà allontanato di uno spazio. Python non sa scrivere in modo ordinato, dobbiamo dirgli noi come visualizzare le informazioni sul video e anche come stamparle.



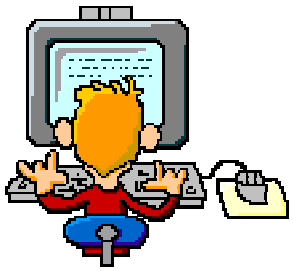
Quando il computer legge la parola input, che in Python identifica una **"funzione"** (vedremo nei prossimi STEP che cos'è una funzione), riceve l'ordine di fermarsi e attende che l'operatore inserisca un numero dalla tastiera.

Per far capire al computer quando il numero e' finito, l'operatore preme il tasto Invio (o Enter). A questo punto il programma riprende e input interpreta ciò che l'operatore ha inserito come un numero e lo mette nella scatola indicata.

Il programma prosegue poi con le istruzioni successive.

Il messaggio tra le due parentesi dopo input, che abbiamo chiamato "prompt", avvisa l'operatore di ciò che deve essere inserito.

Input è molto utile nella costruzione dei programmi perché ci permette di trasmettere dei dati al calcolatore durante l'esecuzione del programma.



Finora abbiamo sempre inserito tutti i dati prima dell'esecuzione di un programma e poi abbiamo eseguito il programma stesso; **con input, invece, i dati possono essere inseriti durante l'esecuzione**.
Vediamo in dettaglio cosa succede nel programma seguente quando usiamo la "funzione" input:

```
Anni = input("Quanti anni hai? ")
print "Tu hai ", Anni, " anni"
```

`Anni = input("Quanti anni hai? ")`	1. il computer mette a disposizione una scatola 2. la battezza scrivendo sul coperchio "Anni" 3. si ferma nell'attesa che venga inserito un dato dalla tastiera 4. inserisce il dato nella scatola indicata
`print "Tu hai ", Anni, " anni"`	Stampa prima la stringa "Tu hai", poi il contenuto della scatola Anni e infine la stringa "anni".

Utilizzando l'interprete di Python, prova ad eseguire il programma.

Adesso prova a descrivere la sequenza di operazioni fatte dal calcolatore per eseguire il programma seguente e poi scrivilo sull'interprete di Python:

Programma 3 – Stampa il triplo di un numero
numero = input ("Introduci un numero ")
numero = numero *3
print "Il triplo del numero introdotto è : ", numero

Prova ora ad inserire dei caratteri che non rappresentino un numero e osserva cosa succede.
Sfortunatamente se i caratteri inseriti dall'operatore non rappresentano un numero, il programma stampa un messaggio d'errore e si blocca perchè **input funziona soltanto con i numeri** e quindi con le variabili numeriche (cioè le variabili di tipo numero). Questo vuol dire che l'interprete raccoglie il dato inserito e cerca di interpretarlo come un numero.
Ad esempio se l'utilizzatore scrive '1 2 3', la funzione input leggerà i tre caratteri e li convertirà nel valore numerico *123*.

raw_input

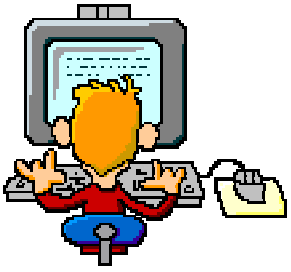
Questa funzione, a differenza della precedente input, **accetta qualunque carattere immesso dall'utilizzatore;**
"raw input" significa letteralmente "inserisci qualcosa".
La differenza consiste nel fatto che raw_input raccoglie i caratteri immessi dall'utilizzatore e li presenta sotto forma di stringa, mentre input li raccoglie e cerca di interpretarli come numero.
Il funzionamento e' lo stesso della funzione input, cambia il valore del risultato. Vediamo qualche esempio.
Il programma seguente:

```
s = raw_input("Come ti chiami? ")
print "Ciao ", s
```

Ricordati sempre lo spazio

darà come risultato:
`Ciao Paola` (o qualunque nome sia stato inserito)
Se manderai in esecuzione il programma tante volte inserendo sempre un nome diverso il risultato sarà:
Ciao Paola
Ciao Alda
Ciao Marco

NOTA BENE!
La scatola è sempre la stessa, nelle esecuzioni successive alla prima il computer cancella il vecchio dato e inserisce quello nuovo.



Utilizzando l'interprete di Python prova a scrivere gli esempi seguenti:

Programma 4

```
print "Alt! "
s = raw_input("Chi va la'? ")
print "Passa pure ", s
```

Programma 5

```
num = input("Scrivi un numero ")
str = raw_input("Scrivi una stringa ")
print "num = ", num
print "num * 2 = ", num * 2
print "str = ", str
print "str * 2 = ", str * 2
```

Esercitiamoci un po'

Ci sono più soluzioni possibili per ognuno degli esercizi proposti; sta a te trovarle e, soprattutto, provarle.

1. Scrivi un programma che chiede un numero e ne calcola il quadrato e il cubo e li visualizza sullo schermo.

2. Scrivi un programma che aggiunge 7 a qualunque numero inserito e visualizza il risultato sullo schermo.

3. Scrivi un programma che chiede due numeri, li somma e visualizza il risultato.

4. Scrivi il programma per calcolare l'area di qualunque rettangolo chiedendo all'utilizzatore la base e l'altezza.

5. Scrivi il programma che chieda tre numeri e ne visualizzi sia la somma sia il prodotto.

6. Scrivi il programma che calcola la metà e il doppio di qualunque numero inserito dall'utente, poi visualizza i risultati.

7. Scrivi il programma che chiede la misura del lato di un quadrato e ne calcola l'area, poi visualizza il risultato.

8. Scrivi il programma che calcola il perimetro del cortile della scuola che è un rettangolo i cui lati misurano rispettivamente 45 m e 65 m e visualizza il risultato. Quindi calcola il perimetro di ogni rettangolo per il quale l'operatore inserisca la misura della base e dell'altezza.

9. Scrivi un programma che chiede tre numeri, ne calcola la somma, la somma dei quadrati e il quadrato della somma. Infine, visualizza i risultati.

Adesso facciamo uno stop

Concediamoci un momento di pausa per giocare un po'.

E' importante che tu capisca che programmare è facile e divertente.
Prima di proseguire il nostro percorso di studio, facciamo un breve gioco.

Giochiamo a:

CACCIA ALL'ERRORE!

Regole del gioco:
In ogni programma è inserito un errore.
Leggi attentamente il listato di ciascun programma, prova a digitarlo utilizzando l'interprete di Python, scopri e correggi l'errore.

Per ogni errore trovato e relativa soluzione corretta assegnati due punti.

Sei pronto?

Tempo massimo per trovare gli errori e completare la scheda: 30 minuti.

Via!

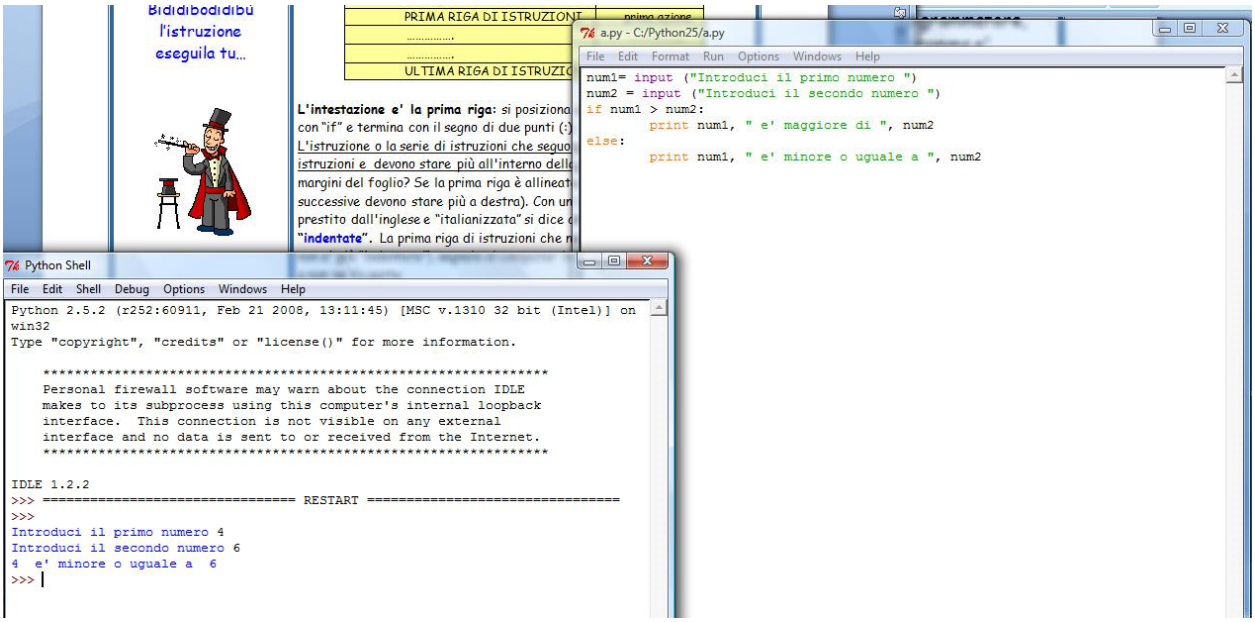
Esercizio	errore/correzione	punti
Es. 1: stampa il nome del tuo cantante preferito. cantante = raw_input ("Scrivi il nome del cantante preferito: ") print "Il mio cantante preferito e' ", cantant		
Es. 2: input di numeri e stringhe Primonumero= input ("Scrivi il primo numero: ") Secondonumero= input ("Scrivi il secondo numero: ") Nome = raw_input ("Scrivi il tuo nome: ") Cognome = raw_input ("Scrivi il tuo cognome: ") Print nome , cognome, "primonumero", "per", secondonumero, "uguale", primonumero*secondonumero		
Es. 3: domanda di filosofia printt " Sai in quale anno e' nato Socrate" sino = raw_input ("si o no") print "Ma certo, nell'anno 469 prima di Cristo"		
Es. 4: disegno un quadrato print "*****" print "* *" print "* *" print "* *" print "*****"		
Es. 5: divisione con resto primo = input ("Inserisci il primo numero") secondo = input ("Inserisci il secondo numero") print primo, "diviso", seco ndo,"si ottiene",primo/secondo print "il resto della divisione e' ", primo % secondo		
	totale	..../10

Se hai totalizzato tra i sei e i dieci punti puoi proseguire nello studio degli step successivi.
Se i punti sono meno di sei devi rivedere meglio gli argomenti trattati ed esercitarti un po' di più.

STEP 8

In questo step impareremo:

* come utilizzare le istruzioni condizionali **if** e **if else**
* come utilizzare gli operatori logici **or, and, not**
* cos'è **un'istruzione indentata**



STEP 8

Ovvero, un passo dopo l'altro impariamo a programmare

Le decisioni con le istruzioni condizionali
if e if else

A volte, nella scrittura di un programma, ci si accorge che se i contenuti delle scatole hanno certi valori si devono fare certe cose, altrimenti se ne debbono fare altre. Vediamo un esempio.
Supponiamo di voler mandare un messaggio gentile a chi usa il nostro programma. Nello scrivere il programma ci accorgiamo che il messaggio è diverso a seconda che il nostro interlocutore sia una femmina oppure un maschio. Allora gli chiediamo se è femmina o maschio e poi scriviamo un messaggio se è una femmina, un messaggio diverso se è un maschio.

if

L'istruzione che ci permette di scegliere cosa fare si chiama **if**, che in inglese significa SE.
Questo è il codice del programma che descrive l'esempio appena fatto (provalo subito con l'interprete di Python) :

```
nome = raw_input("Scrivi il tuo nome ")
utente = raw_input("Sei femmina? ")
if utente == "si":
     print "Cara ", nome, ", sei bravissima!"
if utente == "no":
     print "Caro ", nome, ", sei bravissimo!"
```

if
squadra_del_cuore
== "Juve"
print "Evviva!"

Vediamo altri esempi concreti per capire bene il significato di "scelta":
A casa: SE suonano alla porta vado ad aprire.
In questo caso l'azione di andare ad aprire viene eseguita solo se si verifica una precisa condizione: SE suonano alla porta.
Una mia azione (andare ad aprire la porta) avviene solo se la condizione (SE suonano) si verifica.
A scuola:
SE domani c'è ginnastica devo portare le scarpe da ginnastica
SE hai fatto i compiti puoi andare a giocare
SE i lati e gli angoli di un poligono sono tutti uguali il poligono è regolare
Prova ad analizzare gli esempi indicando per ognuno la condizione iniziale e l'azione conseguente nella tabella seguente e compila la tabella. Ovviamente puoi aggiungere tutti quelli che ti vengono in mente.

Condizione	Azione
domani ginnastica?	portare le scarpe
compiti finiti?	vado a giocare
....	....

if

voto = raw_input ("Che voto hai preso?) "
if voto >= 6 : print "promosso!"
sei promosso se il voto è maggiore o uguale a 6
Analogamente si potrebbe scrivere il programma, dopo aver definito le variabili relative:
if tuo_peso > 100 : print "grassone!"
sei grassone se pesi più di 100 kg
if scatola == 2 : print "numero pari"
la scritta numero pari viene stampata se il numero indicato è 2
if x < 0 : print x, "è negativo"
viene stampata la scritta x è negativo se il numero indicato è <0

L'istruzione if si chiama *istruzione condizionale* perchè pone una certa condizione prima dell'esecuzione dell'istruzione successiva. Quando il computer incontra l'istruzione if esegue un certo lavoro se è vera la condizione specificata, altrimenti passa all'istruzione successiva del programma.

Bididibodidibù l'istruzione eseguila tu...

La frase che segue la if si chiama: *condizione*
Quando la *condizione* è soddisfatta, si esegue l'istruzione che segue i due punti (:), altrimenti non si fa nulla.
Il contenuto della scatola **voto** è > 0 = a 6? Allora sei promosso!
Notate che i due punti (:) sono obbligatori e che l'istruzione sarà:

struttura	esempio
if <condizione> :	if voto >= 6:
<istruzione>	print "promosso"

L'istruzione deve stare più all'interno della prima riga (hai presente i margini del foglio? Se la prima riga è allineata al margine sinistro, le linee successive devono stare più a destra). Con un'orribile parola presa in prestito dall'inglese e "italianizzata" si dice che **devono essere "indentate"**. Dopo la "condizione" anziché una sola istruzione, come negli esempi precedenti, possono essere scritte due o più istruzioni come segue:

struttura	esempio
if <condizione> :	if voto >= 6:
<istruzione1>	print "promosso"
<istruzione2>	print "mamma contenta"

La prima istruzione che non sta più all'interno (che non è più "indentata") segnala al computer la fine del blocco di istruzioni che devono essere eseguite se la condizione è soddisfatta.
(Negli esempi precedenti abbiamo scritto l'azione sulla stessa riga dell'intestazione perché si trattava di programmi molto semplici con una sola azione. Non è vietato scrivere l'azione sulla stessa riga dell'intestazione ma nel caso di programmi più complessi può essere causa di errori).
Non c'è un limite al numero di istruzioni che possono comparire nel corpo di un'istruzione if **ma deve sempre essercene almeno una**.

Prova a scrivere questo programma:

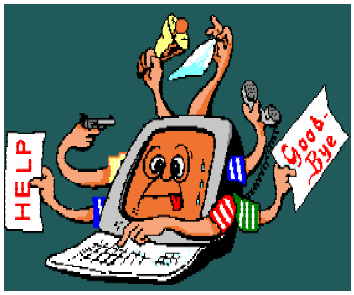
```
if 5 > 10:
    print "sun "
print "moon"
```

il computer scriverà "moon" perchè la linea non fa più parte dell'istruzione if. Ma se la linea print "moon" viene indentata farà parte dell'if e non verrà più stampata.

```
if 5 > 10:
     print "sun"
     print "moon"
```

Questo perchè Python deve sapere dove finisce l'istruzione if e l'unico modo per dirglielo è indentare le istruzioni.

Adesso prova a cambiare il segno > con < ; cosa succede?



Abbiamo visto negli esempi precedenti che, oltre al segno di uguaglianza (==), esistono anche i segni di disuguaglianza e precisamente:

(>) maggiore di

e

(<) minore di

che sicuramente conoscevi già.

Nella tabella seguente trovi elencati tutti gli operatori che servono all'istruzione if per esaminare il contenuto di una variabile.

Operatore	Funzione
= =	uguale
<	minore di
<=	minore o uguale a
>	maggiore di
>=	maggiore o uguale a
!=	diverso da (primo modo)
< >	diverso da (secondo modo)

Per indicare "uguale a " si usa due volte il segno = **(==)**

Questo perché un solo =, come ben sappiamo, significa "metti nella scatola il cui nome è indicato a sinistra il valore indicato a destra" Ad esempio, per assegnare un valore a una variabile si scrive *voto = 7..*

Vediamo un esempio in cui è un po' più complicato prendere una decisione perché consideriamo più condizioni insieme. Infatti il lavoro deve essere svolto se sono soddisfatte contemporaneamente diverse condizioni, come nell'esempio seguente:

```
maggiorenne = raw_input("hai più di 17 anni' ")
prigione = raw_input("Sei già stato in prigione? "
if maggiorenne == "si":
      if prigione == "no":
            print "puoi votare"
```

Operatori logici and or not

Per risolvere il problema precedente dobbiamo introdurre tre nuovi simboli, il cui significato è simile in italiano e in inglese, che si chiamano "operatori logici":

Operatore	*Significato*
or	"oppure"
and	"e inoltre"
not	"non"

L'ultimo programma che abbiamo scritto usando i nuovi **operatori** diventa molto semplice:

```
if maggiorenne == "si" and prigione == "no" :
        print "puoi votare"
```

Vediamo alcuni esempi concreti in cui questi nuovi operatori sono necessari.

and

SE hai fatto i compiti E c'è il sole puoi andare in bicicletta

```
if compiti == "si" and sole == "si":
    print "puoi andare in bicicletta"
```

SE i tre lati di un triangolo sono uguali, il triangolo è equilatero

```
if lato1 == lato2 and lato1 == lato 3:
    print "triangolo equilatero"
```

SE un numero è maggiore di 10 E minore di 20 allora è compreso fra 10 e 20.

```
if num > 10 and num <20:
    print "il numero ", num," è compreso fra 10 e 20"
```

NOTA BENE! Tutte e due le condizioni richieste devono essere soddisfatte, sia quella dopo if che quella dopo and. Il lavoro viene svolto solo in questo caso.

or

SE in un triangolo la lunghezza del lato1 e quella del lato2 sono uguali OPPURE sono uguali le lunghezze del lato1 e quella del lato3 allora il triangolo è isoscele.

```
if lato1 == lato2 or lato1 == lato3:
    print "triangolo isoscele"
```

NOTA BENE: con l'operatore or non è necessario che siano soddisfatte contemporaneamente tutte le condizioni in esame, ma è sufficiente che, fra due condizioni, ne sia verificata solo una perché il lavoro sia svolto.

not

SE NON hai compiti da fare puoi andare a giocare.

```
fatto_compiti = raw_input("hai fatto i compiti? ")
if not fatto_compiti == "si":
      print "Fannullone! non puoi andare a giocare"
```

SE NON è uguale a zero il divisore la divisione è possibile

```
dividendo = input ("indica il dividendo")
divisore = input ("indica il divisore")
if not divisore == 0 :
      print "il risultato è ", dividendo/divisore
```

Riprendiamo l'esempio del triangolo isoscele e scriviamo un'altra versione del programma, questa volta con il not.

```
If lato1 == lato2 and not lato1 == lato3 or lato1 == lato3
                  and not lato1 == lato2 or lato2 == lato3:
    print "Il triangolo è isoscele"
```

Se indichi tre numeri compresi tra 0 e 90 puoi giocarli al Lotto.

```
num1 = input("dimmi un numero ")
num2 = input("dimmi un altro numero ")
num3 = input("dimmi un terzo numero ")
if num1 > 0 and num1 <= 90 and num2 > 0 and num2 <= 90 and
num3 > 0 and num3 <= 90 and not num1 == num2 and not num2
                                == num3 and not num1 == num3:
     print "Gioca i tre numeri al Lotto!"
```

NOTA BENE: l'operatore not richiede che la condizione espressa non sia soddisfatta, perché esso esprime una negazione.

If … else

L'istruzione **if** può essere anche più complicata, in quanto ci consente di dire cosa fare sia quando la condizione è soddisfatta sia quando non è soddisfatta. Spesso ci capita di voler fare una cosa se la condizione è vera e un'altra se la condizione è falsa. Vediamo un semplice esempio:

```
if x < 3:
      print "x è minore di 3"
else:
      print "x non è minore di 3"
```

L'esempio all'inizio dello step 8 (Sono in casa) è stato riscritto con questa nuova possibilità.
Sono in casa: SE suonano alla porta vado ad aprire ALTRIMENTI continuo a leggere.
Con questa istruzione posso decidere di andare ad aprire se suonano alla porta e di continuare a leggere se non suonano alla porta ed è la risposta alla condizione "suonano alla porta?" che decide quale delle due azioni deve essere fatta.

```
ring = raw_input("Suonano alla porta? ")
if ring == "si" :
      print "vai ad aprire"
else:
      print "continua a leggere"
```

Nell'esempio seguente se voto è >= 6 stampo "promosso", se voto è < 6 stampo "bocciato"

```
voto = input("che voto hai preso? ")
if voto >= 6 :
      print "promosso!"
else:
      print "bocciato!"
```

Riprendiamo alcuni esempi precedenti e arricchiamoli con l'istruzione else.

```
fatto_compiti = raw_input("hai fatto i compiti? ")
if not fatto_compiti == "si":
      print "Fannullone! non puoi andare a giocare"
else:
      print "Bravo! vai a giocare"

dividendo = input("indica il dividendo")
divisore = input("indica il divisore")
if not divisore == 0 :
      print "il risultato è ", dividendo/divisore
else:
      print "la divisione per 0 è impossibile"
```

Il meccanismo dell' "indentazione" consente di scrivere tante istruzioni

sia dopo la if ...: sia dopo la else:. Vediamo alcuni esempi:

```
voto = input("che voto hai preso? ")
if voto >= 6 :
      print "promosso"
      print "bravo!"
else:
      print "bocciato"
      print "devi studiare di piu'!"
```

```
nome = raw_input("Come ti chiami? ")
femmina = raw_input ("Sei femmina? ")
if femmina == "si":
      print "Cara ", nome,
      print " , sei bravissima!"
else:
      print "Caro ", nome
      print "  , sei bravissimo!"
```

1) dati due numeri, stabiliamo se il primo è maggiore o minore del secondo.

```
num1= input("Introduci il primo numero ")
num2 = input("Introduci il secondo numero ")
if num1 > num2:
      print num1, " è maggiore di ", num2
else:
      print num1, " è minore o uguale a ", num2
```

2) area del rettangolo

```
risposta = raw_input("Vuoi sapere come calcolare l'area del rettangolo? (S/N) ")
if risposta == "S":
      print "devi moltiplicare la base per l'altezza"
else:
      print "ciao!"
```

3) scrittori

```
risposta1 = raw_input("chi è l'autore dei Promessi Sposi? ")
if risposta1 == "Manzoni":
      print "la risposta è esatta"
      print "bravo!!"
else:
      print "risposta errata!"
      print "la risposta esatta è: Manzoni"
      print "Studia di più"
risposta2 = raw_input("chi è l'autore della Divina Commedia? ")
if risposta2 == "Dante":
      print "la risposta è esatta"
      print "bravo!!"
else:
      print "risposta errata!"
      print "la risposta esatta è: Dante"
      print "Studia di più"
```

Esercitiamoci un po'

Esercizio n. 1
Che cosa significano le due parole if e else?
Esercizio n. 2
Quale parola introduce un lavoro che deve essere svolto come conseguenza di una condizione?
Esercizio n. 3
Che cosa fa il computer quando non è soddisfatta la condizione introdotta da if?
Esercizio n. 4
Scrivi un esempio di scelta condizionata in cui compaia la congiunzione "e" (and), una in cui compaia la congiunzione "oppure" (or) e uno in cui compaia la negazione "non" (not).
Esercizio n. 5
Scrivi il programma relativo alla seguente scelta condizionata:
SE squilla il telefono ALLORA vai a rispondere.
Esercizio n. 6
Spiega il significato delle seguenti istruzioni.
a) if numero <> 20:
        print numero
b) if qui <20:
        quo = 30
Esercizio n. 7
Scrivi il programma per controllare se un numero è positivo.
Esercizio n. 8
Scrivi il programma che controlla il risultato di una addizione, dati due numeri.
Esercizio n. 9
Scrivi il programma che distingue i numeri positivi e i numeri negativi.
Esercizio n. 10
Scrivi un programma che dati due numeri, li visualizza in ordine crescente (o decrescente).
Esercizio n. 11
Scrivi un programma che dia consigli per i vestiti se piove, se nevica e se fa freddo.
Esercizio n. 12
Scrivi il programma che chiede di indicare l'autore di un libro, se è sbagliato stampa "risposta errata", se è corretto stampa "risposta esatta" e prosegue a chiedere un altro autore per un altro libro (puoi ripeterlo quante volte vuoi)
Per esercitarti un po' di piu' puoi scrivere lo stesso programma per i seguenti argomenti:
a) calciatori e squadre di calcio
b) nazioni e capitali
c) citta' e nome degli abitanti.
Esercizio n. 13
Un indovinello. Prova a scrivere un programma per questo indovinello: "a scuola , se ci sei non la fai, se la fai non ci sei: cos'è?" che stampa le due frasi "bravo, hai indovinato!" e "sbagliato, riprova!"

elif

Purtroppo molte volte ci sono più di due possibilità di scelta e quindi dobbiamo trovare ancora un'altra soluzione.

Vediamo un esempio pratico: cosa fa un automobilista che si trova ad un punto in cui la strada si biforca. Pensa: "Se andando a sinistra arrivo a casa, allora vado a sinistra, altrimenti vado a destra".
Supponiamo che decida di andare a sinistra e che ora si trovi ad un altro incrocio. A questo punto dovrà scegliere non solo la strada giusta ma anche la più breve.
Usando l'istruzione if...else scrivere il programma diventa troppo complicato, Per fortuna Python ci aiuta con :

if...elif...else.

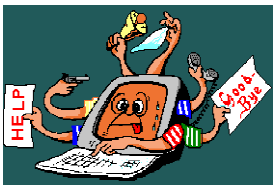
Vediamo alcuni esempi:

```
x = input("indica il numero x ")
y = input("indica il numero y ")
if x < y:
  print x, "è minore di", y
elif x > y:
  print x, "è maggiore di", y
else:
  print x, "e", y, "sono uguali"
```

```
nome = raw_input("qual è il tuo nome? ")
if nome == "Carlo":
    print "il tuo onomastico è il 4 novembre"
elif nome == "Francesca":
    print "il tuo onomastico è il 9 marzo"
elif nome == "Anna":
    print "il tuo onomastico è il 26 luglio"
elif nome == "Andrea":
    print "il tuo onomastico è il 30 novembre"
elif nome == "Stefano"
    print "il tuo onomastico è il 26 dicembre"
else:
    print "non so quando è il tuo onomastico ma spero che
                                ogni giorno sia la tua festa!"
```

Definizione

elif è l'abbreviazione di "else if", che in inglese significa "altrimenti se".
In modo formale questa istruzione viene definita: **condizioni in serie**.



Non c'è alcun limite al numero di istruzioni elif .

È possibile inserire **un'unica** istruzione else **che deve essere l'ultima dell'elenco** e che rappresenta l'azione da eseguire quando nessuna delle condizioni precedenti è stata soddisfatta.

Non é obbligatorio inserire l'istruzione else.

Se nessuna delle precedenti condizioni è vera ed è presente un else verrà eseguito il codice dopo else; se else non è presente non verrà eseguito niente.

Un altro buon esempio è provare ad indovinare il numero indicato.

```
numero = 78
indovina = 0
print "indovina il numero"
indovina = input("indovina il numero: ")
if indovina > numero:
    print "troppo alto"
elif
    indovina < numero
    print "troppo basso"
else
    print "Giusto!!"
```

Ora vediamo un esempio senza else finale.

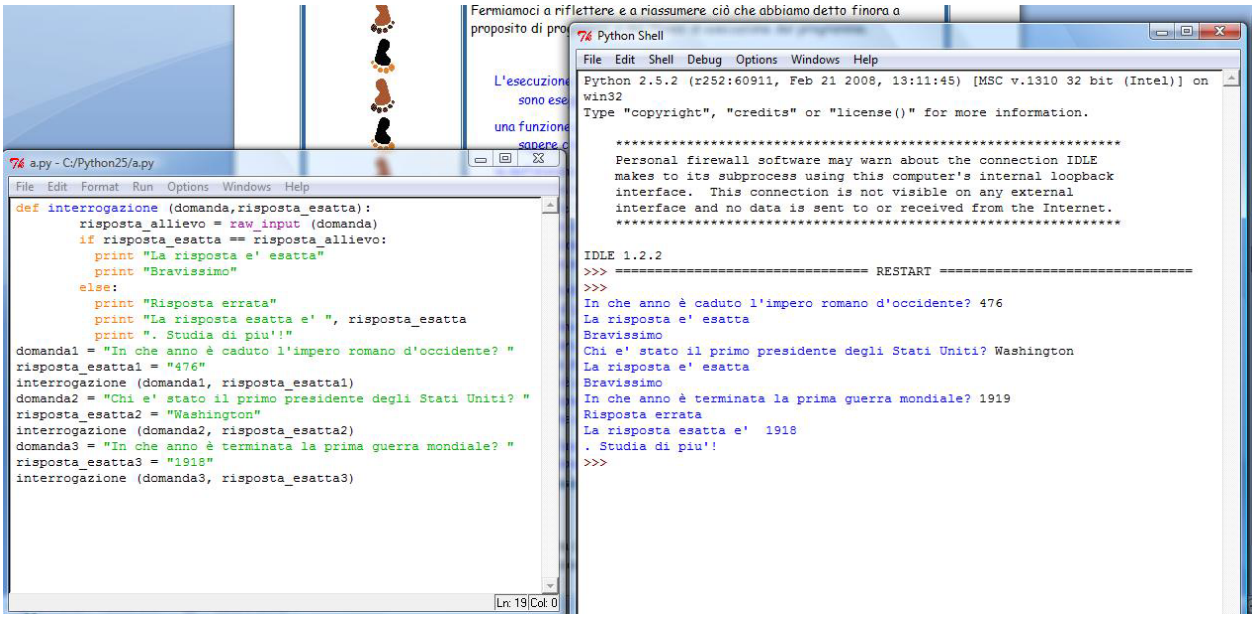
```
print "stampa se un numero é pari o dispari"
numero = input("scrivi un numero: ")
if numero %2 == 0:
      print numero, " é pari"
elif numero %2 == 1:
      print numero, " é dispari"
```

(se non é pari e non é dispari é un numero un po' strano e allora non facciamo nulla)

STEP 9

In questo step impareremo:

* cos'è una **funzione**
* quando è utile usare una funzione
* come si scrive una funzione
* come si fa a chiamare una funzione



STEP 9

Ovvero, un passo dopo l'altro impariamo a programmare

LE FUNZIONI

Una funzione è soltanto un pezzo di programma cui viene dato un nome.

Riprendiamo in esame l'esempio degli "scrittori" esaminato nello STEP precedente:

```
risposta1 = raw_input ("chi e' l'autore dei Promessi Sposi? ")
if risposta1 == "Manzoni":
       print "la risposta e' esatta"
       print "bravo!!"
else:
       print "risposta errata!"
       print "la risposta esatta e': Manzoni"
       print "Studia di piu'"
risposta2 = raw_input ("chi e' l'autore della Divina Commedia? ")
if risposta2 == "Dante":
       print "la risposta e' esatta"
       print "bravo!!"
else:
       print "risposta errata!"
       print "la risposta esatta e': Dante"
      print "Studia di piu'"
```

Hai osservato come il programmatore nel trattare la seconda risposta ripete molte cose che aveva già scritto per la prima ? Questo non è né comodo né veloce. Fortunatamente esiste in Python, come in molti altri linguaggi di programmazione, un meccanismo semplice per specificare una sola volta delle operazioni ricorrenti e per richiamare quelle operazioni ogni volta che ci servono.

Questo meccanismo è la "funzione"

Le funzioni sono utili quando dobbiamo far eseguire al computer piu' volte le stesse operazioni perche' queste si possono scrivere una sola volta nella funzione. Questa sara' identificata con un nome. Ogni volta che abbiamo bisogno di eseguire quelle operazioni chiamiamo la funzione, scrivendo semplicemente il suo nome.

Possiamo immaginare la funzione come un "cameriere" (in gergo tecnico tra gli informatici si chiama "*servente*") al quale si insegnano certe mansioni da svolgere in casa.

Ad esempio, al cameriere si spiega come "preparare la colazione" nel modo seguente:

1. prepara i croissant
2. fai il caffè
3. scalda il latte

Dopo averlo spiegato una volta non e' piu' necessario ripetere tutte le volte gli ordini 1, 2 e 3 ma basta chiedere al cameriere di "preparare la colazione".

Se il cameriere svolge anche le funzioni di cuoco, noi gli spieghiamo una ricetta per fare gli spaghetti alla carbonara che ci piacciono tanto e gliela scriviamo su un foglio. Ogni volta che vogliamo mangiare gli spaghetti alla carbonara gli diciamo semplicemente "spaghetti alla carbonara". A questo punto il cuoco si rilegge la ricetta ed esegue le operazioni elementari la` descritte (fai bollire l'acqua, aggiungi il sale, ecc. ecc.)
Nel caso del nostro esempio del programma degli "scrittori" si deve chiedere al cameriere, ossia alla funzione, di eseguire le seguenti operazioni:

1. porre la domanda e acquisire la risposta
2. verificare se la risposta data e' uguale a quella corretta
3. se la risposta e' corretta, visualizzare il messaggio di congratulazioni
4. se la risposta e' errata, visualizzare il messaggio di rimprovero e il messaggio per ricordare la risposta corretta.

Il nostro "cameriere", ossia la funzione, dovrà eseguire proprio queste operazioni ogni volta che porremo una nuova domanda. Al cameriere, ossia alla funzione, sarà sufficiente indicare la domanda e la risposta esatta.

In pratica dobbiamo scrivere un breve programma che esegua le operazioni che abbiamo elencato, dopo aver dato il nome alla funzione e alle scatole (ossia alle variabili) di cui la funzione avrà bisogno.
La nostra funzione si chiamerà "**interrogazione**".
domanda sarà la scatola contenente la domanda.
risposta_esatta sarà la scatola destinata a contenere la risposta esatta.
Tutto questo andrà scritto su una sola riga, che sarà la prima linea della funzione.

DEFINIZIONE DI FUNZIONE

def interrogazione (domanda, risposta_esatta)

Questa linea indica all'interprete che vogliamo definire un blocco di istruzioni che potrà essere eseguito a richiesta in un qualunque altro punto del programma semplicemente scrivendo il suo nome (interrogazione), dopo aver indicato la domanda e la risposta corretta.

def

Quindi la prima linea della funzione dovrà contenere:

la parolina def (abb. ne di define, definisci)	***def***
il nome che ho deciso di dare alla funzione	*interrogazione*
i nomi delle scatole su cui lavorerà la funzione, separati da virgole e racchiusi da parentesi	*domanda* *risposta_esatta*

Dopo la definizione, occorre scrivere la funzione, cioè il programma che lavorerà sulle scatole indicate. Nel nostro caso:

```
def interrogazione (domanda, risposta_esatta):
risposta_allievo = raw_input(domanda)
if risposta_esatta == risposta_allievo:
            print "La risposta e' esatta"
            print "Bravissimo"
else:
            print "Risposta errata"
            print "La risposta esatta e' ", risposta_esatta
            print "studia di piu'!"
```

Confrontiamolo ora con il pezzo di programma da cui siamo partiti e che non volevamo ripetere tante Volte. Notiamo subito che il codice e' diventato molto piu' generale perchè tutte le risposte sono scritte dentro delle variabili che potranno assumere valori diversi in situazioni diverse.
Mentre il programma valeva solo per domande sugli scrittori, la funzione può essere usata in un programma che pone domande di storia, di geografia, di calcio ecc.

Programma	**Funzione**

`if risposta == "Manzoni":`
`  print "la risposta è esatta"`
`  print "bravo!!"`
`else:`
`  print "risposta errata!"`
`  print "la risposta esatta è: Manzoni"`
`  print "Studia di più"`
	`def interrogazione (domanda,risposta_esatta):`
	`  risposta_allievo = raw_input (domanda)`
	`  if risposta_esatta == risposta_allievo:`
	`  print "La risposta è esatta"`
	`  print "Bravissimo"`
	`  else:`
	`  print "Risposta errata"`
	`  print " La risposta esatta è ",risposta_esatta`
	`  print "studia di più!"`

Puoi usare qualsiasi nome per una funzione,

tranne le parole riservate di Python. Le definizioni di funzione sono delle istruzioni come le altre. Tuttavia, esse saranno eseguite soltanto quando saranno chiamate.

Nota bene!!
Le istruzioni all'interno di una definizione non sono eseguite finchè la funzione non viene chiamata.

CHIAMATA DI FUNZIONE

Ora dobbiamo imparare come chiamare la funzione, ossia ordinare alla funzione stessa di eseguire le operazioni in essa contenute.

In generale, la chiamata di una funzione viene effettuata scrivendo il nome della funzione, seguita dai nomi delle scatole su cui lavorare separati da virgole e racchiuse da parentesi.
Nel caso del programma che avevamo scritto avremo:

PRIMA CHIAMATA

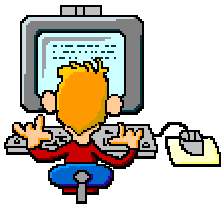
SECONDA CHIAMATA

```
domanda1 = "chi e' l'autore dei Promessi Sposi?"
risposta_esatta1 = "Manzoni"
interrogazione (domanda1, risposta_esatta1)
domanda2 = "chi e' l'autore della Divina Commedia?"
risposta_esatta2 = "Dante"
interrogazione (domanda2, risposta_esatta2)
```

Passo dopo passo…

Fermiamoci a riflettere e a riassumere ciò che abbiamo detto finora a proposito di programmi e del flusso di esecuzione del programma.

- L'esecuzione inizia sempre alla prima riga del programma e le istruzioni sono eseguite una alla volta dall'alto verso il basso.
- Una funzione deve essere definita prima del suo uso (l'interprete deve sapere che la funzione esiste e cosa fa)
- La definizione di funzione non altera il flusso di esecuzione del programma e le istruzioni all'interno della funzione non sono eseguite finché questa non viene chiamata (questo vuol dire che, se definisco la funzione all'inizio del programma, l'interprete legge la definizione ma non fa nulla e prosegue nell'esecuzione del programma finché non trova la chiamata della funzione)
- La chiamata della funzione è una deviazione nel flusso di esecuzione: invece di proseguire con l'istruzione successiva, l'esecuzione salta alla prima riga della funzione chiamata ed esegue tutte le sue istruzioni; alla fine della funzione il flusso riprende dal punto dov'era stato deviato dalla chiamata di funzione. Fortunatamente Python è sufficientemente intelligente da ricordare dove il flusso di esecuzione viene via via interrotto e sa dove riprendere quando una funzione è conclusa
- Quando il flusso del programma giunge all'ultima istruzione, dopo la sua esecuzione il programma è terminato.
- Ricordate che la definizione della funzione termina con l'ultima istruzione indentata.



Vediamo ora il nostro programma scritto con e senza l'uso della funzione "interrogazione":

(il programma è scritto in nero e la definizione di funzione in blu)

senza	con

```
risposta1 = raw_input("chi e' l'autore dei Promessi Sposi? ")
if risposta1 == "Manzoni":
        print "la risposta e' esatta"
        print "bravo!!"
else:
        print "risposta errata!"
        print "la risposta esatta e': Manzoni"
        print "Studia di piu'"
risposta2 = raw_input("chi e' l'autore della Divina Commedia? ")
if risposta2 == "Dante":
        print "la risposta e' esatta"
        print "bravo!!"
else:
        print "risposta errata!"
        print "la risposta esatta e': Dante"
        print "Studia di piu'"
```

```
def interrogazione(domanda,risposta_esatta):
        risposta_allievo = raw_input(domanda)
        if risposta_esatta == risposta_allievo:
          print "La risposta e' esatta"
          print "Bravissimo"
        else:
          print "Risposta errata"
          print "La risposta esatta e' ", risposta_esatta
          print "Studia di piu'!"

domanda1 = "chi e' l'autore dei Promessi Sposi? "
risposta_esatta1 = "Manzoni"
interrogazione (domanda1, risposta_esatta1)
domanda2 = "chi e' l'autore della Divina Commedia? "
risposta_esatta2 = "Dante"
interrogazione (domanda2, risposta_esatta2)
```

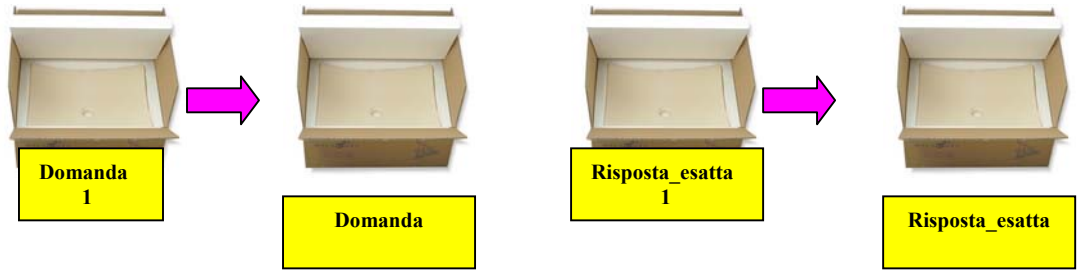
Nota bene!
I nomi delle scatole usati nella istruzione di chiamata della funzione sono diversi dai nomi delle scatole usati nella definizione.

```
def interrogazione (domanda,risposta_esatta):
```
→ DEFINIZIONE

```
interrogazione (domanda1, risposta_esatta1)
```
→ CHIAMATA

Infatti la prima cosa che fa la funzione quando viene chiamata e' prendere le scatole usate nella istruzione con cui e' stata chiamata, scoperchiarle e mettere il loro contenuto nelle corrispondenti scatole usate all'interno della funzione.

Il contenuto di **domanda1** viene scritto in domanda e il contenuto di **risposta_esatta1** viene scritto in risposta_esatta.



Definizione	La funzione è una sequenza di istruzioni che esegue una determinata operazione.

Perché è necessaria questa operazione?

Le scatole all'interno della funzione sono "invisibili" all'esterno e sono utilizzate solo dentro il corpo della funzione.
In questo modo la semplice funzione che abbiamo scritto potrà essere utilizzata da voi per altri programmi o da qualche vostro amico per costruire programmi piu' complicati. E questo e' il secondo motivo, forse il piu' importante, per cui si scrivono le funzioni.
I programmi attuali sono diventati cosi complicati che nessun programmatore, per quanto bravo, riuscirebbe a scriverli da solo. Qualunque programma e' sempre composto per una piccola parte da nuove istruzioni e per la maggior parte da funzioni già scritte in precedenza.

A proposito di funzioni…

Adesso ti propongo un esercizio basato sull'utilizzo delle funzioni

Inserisci un numero e stampa la sua radice quadrata.

Per risolvere questo esercizio devi prima leggere questa breve spiegazione.
Ora che sai cosa sono le funzioni, devi sapere anche che Python è provvisto di una raccolta di funzioni già pronte che ti permettono di eseguire le più comuni operazioni matematiche.
La raccolta di funzioni si chiama ovviamente: math
Prima di poter usare le funzioni fornite da math, devi dire all'interprete di caricare le funzioni in memoria.
Questa operazione, che si chiama "importazione", si scrive:

```
>>> import math
```

A questo punto hai a disposizione tutte le funzioni matematiche tra cui anche la radice quadrata. Per usare la funzione "radice quadrata" dovrai scrivere:

```
nome_modulo.nome_funzione (valore)
```

nel nostro caso sarà: math.sqrt( )

Ad esempio:

```
import math
print math.sqrt(4)
```

Soluzione:

```
import math

numero = input("Scrivi un numero")
```

print "la radice quadrata di", numero, 'è', math.sqrt(numero)

ancora funzioni…ma un po' più difficili
l'istruzione return

In tutti gli esempi precedenti, la funzione chiamata eseguiva completamente una certa attività. A differenza dei casi precedenti, qualche volta la funzione chiamata esegue dei calcoli il cui risultato serve al programma chiamante. Vediamo un esempio:

```
def doppio(numero):
        numero_per_due = numero * 2
        return numero_per_due
```

L'istruzione `return numero_per_due` ordina alla funzione **doppio** di trasmettere al programma chiamante il valore `numero_per_due`.
Nel programma chiamante non si dovrà scrivere `numero_per_due` perché questa è una variabile della funzione e le variabili delle funzioni sono invisibili all'esterno, ma si scriverà un'istruzione del tipo:

```
pippo = 7 + doppio(5)
```

In sostanza tutto avviene come se il contenuto della scatola della funzione `numero_per_due` fosse trasferito nella scatola del programma principale **doppio**.

funzioni che chiamano funzioni

Nel corpo di una funzione si può anche scrivere una o più istruzioni che chiamino altre funzioni. Al limite, una funzione può anche chiamare se stessa.
Vediamo un esempio difficile.
Supponiamo di voler calcolare il fattoriale del numero 7. Occorre scrivere:

```
fatt = 7*6*5*4*3*2
```

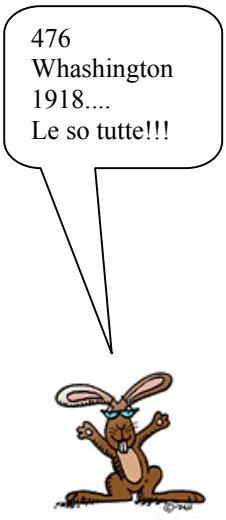
La funzione che calcola il fattoriale di un numero n dovrà essere scritta come n moltiplicato per il fattoriale di (n-1); ad esempio:

```
fatt(7) = 7 * fatt(6)
```

E la funzione sarà:

```
def fatt(n):
        if n == 1:
            return 1
        else:
        return n * fatt(n-1)
```

Dopo una lezione così…….proviamo a fare qualche esercizio insieme prima di proseguire.



Riscriviamo per comodità la funzione interrogazione:

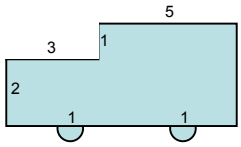
```
def interrogazione (domanda,risposta_esatta):
        risposta_allievo = raw_input (domanda)
        if risposta_esatta == risposta_allievo:
          print "La risposta e' esatta"
          print "Bravissimo"
        else:
          print "Risposta errata"
          print "La risposta esatta e' ", risposta_esatta
          print ". Studia di piu'!"
```

Ora proviamo a inventare altre situazioni in cui questa funzione può tornare utile. Ad esempio, una interrogazione di storia.

```
def interrogazione (domanda,risposta_esatta):
      risposta_allievo = raw_input (domanda)
      if risposta_esatta == risposta_allievo:
        print "La risposta e' esatta"
        print "Bravissimo"
      else:
        print "Risposta errata"
        print "La risposta esatta e' ", risposta_esatta
        print ". Studia di piu'!"
domanda1 = "In che anno è caduto l'impero romano d'occidente?"
risposta_esatta1 = "476"
interrogazione (domanda1, risposta_esatta1)
domanda2 = "Chi fu il primo presidente degli Stati Uniti?"
risposta_esatta2 = "Washington"
interrogazione (domanda2, risposta_esatta2)
domanda3 = "In che anno è terminata la prima guerra mondiale?
"risposta_esatta3 = "1918"
interrogazione (domanda3, risposta_esatta3)
```

Vediamo alcuni esempi di funzioni che utilizzano l'istruzione `return`.

```
print "Programma che calcola il doppio di un numero"
def doppio(numero):
    numero_per_due = numero*2
    return numero_per_due
numero = input("Inserisci un numero ")
doppio(numero)
print "il doppio è: ", numero
```



In questo esempio calcoliamo l'area della figura a fianco.

Scriviamo una funzione che consenta di calcolare l'area di un rettangolo.

```
def area_rettangolo(base, altezza):
        return base * altezza
```

Scriviamo una funzione che calcoli l'area di un cerchio di raggio r.

```
def area_cerchio(raggio):
        return 3,14 * raggio**2
```

Utilizziamo ora le due funzioni che abbiamo scritto per calcolare l'area della figura azzurra.

```
print "Programma che calcola l'area della figura azzurra"
def area_rettangolo(base, altezza):
    return base * altezza
def area_cerchio(raggio):
    return 3,14 * raggio**2
area_figura = area_rettangolo(3,2) +
area_rettangolo(5,3) + 2 * area_cerchio(1)/2
print "l'area della figura azzurra è: ", area_figura
```

Quest'ultimo esempio mostra come calcolare la potenza di un numero senza usare la funzione definita da python ma usando una funzione definita da noi.

```
def potenza(numero, esponente):
    if esponente == 1:
          return numero
    else:
          return potenza(numero,esponente-1) * numero
numero = input("Inserisci un numero ")
esponente = input("Inserisci l'esponente ")
print potenza(numero, esponente)
```

E' arrivato il momento di…	ESERCITARCI CON PYTHON
	Utilizzando Python Shell risolvi gli esercizi. Quando hai finito, vai al fondo del libro dove trovi le soluzioni; confrontale con quello che hai scritto e assegnati 2 punti per ogni esercizio eseguito correttamente, nessun punto se l'esercizio è incompleto o errato. Quando hai finito vai alla pagina dell' autovalutazione. Buon Lavoro.

Esercizio	Punti
1. Scrivi un programma utilizzando la funzione che chiede due numeri e poi li sommi tra loro.	
2. Scrivi un programma utilizzando la funzione che concateni due stringhe (per esempio due nomi Sandro e Silvia).	
3. Scrivi un programma utilizzando la funzione che visualizzi i colori dell'arcobaleno.	

4. Trova l'errore:
`def facciamofesta(musica):`
`  torte = 5`
`  print "Stasera ci sono",torte,"torte, e la musica di", musica`
 	
`musica = raw_input("Qual'è il tuo cantante preferito?")`
`facciamofesta()`	

5. Scrivi un programma utilizzando la funzione che chiede qual è il tuo primo piatto preferito e il programma risponde "A me piacciono i peperoni in carpione" e poi chiede qual è il secondo preferito e risponde sempre "A me piacciono i peperoni in carpione".	
6. Scrivi un programma utilizzando la funzione e poi un altro programma senza utilizzare le funzioni che chiede un numero, ne fa la somma con 22, lo moltiplica per 5 e poi lo divide per 8 e poi fa la stesse operazioni per un secondo numero.	
7. Scrivi un programma utilizzando la funzione che chiede 2 numeri e visualizzi la somma, il valore medio e visualizzi il minimo tra i due.	
8. Scrivi un programma utilizzando la funzione che calcoli l'area e il volume di un parallelepipedo rettangolo.	
Totale punti	…/16

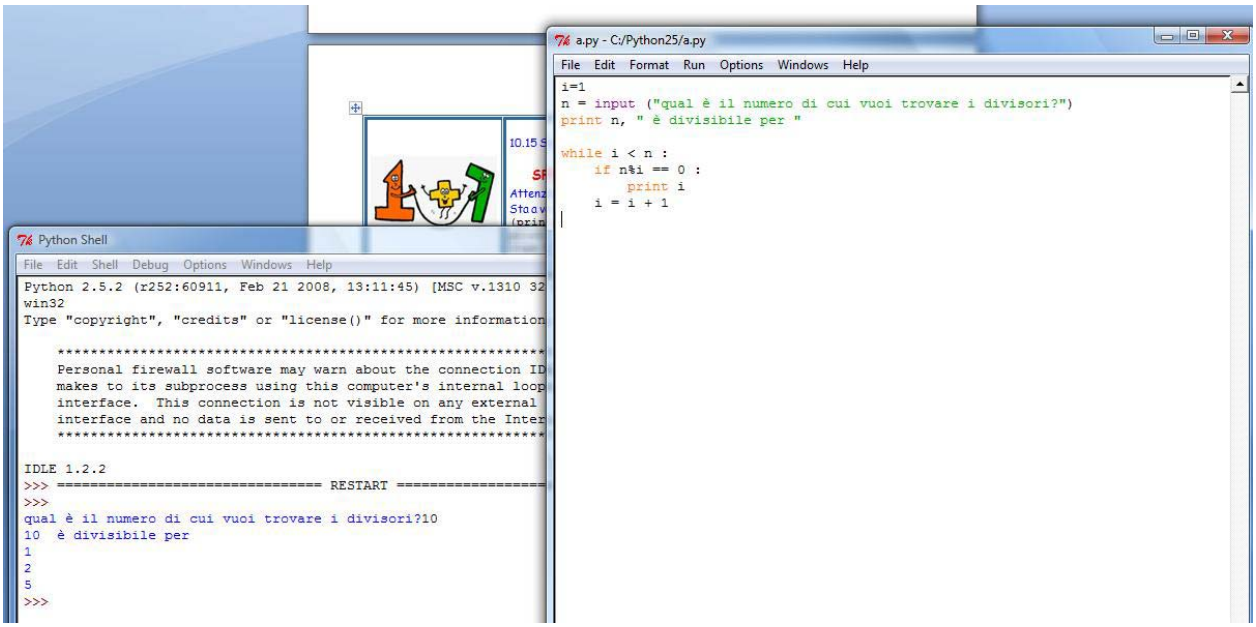
E' arrivato il momento di…	**CONSOLIDARE LE CONOSCENZE**
	Ti sarai ormai accorto che questo è un appuntamento fisso. Ogni tre step approfondiamo un argomento e poi passiamo all'autovalutazione. Questa volta ho chiesto ad Angelo di spiegarti bene che cos'è il software libero e cosa significa esattamente "open source" (la terza domanda del primo step..). Ecco il suo racconto:
	Diciamo subito che software "libero" e software "opensource" non sono sinonimi, anche se sono strettamente legati tra loro. Prima di spiegarti cosa sono ritengo doveroso citare due importanti personaggi del mondo informatico: **Richard Stallman** e **Linus Torvalds**. **Stallman, statunitense,** (New York, 16 marzo 1953) è un programmatore e hacker statunitense. Nel settembre del 1983 lanciò il progetto GNU per creare un **sistema operativo libero**. Con il lancio del Progetto GNU, il movimento del software libero prese vita; nell'Ottobre del 1985 venne fondata la **Free Software Foundation** (FSF). Secondo Richard Stallman **un software per poter essere definito libero deve garantire quattro "libertà fondamentali":** • Libertà di eseguire il programma per qualsiasi scopo (chiamata "libertà 0") • Libertà di studiare il programma e modificarlo ("libertà 1") • Libertà di copiare il programma in modo da aiutare il prossimo ("libertà 2") • Libertà di migliorare il programma e di distribuirne pubblicamente i miglioramenti, in modo tale che tutta la comunità ne tragga beneficio ("libertà 3") **Torvalds** è colui che ha iniziato lo sviluppo del kernel **Linux**. Il sistema operativo GNU/Linux, ottenuto unendo Linux con il sistema GNU, è entrato nella storia come valida alternativa ai sistemi operativi commerciali a licenza chiusa (come per esempio Microsoft Windows). In informatica, **open source** (termine inglese che significa *sorgente aperto*) indica un software rilasciato con un tipo di licenza per la quale il codice sorgente è lasciato alla disponibilità di eventuali sviluppatori, in modo che con la collaborazione (in genere libera e spontanea) il prodotto finale possa raggiungere una complessità maggiore di quanto potrebbe ottenere un singolo gruppo di programmazione. (da: http://it.wikipedia.org)

E' ARRIVATO IL MOMENTO DI……	**AUTOVALUTARCI**
	Adesso proviamo a riflettere sull'attività fin qui svolta. Per verificare se hai fatto un buon lavoro rispondi alle seguenti domande (sinceramente!):
 Vediamo un po'…….	1. che differenza c'è tra input e raw_input? A cosa servono? 2. quali "operatori logici" conosci? 3. qual è l'esatta definizione di funzione? Per ogni risposta corretta hai guadagnato 2 punti. Segna qui a fianco il punteggio ottenuto …/6
Tiriamo le somme	Adesso somma il punteggio ottenuto rispondendo alle domande sopra con quello degli esercizi svolti nella pagina precedente: Esercizi ……/16 + Domande ……/6 = ---------- Totale ……/22 punti Se hai totalizzato meno di 12 punti ….. segui i consigli di Super Teacher : studia di più, esercitati di più, chiedi aiuto al tuo insegnante per capire meglio prima di proseguire l'attività. Se hai ottenuto 12 punti o più puoi passare direttamente allo step 10.
	Qualcosa ti è sembrato difficile? Non hai capito qualche argomento? Annotalo qui sotto e segnalalo al tuo insegnante: ---------- ---------- ---------- Se invece il lavoro che abbiamo fatto per te ti è piaciuto disegna tante meline rosse (da 1 a 10) ----------

STEP 10

In questo step impareremo:

* ad usare i **cicli "loop"**
* a creare cicli annidati
* quando è opportuno utilizzare l'istruzione **`while`** o piuttosto l'istruzione **`for`**
* a conoscere e utilizzare meglio le stringhe



STEP 10

Ovvero, un passo dopo l'altro impariamo a programmare

I Cicli

Prima di proseguire… rinfreschiamoci La memoria

Come già sai è possibile assegnare più valori ad una stessa variabile, con la variabile che assume sempre l'ultimo valore assegnato.
Ad esempio:

```
Num = 3
print Num,
Num = 9
print Num
```

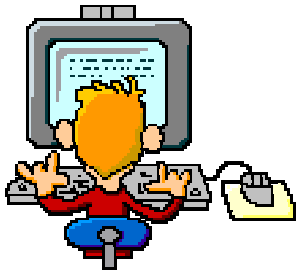
Il risultato di questo programma è 3 9, perché la prima volta che Num è stampato il suo valore e` 3, la seconda 9.

Nota bene: La virgola dopo la prima istruzione `print` evita il ritorno a capo dopo la stampa; in questo modo entrambi i valori appaiono sulla stessa riga.

Detto questo, prova a scrivere il seguente programma e analizza cosa succede.

```
a =1
while a < 4 :
  print a
  a = a + 1
```

Prova a leggere il programma con l'istruzione while come fosse scritto in un linguaggio naturale (***while*** *significa: finchè, fintantoche`*). Otterrai:
"Finché (`while`) a è minore di 4 stampa il valore di a e poi aumentalo di 1. Quando arrivi a 4 esci perché a deve essere minore di 4".



Prova ora a scrivere un programma per stampare:

1. tutti i numeri minori di 10
2. tutti i numeri da 100 a 1000 compresi.

Se non ci riesci prosegui a leggere: troverai i due programmi analizzati in dettaglio più avanti.

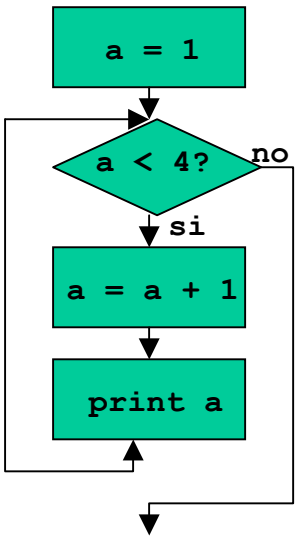
Come hai capito questo comando viene usato per eseguire le azioni ripetitive e i computer sono bravissimi a ripetere le azioni tante volte. Per risolvere questo tipo di problema si usano i **"cicli" o "loop"** (che in inglese significa appunto "ciclo").

Ci sono due tipi di "cicli" e tra i due c'è una differenza molto importante.

Il primo, chiamato **`while`**, viene usato quando non si sa prima quante volte deve ripetere l'azione.

Il secondo, chiamato **`for`**, è usato quando si sa in anticipo quante volte si dovrà ripetere l'azione.

Il programma precedente può essere rappresentato cosi:




```
a =1
while a < 4 :
  print a
  a = a + 1
```

Vediamo ora iinsieme i programmi 1 e 2.

1.Stampa tutti i numeri minori di 10

```
a = 0
while a < 10:
      print a,
      a = a + 1
```

2.Stampa tutti i numeri da 100 a 1000

```
a = 100
while a <= 1000:
      print a,
      a = a + 1
```

Tralasciamo per un attimo il ciclo **for**, che studieremo nel prossimo step, per imparare bene ad utilizzare:

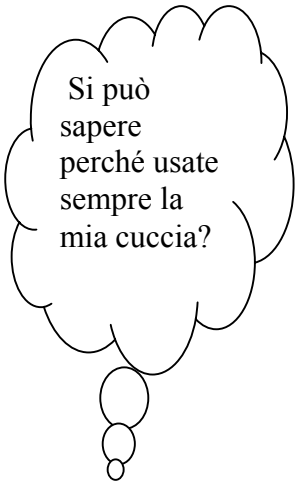
while <relazione>:

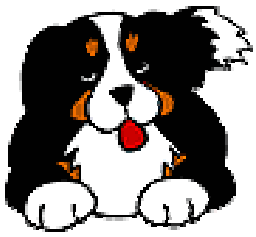
Riprendiamo ora il semplice programmino di prima:	Avete visto che il programma visualizza:

`a =1`
`while a < 4 :`
`  print a`
`  a = a + 1`
	1
	2
	3

Ripetiamo. L'istruzione while significa: "fai tutto quello che segue **fintantoche`** la relazione scritta fra la parola `while` e i due punti è soddisfatta".

Nota bene: i due punti sono obbligatori





In modo più formale ecco il flusso di esecuzione di un'istruzione `while`:

1. Valuta la condizione controllando se essa è vera o falsa.
2. Se la condizione è falsa esci dal ciclo `while` e continua l'esecuzione dalla prima istruzione che lo segue.
3. Se la condizione è vera esegui tutte le istruzioni nel corpo del `while` e torna al passo 1.

Nel nostro caso il programma esegue le istruzioni:

```
print a
a = a + 1
```

tre volte, cioè finchè a è minore di 4.

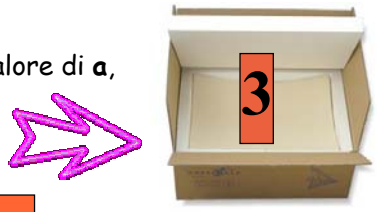
Ecco in dettaglio quello che succede in quelle tre volte.

La prima volta

a vale 1
La relazione **a < 4** è soddisfatta
Il programma esegue prima **print a**
e visualizza **1**
poi esegue **a = a + 1**
e pone il valore **2** nella scatola **a**

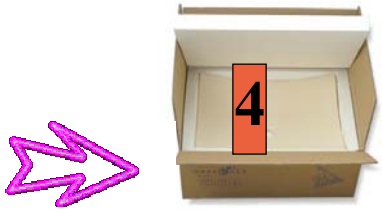
La seconda volta

a vale 2
a < 4 è ancora soddisfatta
Il programma esegue **print a** e visualizza il valore di **a**, cioè **2** e
pone il valore **2 + 1 = 3** nella scatola **a**



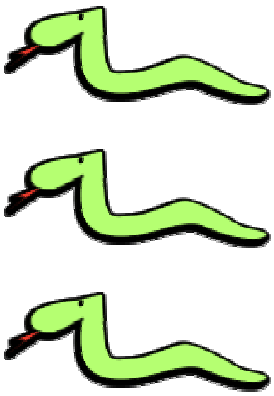
La terza volta

a vale 3
a < 4 è soddisfatta
Il programma visualizza il valore di **a**, cioè **3**,
ed esegue **a = a + 1** ponendo **4** nella scatola **a**.



La quarta volta

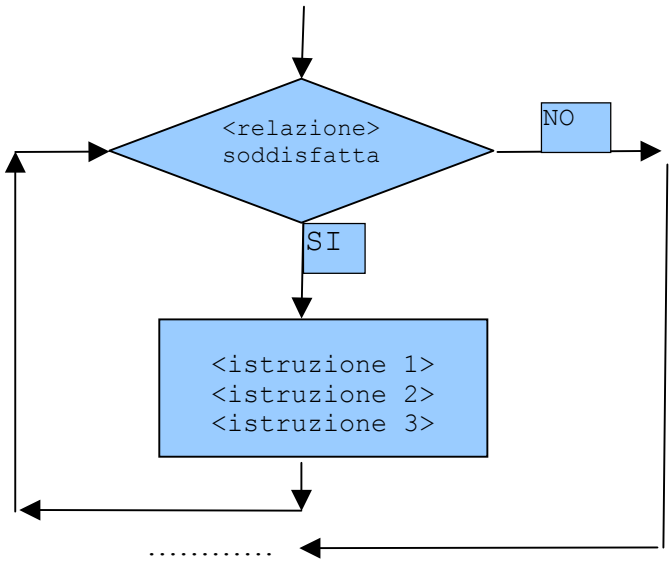
a vale 4
Il programma esce dal ciclo.



Più in generale, se scrivo:

```
while <relazione> :
      <istruzione 1>
      <istruzione 2>
      <istruzione 3>
```

il programma esegue il seguente ciclo:



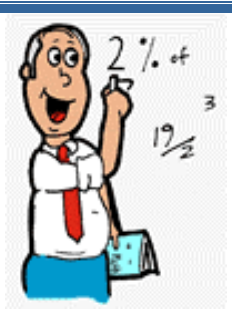
Nota bene: Tutte le istruzioni che seguono la while <relazione> :
devono essere indentate se fanno parte delle cose da fare quando <relazione> è soddisfatta.

Se la condizione è falsa al primo controllo, le istruzioni del corpo non saranno mai eseguite.

Il corpo del ciclo dovrebbe cambiare il valore di una o più variabili così che la condizione possa prima o poi diventare falsa e far così terminare il ciclo. In caso contrario il ciclo si ripeterebbe all'infinito, determinando un **ciclo infinito**.

Esamina ora il seguente programma.

```
a = 3
   while a > 2:
         print a
         a = a + 1
```



Il programma stampa prima 3, poi 4, poi 5 e cosi via, e non si ferma mai.
Perché?

Se volete evitare che il vostro PC visualizzi la successione dei numeri naturali per molti anni, premete contemporaneamente i tasti CTRL e C.

Molto bene, il tuo programma di lavoro sta procedendo regolarmente.
Ora dedica un po' di tempo ad esercitarti, l'esercizio e la pratica sono indispensabili se vuoi diventare un bravo programmatore.

Sei pronto? Vediamo come te la cavi con i prossimi esercizi e con le "sfide" che Mario ti ha preparato.

Esercitiamoci insieme

10.1 Provate ad eseguire questi due programmi.

Programma 1

```
a = 1
while a < 100 :
      a = a+1
```

Programma 2

```
a = 1
while a < 100000 :
      a = a + 1
```

Cosa fanno questi due programmi?
Perchè il secondo dura più del primo?
A cosa possono servire questi programmi?

Qualche volta, nel corso di un programma complesso, può essere opportuno fermarsi un po' di tempo per consentire all'utilizzatore del programma di riflettere prima di proseguire. I due programmi precedenti servono proprio a questo, ossia a perdere tempo.

Quanto tempo?
Dipende dalla velocità del calcolatore. Infatti, un calcolatore molto veloce può eseguire una delle due istruzioni:

```
while a < 100:
    a = a + 1
```

in un decimo di milionesimo di secondo.

Prova a risponder a questa domanda: fa perdere più tempo il programma 1 oppure il programma 2?

10.2 Visualizza tutti i numeri compresi fra 0 e 100.

```
i = 0
 while i <= 100 :
      print i
      i = i + 1
```

10.3 Visualizza tutti i numeri compresi fra 3 e 9.

```
i = 3
while i <= 9 :
      print i
      i = i + 1
```

10.4 Visualizza in ordine decrescente i numeri da 200 a 100.

```
i = 200
while i >= 100 :
      print i
      i = i - 1
```

10.5 Scrivi il programma "Conto alla rovescia" che - finché (while) n è più grande di 0 - stampa il valore di n e poi lo diminuisce di 1. Quando arriva a 0 stampa la stringa "Pronti…VIA!"-.

```
n = 10
while n > 0 :
      print n
      n = n - 1
print "Pronti ...VIA!"
```

SFIDA

10.6 Quando esegui il programma 10.5, scopri che la visualizzazione dei numeri è troppo veloce. Come fai per rallentarla?
(la soluzione l'hai già trovata in questo STEP!)

10.7 Visualizza la tabellina del sette stampando i numeri di seguito sulla stessa riga.

```
i = 1
while i <= 10 :
      print 7*i,
      i = i + 1
```

10.8 Scrivi un programma che attende finche non viene inserita la password corretta.
(La password la decidi tu, ad es. "chicchiricchi").

```
password = "python"
while password != "chicchiricchi":
    password = raw_input("Password:")
print "Entra pure"
```

10.9 Visualizza tutti i numeri dispari compresi fra 1 e 100.

```
i = 1
while i < 100 :
    print i
    i = i + 2
```

10.10 Stampa la frase "Aiuto! Non mi fermo più" infinite volte usando il ciclo while.
(Per fermare il programma usa i tasti CTRL + C)

```
i=1
while i != 0 :
    print "Aiuto! Non mi fermo più"
    i = i + 1
```

e poi stampa il messaggio di seguito, senza saltare una riga fra uno e l'altro e senza andare a capo.
(Per fermare il programma usa i tasti CTRL + C)

```
i=1
while i <> 0 :
    print "Aiuto! Non mi fermo più",
    i = i + 1
```

SFIDA

10.11 Prova a trovare i divisori di un numero usando il ciclo `while`.

```
i=1
n = input("qual è il numero di cui vuoi trovare i divisori? ")
print n, " è divisibile per "
while i < n :
    if n%i == 0 :
        print i,
    i = i + 1
```

10.12 Scrivi un programma che chieda il risultato dell'operazione 15 x 17 tante volte sino a quando non viene indicata la soluzione corretta.

```
corretto = "no"
while corretto == "no":
    risposta = input("quanto vale 15 x 17? ")
    if risposta == 15*17:
        corretto = "si"
        print "Brava!"
    else:
        corretto = "no"
```

I Cicli annidati

Precedentemente abbiamo risolto il programma n. 10.5 ma quando lo eseguiamo scopriamo che il conteggio alla rovescia è molto veloce. Troppo.
Avete trovato la soluzione per rallentarlo?
Se si, bene! Leggete comunque come abbiamo illustrato la soluzione qui di seguito.

Per rallentarlo introduciamo il programma che abbiamo già scritto e che serviva a perdere tempo (esercizio 10.1)

```
a = 1
while a < 100 :
      a = a+1
```

e nella relazione da verificare decidiamo quanto aspettare. La soluzione che conta lentamente non è altro che la fusione dei due programmi.

```
n = 10
while n > 0 :
      print n
      a = 1
      while a < 1000000 :
            a = a + 1
      n = n - 1
print "Pronti ...VIA!"
```

Nota bene: Quest'ultimo programma contiene due cicli `while`, di cui il secondo, quello introdotto per "perdere tempo", è ANNIDATO entro il primo.

Notate bene anche come è stato scritto il programma:
l'istruzione `while a < 1000000` è indentata rispetto a `while n > 0`
mentre l'istruzione `a = a + 1` è indentata rispetto a `while a < 1000000`

SFIDA

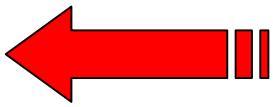
Quante volte viene eseguita l'istruzione a = 1?
Quante volte viene eseguita l'istruzione a = a + 1?

Le istruzioni che possono essere annidate non devono essere necessariamente dello stesso tipo, possono essere disomogenee.
Ad esempio, posso annidare una decisione (IF), oppure un ciclo di altro tipo (FOR), e anche una funzione.

Riprendiamo adesso i due esercizi SFIDA dello STEP 10, l'esercizio 10.11 e l'esercizio 10.12.

La soluzione del primo e' un esempio di annidamento dove l'istruzione annidata è una `if <relazione> :` e la soluzione è la seguente.

```
i=1
n = input("qual è il numero di cui vuoi trovare i divisori?")
print n, " è divisibile per "

while i < n :
    if n%i == 0 :
        print i
    i = i + 1
```



se non ti ricordi più, vai a rivedere il significato dell'operatore %

Nel secondo la `if` annidata è più complessa.

```
corretto = "no"
while corretto == "no":
    risposta = input("quanto vale 15 x 17? ")
    if risposta == 15*17:
        corretto = "si"
        print "Brava!"
    else:
        corretto = "no"
```

Fai attenzione all'**indentazione con cui sono scritte le istruzioni** perchè è fondamentale per avere un'esecuzione corretta del programma!

Proviamo a fare qualche esercizio insieme:

10.13 Scrivi un programma per stampare il quadrato e il cubo dei numeri da 1 a 10.

```
i = 1
while i <= 10 :
    e = 1
    while e <= 3 :
        print i ** e, " ",
        e = e + 1
    print
    i = i + 1
```

10.14 Scrivi un programma che chiede una sequenza di numeri da aggiungere ad una somma. Per terminare inserisci 0.

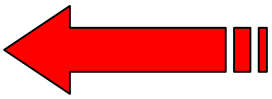
```
a = 1
somma = 0
print 'Inserisci i numeri da aggiungere alla somma '
print 'Quando hai finito inserisci 0'
while a != 0 :
        print 'La somma è:',somma
        a = input('Numero? ')
        somma = somma + a
print 'Totale =',somma
```

10.15 Scrivi un programma per stampare la Tavola Pitagorica.

SFIDA

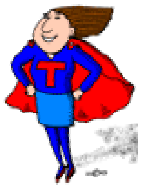
Attenzione: la stampa dei numeri non risulterà bene in colonna.
Sta a voi trovare il modo per stampare bene le colonne.

```
print " TAVOLA PITAGORICA"
print
riga = 1
while riga <= 10 :
    colonna = 1
    while colonna <= 10 :
        print '\t', riga * colonna,
        colonna = colonna + 1
    riga = riga + 1
    print
```

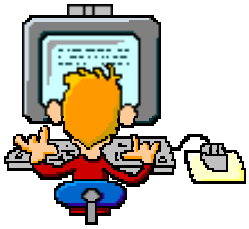


Ora ti spiego un "trucco" per formattare le stampe

Il carattere di backslash '\' indica l'inizio di una sequenza di escape.
Le sequenze di escape sono usate per rappresentare caratteri "speciali" e invisibili come la tabulazione ('\t') e il ritorno a capo ('\n') e possono comparire in qualsiasi punto di una stringa.
Prova ad esempio a stampare una stringa unica che produca questo risultato:



```
vengo
                subito
                                    a casa

(print "vengo \n \t subito \n \t \t a casa")
```



10.16 Indovina un numero! Scrivi un programma per indovinare un numero

```
numero = 27
indovina = 0
while indovina!= numero :
    indovina = input("Indovina un numero: ")
    if indovina > numero:
        print "Troppo grande"
    elif indovina < numero :
            print "Troppo piccolo"

print "BRAVO!!"
```

Prima di proseguire rinfreschiamoci la memoria

….ancora sulle STRINGHE

Sinora abbiamo visto che una stringa è una serie di caratteri, cifre, lettere o altri simboli che si trovano sulla tastiera, cioè un messaggio qualunque. Inoltre, sappiamo che una stringa per essere considerata tale deve essere racchiusa tra virgolette (semplici o doppie) e che si possono fare alcune operazioni con le stringhe.
Possiamo scrivere:

"ciao" * 3 oppure "ciao" + "ciao" + "ciao" oppure "ciao" * 2 + "ciao"

invece

"ciao"/4 oppure "ciao" + 5 oppure "18" + 8

sono sbagliate e generano un `syntax error`.

Vi ricordate come si chiamano le operazioni sulle stringhe?

Che cosa hanno di diverso le stringhe dagli altri tipi di dati (interi, floating point)?

Le stringhe sono qualitativamente diverse dagli altri tipi di dati perchè sono composte di unità più piccole: i caratteri.

Per questo le stringhe sono dati "**composti**" in alternativa ai dati "semplici" che sono gli interi e i floating point.

Questo ci consente di trattare una stringa come fosse una singola entità oppure di agire sulle sue singole parti (i caratteri) a seconda di ciò che stiamo facendo. Come si fa ad agire sui singoli caratteri?

Secondo te quale sarà il risultato di questo programma?

```
squadra = "Juventus"
messaggio = squadra[0] + squadra[2] + squadra[3] + squadra[4] + squadra[5] + squadra[6] + squadra[7]
print squadra
print messaggio


>>>Juventus
>>>Jventus
```

Come si individuano i singoli caratteri.

Nella variabile `squadra` abbiamo messo la stringa tutta intera mentre nella variabile `messaggio` l'abbiamo inserita come sequenza di caratteri.

`squadra[i]` e` una stringa fatta di un solo carattere e non e` altro che il carattere di `Juventus` che occupa la posizione i + 1.

L'espressione tra parentesi quadrate seleziona i caratteri della stringa. Quindi squadra[0] seleziona il primo carattere, squadra[1] il secondo e cosi via.

Ricordati che i calcolatori iniziano quasi sempre a contare da 0.

L'espressione tra parentesi quadrate è chiamata **indice**.

Un indice individua un particolare elemento di una stringa e può essere una qualsiasi espressione intera.

Un'espressione aritmetica negativa come **–i** fra parentesi quadre **[-i]** indica il carattere posto nella posizione (i + 1) a partire dal fondo.

Così, ad esempio,

`squadra[-0]` è uguale al carattere "s"

`squadra[-2]` è uguale al terzultimo carattere, ossia "t".

Cosa otterrai scrivendo il seguente programma?

```
>>>squadra = "Juventus"
>>>messaggio = "VIVA" + " " + squadra[0] + squadra[1] + squadra[2] + squadra[3]
>>>print squadra
>>>print messaggio
Juventus
VIVA Juve
```

Porzioni di stringa

Nell'esempio precedente abbiamo preso una parte della stringa Juventus sommando i primi 4 caratteri; potevamo ottenere lo stesso risultato usando l'espressione

```
messaggio = "VIVA" + squadra[0:4]
```

che vuol dire "seleziona tutto il pezzo di stringa a partire dal carattere [0], il primo, fino al carattere [4], il quinto, escluso.

Secondo te quale sarà il risultato di:

```
squadra[:3]
>>>
squadra[3:]
>>>
squadra[:]
>>>
```

Se non è specificato il primo indice (prima dei due punti :) la porzione parte dall'inizio della stringa. Senza il secondo indice la porzione finisce con il termine della stringa.

Di conseguenza squadra[:] è la stringa intera.

Gli indici di porzione sbagliati vengono gestiti con eleganza: un indice troppo grande viene rimpiazzato con la dimensione della stringa, un indice destro minore di quello sinistro restituisce una stringa vuota.

A me piacciono le stringhe di liquirizia

Lunghezza di una stringa

La lunghezza di una stringa è rappresentato dal numero di caratteri che contiene.

La funzione **len (nome della stringa)** restituisce la lunghezza della stringa, cioè restituisce il numero di caratteri della stringa.

```
>>>squadra = "Roma"

>>>print len(squadra)
4
```

Quanto vale len("Juventus")?

Osserva che abbiamo scritto **len(squadra)** senza virgolette perchè squadra e` il nome di una variabile e invece abbiamo scritto len("Juventus") perchè in questo caso abbiamo inserito direttamente la stringa tra parentesi.

Quanto vale len("squadra")?

Le stringhe sono immutabili.

```
>>>squadra = "roma"
>>>squadra[0] = "t"
TypeError: object doesn't support item assignment
```

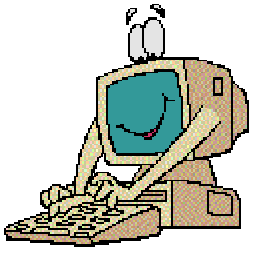
Se proviamo a cambiare il primo carattere della stringa "roma" il risultato non e`toma ma un messaggio di errore perche` **le stringhe non sono modificabili**.

Provate a pensare come si potrebbe ottenere un risultato simile.

La soluzione e`:

```
squadra2 = "t" + squadra[1:4]
```

Abbiamo creato una nuova stringa concatenando la lettera "t" ad una porzione della stringa "roma". Il risultato e` "toma" ma questa operazione non ha avuto alcun effetto sulla stringa originale.

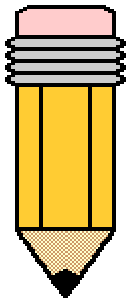


Ora facciamo qualche esercizio per comprendere il significato delle funzioni:

```
>>>3
3
>>>repr(3)                    (converte un intero in una stringa)
'3'
```

Riflettete su questo esempio: il 3 contenuto fra le parentesi è un numero;la risposta del sistema '3' è una stringa costituita da un solo carattere, il carattere '3'.

```
>>>repr(-321)
'-321'
>>>"32"
'32'
```



esercizio

>>>int(32) (converte una stringa in un intero)

32

La funzione **int** produce un numero intero.

Essa può avere come argomento sia una stringa che rappresenti un numero intero sia un'espressione che rappresenti un numero decimale di cui sarà prodotto il valore intero corrispondente, come appare chiaro dagli esempi seguenti.

>>>"32"*2

'3232'

E' questo un risultato che può apparire strano.

Siccome * si applica non al numero 32 ma alla stringa fatta dai due caratteri '3' e '2', * ha il significato di "replica la stringa tante volte", come già sappiamo.

>>> int("32") * 2

64

>>>repr(3.21) (da decimale a string)

'3.21'

>>>int(3.21) (da decimale a intero)

3

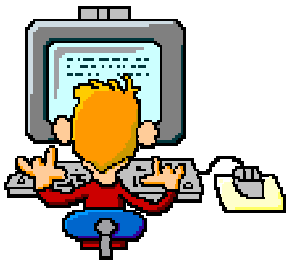
>>>int(-3.21)

-3

>>> float("3.21") (converte una stringa in un decimale)

3.21

Cosa fanno le funzioni repr, int e float?



Che cosa viene visualizzato con il programma?

```
py = "Python"
i=0
while i < 6:
print py[i], '\t' , py[0:i+1], '\t', py[i]+py[i-1]+py[i-2]
i=i+1
```

Ecco il risultato:

```
P       P       Pno
Y       Py      yPn
T       Pyt     tyP
H       Pyth    hty
o       Pytho   oht
N       Python   noh
```

Esercitiamoci un po'

10.17 Un contadino vuole attraversare un fiume portando con sè un lupo, una capra ed un grosso cavolo, ma non dispone che di una barchetta così piccola da non poter contenere che lui e una sola di tali cose. Cosa dovrà fare per evitare che il lupo mangi la capra o questa il cavolo? Risolvi l'indovinello e poi…
Scrivi il programma che lo risolva.

10.18 Scrivi un programma che stampi:

bravo
bravo
peperone
bravo
bravo
peperone peperone
bravo
bravo
peperone peperone peperone

10.19 Scrivi un programma che stampi:

5
4
3
2
1

10.20 Scrivi un programma che stampi:

5
4
3
2
1
2
3
4
5

10.21 Scrivi un programma che simuli un semaforo:
verde 6 cicli
giallo 2 cicli
rosso 4 cicli
(ricorda che la sequenza del semaforo è verde – giallo – rosso – verde)

10.22 Scrivi un programma utilizzando un ciclo while e una funzione per visualizzare la tabellina del 2

10.23 Scrivi un programma utilizzando un ciclo while e una funzione che chiede: "Perché il linguaggio di programmazione di chiama Python?" e ci sono 3 possibilità:

- Il nome viene da pitone
- Il nome viene dai comici inglesi "Monty Python"
- Il nome viene da un personaggio di Harry Potter

10.24 Con un ciclo while stampa:

```
2  3  4  5
4  6  8 10
6  9 12 15
8 12 16 20
10 15 20 25
```

10.25 Trova l'errore

```
i=5
while i < 5
        print "Che bella partita di calcio"
        i=i+1
```

10.26 Trova l'errore

```
def tabellinadel4():
   i=1
   while i <=10:
      print i, i*4
      i=i+1
```

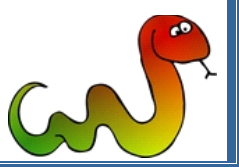
tabellina del 4(4)

10.27 Indovina un numero! Scrivi un programma per indovinare un numero

10.28 Nell'esercizio 10.22 dovevi stampare i multipli di 2; ora prova a generalizzare, scrivendo un programma che calcoli i multipli di un qualsiasi numero intero (ricordati delle funzioni).

Rifletti ancora:

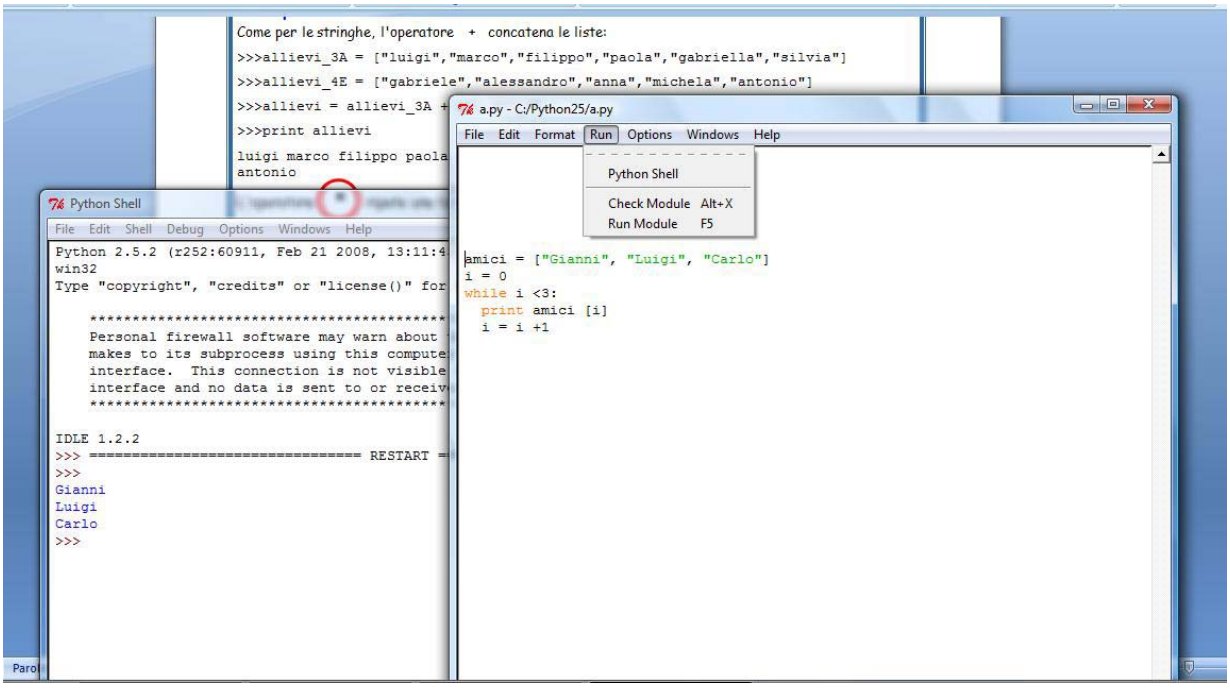
Cosa ottieni se chiami la funzione con il parametro 2?
Come farai a stampare la Tavola Pitagorica?



STEP 11

In questo step impareremo:

* cos'è una **lista**
* quando è utile utilizzare le liste
* a **creare** liste annidate
* ad analizzare tutti gli elementi di una lista utilizzando il ciclo "for"



STEP 11
Ovvero, un passo dopo l'altro impariamo a programmare

LE LISTE

Cos'è una lista? È come quella che si fa quando bisogna fare la spesa? O come quando si fa l'elenco degli amici da invitare alla festa per il compleanno? Esatto!

Definizione

Una lista è un insieme ordinato di valori di qualunque tipo, proprio come la lista della spesa.

Vediamo qualche esempio:

```
[3,6,9,12,15]
```
lista di tutti numeri interi

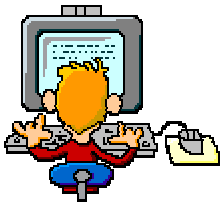
```
["Luigi","Mario","Nicola","Giuseppina"]
```
lista di tutte stringhe

```
["pane","latte","zucchero",1,15,230,"bicchieri",1.5,2.5]
```
lista mista: 3 stringhe, 3 numeri interi, 1 stringa, 2 numeri decimali

I valori della lista sono chiamati *"elementi"*.
Le parentesi quadrate [ ] iniziano e finiscono la lista e la virgola (",") separa un elemento della lista dall'altro.
Come abbiamo visto in uno degli esempi precedenti, non è obbligatorio che gli elementi di una lista siano tutti dello stesso tipo, tutti numeri o tutte stringhe, ossia **una lista può essere non omogenea**.

Esiste poi una lista speciale chiamata "**lista vuota**" e si indica con **[ ]**.
La lista vuota non contiene alcun elemento.
Esempi:

```
["pane","latte","zucchero",1,15,230,"bicchieri",1.5,2.5]
[1,"due",3.0]
[]
["tabellina del cinque",5,10,15,20,25]
```



Ovviamente, alle liste possiamo dare un nome.

```
spesa
=["pane","latte","zucchero",1,15,230,"bicchieri",1.5,2.5]
vocabolario = ["bicicletta","casa","scuola"]
dati_di_delpiero = ["Del Piero", "Alessandro", 1974]
lista_vuota = []
```
Prova a visualizzare sul video il contenuto delle liste precedenti.

Ma le liste sono un tipo di variabile?

Si, le liste sono un tipo di variabile. Variabili "composte".
Le variabili ordinarie immagazzinano un singolo valore.
Le liste sono variabili che possono contenere più di un valore.
Possiamo pensarle come composte da una serie di scatole.

Vediamo ora come si fa ad accedere ai singoli elementi di una lista.
Prova a scrivere il seguente programma e analizza attentamente il risultato:

```
Squadre = ["Juventus", "Inter", "Milan", "Roma"]
i = 0
while i < 4:
  print Squadre[i]
  i = i + 1
```

Questo programma assegna alla lista `squadre` alcuni valori e con il ciclo `while` "attraversa" la lista e ne visualizza un elemento alla volta.
Per farlo usa un indice (nel nostro caso "i").

`Squadre[i]` individua l'elemento **i** della lista `squadre`.

Quindi è possibile accedere separatamente a ciascun elemento della lista.

Ricordi che abbiamo parlato di insieme ordinato di valori?

Gli elementi di una lista sono valori numerati e identificati da un indice.
Un indice identifica un particolare elemento di un insieme ordinato che nel nostro caso è l'insieme degli elementi della lista.
La numerazione degli elementi della lista inizia da 0.

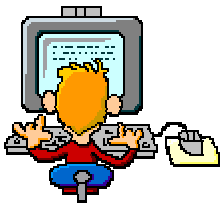
```
Squadre[0] è la stringa "Juventus"
Squadre[1] è la stringa "Inter"
Squadre[2] è la stringa "Milan"
Squadre[3] è la stringa "Roma"
```

oppure

```
dati_di_delpiero[0] è la stringa "Del Piero"
dati_di_delpiero[2] è la stringa "1974"
```

Appare chiaro da questi esempi che l'elemento [0] è il primo e l'elemento [2] è il terzo.

Nota bene: Fai attenzione al fatto che gli indici di una lista, cosi come gli indici di una stringa, iniziano sempre da zero; può sembrarti strano, ma i calcolatori preferiscono iniziare a contare da zero. Quindi per accedere al primo elemento di una lista dobbiamo scrivere [0] e non [1].



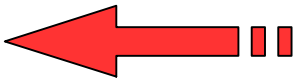
Esercitati a selezionare gli elementi di tutte le liste che abbiamo usato come esempio e prova a scrivere un ciclo WHILE per attraversarle e visualizzare gli elementi su video, come nell'esempio seguente, con la lista: `[3,6,9,12,15]`

1. Innanzitutto dobbiamo dare un nome alla lista:

```
tabellina = [3,6,9,12,15]
```

2. Quindi selezionare gli elementi:

```
>>>print tabellina[0]
3
.......
.......
.......
```



3. Ora scriviamo il ciclo per *attraversare* la lista:

```
i = 0
while i < 5:
      print tabellina[i]
      i = i + 1
```

Ora che hai una buona dimestichezza con le liste, specifichiamone meglio l'uso. Ad esempio:

Cosa succede con questa espressione?

```
dati_di_delpiero[1] = "Ale"
```

L'operatore porzione [ ] può comparire sia a destra che a sinistra del simbolo = di un'espressione. A destra indica l'elemento specificato, a sinistra cambia uno degli elementi della lista (nell'esempio all'elemento 1 viene assegnato il valore "Ale"). Quindi il secondo elemento della lista `dati_di_delpiero[1]` diventa: `"Ale"`.

A differenza delle stringhe, le liste sono "**mutabili**", cioè gli elementi della lista possono essere sostituiti da valori diversi da quelli iniziali. È quindi possibile cambiare il valore di un elemento di una lista con un'istruzione di assegnazione.

Se a `dati_di_delpiero` aggiungiamo un quarto elemento per indicare i gol fatti, scriveremo:

```
dati_di_delpiero = ["Alessandro","Del Piero", 32, 205]
```

Ogni volta che Del Piero segnerà un goal scriverò:

```
dati_di_delpiero[3] = dati_di_delpiero[3] + 1
```

e, ogni anno, il giorno del suo compleanno, il 9 novembre, scriverò

```
dati_di_delpiero[2] = dati_di_delpiero[2] + 1
```

Nelle pagine successive approfondiamo insieme a Martina l'argomento.

1 - Indici

Si accede agli elementi di una lista nel modo già visto per accedere ai caratteri di una stringa.

`print dati_di_delpiero[1]` Visualizza: il secondo elemento della lista, ossia `Alessandro`

Una qualsiasi espressione che produca un intero può essere un indice.

Ad esempio: `tabellina[5-1]` visualizza: 12

Prova a usare un numero decimale come indice e analizza il messaggio di errore che compare:

```
TypeError: sequence index must be integer
```

Prova a leggere o a modificare un elemento che non esiste. Ad esempio:

```
tabellina[7] = 2
```

L'interprete cerca di assegnare a `tabellina[7]` il valore 2.
Anche qui analizza attentamente il messaggio di errore:

```
IndexError: list assignment index out of range
```

Cosa succede se l'indice ha valore negativo? ad esempio: **`tabellina[-2]`**

Se un indice ha valore negativo il conteggio parte dalla fine della lista.
Per verificarlo prova a scrivere: `dati_di_delpiero[-3]`
Questo metodo di usare l'indice negativo si rivela molto comodo quando non si ricorda la lunghezza della lista. Sappiamo che se scriviamo [-1] accederemo all'ultimo elemento, [-2] al penultimo e così via.
Infine, come abbiamo visto all'inizio, una variabile di ciclo può essere usata come indice di lista. Esempio:

```
amici = ["Gianni", "Luigi", "Carlo"]
i = 0
while i <3:
  print amici[i]
  i = i +1
```

2 - Lista speciale

È una lista composta da tutti numeri interi consecutivi, talmente comune che Python fornisce un modo speciale per crearla. La funzione **`range`** è lo strumento usato per crearla. Infatti, scrivere `[0,1,2,3,4]` è equivalente a scrivere `range[5]`

Cioè, la funzione `range` crea una lista che parte da 0 e arriva fino al numero indicato dall'indice.

`range[10]` è equivalente a `[0,1,2,3,4,5,6,7,8,9]`

Con `range` si può scrivere: `range[1,5]` che è equivalente a `[1,2,3,4]`

La funzione `range` in questo caso legge dalla lista di interi gli elementi che partono dal primo indice incluso e arrivano all'ultimo indice escluso.

Con range possiamo scrivere anche espressioni del tipo:

`range[1,10,2]` che è equivalente a `[1,3,5,7,9]`

Il terzo indice si chiama "passo" e indica con quale intervallo leggere i numeri dalla lista di interi partendo da 1 e arrivando a 10 escluso. In questo caso legge il primo, il terzo, il quinto e cosi via fino a 10-1, ottenendo una lista di numeri dispari.

Che lista ottieni con questa funzione, range[2,10,2] ? [2, 4, 6, 8]

3 – Lunghezza di una lista

La funzione **len** applicata ad una lista produce il numero di elementi di una lista, come nelle stringhe.

```
>>>allievi=["luigi","marco","filippo","paola","gabriella","silvia"]

>>>i = 0
>>>while i < len(allievi):
>>>   print allievi[i]
>>>   i = i + 1

luigi marco filippo paola gabriella silvia
```

4 – Operazioni sulle liste

Come per le stringhe, l'operatore **+** concatena le liste:

```
>>>allievi_3A = ["luigi","marco","filippo","paola","gabriella","silvia"]
>>>allievi_4E = ["gabriele","alessandro","anna","michela","antonio"]
>>>allievi = allievi_3A + allievi_4E
>>>print allievi

luigi marco filippo paola gabriella silvia gabriele alessandro anna michela
antonio
```

L'operatore ***** ripete una lista un dato numero di volte.

```
>>>numeri * 4

3 6 9 12 15 3 6 9 12 15 3 6 9 12 15 3 6 9 12 15

>>>allievi_4E * 2

gabriele alessandro anna michela antonio gabriele alessandro anna michela
antonio
```

5 – Sezione di lista

Una sezione di lista è l'analogo di una sezione di stringa e ha la stessa logica di funzionamento. Infatti, come per le stringhe, si adotta l'operatore porzione **:**

```
>>>amici = ["Gianni", "Luigi", "Carlo")
>>>amici[0:3]
gianni luigi carlo
>>amici[1:3]
luigi carlo
>>>allievi_4E =
["gabriele","alessandro","anna","michela","antonio"]
>>>allievi_4E[1:3]
[alessandro,anna ]
>>>allievi[1:2]
[alessandro]
```

Ricordati: il primo indice incluso, il secondo indice escluso.

Con questo operatore possiamo eliminare uno o più elementi di una lista assegnando alla corrispondente sezione una stringa vuota.

```
>>>allievi_4E = ["gabriele","alessandro","anna","michela","antonio"]
>>>allievi_4E[1:3] = []
>>>print allievi_4E
gabriele michela antonio
```

Ma non è così pratico ed è facile sbagliare. Con la funzione **del** è molto più facile cancellare un elemento da una lista.

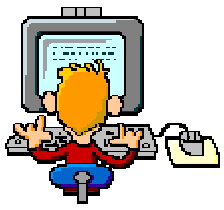
```
>>>allievi_4E = "gabriele","alessandro","anna","michela","antonio"]
>>> del allievi_4E[1:3]
>>>print allievi_4E
gabriele michela antonio
```

Vediamo un altro esempio:

```
>>>Squadre = ["Juventus", "Inter", "Milan", "Roma"]
>>>del Squadre[1]
>>>print Squadre
Juventus Milan Roma
```

Analogamente possiamo inserire uno o più elementi in una lista inserendoli in una sezione vuota nella posizione desiderata, come nella terza istruzione del programma seguente:

```
>>>allievi_4E = ["gabriele","alessandro","anna","michela","antonio"]
>>>print allievi[2:2]
[]
>>>allievi_4E[2:2] = ["sandra", "andrea"]
>>>print allievi_4E
gabriele alessandro sandra andrea anna michela antonio
```



Cosa ottengo se scrivo quanto segue?

```
>>>allievi_4E =
["gabriele","alessandro","anna","michela","antonio"]
>>>print allievi_4E[:4]
>>>print allievi_4E[3:]
>>>print allievi_4E[:]
```

Se non viene specificato il primo indice la porzione parte dall'inizio della stringa.
Senza il secondo indice la porzione finisce con il termine della stringa.
Se mancano entrambi gli indici si intende tutta la lista.

LISTE annidate

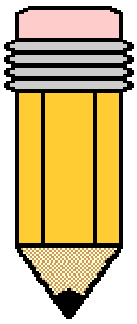
Un elemento di una lista può essere a sua volta una lista.

```
>>>tabellina = [3,6,9,12,15,[10,20,30]]
>>>print tabellina[5]
10 20 30
```

Il sesto elemento della lista `tabellina` è a sua volta una lista.

Nell'esempio la lista `allievi_4E` è diventata una lista di liste:

```
>>>allievi_4E =
[["bianchi","gabriele"],["verdi","alessandro"],["rossi","anna"],["neri","michela
"],["viola","antonio"]]
>>>print allievi_4E[2]
rossi anna
```



Cosa ottengo se scrivo `print allievi_4E[2][0]`?

Ottengo il risultato `[rossi]`.

Posso <u>estrarre un elemento da una lista annidata</u> con due metodi differenti.

Il primo è il seguente:

```
>>>allievo = allievi_4E[2]
>>> allievo[0]
rossi
```

Creo una lista che si chiama allievo e prendo il primo elemento di quella nuova lista.

Il secondo mi permette di scrivere direttamente:

```
>>>allievi_4E[2][0]
```

Questa espressione deve essere letta da sinistra verso destra: trova il terzo elemento (2) della lista `allievi_4E`

ed essendo questo elemento a sua volta una lista, ne estrae il primo elemento (0).

Ancora una domanda difficile. Qual è la lunghezza della lista seguente?

```
allievi_4E =
[["bianchi","gabriele"],["verdi","alessandro"],["rossi","anna"
],["neri","michela"],["viola","antonio"]]
```

Il ciclo FOR (ancora più semplice *attraversare* una lista)

Riprendiamo l'esempio:

```
allievi = ["luigi","marco","filippo","paola","gabriella","silvia"]
i = 0
while i < len(allievi):
      print allievi[i],
      i = i + 1

luigi marco filippo paola gabriella silvia
```

Questo ciclo può essere scritto anche in questo modo:

```
allievi = ["luigi","marco","filippo","paola","gabriella","silvia"]
for allievo in allievi:
      print "Ora stampo l'allievo: ", allievo
```

Possiamo leggere quasi letteralmente: "Fai quello che segue per ciascun allievo nella lista allievi, lavorando la prima volta sul primo elemento, la seconda sul secondo, e cosi via.

Stesso risultato, ma il programma è molto più conciso.

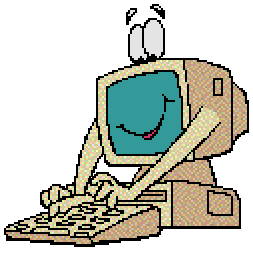
In termini generali devo scrivere:

`for <nome di variabile> in <nome di lista>:`

Definizione

L'istruzione **for** significa:

per ciascuna variabile della lista fai tutto quello che segue



In modo più formale la lista viene analizzata dal primo elemento sino all'ultimo. La prima volta:

1. alla variabile `allievo` assegna il valore `allievi[0]`
2. esegue i comandi che seguono i due punti
3. ritorna all'istruzione `for`

e prosegue fin quando trova un elemento nella lista.

Negli esempi che seguono cerchiamo di capire a cosa può servire il ciclo **`for.`**

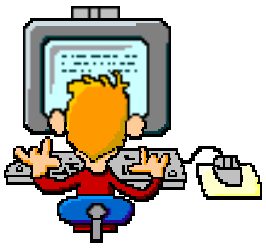
Stampiamo il quadrato dei primi 10 numeri interi.

```
for numero in range(11):
      print "il quadrato di ", numero, " è ", numero * numero
```

```
il quadrato di 0 è  0
il quadrato di 1 e` 1
il quadrato di 2 e` 4
il quadrato di 3 e` 9
il quadrato di 4 e` 16
il quadrato di 5 e` 25
il quadrato di 6 e` 36
il quadrato di 7 e` 49
il quadrato di 8 e` 64
il quadrato di 9 e` 81
il quadrato di 10 e` 100
```

Perchè ho scritto range (11)?

Ti ricordi come si fa a modificare il programma per ottenere un elenco in colonna?



Misuriamo ora la lunghezza di alcune parole:

```
vocabolario=['alba','banana','carro','dodici','giacca','finestra']
for parola in vocabolario:
     print parola, len(parola)
```

```
alba 4
banana 6
carro 5
dodici 6
giacca 5
finestra 8
```

Per uscire immediatamente da un ciclo for o while puoi usare l'istruzione break.

Vediamo un esempio semplice: un programma che stampa tutti i numeri positivi fino a 100 ma se viene inserito 0 si interrompe.

```
i = 1
while i < 100:
      i = input('inserisci un numero da 0 a 100 ')
      if i == 0:
          break
      print "il numero è ", i
      i = i + 1
print 'Finito'
```

Ricordati che se il ciclo avesse un else questo non verrebbe mai eseguito.

Esercitiamoci un po'

1. Scrivi un programma che analizzi le seguenti stringhe e per ognuna stampi le lettere una ad una: banana, cipolla, bambino, finestra, girotondo.

2. Scrivi un programma che sommi tutti gli elementi di una lista data.

3. Scrivi un programma che usi il concatenamento e un ciclo for per generare una lista di valori nei quali gli elementi appaiono in ordine alfabetico. Per esempio:

4. Scrivi un programma che stampi la "serie di Fibonacci". Cioè, la somma di due elementi definisce l'elemento successivo (1,2,3,5,8,13…..)

5. Scrivi un programma per trovare i numeri primi in una lista di numeri che va da 2 a 20.

6. Scrivi un programma che conta il numero di volte in cui la lettera 'a' compare in una stringa, usando un contatore.

7. .Scrivi un programma che esamini una lista di numeri e conti il numero di volte in cui uno di questi numeri cade in un determinato intervallo, ossia è superiore a un valore indicato ed è inferiore ad un altro valore assegnato. Non è molto diverso dal programma precedente che conta le lettere.

8. Prova a incapsulare il codice del programma in una funzione chiamata "intervallo" che ha come parametri la lista da controllare ed i valori Inferiore e Superiore.

9. Prova a svolgere l'esercizio 4 usando il `ciclo FOR`.

E' arrivato il momento di…	**Fare le cose "con stile"**
	Prima di passare allo step successivo, dedicato interamente alla grafica, vediamo insieme qualche accorgimento utile per facilitarci nella scrittura di programmi più complessi, con molte righe di codice.
	L'osservazione di Superteacher può sembrare un po' strana, ma i programmi che eseguono operazioni complesse hanno tantissime righe di codice e a volte può essere veramente difficile ricordarsi la logica con cui sono state scritte. I programmatori hanno una regola fondamentale quando scrivono i loro programmi: la coerenza. E' molto importante che il codice del programma sia scritto in modo da rispettare alcune regole che il programmatore decide all'inizio cosi poi gli sara' facile leggerlo in seguito quando dovra' correggerlo o cambiarlo. Cominciamo con le ...raccomandazioni.

Indentazioni
Quando inizi a scrivere decidi quanti spazi lasciare nelle indentazioni delle istruzioni e lascia sempre quelli.
Ricordati di usare sempre e solo la sbarra spaziatrice.
Se utilizzassi il tasto di tabulazione potresti avere dei problemi; infatti, i computer seguono formattazioni diverse per il TAB.

Nell'esempio seguente lasciamo sempre 4 spazi:

```
def interrogazione(domanda,risposta_esatta):
    risposta_allievo = raw_input(domanda)
        if risposta_esatta == risposta_allievo:
            print "La risposta e' esatta"
            print "Bravissimo"
        else:
            print "Risposta errata"
            print "La risposta esatta e' ", risposta_esatta
            print ". Studia di piu'!"
```

Separazioni
E' una buona regola separare le definizioni di funzioni lasciando due righe bianche.
Puoi usare le righe bianche anche all'interno del programma e delle funzioni, moderatamente, per indicare le sezioni logiche, cioe' le parti di programma che risolvono problemi diversi.

Spazi bianchi in Espressioni e Istruzioni sono da evitare

- immediatamente entro parentesi tonde, quadre o graffe
- subito prima di virgola, punto e virgola, o due punti
- subito prima della parentesi aperta che inizia la lista degli argomenti in una chiamata
- subito prima della quadra aperta che inizia un indicizzazione
- più di uno spazio attorno a un assegnamento o operatore per allinearlo con altri, tipo:
- x = 1
- y = 2

Lunghezza massima delle righe

Molto spesso sui PC di uso comune le righe sono di 80 caratteri.
Se scrivi linee più lunghe, queste verranno "ripiegate" e risulteranno brutte e soprattutto difficili da leggere.
Cerca quindi di limitare ogni linea a un massimo di 79 caratteri (una linea di esattamente 80 caratteri rischierebbe infatti di venire comunque ripiegata).
Comunque, se proprio non ci riesci, il modo migliore di dividere linee lunghe è come vedi nell'esempio seguente:

```
print altezza_media_femmine
altezza_media_classe =((altezza_media_femmine*femmine)+
                       (altezza_media_maschi*maschi))/alunni
print altezza_media_classe
```

Ricordati di indentare bene la riga che continua a capo e sempre nello stesso modo.

E ora ...i commenti

I "commenti" sono parole, frasi o proprio testi brevi che servono a spiegare bene le operazioni che il codice esegue. I commenti non sono esaminati dall'interprete che quando li incontra passa oltre.

E' molto importante aggiungere nei programmi alcune osservazioni che spiegano le operazioni che vengono fatte perchè il codice risulta molto piu' comprensibile anche allo stesso autore, che dopo un po' di mesi puo' anche non ricordarsi più perchè aveva scritto le istruzioni in quel modo.
Attenzione a scrivere bene i commenti: inserire dei commenti che contraddicono il codice è molto peggio che non avere commenti!

Anche per i commenti valgono le raccomandazioni precedenti.

I commenti possono essere "a blocco" e allora si riferiscono al codice che li segue.

- Se il codice è indentato anche i commenti a blocco vanno indentati esattamente come quel codice.
- Ogni riga di un commento a blocco inizia con # .
- Si possono separare i paragrafi entro un commento a blocco con una riga vuota che contiene soltanto un #.
- Il commento puo' essere lungo quanto si vuole, non ci sono limiti sulla lunghezza.
- È meglio circondare i commenti a blocco con una riga bianca sopra e una sotto per renderli più visibili.

I commenti possono essere anche "in linea" ovvero sulla stessa riga dell'istruzione.

Questi commenti non dovrebbero dire delle cose ovvie, ad esempio:

```
x = x+1    # Incremento x
```

è un commento ovvio e quindi inutile e soprattutto distrae chi legge il programma!!!

I commenti in linea dovrebbero essere separati da almeno due spazi dall'istruzione e devono iniziare con un # seguito da un singolo spazio. Ricordati di non usare spesso i commenti in linea perchè l'effetto finale è di un programma poco leggibile, meglio usare alcuni blocchi anche lunghi ma messi nei punti cruciali del programma.

Vediamo un esempio ripreso dallo STEP9 prima senza i commenti e poi con alcuni commenti.

```
def interrogazione(domanda,risposta_esatta):

      risposta_allievo = raw_input(domanda)
       if risposta_esatta == risposta_allievo:
            print "La risposta e' esatta"
            print "Bravissimo"
      else:
             print "Risposta errata"
            print "La risposta esatta e' ", risposta_esatta
            print ". Studia di piu'!"
```

```
domanda1 = "In che anno è caduto l'impero romano d'occidente? "
risposta_esatta1 = "476"
interrogazione(domanda1, risposta_esatta1)
domanda2 = "Chi e' stato il primo presidente degli Stati Uniti? "
risposta_esatta2 = "Washington"
interrogazione(domanda2, risposta_esatta2)
domanda3 = "In che anno è terminata la prima guerra mondiale? "
risposta_esatta3 = "1918"
interrogazione(domanda3, risposta_esatta3)
```

Ora rivediamo lo stesso esercizio con i commenti:

```
# Programma "interrogazione di storia"
# a ogni domanda chiama la funzione
# interrogazione

# funzione che controlla la risposta dell'allievo, confrontandola con la
# risposta esatta
def interrogazione(domanda,risposta_esatta):
    risposta_allievo = raw_input(domanda)
    if risposta_esatta == risposta_allievo:
        print "La risposta e' esatta"
        print "Bravissimo"
    else:
        print "Risposta errata"
        print "La risposta esatta e' ", risposta_esatta
        print ". Studia di piu'!"

# prima domanda
domanda1 = "In che anno e` caduto l'impero romano d'occidente? "
risposta_esatta1 = "476"
interrogazione(domanda1, risposta_esatta1)     # prima chiamata di funzione

# seconda domanda
domanda2 = "Chi e' stato il primo presidente degli Stati Uniti? "
risposta_esatta2 = "Washington"
interrogazione(domanda2, risposta_esatta2)..... # seconda chiamata di funzione

# terza domanda
domanda3 = "In che anno e` finita la prima guerra mondiale? "
risposta_esatta3 = "1918"
interrogazione(domanda3, risposta_esatta3      # terza chiamata di funzione
```

Scoprire gli errori (debugging)

Può capitare che un programma funzioni correttamente ma che il risultato sia diverso da quello che ti aspettavi oppure che un programma non faccia nulla o ancora che non si fermi più quando va in esecuzione; insomma spesso i programmi contengono degli errori che non è sempre facile scoprire. In inglese gli errori si chiamano "bug", insetti che si infiltrano tra le righe del programma, ed è per questo che la tecnica di scoprire gli errori si chiama "debugging".
Il primo passo per scoprire un errore è capire con che tipo di errore hai a che fare.

Un tipo di errore molto comune è l'errore di sintassi. Si verifica quando sbagli a scrivere un istruzione (ad esempio mancano i due punti alla fine di una istruzione if) e compaiono i messaggi: `SyntaxError`, `invalid syntax`. Python può eseguire un programma solo se il programma è sintatticamente corretto, altrimenti il programma non è eseguito e l'interprete trasmette un messaggio d'errore.

Eccoti alcune raccomandazioni per evitare i più comuni errori di sintassi.

- controlla di non usare parole riservate per dare i nomi alle variabili
- controlla i due punti alla fine delle istruzioni dove sono necessari
- controlla l'indentazione
- controlla che le parentesi siano appaiate correttamente (una parentesi non chiusa costringe Python a cercare la fine nelle righe successive...)
- controlla che non ci sia un "=" anzichè un "=="

Un altro tipo di errore è l'errore in esecuzione (runtime error).
Cioè il programma non fa nulla oppure si blocca, oppure continua all'infinito, oppure ancora stai cercando di dividere un numero per 0. In tutti questi casi la sintassi è corretta ma qualcosa va storto. E qui il controllo diventa più difficile. Vediamo i casi possibili:

- *Il programma non fa nulla*: magari hai delle funzioni ma ti sei dimenticato di mettere nel programma la chiamata di funzione. Controlla bene che ci siano tutte le chiamate alle funzioni.
- *Il programma si blocca o continua all'infinito*. Controlla i cicli, forse il programma non esce mai da un ciclo.

Un metodo semplice e utile di scoprire gli errori nei cicli è quello di stampare i risultati di ogni operazione usando l'istruzione print immediatamente prima di entrare nel ciclo e una subito dopo l'uscita. Ad esempio: `print "entrata nel ciclo"` e `print "uscita dal ciclo"`
Cosi se eseguendo il programma vedi il primo messaggio ma non il secondo sei sicuro che il ciclo è infinito!

Inserire le istruzioni di "print" per scoprire gli errori in esecuzione è un metodo ottimo e può essere usato come nell'esempio di prima oppure per:

- stampare i valori delle variabili prima e dopo una condizione (if, while) per verificare che cambino.
- stampare i parametri di una funzione
- aggiungere le istruzioni print lungo il flusso del programma ti permette di seguire la traccia di cosa sta facendo Python ed eventualmente dove sbaglia

Attenzione a non farti sommergere dalle "print". Devi metterne poche e quando non servono più devi ricordarti di toglierle.
Spesso vedere stampati i dati ti aiuta a scoprire indizi utili a trovare la soluzione agli errori.

Ma non disperarti se sei davvero bloccato! Anche ai migliori succede. In questo caso la soluzione migliore è smetterla di guardare il codice, andarsi a mangiare un buon gelato e poi tornare con la pancia piena e gli occhi "freschi" . A volte ci vuole molto tempo per trovare un errore ed è meglio fare la ricerca con la mente riposata.
I migliori hacker sostengono che i posti sicuri per trovare i bug sono i treni, le docce, il letto....non ho mai capito se si riferiscono agli errori o ai veri bug biologici!!
Il lavoro di debugging a volte può essere frustrante ma è anche una delle parti più creative, stimolanti ed interessanti della programmazione.
In un certo senso per fare il debug devi diventare un piccolo detective: sei messo di fronte agli indizi e devi ricostruire i processi e gli eventi che hanno portato ai risultati che hai ottenuto.
Come diceva Sherlock Holmes "Quando hai eliminato l'impossibile ciò che rimane, per quanto improbabile, deve essere la verità"

(A.Conan Doyle, *Il segno dei quattro*)

Vediamo un esempio di debugging così capirai più facilmente l'uso dell'istruzione print come strumento di debugging. Riprendiamo l'esercizio n°8 dello step 3.
La soluzione corretta, con le "print" per il debugging e i commenti è:

```
maschi = 8
femmine =10
alunni = maschi + femmine
print  alunni
M = 1.55                              #altezza di Mario
FML = 1.60*3                          #altezza di Fabio, Matteo, Luca
AAGG = 1.50*4                         #altezza di Aldo Andrea Giovanni Giacomo
altezza_media_maschi = (M + FML+ AAGG)/  maschi
print altezza_media_maschi
MGEF = 1.55*4                         #altezza di Marta Giovanna Elisabetta Francesca
SCS = 1.50*3                          #altezza di Stefania Chiara Simonetta
A = 1.68                              #altezza di Arianna
DD = 1.63*2                           #altezza di Daria, Domitilla
altezza_media_femmine =  (MGEF+ SCS + A + DD)/femmine
print altezza_media_femmine
altezza_media_classe =((altezza_media_femmine*femmine)+(altezza_media_maschi*maschi))/
                      alunni
print altezza_media_classe
```

Facciamo un errore:

```
maschi = 8
femmine =10
alunni = maschi + femmine
print  alunni
M = 1.55
FML = 1.60*3
AAGG = 1.50*4
altezza_media_maschi = (M + FML+ AAGG)/  maschi
print altezza_media_maschi

MGEF = 1.55*4
SCS = 1.50*3
A = 1.68
```

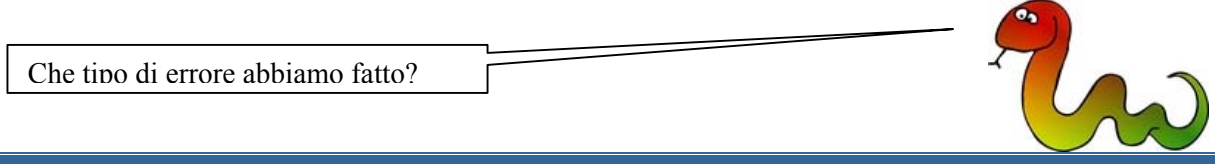
DD = 1.63*5 (errore inserito)

```
altezza_media_femmine =  (MGEF+ SCS + A + DD)/femmine
print altezza_media_femmine
altezza_media_classe =((altezza_media_femmine*femmine)+(altezza_media_maschi*mas
                                                                        chi))/alunni
print altezza_media_classe
```

Dopo aver verificato che l'altezza media delle femmine ha un valore che è sbagliato, proviamo ad inserire alcune istruzioni di stampa (print) per controllare il valore delle variabili che utilizziamo per calcolare il valore - altezza media delle femmine. Così si vedrà che il valore della variabile DD è sbagliato e sarà facile correggere l'errore:

```
maschi = 8
femmine =10
alunni = maschi + femmine
print  alunni
M = 1.55
FML = 1.60*3
AAGG = 1.50*4
altezza_media_maschi = (M + FML+ AAGG)/  maschi
print altezza_media_maschi

MGEF = 1.55*4
SCS = 1.50*3
A = 1.68
DD = 1.63*5
print AA
print SCS
print A
print DD
altezza_media_femmine =  (MGEF+ SCS + A + DD)/femmine
print altezza_media_femmine

altezza_media_classe
=((altezza_media_femmine*femmine)+(altezza_media_maschi*maschi))/alunni
print altezza_media_classe
```

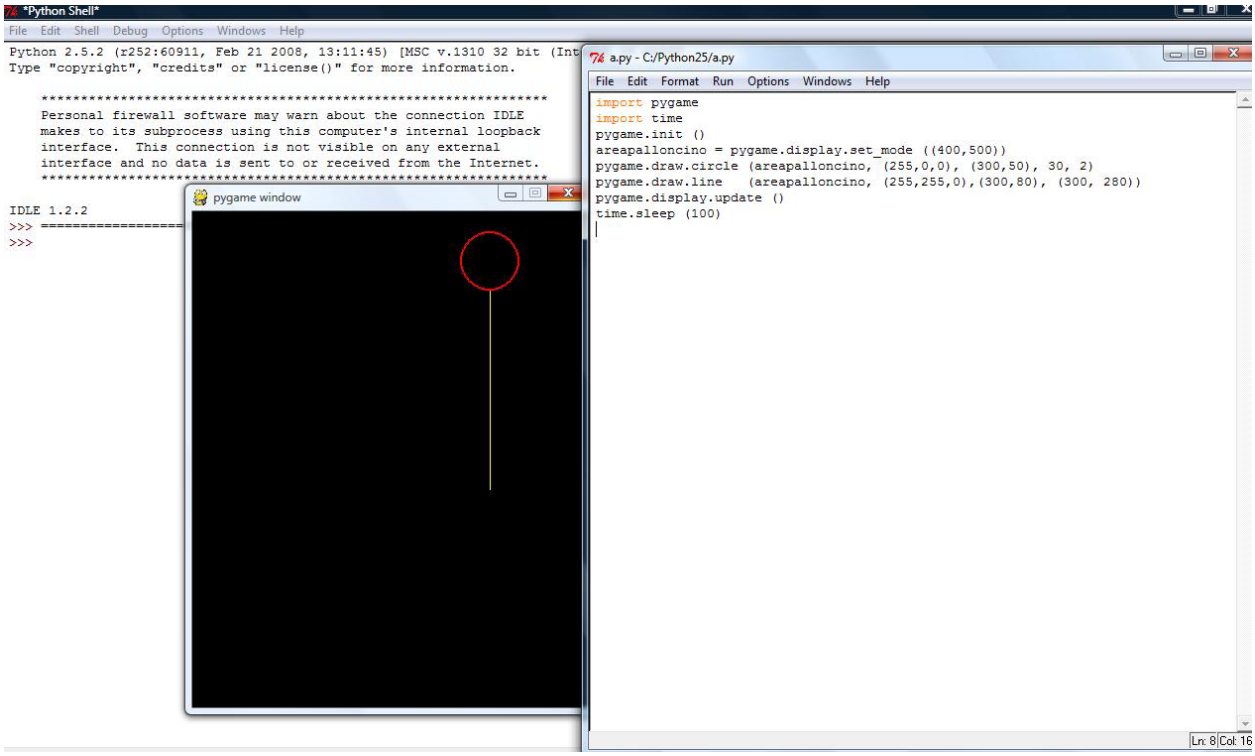
Inserendo queste quattro istruzioni di print potremo accorgerci dell'errore che abbiamo fatto.



STEP 12

In questo step impareremo:

* come si formano le immagini su un monitor
* cos'è una **libreria** e come si importa
* come si realizza un semplice disegno
* come si crea un'**animazione**

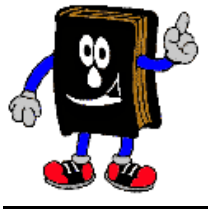


STEP 12

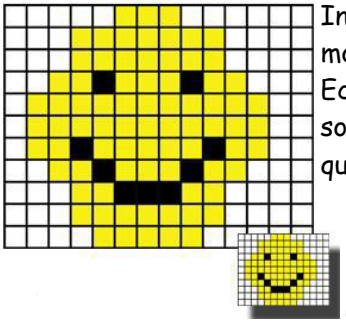
Ovvero, un passo dopo l'altro impariamo a programmare

PRODURRE UN DISEGNO SU VIDEO

Ancora un passo, ma molto importante, dedicato alla grafica!
In questo step infatti imparerai a scrivere programmi in Python che ti consentiranno di produrre semplici disegni e animazioni sul video.
Per comprendere bene come funziona il video di un personal computer e come scrivere poi le istruzioni che ti consentiranno di disegnare sul video, devi prima ascoltare il racconto che segue, un po' lungo e noioso ma utile a capire come è fatto il video di un PC.



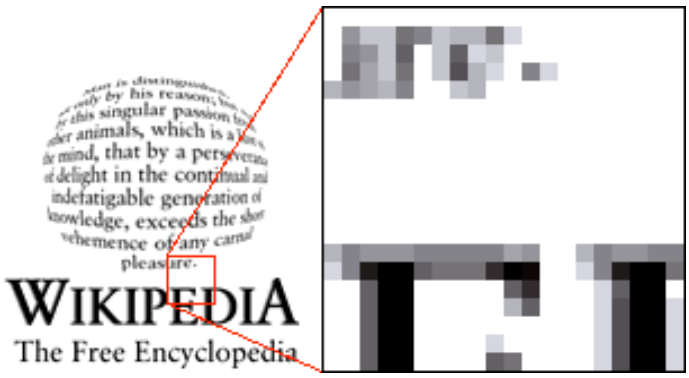
Cominciamo con la codifica delle immagini.
Osserva attentamente le immagini della televisione o del video del PC e vedrai che sono formate da una serie di puntini uno vicino all'altro: sono in realtà dei quadratini molto piccoli.



Ingrandendo l'immagine i quadratini si vedranno molto bene.
Ecco cosa succede all'immagine di un volto che sorride se la ingrandiamo tantissime volte. I quadratini da cui è formata sono evidentissimi.

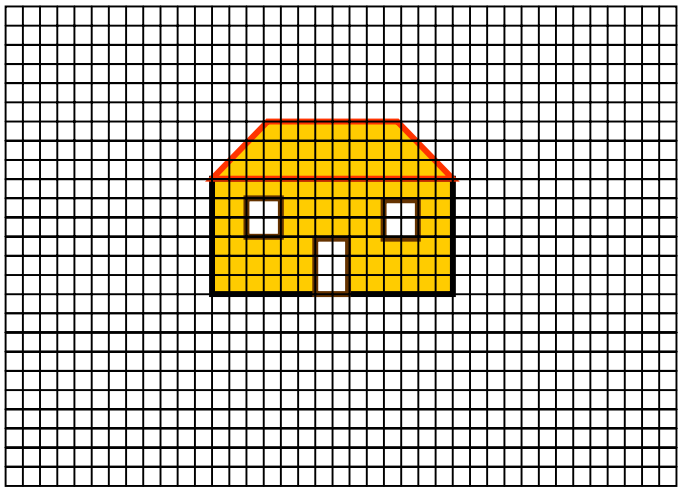
(originale)

E se ingrandiamo un pezzo della scritta Wikipedia ?



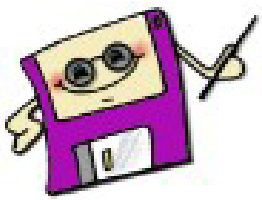
Nella TV i puntini sono circa 500 su ogni riga e ci sono circa 300 righe sullo schermo, quindi in tutto ci sono 150.000 puntini. I puntini sono di materiale fosforescente. Questi puntini si chiamano **"pixel" (picture element),** o "punto di un'immagine", e sia sul PC che sulla TV possono essere in bianco e nero o colorati. Adesso comincia ad analizzare i puntini in bianco e nero perchè quelli colorati sono un poco più complicati. Inoltre, supporrai quasi sempre che un puntino sia bianco oppure nero. Nella grafica più raffinata un puntino può essere non soltanto bianco e nero ma anche grigio più o meno scuro.

Prendi l'immagine di una casetta e pensa di rappresentarla sul monitor del calcolatore. Sovrapponi poi al disegno una griglia di quadratini ognuno dei quali rappresenta un "pixel". Scrivi **0** se nel quadratino c'è più bianco che nero, oppure **1** se c'è più nero (o colore).
Ora prova a farlo con due griglie diverse: la prima ha i quadratini di 1 cm e la seconda di 0,5 cm.
Prova a riempire di 1 e 0 le due griglie e conta quanti puntini ci sono nel primo esempio e quanti nel secondo.



Verificherai che se i quadratini sono più piccoli il disegno verrà meglio; in linguaggio tecnico si dice che è più definito.

La definizione di un'immagine si chiama *risoluzione* ed è il numero di quadratini che ci sono in una certa area.
Un'immagine ha una risoluzione più alta quando i puntini per area sono tanti, ha una risoluzione bassa se i puntini sono pochi.
La risoluzione di uno schermo si misura moltiplicando il numero di pixel di ogni riga per il numero di righe.
Ad esempio, la risoluzione dello schermo televisivo, che è 500 x 300, corrisponde a 300 righe ciascuna delle quali è composta da 500 pixel.
La risoluzione dello schermo TV è molto bassa, sui video dei PC usati per applicazioni di grafica si arriva a risoluzioni di 4096 x 3300 pixel.



Come hai appena visto, un'immagine viene rappresentata con dei puntini e la perfezione del disegno dipende dal numero dei puntini e quindi dai bit. Servono tantissimi bit per rappresentare bene un'immagine.
L'enorme quantità di bit necessaria per rappresentare bene un'immagine è il motivo per cui ci vuole tanto tempo per "scaricare" un'immagine da Internet. Poichè le immagini sul video sono composte da tanti puntini uno vicino all'altro, o se preferisci da tanti bit, anche un'immagine può essere rappresentata sul calcolatore come una lunga fila di bit. Si dice che i pixel possono essere accesi (quando il quadratino è nero o colorato) o spenti (quando il quadratino è bianco) e l'immagine si può rappresentare semplicemente dicendo che un pixel e' acceso e l'altro e' spento.
Questo si può dire con un codice, ad esempio 0 = spento, 1 = acceso.

Per gli schermi a colori il discorso è un po' più complesso, per ora quindi cerca di comprendere come funziona lo schermo in bianco e nero. Inoltre supponi che il puntino possa essere soltanto bianco o nero, ma non grigio.
L'immagine della casetta si è trasformata in una lunga fila di 1 e 0 (di bit). Se pensi questi bit come dei puntini fosforescenti, puoi dire che = 1 significa puntino acceso, = 0 significa puntino spento (in parole povere luce o non luce su un certo puntino dell'immagine).
Ricorda che con 150.000 bit puoi descrivere un'immagine in bianco e nero su uno schermo televisivo, ma ne hai bisogno di 500.000 per descrivere la stessa immagine sul PC. Questo perchè il video del PC ha molte più righe e molte più colonne, anzi sul PC puoi decidere tu quante righe e colonne usare. Partendo da 800 colonne x 600 righe, puoi arrivare fino ai più potenti video che hanno 1280 colonne x 1024 righe.
Un'**immagine digitale** e' composta da una serie di bit (0 e 1) che rappresentano il valore dei pixel corrispondenti.
L'immagine nel calcolatore sarà una lunga sequenza di 0 e 1, che saranno usati proprio come comandi per decidere se accendere o spegnere i pixel da una apposita "memoria" del PC; infatti il video del PC ha una memoria che si chiama **"memoria video"**, dove sono scritte le sequenze di 0 e 1 che comandano la luce sui pixel.
Quindi la memoria video, in cui è descritta l'immagine come una sequenza di 0 e 1, trasmette al video i comandi da eseguire per rappresentare l'immagine.
Quando facciamo un disegno sul PC, questo viene disegnato a puntini e la perfezione del disegno dipende dal numero dei puntini. *Ti ricordi cos'è la risoluzione?* (…se usi dei puntini grossi il disegno viene male e non si riconosce più, se usi dei puntini più piccoli il disegno e' più definito).
Questa operazione si chiama anche **"discretizzazione"** di un'immagine: l'immagine può cambiare la sua intensità, cioè può essere semplificata, approssimata o complicata. La discretizzazione di un'immagine è la "numerizzazione" o "digitalizzazione" dell'immagine, cioè la trasformazione dell'immagine in bit (0 e 1). Naturalmente tanto più numerosi sono i pixel tanto migliore sarà la qualità dell'immagine.
"Immagine digitale", "TV digitale", "macchina fotografica digitale" ecc., con queste espressioni indichiamo tutti gli apparati le cui immagini sono rappresentate con dei numeri (dei puntini).

VOCABOLARIO ELETTRONICO

PIXEL = picture element = punto di un'immagine
RISOLUZIONE = numero di pixel per unità di lunghezza, si misura in DPI (dot X inch, punti per pollice).
IMMAGINE DIGITALE = immagine rappresentata con 0 e 1 (puntini o pixel)
MEMORIA VIDEO = memoria apposita del PC per il comando del video
DISCRETIZZARE = trasformare un'immagine in bit (0 e 1), si può dire anche "numerizzare" o "digitalizzare".

In conclusione, potresti produrre un disegno sul video scrivendo un programma fatto da molte istruzioni che precisino i punti dello schermo che vuoi illuminare uno alla volta. Ma occorrerebbero troppe istruzioni e molto tempo per produrre disegni anche molto semplici.
Fortunatamente altri programmatori prima di te hanno scritto delle funzioni che ti consentono di disegnare parti di un disegno con uno sforzo molto piccolo.
Queste funzioni sono state raccolte in una **"libreria"**, che è appunto un insieme di funzioni che il programma principale potrà richiamare quando ne ha bisogno.
La libreria che useremo per disegnare è:

In Python le raccolte di funzioni si chiamano "librerie" e non sono niente altro che programmi già scritti e funzionanti pronti per l'uso.
Non importa se ora non ti è del tutto chiaro che cosa sono le librerie.
Usandole capirai meglio.
Sai rispondermi se ti chiedo come è nato il nome Pygame?

Il nome di questa libreria è stato generato fondendo le parole "python" e "game", per ricordarci che essa potrà essere utilizzata per programmare anche videogame.

Il primo disegno!

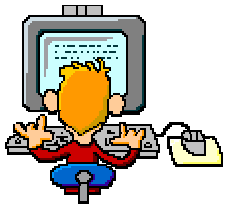
Scriviamo insieme un primo semplice programma che servirà a disegnare un palloncino.
Il programma inizierà con un'istruzione un po' strana:

```
import pygame
```

che significa:
"Caro Python, ho bisogno della libreria Pygame. Per favore mettimi a disposizione ("`import`") <u>tutte</u> le funzioni di questa libreria."
Ricordati di dire sempre a Python in quale libreria dovrà cercare le funzioni che ti servono, altrimenti si lamenterà di non conoscerne il nome!
Successivamente dobbiamo scrivere:

```
import time
```

che è l'ordine impartito a Python di mettere a disposizione una seconda libreria, chiamata "time" (tempo). Come vedrai questa libreria contiene funzioni che consentono di far comparire un disegno sul video per un certo numero di secondi, in modo da cambiare disegno a intervalli ben definiti di tempo, come è necessario se vuoi disegnare un oggetto che si sposta sul video.



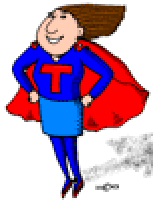
Quindi scriviamo:

pygame.init ()

che è la richiesta di attivare la funzione "init()" della libreria "pygame". Questa funzione è equivalente all'ordine che dà la professoressa quando dice ad un allievo *"Pulisci la lavagna che dobbiamo fare un disegno".* Infatti, il video sarà pulito per poter fare il disegno.
A questo punto sei in grado di disegnare, ma come prima operazione devi dare un nome all'area del video che conterrà il nuovo disegno e decidere quanto sarà grande questa area. Per questo dovrai scrivere, ad esempio:
areapalloncino = pygame.display.set_mode ((400,500))
Cosa significano quei numeri tra le parentesi tonde?
Sono le dimensioni dell'area dove disegneremo.
L'istruzione servirà a chiamare la funzione "set_mode" della libreria "display" di Pygame, e ci consentirà di:

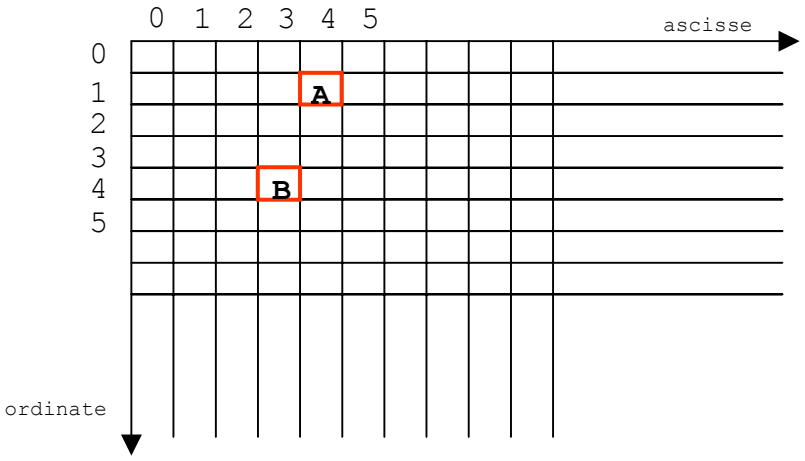
1. dare un nome ("areapalloncino") all'area del video dove disegneremo;
2. decidere le dimensioni di quest'area, che sarà un rettangolo largo 400 pixel (o quadratini") e alto 500 pixel.

In pratica è come prendere un foglio da disegno a quadretti e poi tagliarlo con le forbici della dimensione adatta al tuo disegno. Quel foglietto sarà chiamato "area palloncino".

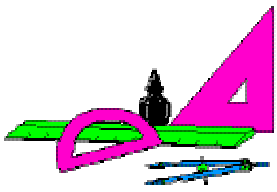


Adesso sei in grado di disegnare il tuo palloncino, ma per continuare è importante sapere cosa sono le "coordinate". Se ancora non lo sai, chiedi alla tua insegnante di matematica di spiegarti cosa sono le coordinate.
Attenzione però, le coordinate del video sono un po' strane!!
Abbiamo detto che i pixel del video sono individuati dalle coordinate (riga e colonna) ma attenzione al fatto che *sul video il punto 0,0 è in alto a sinistra, e quindi si conta a partire dall'alto verso il basso*.

Cosi nella figura che vediamo,



il pixel A ha "ascissa" 4, perchè sta nella colonna 4 e "ordinata" 1, perchè sta nella riga 1 e la sua posizione si indica con A(4,1).
Invece, il pixel B avrà ascissa uguale a 3 e ordinata uguale a 4, e quindi B(3,4).



...continuiamo a disegnare..

Il nostro programma ha già quattro istruzioni. (Un po' strane per ora).

```
import pygame
import time
pygame.init()
areapalloncino = pygame.display.set_mode((400,500))
```

(e hai solo preparato il foglio da disegno!! Immagina se avessi dovuto disegnare un foglio a quadretti composto da 400 colonne e 500 righe dando un comando per ogni pixel!!!).

Ora disegniamo il palloncino. Useremo la funzione: pygame.draw.circle

Chi sa un po' l'inglese avrà già capito che questa funzione serve a disegnare ("draw") un cerchio ("circle").

L'istruzione che dovrai scrivere non è proprio facilissima:

```
pygame.draw.circle(areapalloncino, (255,0,0), (300,50), 30, 2)
```

1 2 3 4 5

Ora con un po' di pazienza cerchiamo di capire come funziona questa istruzione.

1 **areapalloncino:** questo è il nome dell'area del disegno che conterrà il palloncino.

2 **(255, 0, 0)** specifica il colore del palloncino.
Più avanti imparerai il codice universale dei colori usato dai computer, per ora basta sapere che (255, 0, 0) indica il ROSSO.

3 **(300, 50)** sono le coordinate del centro del cerchio. In pratica, il centro del cerchio starà sulla 300-esima colonna e sulla 50-esima riga (partendo dall'alto!).

4 **30** è il raggio del cerchio che è 30 quadratini (pixel).

5 **2** è lo spessore della linea usata per disegnare il cerchio. Lo spessore è indicato in pixel, quindi nel nostro caso la linea che disegna il cerchio sarà spessa come 2 quadratini.
(è facile! come quando scegliamo con cosa disegnare: una matita, un pastello, un pennarello....)

*E' importante sapere che se invece di **2** si scrivesse **0**,*
oppure, più semplicemente, non si scrivesse nulla dopo la misura del raggio,
tutto il cerchio sarebbe colorato e non soltanto la linea di contorno

Due ultime istruzioni devono essere scritte per chiudere il programma:
pygame.display.update() e time.sleep(10)
La prima di queste due istruzioni: `pygame.display.update()`
è l'ordine di aggiornare il disegno (update in inglese significa aggiornare) sulla base delle istruzioni precedenti. Se, ad esempio, nelle due istruzioni precedenti, del tipo:

```
pygame.draw.circle(areapalloncino, (255,0,0), (300,50),30, 2)
```

abbiamo definito due cerchi, questi saranno disegnati insieme solo dopo l'esecuzione dell'istruzione update.

L'istruzione time.sleep() è l'ordine di "dormire" per un certo periodo di tempo espresso in secondi e frazioni di secondo prima di riprendere l'esecuzione del programma. Vedremo che questa istruzione, che ora non serve a nulla, sarà invece utile per realizzare disegni animati. Ecco il nostro programma completo:

```
import pygame
import time
pygame.init()
areapalloncino = pygame.display.set_mode((400,500))
pygame.draw.circle(areapalloncino,(255,0,0),(300,50),30,2)
pygame.display.update()
time.sleep(10)
```

Esercitati a disegnare, magari giocando un po' con la dimensione e la posizione del palloncino. Cambia le coordinate del centro e la dimensione del raggio. Ad esempio:
(50,300) e 10
(250,250) e 50
e così via...

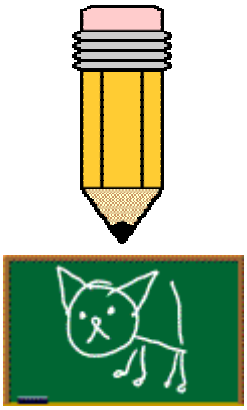
Adesso giochiamo con i colori

```
pygame.draw.circle(areapalloncino,(255,0,0),(300,50),30, 2)
```

Hai visto che nell'istruzione riportata sopra la parte (255, 0, 0) indica il codice del colore ROSSO. In questo modo, con i tre numeri scritti fra le due parentesi puoi indicare moltissimi colori diversi.
Questi tre numeri indicano, nell'ordine, le componenti dei tre colori fondamentali – rosso, verde, blu – (in inglese: R.G.B., come Red, Green, Blue). La scala dei valori dei colori va da 0 a 255. Quindi (255,0,0) è un palloncino di colore rosso (il rosso più intenso che esiste tra i tuoi colori), mentre se scrivo (0,50,255) il colore del palloncino sarà blu intenso con una leggera sfumatura di verde.
Prova a disegnare il palloncino cambiando i colori, ad esempio sostituendo (255, 0, 0) con:
(200,50,50)
(50,200,50)
(50,50,50)
e ancora
(0,255,0)
(0,0,255)

Puoi disegnare diversi palloncini, di colori diversi, cambiando i valori delle tre variabili R, G, B, cioè cambiando il codice del colore, come nella variante del nostro programmino:

```
import pygame
import time
R = 100
G = 255
B = 20
pygame.init ()
areapalloncino = pygame.display.set_mode((400,500))
pygame.draw.circle(areapalloncino,(R,G,B),(300,50),30,0)
pygame.display.update()
time.sleep(10)
```



Tutto avviene come se avessi tre matite colorate, una rossa, una verde e una blu, e decidessi di mescolare i colori disegnando un palloncino, ad esempio, con poco rosso, poco verde e tanto blu (50,50,200).
N. B. che nella variante del programmino ho scritto 0 al fondo dell'istruzione pygame.draw.circle(...) e quindi il palloncino sarà colorato per intero, non solo sul bordo.
Per verificare se sei diventato abbastanza bravo, disegna un palloncino giallo di grandi dimensioni e uno viola piccolo piccolo.
Prova anche a disegnare il palloncino di codice (0,0,0), che cosa vedrai?
E se invece il codice sarà (255,255,255), come sarà il palloncino?
Nel primo caso non vedrai nulla perchè il palloncino sarà nero e lo sfondo dell'area su cui disegni è anch'esso nero.
Nel secondo caso otterrai un palloncino bianco, perchè il bianco si trova appunto mescolando in parti uguali i tre colori fondamentali.

Ora esaminiamo una nuova istruzione che ci permetterà di disegnare le linee.

```
import pygame
import time
pygame init()
areasegmenti = pygame.display.set_mode((400, 500))
pygame.draw.line(areasegmenti,(255,0,0),(50,200), 250,300))
pygame.display.update()
time.sleep(10)
```

L'istruzione scritta in rosso nel programma:

```
pygame.draw.line(areasegmenti,(255,0,0),(50,200), 250,300))
```

1 2 3 4 5

1 è il nome della nuova istruzione che serve a disegnare un segmento

2 è il nome dell'area dove disegnare il segmento

3 è il codice del colore. In questo caso il segmento sarà rosso, senza alcuna componente verde o blu.

4 sono le due coordinate di un estremo del segmento, che quindi partirà dal quadratino che sta nella

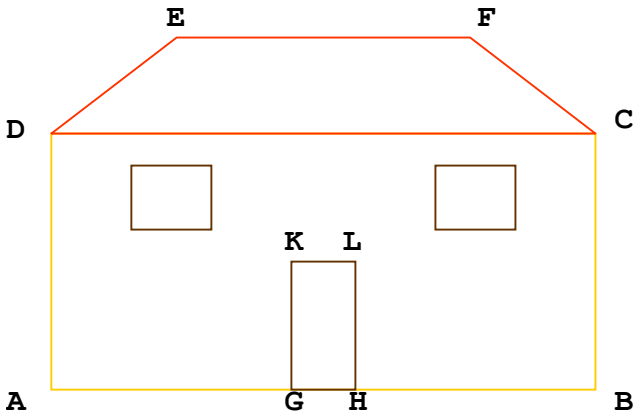
colonna 50 e nella riga 200

5 sono le due coordinate dell'altro estremo del segmento, che quindi finirà nel quadratino che sta nella

colonna 250 e nella riga 200.

A questo punto potrai disegnare tutti i segmenti che vorrai, in tutte le direzioni e... anche case, scatole, tavoli e decine di altri oggetti. Prova.

Con le istruzioni che abbiamo visto siamo già in grado di disegnare una casetta come questa. Non è bellissima ma è un disegno!

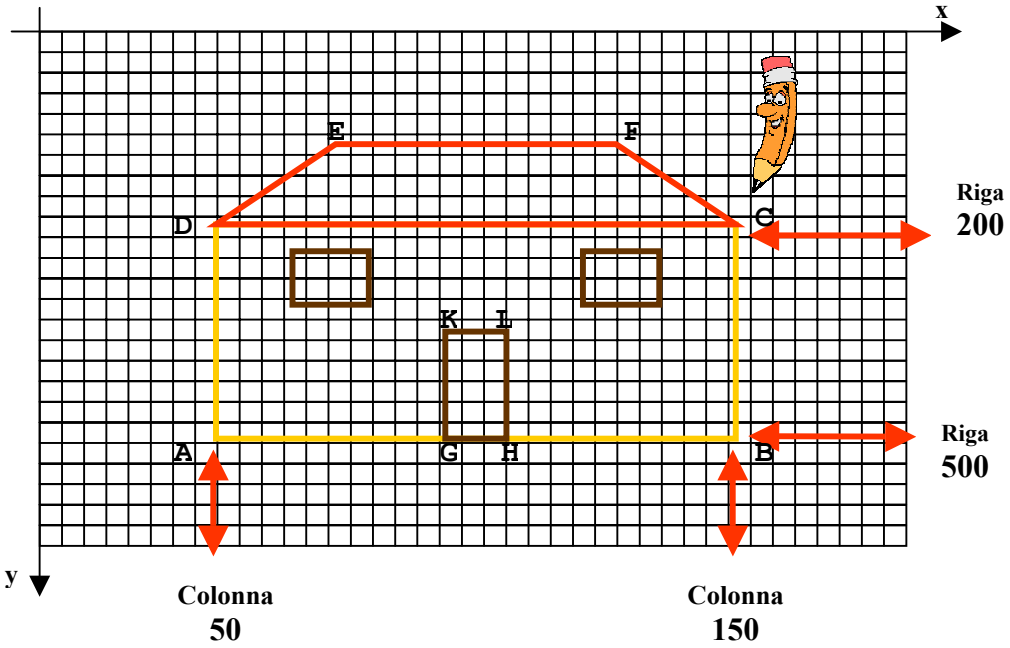


Cominciamo con disegnare il segmento AB partendo dall'ipotesi che A stia nella colonna 50 a partire da sinistra e nella riga 500 a partire dall'alto e che B sia nella colonna 150 e nella riga 500.
Con parole più eleganti, anche se più difficili, si dovrebbe dire che:
il punto A ha ascissa = 50 e ordinata = 500;
il punto B ha ascissa = 150 e ordinata = 500.
Sinteticamente possiamo scrivere:
A (50, 500)
B (150,500)

Se ci riesci sei un DRAGO...

Per disegnare il segmento AB scriveremo allora:

```
pygame.draw.line(areapalloncino,(255,255,0),(50,500),(150,500))
```



Disegnamo ora il segmento BC supponendo C(150,200). Basterà scrivere:

```
pygame.draw.line(areapalloncino, (255,255,0),(150, 500), (150,200))
```

Continuiamo disegnando i segmenti CD e DA con le istruzioni seguenti:

```
pygame.draw.line(areapalloncino, (255,0,0),(150, 200), (50,200))
pygame.draw.line(areapalloncino, (255,255,0),(50,200), (50,500))
```

Hai notato che ho cambiato colore al segmento CD.
Sai dire perchè?

Disegniamo ora il tetto supponendo E(80, 170) e F(370, 170).
Partendo dal punto D, disegniamo DE, EF e FC scrivendo:

```
pygame.draw.line(areapalloncino, (255,0,0),(50,200),(80,170)))
pygame.draw.line(areapalloncino, (255,0,0),(80,170),(370,170))
pygame.draw.line(areapalloncino, (255,0,0),(370,170),(150,200))
```

Ora sei sicuramente in grado di disegnare da solo la porta e le due finestre.
Puoi iniziare con la porta, disegnando i tre segmenti GK, KL, LH. Quindi potrai passare alle finestre.
Ricordati di cambiare i colori!

Segmenti e casette

Riprendi ora il programma che disegna il palloncino e aggiungi l'istruzione scritta in rosso nell'esempio esattamente dove indicato:

```
import pygame
import time
pygame.init()
areapalloncino = pygame.display.set_mode ((400,500))
pygame.draw.circle(areapalloncino, (255,0,0), (300,50), 30, 2)
pygame.draw.line(areapalloncino, (255,255,0),(300,80), (300, 280))
pygame.display.update()
time.sleep(10)
```

La nuova istruzione serve ad aggiungere il cordino al palloncino!
Più in generale l'istruzione `pygame.draw.line` serve a disegnare un segmento.
Esaminiamo i parametri dell'istruzione che abbiamo scritto:

```
pygame.draw.line(areapalloncino, (255,255,0),(300,80), (300, 280))
```

1 **2** **3** **4**

1 "areapalloncino" ha il significato che già conosci

2 questo è il codice del colore del segmento che vogliamo disegnare (il cordino del palloncino)

3 queste sono le coordinate del punto di partenza del segmento (300-esima colonna e 80-esima riga)

4 queste sono le coordinate del punto di arrivo del segmento (300-esima colonna e 280-esima riga)

Nota bene:
per disegnare un segmento devi indicare le coordinate del punto di partenza e del punto di arrivo. Se il segmento è verticale per entrambi i punti la colonna sarà la stessa mentre la riga del punto di arrivo sarà data dalla coordinata del punto di partenza più la lunghezza del segmento (che decidi tu).
Nel nostro esempio il cordino del palloncino è lungo 200 pixel.

Prova ora a rispondere a queste due domande (non proprio facili):
- Come mai le coordinate del punto di partenza del cordino sono (300,80)?
- Quali saranno le coordinate di un segmento orizzontale che parte dalla colonna 150 ed è lungo 50 (decidi tu su quale riga disegnarlo)?

Perchè il punto di partenza è perpendicolare al centro del cerchio (300,50) ed è sulla circonferenza quindi al centro del cerchio bisogna aggiungere la dimensione del raggio (30).
Le coordinate potrebbero essere, ad esempio: (150, 100) – (200, 100)

...finalmente: Animazione!

Sai come si fa a costruire un cartone animato sul video?
Un cartone animato è costituito da una successione di disegni, leggermente diversi uno dall'altro, che quando sono proiettati in sequenza danno l'impressione del movimento.
Riprendiamo il programma che disegna il palloncino e con poche istruzioni in più … facciamolo volare verso l'alto.

```
import pygame
import time
pygame.init()
areapalloncino = pygame.display.set_mode((400,500))
pygame.draw.circle(areapalloncino, (255,0,0), (300,350), 30)
pygame.draw.line(areapalloncino, (255,255,0),(300,380), (300, 580))
pygame.display.update()
time.sleep(0.30)
areapalloncino.fill(255,255,255)
pygame.draw.circle(areapalloncino, (255,0,0), (300,250), 30)
pygame.draw.line(areapalloncino, (255,255,0),(300,280), (300, 480))
pygame.display.update()
time.sleep(0.30)
areapalloncino.fill(255,255,255)
pygame.draw.circle(areapalloncino, (255,0,0), (300,150), 30)
pygame.draw.line(areapalloncino, (255,255,0),(300,180), (300, 380))
pygame.display.update()
time.sleep(0.30)
areapalloncino.fill(255,255,255)
```

Per far volare il palloncino, abbiamo dovuto disegnare il nostro palloncino e il cordino per tre volte, in tre posizioni diverse, spostando il cerchio e il segmento di 100 pixel verso l'alto la seconda e la terza volta (da 350 il centro del cerchio che rappresenta il pallone si sposta a 250 e poi a 150).

Esaminiamo con attenzione il programma che abbiamo scritto. Le prime quattro istruzioni hanno il solito significato.

La quinta istruzione:
`pygame.draw.circle(areapalloncino, (255,0,0), (300,350), 30)`
significa: preparati a disegnare un palloncino con centro nella posizione (300, 350)

La sesta istruzione:
`pygame.draw.line(areapalloncino, (255,255,0),(300,380), (300, 580))`
significa: preparati a disegnare il cordino di quel palloncino

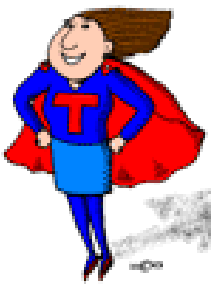
La settima istruzione:
`pygame.display.update()`
equivale al seguente ordine " proietta sul video i due disegni del palloncino e del cordino che hai preparato".

L'ottava istruzione:
`time.sleep(0.30)`
equivale all'ordine di aspettare, senza far nulla, un tempo dell'ordine di 0,30 secondi, ovvero 3 decimi di secondo.
Sfortunatamente questa istruzione potrebbe presentare qualche problema, in quanto è stata pensata per un calcolatore di velocità media. Se il tuo calcolatore è veloce, potrebbe succedere che il programma aspetti meno di 0,30 secondi, per cui probabilmente dovrai cambiare il tempo di attesa.

La nona istruzione:
`areapalloncino.fill((255,255,255))`
è un'istruzione nuova e anch'essa molto importante. Ordina di coprire tutta l'area del disegno (il nostro foglio "areapalloncino") con il colore (255,255,255), cioè ordina di riempire tutto l'area di pixel bianchi. Ed e' importante per evitare che i palloncini siano disegnati sullo stesso "foglio", altrimenti noi vedremmo tre palloncini invece di un palloncino che vola.

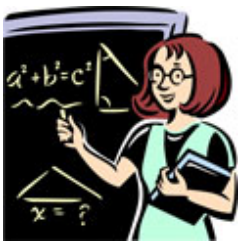
In sostanza questa istruzione è equivalente all'ordine di cancellare tutto il video e ricominciare a disegnare.

E' importante quello che succede nelle istruzioni seguenti, con le quali ridisegno il palloncino e il cordino in una nuova posizione e aspetto nuovamente un pò prima di procedere al nuovo disegno del palloncino.



Segmenti, casette, palloncini….
Ok, non è facile, ma considera che è la prima volta che ti cimenti in un lavoro così impegnativo.

Adesso prova a inventare qualche esercizio per impratichirti un po'.
Qualche suggerimento?
Inizia con il variare colori e dimensioni, a riempire di colore i disegni che ottieni….



....continua l'animazione...

Il programma che abbiamo scritto è un programma di animazione molto rudimentale, perchè l'immagine del palloncino si muove a scatti, spostandosi ogni volta di 100 pixel. In effetti, più che volare il nostro palloncino saltella!

Ora proviamo a scrivere il programma seguente che fa le stesse cose del precedente ma … meglio.

```
import pygame
import time
pygame.init()
areapalloncino = pygame.display.set_mode((400,500))
i = 0
while i < 600:
    pygame.draw.circle(areapalloncino, (255,0,0), (300,350), 30)
    pygame.draw.line(areapalloncino,(255,255,0),(300,380),(300, 580))
    pygame.display.update()
    time.sleep(0.03)
    areapalloncino.fill((255,255,255))
    i = i + 1
```

Nota bene: La struttura del programma ora è la seguente:

```
i = 0
while < 600:
    #disegna il palloncino spostato di i pixel verso l'alto
    pygame.display.update()
    time.sleep(0.03)
    areapalloncino.fill((255,255,255))
    i = i + 1
```

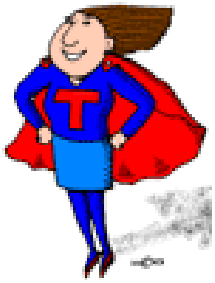
In sostanza, quando:
i = 0 (la prima volta nel ciclo while): il palloncino è in posizione di partenza
i = 1 (la seconda volta " " "): il palloncino è spostato di 1 pixel verso l'alto
i = 2 (la terza volta " " "): si ridisegna il palloncino spostato verso l'alto di 2 pixel
..... e cosi via per 600 volte!

E' importante sottolineare che il palloncino non si muove ma viene disegnato 600 volte in 600 posizioni diverse, per dare l'illusione del movimento.
Per quanto tempo il palloncino rimane sullo schermo nella stessa posizione? E sai anche dire perchè abbiamo scelto proprio quel tempo?
L'istruzione time.sleep (0.03) lascia l'immagine sul video solo 3 centesimi di secondo. Siccome il nostro occhio conserva un'immagine che colpisce la retina per un tempo di poco superiore a quei 3 centesimi di secondo, ora non ci accorgiamo più del fatto che il movimento avviene a scatti, ma abbiamo l'impressione che il movimento sia continuo. Una buona animazione deve avere tempi di aggiornamento dell'immagine dell'ordine di centesimi di secondo.

RAGGIUNTA LA META !

In dodici passi abbiamo fatto…un grande salto!
Ovvero: imparato le basi della programmazione in Python.

Fantastico!!!
Sono riuscito a far volare il palloncino rosso.
Adesso cosa faccio?



Ricordati che quello che hai imparato negli step precedenti costituisce solo un inizio per saper programmare e pertanto, se vuoi, dovrai approfondire l'argomento in tempi successivi.

Intanto consolida le tue conoscenze ed esercitati utilizzando quello che hai finora appreso.

Come fare?

Prova a inventare tu nuovi programmi.
Scoprirai con piacere di essere già in grado di fare moltissime cose.
Segui le indicazioni che ti hanno dato Angelo, Martina e Mario ma soprattutto, divertiti a programmare e ad inventare nuove applicazioni.

Quando ti senti pronto passa alla verifica finale.

Buon lavoro!

La nostra fatica è finita, prova a dare un voto al lavoro che abbiamo fatto per te.

Se ritieni di essere diventato un bravo apprendista programmatore disegna tante meline rosse nello spazio apposito. Se hai trovato difficile lavorare con noi…una melina sola.
Se invece ti sei annoiato tanto o non hai capito….fai tu!

……

E' arrivato il momento di…	fare la verifica finale
	**Arrivati alla fine del nostro percorso, ti proponiamo tre esercizi per verificare la tua abilità come programmatore.** **Per svolgere il primo esercizio devi aver studiato i primi undici step; per gli altri due ti occorre conoscere anche il dodicesimo step.** **Buon Lavoro.**

PRIMO ESERCIZIO

I numeri di Martina: ordina questi tre numeri in ordine decrescente:

61, 5, 23

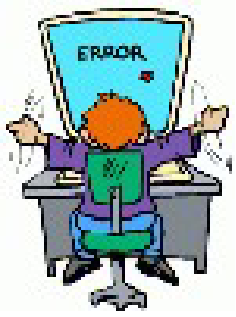
(Il programma può essere risolto in vari modi, noi ti proponiamo una soluzione classica che utilizza un algoritmo molto noto).

SECONDO ESERCIZIO

Il ciao di Clotilde: utilizzando la funzione time.sleep di pygame visualizza la lettera C poi la I, a seguire la A e la O, facendo in modo che appaiano una dopo l'altra a distanza di alcuni secondi.

TERZO ESERCIZIO

L'orologio di Angelo: utilizzando pygame, cerchi e segmenti, disegna un orologio rotondo con una lancetta che cambi posizione di 90° ogni 15". Ad esempio, l'orologio sarà un cerchio e la lancetta un segmento che cambierà posizione di 90° ogni 15 secondi.
(Ricordati di misurare il tempo corretto per la funzione time.sleep)



Infine…..

Se riuscito a risolvere tutti e tre gli esercizi?

Bravissimo, ormai sei un programmatore esperto!!

Hai risolto solo parte degli esercizi oppure ti sei trovato in difficoltà?

Torna ad indossare il cappello pensante;
vai alla pagina delle soluzioni e confronta il tuo lavoro con i programmi corretti, rifletti sulle cose che non hai saputo fare e riprova a scrivere i programmi, soprattutto non arrenderti, vedrai che ce la farai…infine, se vuoi e se proprio non sei riuscito a risolvere gli esercizi da solo, vai alla pagina delle soluzioni e studiati il listato dei programmi .

Se non sei riuscito a risolvere nessun esercizio…. prova con il cappello magico e divertiti con i giochi di Magogian!!!

Magogian propone:
La matemagica

Ovvero: giochiamo un po' per continuare ad esercitarci

1. Quanti soldi hai in tasca?

Vuoi scoprire quanti soldi ha in tasca un tuo amico?
Chiedigli di:
- partire dalla somma (s) dei soldi che ha in tasca (ad es. € 35)
-Moltiplicarla per 2: 35x2=70 (2s)
-Aggiungere tre al prodotto: 70+3= 73 (2s+3)
-Moltiplicare la somma per 5: 5x73=365 5(2s+3)=10s+15
-Sottrarre 6: 365-6=359 10s+15-6=10s+9
-Darti il risultato finale: 359
Togli l'unità e saprai che ha in tasca 35 euro!
Il gioco si basa sulle proprietà algebriche dei numeri: si sa che in algebra **x** può assumere qualsiasi valore numerico. Se io scrivo l'espressione algebrica
10**x**+9 e poi assegno a x successivamente il valore 1, 4, 20
per x=1 il risultato sarà 19
per x=4 " " 49
per x=20 " " 209
Cosa noti?
Il numero scelto compare nelle decine tolto il numero finale e pertanto è sufficiente togliere le unità per indovinare il gioco.

Proviamo a trasformare questo gioco in un programma per il computer:

```
# quanti soldi hai in tasca

print "Vediamo se indovino subito quanti soldi hai in tasca"

print "Moltiplica il numero dei soldi che hai in tasca per 2"

print "Ora aggiungi 3 al prodotto"

print "Ora moltiplica per 5"

print "Sottrai 6"

soldiintasca = input ("Inserisci ora il risultato ottenuto")

soldiintasca = str(soldiintasca)

print "Secondo me hai in tasca", soldiintasca[1:3], "euro"
```

E se utilizzo due o più incognite?

Si possono realizzare giochi che consentono di trovare più numeri.
Nella pagina seguente troverai altri due giochi che ti potranno sembrare….sorprendenti! Per prima cosa devi assegnare un numero a ogni mese dell'anno: 1 gennaio, 2 febbraio, 3 marzo ….fino a 12 per dicembre. Poi devi stabilire le due variabili: supponiamo che m sia uno qualsiasi dei mesi dell'anno e g il numero che esprime il giorno ….segui attentamente le istruzioni e sarai in grado di scoprire la data del compleanno dei tuoi amici.

2. Indovina la data del compleanno

Chiedi a una tua amica (se non è forte in matematica procurati una calcolatrice…)
di moltiplicare mentalmente per 5 il numero del mese in cui è nata (5m). Supponiamo che sia nata il 13 giugno (5x6=30).

Poi chiedile di :
-aggiungere 7: 30+7=37 (5m+7)
-moltiplicare per 4: 37x4=148 (4(5m+7)=20m+28)
-aggiungere 13: 148+13=161 (20m+28+13=20m+41)
-moltiplicare per 5: 161x5=805 (5(20m+41)=100m+205)
-aggiungere il giorno del mese in cui è nata: 805+13=818 (100m+205+g)
-sottrarre 205 dal numero ottenuto: 818-205=613 (100m+g+205-205).

Chiedi alla tua amica di dirti il numero ottenuto (613): dalle centinaia otterrai il mese e dalle unità il giorno.

Non ti resta che di ricordarti di augurarle, a tempo debito, Buon compleanno!

```
# indovina la data del compleanno

print "Vediamo se indovino subito la data del compleanno\n"
print "Moltiplica il numero del mese in cui sei nata per 5\n"
print "Ora aggiungi 7\n"
print "Ora moltiplica per 4\n"
print "Aggiungi 13\n"
print "Moltiplica per 5\n"
print "Aggiungi il giorno del mese in cui sei nato\n"
print "Sottrai 205 dal numero ottenuto\n"
compleanno = input ("Inserisci ora il risultato ottenuto\n")
compleanno = int(compleanno)
x= 999
if compleanno > x:
  comp = str(compleanno)              # trasforma il numero in stringa
else:
  comp = '0' + str(compleanno)
if comp[0:2] == "01":
   mese = "gennaio"
elif comp[0:2] == "02":
   mese = "febbraio"
elif comp[0:2] == "03":
   mese = "marzo"
elif comp[0:2] == "04":
   mese = "aprile"
elif comp[0:2] == "05":
   mese = "maggio"
elif comp[0:2] == "06":
   mese = "giugno"
elif comp[0:2] == "07":
   mese = "luglio"
elif comp[0:2] == "08":
   mese = "agosto"
elif comp[0:2] == "09":
   mese = "settembre"
elif comp[0:2] == "10":
   mese = "ottobre"
elif comp[0:2] == "11":
   mese = "novembre"
elif comp[0:2] == "12":
   mese = "dicembre"
print "Sei nata il ", comp[2:4], "del mese di", mese
```

3. Indovina la data della Rivoluzione Francese

Vuoi stupire la tua insegnante di storia con un gioco di matemagica?

Chiedile di:
- moltiplicare il numero del mese per 5: 5x7=35
(5m)
- sottrarre 3 al risultato:
35-3=32(5m-3)
- raddoppiare il numero ottenuto:
32x2=64
(2(5m-3)=10m+6)
- moltiplicarlo per 10:
64x10=640
(100m-60)
- aggiungere il giorno:
640+14= 654
(100m-60+g)
-dirti il numero ottenuto:654
Aggiungi 60 e otterrai 714
(100m+g+60-60)
La data dell'inizio della Rivoluzione Francese è 14 luglio

```
# la Rivoluzione Francese

print "Vediamo se indovino la data della Rivoluzione Francese\n"

print "Moltiplica il numero del mese corrente per 7\n"

print "Ora sottrai 3 al risultato\n"

print "Raddoppia il numero ottenuto (moltiplica per 2)\n"

print "Moltiplica ora per 10\n"

print "Ora aggiungi il giorno dell'inizio della Rivoluzione Francese\n"

rivoluzionefrancese = input ("Inserisci ora il risultato ottenuto\n ")

# trasformo il numero in stringa

rivoluzionefrancese = str(rivoluzionefrancese)

print "Secondo me la Rivoluzione Francese è iniziata il",
rivoluzionefrancese[1:3], "Luglio
```

Nota bene: le istruzioni in rosso indicano la forma algebrica che tu devi applicare, quelle in nero le operazioni che svolge il tuo amico

Adesso tocca a te inventare nuove soluzioni.
Prova, ad esempio, a indovinare l'età e il numero di scarpe del tuo vicino di casa (con discrezione, naturalmente).
Un consiglio da mago?
Se vuoi confondere un po' le idee ai tuoi spettatori, ti consiglio di inserire qui e là qualche numero…inutile o qualche, altrettanto inutile, formula magica. Si sa che oggi…tutto quanto fa spettacolo!
Ricorda però che la chiave del gioco poggia sul fatto che g può assumere solo valori che si collocano nelle decine e nelle unità mentre 100m assume valori che si collocano nelle centinaia. Questo pone dei limiti ai tuoi giochi, adatti a valori di g sotto le centinaia; ecco perché ti sono stati proposti esempi relativi a date, età, numeri di scarpe, etc.

Cfr: *Giochi logici e matematici*, Franco Agostini, Arnoldo Mondatori Editore

Alla fine dei 12 passi...

e dopo lunghe discussioni, il consiglio di classe ha deciso che è arrivato per te il momento di giocare un po'.

Ci siamo ricordati che quando avevamo la tua età ci piaceva molto, soprattutto a scuola durante le ore di supplenza, giocare a "Tris". Noi avevamo carta e penna, tu ora hai il tuo PC.

Con un gioco di squadra e con l'aiuto di Python proviamo a realizzare il gioco del **TRIS** sul PC.

"Materiale" occorrente:

- Python finestra per scrivere il programma (IDLE)
- Conoscenza del contenuto dei primi undici step
- Conoscere le regole del Tris

Oltre a una buona dose di pazienza!!!

Questo programma ti consente di realizzare sul tuo computer un gioco che hai fatto tante volte con carta e matita, qualche volta addirittura in spiaggia, disegnando sulla sabbia con il bastoncino di legno del ghiacciolo appena consumato.

Non vorrei essere troppo presuntuoso spiegandoti le regole del gioco del Tris ma … è sempre meglio ripassare.

I giocatori, che sono sempre solo due, scelgono ciascuno una lettera o un simbolo. Sul computer è meglio scegliere una lettera perché è più facile da rappresentare, quindi il primo giocatore sceglierà la N e il secondo la R. Per giocare avranno a disposizione una scacchiera composta da 9 riquadri disposti su 3 file con il seguente schema:

	1 2 3
a	x x x
b	x x x
c	x x x

Inseriranno a turno un simbolo nello spazio scelto ancora libero. Vincerà chi per primo riuscirà ad inserire 3 simboli consecutivi in orizzontale oppure in verticale o in diagonale.

Ad esempio:

	1 2 3
a	N N N
b	R x x
c	x x R

	1 2 3
a	N x R
b	R N x
c	x x N

	1 2 3
a	N x x
b	N R x
c	N x R

Ora con molta attenzione impostiamo le varie parti del programma.

La scacchiera

Per rappresentare la scacchiera utilizzeremo nove variabili che chiameremo a1, a2, a3, b1, b2, b3, c1, c2, c3.
Ciascuna di queste variabili può assumere soltanto tre valori: "R" oppure "N" oppure "X".
Inizialmente tutte le variabili assumono il valore X; successivamente, se, ad esempio, il giocatore che ha scelto "N" come suo simbolo mette quel simbolo nella riga a in corrispondenza della colonna 2, la variabile a2 assumerà il valore "N".

La visualizzazione della scacchiera

Per visualizzare la scacchiera è sufficiente scrivere questo breve programma:

```
print '\t 1 \t 2 \t 3 \n'
print 'a \t', a1, '\t', a2, '\t', a3, '\n'
print 'b \t', b1, '\t', b2, '\t', b3, '\n'
print 'c \t', c1, '\t', c2, '\t', c3, '\n'
```

Avremo bisogno di visualizzare la scacchiera molte volte nell'arco della partita, quindi converrà scrivere una funzione per la visualizzazione.

```
def scacchiera():
    print '\t 1 \t 2 \t 3 \n'
    print 'a \t', a1, '\t', a2, '\t', a3, '\n'
    print 'b \t', b1, '\t', b2, '\t', b3, '\n'
    print 'c \t', c1, '\t', c2, '\t', c3, '\n'
```

Il controllo della vittoria

Questa operazione dovrà essere ripetuta dopo ogni mossa nel corso della partita e quindi converrà scrivere una funzione che chiameremo "vittoria".

La funzione "vittoria" avrà un'unica variabile su cui operare, il nome "N" oppure "R" del giocatore che ha fatto la sua giocata. A questa variabile diamo il nome "n".

```
def vittoria(n):
     if ((a1 == n and a2 == n and a3 == n) or (b1 == n and b2 == n and b3 == n) or
         (c1 == n and c2 == n and c3 == n) or
       (a1 == n and b1 == n and c1 == n) or (a2 == n and b2 == n and c2 == n) or
       (a3 == n and b3 == n and c3 == n) or
       (a1 == n and b2 == n and c3 == n) or (a3 == n and b2 == n and c1 == n)):
          EsisteVincitore = 'si'
     else:
          EsisteVincitore = 'no'
     return EsisteVincitore
```

variabili globali, variabili locali

La funzione "vittoria" contiene una novità importante su cui dobbiamo riflettere insieme.

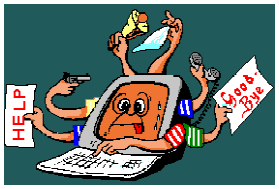
In una funzione di Python si possono introdurre variabili che appartengono al programma principale, come a1,..., c3..., che sono chiamate "**variabili globali**", perchè hanno un valore globale indipendentemente da dove sono scritte. Inoltre, vi sono variabili come "EsisteVincitore" che sono valide soltanto entro il corpo della funzione e che sono chiamate "**variabili locali**".
Le variabili locali **non** sono utilizzabili fuori dal corpo della funzione; inoltre, entro una funzione **non** si può modificare il valore di una variabile globale.
Se vogliamo utilizzare il valore di una variabile locale entro il programma principale dobbiamo scrivere, come abbiamo fatto in "vittoria", una istruzione come

```
return EsisteVincitore
```

L'istruzione "return" trasferisce il valore della variabile locale "EsisteVincitore" nella scatola che ha il nome della funzione (nel nostro caso, "vittoria"). Quando poi, nel programma principale, scriveremo, ad esempio,

```
partitafinita = vittoria('R')
```

il contenuto di "vittoria" sarà trasferito nella variabile globale "partitafinita". In sostanza, il contenuto della variabile locale "EsisteVincitore" (che può essere "si" oppure "no") viene dapprima trasferito nella scatola di nome "vittoria" e poi nella scatola di nome "partitafinita".



Il programma principale

Per scrivere il programma principale serviranno molte testoline pensanti perchè è un programma un po' lungo e non sempre facilissimo.
Accetta la sfida! Ormai sei perfettamente in grado di risolvere questi problemini e comunque, se proprio non ci riesci, alla fine del capitolo abbiamo messo una delle soluzioni possibili che potrai studiare con attenzione, ricopiare sul PC e poi .....GIOCARE!!!!

Il programma inizia ponendo tutte le nove variabili uguali a 'x'.

```
a1 = 'x'
a2 = 'x'
a3 = 'x'
b1 = 'x'
b2 = 'x'
b3 = 'x'
c1 = 'x'
c2 = 'x'
c3 = 'x'
```

Poi visualizza la scacchiera

```
scacchiera()
```

Quindi inizializza una variabile di nome "nmosse" che conta il numero delle mosse fatte dai giocatori e la variabile "partitafinita" che indica quando la partita è finita.

```
nmosse = 0
partitafinita = 'no'
```

Poi informa che muoverà prima "N".

```
print 'Il primo giocatore sarà N \n'
```

A questo punto il programma entra in un ciclo che sarà percorso più volte. Poichè il numero totale delle mosse non può superare 9 (il numero delle posizioni sulla scacchiera) il ciclo sarà controllato dalla variabile "nmosse", che abbiamo posto = 0. Dopo la prima mossa, nmosse sarà posta = 1 e dopo la seconda mossa nmosse sarà posta = 2. A questo punto si torna all'inizio del ciclo che quindi sarà descritto al massimo 5 volte.

Prima di tutto si invita il giocatore a fare la propria mossa. Il programma verificherà che la mossa sia corretta con le istruzioni:

```
print 'Muova N \n'
mossafatta = 'no'
x = raw_input('Dove vuoi scrivere N: a1, a2, a3, b1, b2, b3, c1, c2, c3? \n')
if x == 'a1' and a1 == 'x':
   a1 = 'N'
   mossafatta = 'si'
```

In sostanza il programma verifica che la posizione dove il giocatore intende collocare il proprio simbolo contenga una "x" e sostituisce quella "x" con il simbolo ("N" o "R") del giocatore stesso. La variabile "mossafatta" viene inizializzata al valore "no" e viene posta = "si" se il giocatore ha indicato una mossa corretta, ossia uno dei simboli a1,a2,.....c3, come propria risposta e non ha chiesto di sostituire il simbolo dell'avversario con il proprio. Se il giocatore richiede una mossa non legittima o dà una risposta non corretta, verrà squalificato.
Nota bene che quando si scopre che c'è un vincitore (EsisteVincitore = 'si') si pone nmosse = 10 per uscire subito dal ciclo. Se il numero delle mosse fatte è = 0 e non c'è vincitore, il programma decide il pareggio e pone nmosse = 10 per uscire dal ciclo.

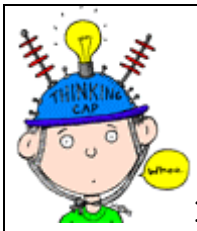
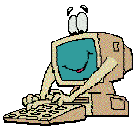
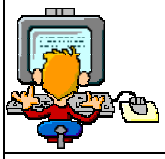
```
while nmosse < 9 :
    print 'Muova N \n'
    mossafatta = 'no'
    x = raw_input('Dove vuoi scrivere N: a1, a2, a3, b1, b2, b3, c1, c2, c3?
                                                                    \n')
    if x == 'a1' and a1 == 'x':
       a1 = 'N'
       mossafatta = 'si'
    elif x == 'a2' and a2 == 'x':
       a2 = 'N'
       mossafatta ='si'
    elif x == 'a3' and a3 == 'x':
       a3 = 'N'
       mossafatta ='si'
    elif x == 'b1' and b1 =='x':
       b1 = 'N'
       mossafatta ='si'
    elif x == 'b2' and b2 == 'x':
       b2 = 'N'
       mossafatta ='si'
    elif x == 'b3' and b3 == 'x':
       b3 = 'N'
       mossafatta ='si'
    elif x == 'c1' and c1 == 'x':
       c1 = 'N'
       mossafatta ='si'
    elif x == 'c2' and c2 == 'x':
       c2 = 'N'
       mossafatta ='si'
    elif x == 'c3' and c3 == 'x':
       c3 = 'N'
       mossafatta ='si'

    if mossafatta == 'no':
       print 'SQUALIFICATO!'
       partitafinita = 'si'
       nmosse = 10
    scacchiera()
    nmosse = nmosse + 1
    partitafinita = vittoria('N')

    if partitafinita == 'si':
       print 'Ha vinto N \n'
       nmosse = 10
    elif   nmosse == 9:
       print 'PAREGGIO'
       nmosse = 10
```

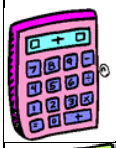
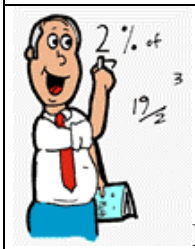
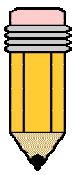
Se non si sono fatte tutte le mosse, il programma invita il giocatore "R" a fare la sua mossa e procede al controllo di questa mossa con la stessa logica che è stata adottata per "N".

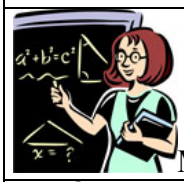
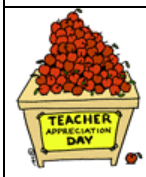
```
if partitafinita == 'no' and nmosse < 9:
   nmosse = nmosse +1
   mossafatta = 'no'
   print 'La mossa tocca a R \n'
   x = raw_input('Dove vuoi scrivere R: a1, a2, a3, b1, b2, b3, c1, c2, c3? \n')
   if x == 'a1' and a1 == 'x':
      a1 = 'R'
      mossafatta = 'si'
   elif x == 'a2'and a2 == 'x':
      a2 = 'R'
      mossafatta = 'si'
   elif x == 'a3'and a3 == 'x':
      a3 = 'R'
      mossafatta = 'si'
   elif x == 'b1'and b1 == 'x':
      b1 = 'R'
      mossafatta = 'si'
   elif x == 'b2'and b2 == 'x':
      b2 = 'R'
      mossafatta = 'si'
   elif x == 'b3'and b3 == 'x':
      b3 = 'R'
      mossafatta = 'si'
   elif x == 'c1'and c1 == 'x':
      c1 = 'R'
      mossafatta = 'si'
   elif x == 'c2'and c2 == 'x':
      c2 = 'R'
      mossafatta = 'si'
   elif x == 'c3'and c3 == 'x':
      c3 = 'R'
      mossafatta = 'si'
   if mossafatta == 'no':
      print 'SQUALIFICATO!'
      partitafinita = 'si'
      nmosse = 10
   scacchiera ()
   partitafinita = vittoria('R')
   if partitafinita == 'si':
      print 'Ha vinto R \n'
      nmosse = 10
```

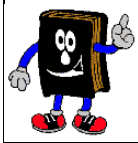
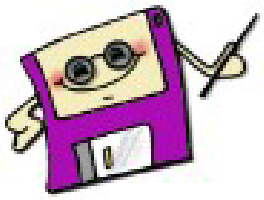
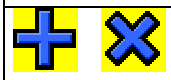
Quella che ti ho preparato è la versione base del gioco.
Utilizzandola per giocare con i tuoi amici ti accorgerai di aver bisogno di perfezionarla e, se vuoi, puoi provare a svilupparla aggiungendo altre funzioni.
Ti ricordi che i programmi si dividono in interpretati e compilati?
Python appartiene alla prima categoria e pertanto ogni volta che vuoi giocare devi scrivere: **python tris.py**

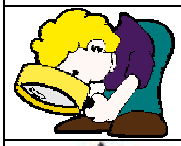
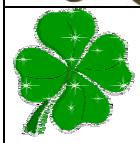
CLIPART

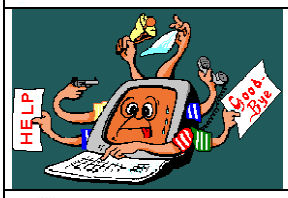
 1	http://school.discovery.com/clipart/index.html "Clip art licensed from the Clip Art Gallery on DiscoverySchool.com"
	http://emiglia.blog.dada.net/img/scatola.jpg
 2	http://school.discovery.com/clipart/index.html "Clip art licensed from the Clip Art Gallery on DiscoverySchool.com"
	http://www.clipart.co.uk/cgi-bin/animdisplay2.cgi?computer/an2.gif http://www.free-clipart.net/
	http://www.teachnet.com/ **Copyright © 1995-2007. All rights reserved.-Teachnet.Com-120 Kodiak-Kechi KS 67067-U.S.A.** Teachnet.Com gives you permission to use, copy and distribute documents and related graphics from Teachnet.Com, so long as:**1.** the above copyright notice appears in all copies and that both the copyright notice and this permission notice appear, and **2.** you do not modify or change, in any way, any of the documents or related graphics available from this Server, and **3.** you do not use, copy or distribute, separately from its accompanying text, any graphics available from this Server.
 3	http://school.discovery.com/clipart/index.html "Clip art licensed from the Clip Art Gallery on DiscoverySchool.com"
	http://www.teacherfiles.com/animated/
	http://www.teacherfiles.com
 4	http://school.discovery.com/clipart/small/bus-stop2-color.gif "Clip art licensed from the Clip Art Gallery on DiscoverySchool.com"

 5	http://school.discovery.com/clipart/index.html "Clip art licensed from the Clip Art Gallery on DiscoverySchool.com"
	http://www.teacherfiles.com/clip_words.htm
 6	http://school.discovery.com/clipart/category/math.html "Clip art licensed from the Clip Art Gallery on DiscoverySchool.com"
	http://teachnet.com/how-to/clipart/animals **Copyright © 1995-2007. All rights reserved.-Teachnet.Com-120 Kodiak-Kechi KS 67067-U.S.A.** Teachnet.Com gives you permission to use, copy and distribute documents and related graphics from Teachnet.Com, so long as:**1.** the above copyright notice appears in all copies and that both the copyright notice and this permission notice appear, and **2.** you do not modify or change, in any way, any of the documents or related graphics available from this Server, and **3.** you do not use, copy or distribute, separately from its accompanying text, any graphics available from this Server
	http://www.hotdogroundup.com/files/dog%20wagging%20tail.gif
	http://www.clipart.co.uk/html/drawshop.shtml
	http://www.teacherfiles.com/clip_math.htm
 7	http://school.discovery.com/clipart/category/new.htm "Clip art licensed from the Clip Art Gallery on DiscoverySchool.com"
 **Mario**	http://www.teacherfiles.com/clip_teachers.htm
	http://www.theteachersguide.com

 8	http://school.discovery.com/clipart/category/new.htm "Clip art licensed from the Clip Art Gallery on DiscoverySchool.com"
 **Angelo**	http://www.theteachersguide.com/Edgraphicscolor.htm
 **Martina**	http://www.fotosearch.com/clip-art/teacher.html
 9	http://school.discovery.com/clipart/category/spec.html "Clip art licensed from the Clip Art Gallery on DiscoverySchool.com"
	http://www.teacherfiles.com/clip_words.htm
	http://www.teacherfiles.com/clip_sports.htm
	http://www.hellasmultimedia.com/webimages/education/education_images_6.htm Terms of use: (http://www.hellasmultimedia.com/webimages/termsofuse.htm)
	http://teacherfiles.com/clip_supplies.htm

	http://www.teacherfiles.com/clip_art_books.htm http://www.teacherfiles.com/clip_supplies.htm
	http://www.teacherfiles.com/clip_math.htm
	http://www.teacherfiles.com/clipart_computers.htm
	http://www.teacherfiles.com/clip-art_computer.htm
	www.teachnet.com **Copyright © 1995-2007. All rights reserved.-Teachnet.Com-120 Kodiak-Kechi KS 67067-U.S.A.** Teachnet.Com gives you permission to use, copy and distribute documents and related graphics from Teachnet.Com, so long as:**1.** the above copyright notice appears in all copies and that both the copyright notice and this permission notice appear, and **2.** you do not modify or change, in any way, any of the documents or related graphics available from this Server, and **3.** you do not use, copy or distribute, separately from its accompanying text, any graphics available from this Server
	http://www.hellasmultimedia.com/webimages/education/education_images_1.htm Terms of use: (http://www.hellasmultimedia.com/webimages/termsofuse.htm)
	http://www.theteachersguide.com/Edgraphicscolor.htm)
	http://www.teacherfiles.com/animated/animated_computer.htm
	http://www.webweaver.nu/clipart/education3.shtml

	http://teachnet.com/how-to/clipart/misc/index.html **Copyright © 1995-2007. All rights reserved.-Teachnet.Com-120 Kodiak-Kechi KS 67067-U.S.A.** Teachnet.Com gives you permission to use, copy and distribute documents and related graphics from Teachnet.Com, so long as:**1.** the above copyright notice appears in all copies and that both the copyright notice and this permission notice appear, and **2.** you do not modify or change, in any way, any of the documents or related graphics available from this Server, and **3.** you do not use, copy or distribute, separately from its accompanying text, any graphics available from this Server
	http://teachnet.com/how-to/clipart/misc/index.html **Copyright © 1995-2007. All rights reserved.-Teachnet.Com-120 Kodiak-Kechi KS 67067-U.S.A.** Teachnet.Com gives you permission to use, copy and distribute documents and related graphics from Teachnet.Com, so long as:**1.** the above copyright notice appears in all copies and that both the copyright notice and this permission notice appear, and **2.** you do not modify or change, in any way, any of the documents or related graphics available from this Server, and **3.** you do not use, copy or distribute, separately from its accompanying text, any graphics available from this Server
	http://www.teacherfiles.com/science_clipart.htm
	http://www.webweaver.nu/clipart/education3.shtml
	http://www.hellasmultimedia.com/webimages/anim-htm/anim8.htm Terms of use: (http://www.hellasmultimedia.com/webimages/termsofuse.htm)
	http://www.teacherfiles.com/icons_signs.htm
 10	http://school.discovery.com/clipart/category/seas.html
	**http://jwikert.typepad.com/the_average_joe/2007/06/index.html**
	www.FLMNetwork.com

	http://www.webweaver.nu/legal.shtml http://www.webweaver.nu/clipart/food-italian.shtml
	http://www.hellasmultimedia.com/webimages/anim-htm/anim16.htm Terms of use: (http://www.hellasmultimedia.com/webimages/termsofuse.htm)
	http://www.free-clipart.net
	http://www.veggierevolution.com/gifs/waiter.gif There is **no copyright** on any information, with the exception of a few of the recipes where permission has been obtained to reproduce them here. You are welcome to use any other information from the site to promote vegetarianism.
	http://www.hasslefreeclipart.com/clipart_cartanim/page1.html
	http://www.hasslefreeclipart.com/clipart_compusers/error.html **http://www.hasslefreeclipart.com/**
	http://www.dover.lib.nh.us/Childrens'Room/summerpics/clipart-detective.gif
	http://www.clipartguide.com/_thumbs/0041-0609-2517-5029.jpg **Artwork by Dennis Cox** Image Number: 0041-0605-2510-0311 Model Release: No ; Property Release: No; Commercial use is allowed This image is available under: **'Royalty Free'** licensing and is available for immediate download.

	http://www.giovannadarcoilmusical.it/busruler.gif http://www.giovannadarcoilmusical.it/raccontamidipiu-giovannaperlescuole2.htm (sito scolastico libero)
	http://www.cartooncottage.com/html/dragons.html http://www.cartooncottage.com/html/terms.html (condizioni d'uso)
  	http://www.hellasmultimedia.com/webimages/education/education_images_6.htm http://www.hellasmultimedia.com/webimages/education/education_lines_1.htm http://www.hellasmultimedia.com/webimages/education/education_lines_5.htm Terms of use: (http://www.hellasmultimedia.com/webimages/termsofuse.htm)

Link utili

In questo elenco abbiamo riportato i link ufficiali dove poter approfondire la conoscenza del linguaggio Python e alcuni link utili alla didattica collaborativa e all'uso del software libero nella scuola che noi abbiamo consultato per scrivere questo libro.

http://www.python.org/ sito ufficiale del linguaggio di programmazione Python

http://www.python.it/ sito italiano del linguaggio diprogrammazione Python

http://docs.python.it/html/ref/ Il manuale di riferimento di Python **Guido van Rossum Python Software Foundation** Traduzione presso **http://www.zonapython.it** Fred L. Drake, Jr., editor **Versione 2.3.4**

http://docs.python.it/html/lib/ La libreria di riferimento di Python **Guido van Rossum Python Software Foundation** Traduzione presso **http://www.zonapython.it** *Fred L. Drake, Jr., editor***Versione 2.3.4**

http://www.pygame.org/news.html **Pygame** è una raccolta di moduli Python scritti per progettare giochi.

http://groups.google.com/group/comp.lang.python/topics gruppo di discussione per utilizzatori di Python.

http://www.pythonware.com/ distribuzione di software per la community di python

http://it.diveintopython.org/index.html *Dive Into Python* è un libro gratuito sul linguaggio di programmazione Python, per programmatori esperti.

http://www.gentoo.it/Programmazione/byteofpython/ *A Byte of Python* Swaroop C H -Traduttori: Enrico Morelli, Massimo Lucci Version 1.20 Copyright © 2003-2005 Swaroop C H. Questo libro è rilasciato sotto la licenza Creative Commons Attribution-NonCommercial-ShareAlike License 2.0 .

Python, Marco Beri, Apogeo. Copyright ©2007 Apogeo srl – ISBN 978-88-503-2599-3

http://www.linuxdidattica.org/ il portale del software libero nella scuola

http://www.apprendimentocooperativo.it/

http://openbookproject.net/

http://www.livewires.org.uk/home

http://www.creativecommons.it/ il sito italiano dell'organizzazione non-profit Creative Commons per un copyright flessibile per opere creative

http://creativecommons.org/international/ il sito ufficiale dell'organizzazione non-profit Creative Commons

http://hackerforum.devil.it/ Ottimo forum italiano sull'hacking white hat: Hardware, Software, Programmazione , Reti, Sicurezza Informatica e Filosofia Hacker.

http://moodle.org/ (CMS) piattaforma web open source per l'e-learning

STEP1 Esercitiamoci un po' pagina 5

1.Prova a calcolare 153/4 e 73.0/8: che differenza c'è tra queste due operazioni?
Soluzione:
>>> 153/4
38
>>> 73.0/8
9.125
La prima è la divisione tra numeri interi, la seconda, poichè 73.0 ha il punto è una divisione tra numeri reali

2. a) 5*6+(4,5*6)
b) 4+4.5-(6*4/2)
quale di queste due espressioni contiene un errore?
Soluzione:
la prima espressione poiché non si possono scrivere numeri con la virgola come 4,5

3. Scrivi l'espressione per calcolare "quanti mesi hai".
Soluzione:
supponiamo di essere nato il 5 marzo 1990 e supponiamo che oggi sia il 3 febbraio 2010
>>> (2010 - 1990) * 12 + 3 -2
241

4. Inventa un'espressione che dia come risultato 48 ed una che dia come risultato 11.
Soluzione:
>>> 6*7 + 25/4
48
>>> ((2*5)+ (8+4))/2
11

5. a) Per andare da casa di Sandrone a casa di Giulia ci sono 3 km
b)per andare da casa di Giulia a casa di Clotilde ci sono 4 km
scrivi un programma che calcoli quanti km deve fare Sandrone per andare a trovare Giulia e Clotilde e tornare a casa ripassando da casa di Giulia.
Soluzione:
>>> (3+4)*2
14

6. Ora calcola quanti km ci vogliono per andare a trovare i tuoi 4 migliori amici e poi tornare a casa.
Soluzione:
supponiamo che l'amica Sophia abiti a 2 km da casa mia, Federico abiti a 1 km da casa di Sophia, Sandra invece abiti a 5 km da Federico e infine Luca sia a 2 km da Sandra allora:
>>> (2+1+5+2)*2
20

7.Calcola la lunghezza della tua "spanna"(la distanza tra il pollice e il mignolo in estensione) ed ora misura con la tua spanna il banco di scuola. Trova l'area del ripiano del banco.
Soluzione:
La mia spanna è 23 cm, un mio dito è circa 1.5 cm

il mio banco è 4 spanne e 3 dita di lunghezza e 2 spanne e 10 dita di larghezza.
lunghezza del banco
>>> 4*23 + 3*1.5
96.5
larghezza del banco
>>> 2*23 + 10*1.5
61.0
Area del banco
>>> 96.5*61.0
5886.5

8. Prendi il numero di telefono della mamma o del papà e prendi il numero di telefono della scuola:
- Moltiplica i due numeri
- Dividi i due numeri
- Sottrai i due numeri
- Eleva al quadrato i due numeri

Soluzione:
supponiamo che il numero di cellulare della mamma sia 323123123 e il numero della scuola sia 011123123
allora:
>>> 323123123 * 011123123
776101351055625L
>>> 323123123 / 011123123
134
>>> 323123123-011123123
320721248
>>> 323123123**2
104408552617273129L
>>> 011123123**2
5769003515625L

9. Calcola l'area della cucina di casa tua.
Soluzione:
Supponiamo che la cucina di casa sia larga 4m e lunga 5m
Allora l'area della cucina è:
>>> 4*5
20

10. Calcola il diametro, la circonferenza, l'area di un cerchio di raggio 2.5 cm
Soluzione:
diametro = raggio *2
>>> 2.5 *2
5.0
circonferenza = 2 * pigreco * raggio
>>> 2* 3.14 *2.5
15.700000000000001
area = pigreco * raggio * raggio
>>> 3.14*2.5*2.5
19.625

STEP2 Esercitiamoci un po' utilizzando Python Shell pagina 9

1. Puoi dare a una scatola il nome 10ART?
   Soluzione:
   No perchè le scatole devono sempre iniziare con una lettera

2. Quale di questi nomi è sbagliato?
Cane_M_4 CaneM4 4cane_M CANE_M4
Soluzione:
4cane_M

3. Puoi dare a due scatole diverse i nomi SCATOLA_1 e Scatola_1?
Soluzione:
Si, perchè la prima ha il nome in maiuscolo e la seconda in minuscolo

4. Puoi chiamare una scatola Print?
Soluzione:
Si, mentre non posso chiamare una variabile print perchè è una parola riservata

5. Se chiami una scatola Zio Pippo, cosa succede?
   Soluzione:
   Viene segnalato errore perchè non sono corretti i nomi delle scatole che hanno degli spazi

STEP3 Esercitiamoci un po'. pagina 14

1. Scrivi in Python la seguente istruzione per il calcolatore: dammi la scatola GAIA3 e mettici dentro il numero 10.
Soluzione:
>>> GAIA3=10

2. Illustra i passaggi attraverso i quali il calcolatore inserisce un numero in una scatola che era già stata usata in precedenza.
Soluzione:
Il computer fa quanto segue:
1. cerca la scatola che ha nome SCATOLA1 (per esempio),
2. apre la scatola e toglie il contenuto lasciandola vuota,ovvero toglie il foglietto che era stato introdotto prima,
4. mette nella scatola il nuovo contenuto che é il foglietto con scritto nuovo numero

3. Qual e' il significato dell'istruzione print?
Soluzione:
print stampa sullo schermo il contenuto delle variabili e di espressioni che seguono l'istruzione

4. Se hai la scatola MELA2 che contiene il numero 7, quale istruzione devi scrivere per visualizzare il contenuto di MELA2?
Soluzione:
print MELA2

5. Cosa visualizza il calcolatore se scrivo:
>>>SCATOLA2 = 10-3
>>>print SCATOLA2

>>>SCATOLA3 = 7 * 4
>>>print SCATOLA3

>>>SCATOLA4 = 24/8
>>>print SCATOLA4
Soluzione:
7
28
3

6. Se adesso scrivo: SCATOLA3 = 20/2 * 5
Print SCATOLA3
Cosa contiene SCATOLA3? E se scrivo PRINT SCATOLA3 ?
Soluzione:
avrò in entrambi i casi un errore perchè Print e PRINT non sono istruzioni, mentre SCATOLA3 contiene il valore 50

ESERCITARCI CON PYTHON pagina 16

Esercizio n°1:
Il mago Silvan fa tanti giochi di magia: dal suo cappello a cilindro escono tre conigli bianchi e due neri. Quanti conigli sono nascosti nel cappello?
Soluzione:
5 conigli

```
>>> coniglibianchi=2
>>> coniglineri=3
>>> conigli_nel_cappello = coniglibianchi + coniglineri
>>> print conigli_nel_cappello
5
```

Esercizio n°2:
Al mago Berri invece piace fare le magie con le maxi-carte: sono così grandi che quasi non stanno sul tavolo! Se ciascuna carta è lunga cm. 45 e larga cm. 30, quanto è grande la superficie di ciascuna carta?
Soluzione:
1350 centimetriquadrati

```
>>> lunghezza  = 45
>>> larghezza  = 30
>>> superficie_carta = lunghezza*larghezza
>>> print superficie_carta
1350
```

Esercizio n°3:
Quale superficie del tavolo occupano i quattro assi usati dal mago Berri per i suoi giochi di magia, affiancati per il lato più lungo?
Soluzione:
5400 centimetriquadrati

```
>>> lunghezza = 45
>>> larghezza = 30*4
>>> superficie_occupata = lunghezza*larghezza
>>> print superficie_occupata
5400
```

Esercizio n°4:
Il mago Gian ha un bellissimo mantello di seta nera ricamato con tante stelle argentate. Per farlo il sarto ha utilizzato ben 5 metri di stoffa. Se la stoffa costava 13 € al metro, quanto ha speso per comprarla?
Soluzione:
65 €

```
>>> lunghezza_stoffa = 5
>>> costo_stoffa = 13
>>> spesa = lunghezza_stoffa*costo_stoffa
>>> print spesa
65
```

Esercizio n°5:
Se un mantello costa 80 €, un cappello a cilindro 45 €, una bacchetta magica 20 €, un mazzo di maxi-carte 13 €, quanto costa l'attrezzatura per fare il mago?
Soluzione:
158 €
```
>>> mantello = 80
>>> cilindro = 45
>>> bacchetta = 20
>>> carte = 13
>>> costo_attrezzatura_mago = mantello+cilindro+bacchetta+carte
>>> print costo_attrezzatura_mago
158
```

Esercizio n°6:
Se ho 5 € di paghetta settimanale,
a quanto ammonta la mia paghetta mensile? Quanti € posso spendere in un anno?
Se un CD musicale costa 15 €, quanti ne posso comprare con la paghetta del mese di aprile?
Soluzione:
la paghetta mensile è 20 €
all'anno posso spendere 240 € (in un anno ci sono 52 settimane)
con la paghetta del mese di aprile posso comprare un cd
```
>>> paga_settimanale = 5
>>> paga_mensile = paga_settimanale*4
>>> print paga_mensile
20
>>> paga_annuale = paga_mensile*12
>>> print paga_annuale
240
>>> cd = 15
>>> numero_cd = paga_mensile/cd
>>> print numero_cd
1
```

Esercizio n°7:
Se il cortile della scuola misura m. 75 di lunghezza e m. 50 di larghezza, qual è la sua superficie?
Soluzione:
3750 mquadrati
```
>>> lunghezza = 75
>>> larghezza = 50
>>> superficie_cortile = lunghezza*larghezza
>>> print superficie_cortile
3750
```
Sapendo che un campo di calcetto misura m. 40 di lunghezza e m. 20 di larghezza, quanti campi di calcetto potrai ricavare dal cortile della scuola?
Soluzione: 4 campi di calcetto
```
>>> lunghezza_calcetto = 40
>>> larghezza_calcetto = 20
>>> superficie_calcetto = lunghezza_calcetto*larghezza_calcetto
>>> print superficie_calcetto
```

```
800
>>> numero_campi = superficie_cortile/superficie_calcetto
>>> print numero_campi
4
```

Esercizio n°8:
Nella tua classe ci sono 8 maschi e 10 femmine. Se Mario è alto m. 1.55, Fabio, Matteo e Luca sono alti m. 1.60, Andrea, Aldo, Giovanni e Giacomo m. 1.50, qual è l'altezza media dei maschi della classe? Se Marta, Giovanna, Elisabetta e Francesca sono alte come Mario, Stefania, Chiara e Simonetta sono alte m. 1.50, Daria e Domitilla sono 5 cm più piccole di Arianna che è alta m. 1,68, qual è l'altezza media delle femmine della classe?
Qual è l'altezza media della classe?
Soluzione:
l'altezza media dei maschi della classe è 1.54375 m

```
>>> maschi = 8
>>> femmine =10
>>> alunni = maschi + femmine
>>> print  alunni
>>> M = 1.55
>>> FML = 1.60*3
>>> AAGG = 1.50*4
>>> altezza_media_maschi = (M + FML+ AAGG)/  maschi
>>> print altezza_media_maschi
1.54375
```

l'altezza media delle femmine è 1.564 m

```
>>> MGEF = 1.55*4
>>> SCS = 1.50*3
>>> A = 1.68
>>> DD = 1.63*2
>>> altezza_media_femmine =  (MGEF+ SCS + A + DD)/femmine
>>> print altezza_media_femmine
1.564
```

l'altezza media della classe è: 1.555 m

```
>>> altezza_media_classe =((altezza_media_femmine*femmine)+(altezza_media_maschi*maschi))/alunni
>>> print altezza_media_classe
1.555
```

STEP4 Esercitiamoci un po' pagina 24

1. Cinque per tre e' uguale a 15.
   Puoi ottenere questo risultato con o senza scatole, ossia le variabili.
   Trova le due soluzioni.

   Soluzione:

```
>>> print 5*3
15
>>> moltiplicazione = 5*3
>>> print moltiplicazione
15
```

2. scrivi tutte le sequenze di istruzioni possibili per visualizzare il messaggio "Buon Compleanno" cinque volte.

   Soluzione:

   - assegno ad una variabile "Buon Compleanno"
   - stampo la variabile *5
   - stampo la variabile+ variabile+ variabile+ variabile+ variabile
   - stampo Buon Compleanno 5 volte

```
>>> saluto = "Buon Compleanno "
>>> print saluto*5
Buon Compleanno Buon Compleanno Buon Compleanno Buon Compleanno Buon Compleanno
>>> print saluto+saluto+saluto+saluto+saluto
Buon Compleanno Buon Compleanno Buon Compleanno Buon Compleanno Buon Compleanno
>>> print "Buon Compleanno Buon Compleanno Buon Compleanno Buon Compleanno Buon Compleanno"
```

3. Scrivi la sequenza di istruzioni per visualizzare il tuo nome e cognome in due stringhe separate.

   Soluzione:

   - assegno ad una variabile il nome
   - assegno ad una variabile il cognome
   - stampo le due variabili separate dalla virgola

```
>>> nome = "Rosanna"
>>> cognome = "Bianchi"
>>> print nome, cognome
Rosanna Bianchi
```

4. Scrivi la sequenza di istruzioni per ottenere il messaggio seguente utilizzando le variabili: l'area del rettangolo e' uguale a 50.

   Soluzione:

   - assegno ad una variabile la stringa "l'area del rettangolo e' uguale a"
   - assegno ad una variabile 50
   - stampo le due variabili separate dalla virgola

```
>>> stringa = "l'area del rettangolo e' uguale a"
>>> cinquanta = 50
>>> print stringa, cinquanta
l'area del rettangolo e' uguale a 50
```

5. Scrivi la sequenza di istruzioni per ottenere il perimetro e l'area di un rettangolo.

Soluzione:
- assegno ad una variabile, il lato maggiore, un valore
- assegno ad una variabile, il lato minore, un valore
- perimetro uguale lato maggiore + lato minore il risultato moltiplicato per due
- stampo il perimetro
- area del rettangolo uguale lato maggiore * lato minore
- stampo l'area

```
>>> latomaggiore = 5
>>> latominore = 3
>>> perimetro = (latominore+latomaggiore)*2
>>> print "perimetro del rettangolo=" , perimetro
16
>>> area = latominore*latomaggiore
>>> print "area del rettangolo =", area
15
```

6. Scrivi le istruzioni per un programma che concateni due variabili stringa (attenti agli spazi) e moltiplichi due variabili numeriche (intere). Infine visualizza il risultato sullo schermo.

Soluzione:
- assegno ad una variabile una stringa
- assegno alla seconda variabile una stringa
- stampo sullo schermo le due variabili e tra una una e l'altra inserisci il +
- assegno ad una variabile un numero intero
- assegno alla seconda variabile un numero intero
- stampo sullo schermo moltiplicando le due variabili

```
>>> zia = "Anna "
>>> zio = "Armando"
>>> print zia + zio
Anna Armando
>>> cinque = 5
>>> sei = 6
>>> print cinque * sei
30
```

7. Scrivi un programma che faccia un disegno con alcuni caratteri; per esempio prova a copiare il programma:

```
print "0  0"
print "   |..."
print " \_/ "
```

e poi scrivine uno tu.

8. Trova l'errore:

a.print ciao+4

b.print "ciao+4"

Soluzione: nella **a** perchè ciao è una scatola che non è definita, non ha un contenuto e non si può sommare a un numero

9. Trova l'errore:

a.
```
6scatola = "che bello il telefonino"
print 6scatola
```

b.
```
farfalla = "cane"
print "è bello il mio", farfalla, "Joe!"
```

Soluzione:
l'errore è nell'esercizio **a** perchè una scatola non può avere un nome che inizia con un numero

10. Trova l'errore:

a.
```
scatola = "viva il calciobalilla"*3
print scatola
```

b.
```
farfalla = "cane"/5
print "è bello il mio", farfalla, "Joe!
```

Soluzione:
l'errore è nell'esercizio **b** perchè non è valida l'operazione "cane"/5

STEP5

Esercitiamoci un po' pagina 29

		Dati:	Operazioni:
		4 bicchieri	dispongo la tovaglia
Esempio	Scrivi l'elenco dei dati e delle	4 piatti	posiziono i 4 piatti
	operazioni necessarie per	4 forchette	posiziono le 4 forchette
	preparare la tavola	4 cucchiai	posiziono i 4 coltelli
		4 coltelli	posiziono i 4 cucchiai
		4 tovaglioli	posiziono i 4 cucchiai
		1 tovaglia	posiziono i 4 tovaglioli

LE SOLUZIONI DEGLI ESERCIZI DA 1 A 7 NON VENGONO SCRITTE QUI PERCHE' MOLTO SEMPLICI, E' SUFFICIENTE SEGUIRE LE INDICAZIONI DELL'ESEMPIO.

Riepilogando...

8. Scrivi tutte le sequenze di istruzioni possibili per visualizzare il messaggio "Ho conosciuto una principessa" cinque volte.

Soluzione:

- assegno ad una variabile "Ho conosciuto una principessa"
- stampo la variabile *5
- stampo la variabile+ variabile+ variabile+ variabile+ variabile
- stampo Ho conosciuto una principessa 5 volte

9. Scrivi l'elenco dei dati, le operazioni e le istruzioni per ottenere che nella scatola1 ci sia 30, nella scatola2 ci sia anni e stampi *hai 30 anni domani* .

Soluzione:
Dati:
scatola1 30
scatola2 "anni"

Operazioni
assegno 30 a scatola1
assegno "anni" a scatola2
stampo "hai", scatola1, scatola2, "domani"

Istruzioni:

```
scatola1 = 30
scatola2 = "anni"
print "hai", scatola1, scatola2, "domani"
```

10. Scrivi la sequenza di istruzioni per ottenere il messaggio seguente utilizzando le variabili: l'area del rettangolo e' uguale a 20 e l'area del triangolo rettangolo è uguale a 20

Soluzione:

```
scat1 = "l'area del "
scat2 = "è uguale"
area1 = 20
area2 = 20
print scat1, "rettangolo", scat2, area1, scat1, "triangolo rettangolo", scat2, area2
```

ESERCITARCI CON PYTHON pagina 36

Esercizio n. 1: scrivi un programma per sommare i primi dieci numeri pari.
Soluzione:
Scrivo il programma:

```
sommamumeripari = 2+4+6+8+10+12+14+16+18+20
print "La somma dei primi 10 numeri pari è". sommamumeripari
```

e eseguendo l'interprete dopo aver salvato il file ottengo:

```
>>>
La somma dei primi 10 numeri pari è 110
```

Esercizio n. 2: scrivi un programma che visualizzi cinque stati e le corrispondenti capitali in colonna.
Soluzione:
Scrivo il programma:

```
print "Italia   ", "Roma"
print "Canada   ", "Ottawa"
print "Venezuela", "Caracas"
print "Egitto   ", "Il Cairo"
print "India    ", "Delhi"
```

e eseguendo l'interprete dopo aver salvato il file ottengo:

```
>>>
Italia       Roma
Canada     Ottawa
Venezuela Caracas
Egitto       Il Cairo
India        Delhi
```

Esercizio n. 3: scrivi il programma che calcola e visualizza il quadrato e il cubo di 3.
Soluzione:
Scrivo il programma:

```
quadratodi3 = 3**2
cubodi3 = 3**3
print "il quadrato di 3 è", quadratodi3
print "il cubo di 3 è", cubodi3
```

e eseguendo l'interprete dopo aver salvato il file ottengo:

```
>>>
il quadrato di 3 è 9
il cubo di 3 è 27
```

Esercizio n. 4: scrivi il programma per calcolare l'area di un quadrato avente il lato uguale a 5 cm.
Soluzione:
Scrivo il programma:

```
areaquadrato=5*5
print "l'area del quadrato è", areaquadrato, "centimetri quadrati"
```

e eseguendo l'interprete dopo aver salvato il file ottengo:

```
>>>
l'area del quadrato è 25 centimetri quadrati
```

Esercizio n. 5: osserva il seguente programma che visualizza un rettangolo:

```
print "*******"
print "*******"
print "*******"
print "*******"
print "*******"
```

Prova a mandarlo in esecuzione. Scrivi quindi un programma che visualizzi un triangolo rettangolo formato da asterischi.

Soluzione:

```
>>>
*******
*******
*******
*******
*******
```

Scrivo il programma per visualizzare un triangolo rettangolo

```
print "*"
print "**"
print "***"
print "****"
print "*****"
print "******"
print "*******"
```

Esercizio n. 6: scrivi un programma che visualizzi un trapezio formato da asterischi.

Soluzione:
Scrivo il programma:

```
print "****"
print "*****"
print "******"
print "*******"
print "********"
print "*********"
print "**********"
```

e eseguendo l'interprete dopo aver salvato il file ottengo:

```
>>>
****
*****
******
*******
********
*********
**********
```

Esercizio n. 7: crea tre scatole: Qui= 6, Quo= 12, Qua= 8
Scrivi un programma che visualizzi la somma del contenuto delle scatole e inserisci il risultato in un'altra scatola.

Soluzione:
Scrivo il programma:

```
Qui= 6
Quo= 12
Qua= 8
somma = Qui+Quo+Qua
print "la somma è", somma
```

e eseguendo l'interprete dopo aver salvato il file ottengo:

```
>>>
```

la somma è 26

Esercizio n. 8: scrivi un programma che visualizzi le potenze del 2.
(*Quest' ultimo esercizio ti sembrerà un po' difficile, ma prova lo stesso a risolverlo, tuffati senza paura nel mare della programmazione!)*

Soluzione:
prima soluzione:
print "le potenze di 2 sono"
print 2**0
print 2**1
print 2**2
print 2**3
print 2**4
print 2**5
	Seconda soluzione che vedremo più avanti:
	print "le potenze di 2 sono"
	i=0
	while i < 6:
	  print i, 2**i
	  i=i+1

Risultato di entrambe le soluzioni:
>>> le potenze di 2 sono
1
2
4
8
16
32

AUTOVALUTARCI pagina 37

1. Che cos'è una stringa? Da cosa può essere composta?

Soluzione:

Una STRINGA e' una serie di caratteri, lettere, numeri o altri simboli, racchiusi tra virgolette, che viene memorizzata dal calcolatore come un unico elemento.

2. Che differenza c'è tra un programma compilato ed uno interpretato?

Soluzione:

Il compilatore: legge il programma di alto livello e lo traduce tutto in linguaggio macchina, prima che il programma possa essere eseguito. In questo caso il programma di alto livello viene chiamato codice sorgente ed il programma tradotto codice oggetto.

L'interprete: legge il programma di alto livello e lo esegue trasformando ogni riga di istruzione in un'azione. L'interprete elabora il programma un po' alla volta, alternando la lettura delle istruzioni all'esecuzione dei comandi che le istruzioni descrivono.

Il programma che traduce il linguaggio intermedio (di alto livello) in binario si chiama proprio INTERPRETE.

3. Python è un programma interprete? Perché?

Soluzione:

Si, Python è un programma interpretato.

Python è considerato un linguaggio interpretato perché i programmi scritti in Python sono eseguiti da un interprete.

STEP7 Prima di proseguire....esercitiamoci un po' pagina 41

Esercizio n. 1
Se CLICK1 = 24 e SLAM1 = 32 come faccio per copiare il contenuto di CLICK1 in SLAM1? E quando l'ho copiato come faccio per rimettere nelle due scatole il contenuto originale? Prova a illustrare i vari passaggi attraverso i quali il calcolatore copia il contenuto di una scatola in un'altra.
Soluzione:
1) devo prima copiare il contenuto di SLAM1 in una scatola di passaggio che chiameremo PARK
2) copio il contenuto di PARK in SLAM1
3) Supponiamo di avere a disposizione due scatole di nome PIPPO e PLUTO, con PIPPO = 5 e PLUTO = 15. Vediamo che cosa succede quando diamo al computer un'istruzione come questa:
PIPPO = PLUTO
Il computer fa le seguenti operazioni:
1. mette a disposizione la scatola di nome PIPPO
2. mette a disposizione la scatola di nome PLUTO
3. legge il contenuto di PLUTO e lo mette nella scatola PIPPO
4. Per rimettere il contenuto originale devo effettuare nuovamente le operazioni 1. 2. e 3.

Esercizio n. 2
Scrivi un programma per scambiare il contenuto delle due scatole seguenti:
PLUTO = "America" PIPPO = "Asia"
Soluzione:

```
PLUTO = "America"
PIPPO = "Asia"
print "Il contenuto di PLUTO E PIPPO è", PLUTO, PIPPO
PARK = PIPPO
PIPPO = PLUTO
PLUTO = PARK
print "Ora il contenuto di PLUTO e PIPPO è ", PLUTO, PIPPO
```

Esercizio n. 3
La scatola STAR contiene il numero 8. Come posso ordinare al computer di svuotarla e di mettere 15 al posto di 8?
Soluzione:

```
STAR = 8
STAR = 15
```

Esercizio n. 4
La scatola BLAM contiene il numero 2. Scrivi il programma che calcola il cubo del contenuto e lo mette nella scatola BLAM3.
Soluzione:

```
BLAM = 2
BLAM3 = BLAM **3
print "Il cubo di", BLAM, "è ", BLAM3
```

STEP7 Esercitiamoci un po' pagina 45

1. Scrivi un programma che chiede un numero e ne calcola il quadrato e il cubo e li visualizza sullo schermo.
Sluzione:

```
numero = input ("Scrivi  un numero  ")
quadrato = numero ** 2
cubo = numero ** 3
print "Il quadrato di ", numero, "è ", quadrato
print "Il cubo di ", numero, "è ", cubo
```

2. Scrivi un programma che aggiunge 7 a qualunque numero inserito e visualizza il risultato sullo schermo.
Soluzione:

```
numero = input ("Scrivi  un numero  ")
piusette = numero + 7
print "Se sommiamo 7 a ", numero, "si ottiene", piusette
```

3. Scrivi un programma che chiede due numeri, li somma e visualizza il risultato.
Soluzione:

```
numerouno = input ("Scrivi il primo numero  ")
numerodue = input ("Scrivi il secondo numero  ")
somma = numerouno + numerodue
print "Se sommiamo ", numerouno, "e ", numerodue, "si ottiene", somma
```

4. Scrivi il programma per calcolare l'area di qualunque rettangolo chiedendo all'utilizzatore la base e l'altezza.
Soluzione:

```
base = input ("Scrivi il valore della base del rettangolo  ")
altezza = input ("Scrivi il valore dell'altezza del rettangolo  ")
area = base*altezza
print "L'area del rettangolo di base ", base, "e altezza ", altezza, "è ", area
```

5. Scrivi il programma che chieda tre numeri e ne visualizzi sia la somma sia il prodotto.
Soluzione:

```
uno = input ("Scrivi il primo numero  ")
due = input ("Scrivi il secondo numero  ")
tre = input ("Scrivi il terzo numero  ")
somma = uno + due + tre
prodotto = uno * due * tre
print "Se sommiamo ", uno, "e ", due, "e ", tre, "si ottiene ", somma
print "Se moltiplichiamo ", uno, "e ", due, "e ", tre, "si ottiene ", prodotto
```

6. Scrivi il programma che calcola la metà e il doppio di qualunque numero inserito dall'utente, poi visualizza i risultati.
Soluzione:

```
numero = input ("Scrivi il numero  ")
metà = numero/2.
doppio = numero*2
print "La metà di ", numero, "è ", metà
```

```
print "Il doppio di ", numero, "è ", doppio
```

7. Scrivi il programma che chiede la misura del lato di un quadrato e ne calcola l'area, poi visualizza il risultato.

Soluzione:

```
lato = input("Inserisci il valore del lato del quadrato ")
area = lato * lato
print "L'area del quadrato di lato ", lato, "è ", area
```

8. Scrivi il programma che calcola il perimetro del cortile della scuola che è un rettangolo i cui lati misurano rispettivamente 45 m e 65 m e visualizza il risultato. Quindi calcola il perimetro di ogni rettangolo per il quale l'operatore inserisca la misura della base e dell'altezza.

Soluzione:

```
perimetro = 45*2 + 65*2
print "Il perimetro del cortile è ", perimetro
base = input("Inserisci il valore della base ")
altezza = input("Inserisci il valore dell'altezza ")
perimetro = (base+altezza)*2
print "Il perimetro del rettangolo di base ",base, "e altezza ",altezza,"è ", perimetro
```

9. Scrivi un programma che chiede tre numeri, ne calcola la somma, la somma dei quadrati e il quadrato della somma. Infine, visualizza i risultati.

Soluzione:

```
uno = input ("Scrivi il primo numero  ")
due = input ("Scrivi il secondo numero  ")
tre = input ("Scrivi il terzo numero  ")
somma = uno + due + tre
sommaquadrati = uno**2 + due**2 + tre**2
quadratosomma = (uno + due + tre)**2
print "Se sommiamo ", uno, "e ", due, "e ", tre, "si ottiene", somma
print "La somma dei quadrati dei 3 numeri è ", sommaquadrati
print "Il quadrato della somma dei 3 numeri è ", quadratosomma
```

Adesso facciamo uno stop pagina 46

Esercizio	errore/correzione	punti
Es. 1: stampa il nome del tuo cantante preferito. cantante = raw_input ("Scrivi il nome del cantante preferito: ") print "Il mio cantante preferito e' ", cantant	cantant/cantante	
Es. 2: input di numeri e stringhe Primonumero= input ("Scrivi il primo numero: ") Secondonumero= input ("Scrivi il secondo numero: ") Nome = raw_input ("Scrivi il tuo nome: ") Cognome = raw_input ("Scrivi il tuo cognome: ") Print nome , cognome, "primonumero", "per", secondonumero, "uguale", primonumero*secondonumero	Print/print	
Es. 3: domanda di filosofia printt " Sai in quale anno e' nato Socrate" sino = raw_input ("si o no") print "Ma certo, nell'anno 469 prima di Cristo"	printt/print	
Es. 4: disegno un quadrato print "*****" print "* *" print "* *" print "* *" print "*****"	print "* *"/print "* *"	
Es. 5: divisione con resto primo = input ("Inserisci il primo numero") secondo = input ("Inserisci il secondo numero") print primo, "diviso", seco ndo,"si ottiene",primo/secondo print "il resto della divisione e' ", primo % secondo	seco ndo/secondo	

STEP8 Esercitiamoci un po' pagina 56

Esercizio n. 1
Che cosa significano le due parole if e else?
Soluzione:
if in inglese significa SE
else in inglese significa ALTRIMENTI

Esercizio n. 2
Quale parola introduce un lavoro che deve essere svolto come conseguenza di una condizione?
Soluzione:
ALLORA

Esercizio n. 3
Che cosa fa il computer quando non e' soddisfatta la condizione introdotta da if?
Soluzione:
quando non e' soddisfatta la condizione introdotta da if passa all'istruzione successiva del programma.

Esercizio n. 4
Scrivi un esempio di scelta condizionata in cui compaia la congiunzione "e" (and), una in cui compaia la congiunzione "oppure" (or) e uno in cui compaia la negazione "non" (not).
Soluzione:
1) Se ho i pantalocini e le scarpe da calcio allora oggi gioco nella squadra di calcio
2) se sai cantare oppure suonare uno strumento allora entri nel coro
3) se non sai nuotare non fai il bagno

Esercizio n. 5
Scrivi il programma relativo alla seguente scelta condizionata:
SE squilla il telefono ALLORA vai a rispondere.
Soluzione:

```
telefono = raw_input("Squilla il telefono? SI/NO\n")
if telefono == 'SI':
   print 'Vai a rispondere'
```

Esercizio n. 6
Spiega il significato delle seguenti istruzioni.

```
a) if numero <> 20:
        print numero
b) if qui <20:
        quo = 30
```

Soluzione:
a) se il contenuto della scatola numero è diverso da 20 allora:
stampa il contenuto della scatola numero
b) se il contenuto della scatola qui è minore di 20 allora:
alla scatola quo viene assegnato il valore 30

Esercizio n. 7

Scrivi il programma per controllare se un numero e' positivo.

Soluzione:

```
numero = input ("indica un numero")
if numero > 0:
        print numero, " e' positivo"
```

Esercizio n. 8

Scrivi il programma che controlla il risultato di una addizione, dati due numeri.

Soluzione

```
print "vediamo come te la cavi con le addizioni"
numero1 = input ("scrivi il primo addendo ")
numero2 = input ("scrivi il secondo addendo ")
somma = input ("scrivi il risultato della somma ")
if somma = numero1+numero2:
        print "bravo! risposta esatta"
else:
        print "la risposta e' sbagliata!"
```

Esercizio n. 9

Scrivi il programma che distingue i numeri positivi e i numeri negativi.

Soluzione

```
num = input ("Inserisci un numero (<> da 0)")
if num > 0 :
        print num, " e' positivo"
else:
        print num, " e' negativo"
```

Esercizio n. 10

Scrivi un programma che dati due numeri, li visualizza in ordine crescente (o decrescente).

Soluzione

```
num1 = input ("scrivi il primo numero")
num2 = input ("scrivi il secondo numero")
print "Ordine crescente:"
if num1 < num2 :
        print num1, " ", num2
else:
        print num2, " ", num1
```

Esercizio n. 11

Scrivi un programma che dia consigli per i vestiti se piove, se nevica e se fa freddo.

Soluzione

```
pioggia = raw_input ("piove ? (s/n))
neve = raw_input ("nevica ? (s/n))
freddo = raw_input ("fa freddo ? (s/n))
if pioggia = 's' or neve = 's'
        print "prendi l'ombrello"
if freddo = 's'
        print "metti il cappotto"
```

```
if pioggia = 's' and not "freddo ='s'
        print "metti la giacca"
if not pioggia = 's' and not neve = 's' and not freddo = 's'
        print "vestiti leggero"
```

Esercizio n. 12
Scrivi il programma che chiede di indicare l'autore di un libro, se e' sbagliato stampa "risposta errata", se e' corretto stampa "risposta esatta" e prosegue a chiedere un altro autore per un altro libro (puoi ripeterlo quante volte vuoi)
Soluzione

```
print "scrittori"
risposta = raw_input ("chi e' l'autore dei Promessi Sposi (cognome) ?")
if risposta == "manzoni":
        print "Bravo! risposta esatta. Proseguiamo"
else:
        print "risposta errata"
risposta = raw_input (chi e' l'autore dell'Orlando Furioso (cognome)? ")
if risposta == "ariosto":
        print "Bravo! risposta esatta. Proseguiamo"
else:
        print "risposta errata"
risposta = raw_input (chi e' l'autore della Divina Commedia (cognome)? ")
if risposta == "alighieri":
        print "Bravo! risposta esatta."
else:
        print "risposta errata"
```

Per esercitarti un po' di più puoi scrivere lo stesso programma per i seguenti argomenti:
a) calciatori e squadre di calcio
b) nazioni e capitali
c) città e nome degli abitanti.

Esercizio n. 13
Prova a scrivere un programma per questo indovinello: "a scuola , se ci sei non la fai, se la fai non ci sei: cos'e'?" che stampa le due frasi "bravo, hai indovinato!" e "sbagliato, riprova!"
Soluzione

```
risposta = raw_input ("a scuola , se ci sei non la fai, se la fai non ci sei: cos'e'? ")
if risposta == "assenza" or risposta == "assenza" :
        print "bravo, hai indovinato!"
else:
        print "sbagliato, riprova!"
```

STEP9 ESERCITARCI CON PYTHON pagina 71

1. Scrivi un programma utilizzando le funzioni che chiede due numeri e poi li somma tra loro.
Soluzione:

```
def somma(uno, due):
   somma = uno + due
   print "La somma di ", uno, "più ", due, "è ", somma
uno = input("Inserisci il primo numero ")
due = input("Inserisci il secondo numero ")
somma(uno,due)
```

2. Scrivi un programma utilizzando le funzioni che concateni due stringhe (per esempio due nomi Sandro e Silvia).
Soluzione:

```
def somma(uno, due):
   somma = stringauno + stringadue
   print "La concatenazione di ", stringauno, "e ", stringadue, "è ", somma
stringauno = raw_input("Inserisci il primo nome ")
stringadue = raw_input("Inserisci il secondo nome ")
somma(stringauno,stringadue)
```

3. Scrivi un programma utilizzando le funzioni che visualizzi i colori dell'arcobaleno.
Soluzione:

```
def arcobaleno():
   print "I colori dell'arcobaleno sono: "
   print "rosso, arancione, giallo, verde, azzurro, indaco e violetto. "
arcobaleno()
```

4. Trova l'errore:

```
def facciamofesta(musica):
   torte = 5
   print "Stasera ci sono",torte,"torte, e la musica di", musica
musica = raw_input("Qual'è il tuo cantante preferito?")
facciamofesta()
```

Soluzione:
La chiamata alla funzione facciamofesta() deve avere un argomento:

```
facciamofesta(musica)
```

5. Scrivi un programma utilizzando le funzioni che chiede qual è il tuo primo piatto preferito e il programma risponde "A me piacciono i peperoni in carpione" e poi chiede qual è il secondo preferito e risponde sempre "A me piacciono i peperoni in carpione".
Soluzione:

```
def piattopreferito(piatto):
   print "A me piacciono i peperoni in carpione"
piatto = raw_input("Qual'è il tuo primo piatto preferito? ")
piattopreferito(piatto)
piatto = raw_input("Qual'è il tuo secondo piatto preferito? ")
piattopreferito(piatto)
```

6. Scrivi un programma utilizzando le funzioni e poi un altro programma senza utilizzare le funzioni che chiede un numero ne fa la somma con 22, lo moltiplica per 5 e poi lo divide per 8 e poi fa la stesse operazioni per un secondo numero.

Soluzione con la funzione:

```
def calcolo(numero):
   somma = numero + 22
   moltiplicazione = numero * 5
   divisione = numero / 8
   print "il risultato è", somma, moltiplicazione,
                                              divisione
numero = input("Inserisci un numero ")
calcolo(numero)
numero = input("Inserisci un secondo numero ")
calcolo(numero)
```

Soluzione senza la funzione:

```
numero = input("Inserisci un numero ")
somma = numero + 22
moltiplicazione = numero * 5
divisione = numero / 8
print "il risultato è", somma, moltiplicazione, divisione
numero = input("Inserisci un secondo numero ")
somma = numero + 22
moltiplicazione = numero * 5
divisione = numero / 8
print "il risultato è", somma, moltiplicazione, divisione
```

7. Scrivi un programma utilizzando le funzioni che chiede 2 numeri e visualizzi la somma, il valore medio e visualizzi il minimo tra i due.

Soluzione:

```
def calcolo(numerouno, numerodue):
   somma = numerouno + numerodue
   valormedio = somma / 2.
   minimo = numerouno
   if numerouno > numerodue:
      minimo = numerodue
   print "La somma di ", numerouno, numerodue, "è ", somma
   print "Il valor medio di ", numerouno, numerodue, "è ", valormedio
   print "La numero minimo di ", numerouno, numerodue, "è ", minimo
numerouno = input("Inserisci un numero ")
numerodue = input("Inserisci un secondo numero ")
calcolo(numerouno, numerodue)
```

8. Scrivi un programma utilizzando la funzione che calcoli l'area e il volume di un parallelepipedo rettangolo.

Soluzione:

```
def calcolo(uno, due, tre):
   area = 2*(uno*due + due*tre + uno*tre)
   volume = uno * due * tre
   print "La superficie del parallelpipedo di dimensioni ", uno, due, tre, "è ", area
   print "il volume del parallelpipedo di dimensioni ", uno, due, tre, "è ", volume
uno = input("Inserisci la prima dimensione del parallelpipedo rettangolo ")
due = input("Inserisci la seconda dimensione del parallelpipedo rettangolo ")
tre = input("Inserisci la terza dimensione del parallelpipedo rettangolo ")
calcolo(uno, due, tre)
```

9. Scrivi un programma che chiede di inserire il quadrato di un numero qualunque, trova il numero e lo visualizza.

Soluzione:

```
import math
numero = input ("Scrivi il quadrato di un numero  ")
radice = math.sqrt(numero)
print "La radice quadrata di ", numero, "è ", radice
```

AUTOVALUTARCI pagina 73

1. Che differenza c'è tra input e raw_input? A cosa servono?

La differenza consiste nel fatto che raw_input raccoglie i caratteri immessi dall'utilizzatore e li presenta sotto forma di stringa, mentre input li raccoglie e cerca di interpretarli come numero.
input e raw_input servono a fare in modo che l'operatore possa inserire i dati utilizzando la tastiera

2. Quali "operatori logici" conosci?

Abbiamo studiato gli operatori:
or, and, not

3. Qual è l'esatta definizione di funzione? Fai un esempio di funzione.

Una funzione è un pezzo di programma cui viene dato un nome. Ad esempio:
def interrogazione (domanda, risposta_esatta)
Questa linea indica all'interprete che vogliamo definire un blocco di istruzioni che potrà essere eseguito a richiesta in un qualunque altro punto del programma semplicemente scrivendo il suo nome (interrogazione).

STEP10 Esercitiamoci un po' pagina 89

10.17 Un contadino vuole attraversare un fiume portando con sè un lupo, una capra ed un grosso cavolo, ma non dispone che di una barchetta così piccola da non poter contenere che lui e una sola di tali cose. Cosa dovrà fare per evitare che il lupo mangi la capra o questa il cavolo? Risolvi l'indovinello e poi…
Scrivi il programma che lo risolva.

Soluzione:

Il contadino dovrà prima attraversare il fiume con la capra, lasciando sull'altra sponda il lupo che non può mangiare il cavolo. Quindi dovrà tornare indietro per caricare il cavolo che trasporterà dall'altra parte. Qui la capra, che non potrà mangiare il cavolo appena trasportato perché controllata dal contadino, dovrà essere riportata indietro dal contadino che la sbarcherà nuovamente sull'altra riva, per poi caricare sul traghetto il lupo. Una volta sbarcato il lupo potrà rimanere tranquillamente da solo con il cavolo senza mangiarlo e così il contadino potrà tornare tranquillamente indietro per prendere anche la capra, completando così il traghettamento di lupo, capra e cavolo.

```
def risposta(barca, soluzione, string):
   while barca != soluzione:
      print 'Ritenta'
      barca = raw_input (string)
print 'Un contadino vuole attraversare un fiume portando con sè un lupo, una capra ed un grosso cavolo, ma non dispone che di una barchetta così piccola da non poter contenere che lui e una sola di tali cose. Cosa dovrà fare per evitare che il lupo mangi la capra o questa il cavolo?'
string = 'Chi porti al di là del fiume? lupo, capra, cavolo '
barca = raw_input (string)
soluzione = 'capra'
risposta(barca, soluzione, string)
string = 'Ora chi porti indietro? nessuno, capra'
barca = raw_input (string)
soluzione = 'nessuno'
risposta(barca, soluzione, string)
string ='Ora chi porti al di là del fiume? lupo, cavolo '
barca = raw_input (string)
soluzione = 'cavolo'
risposta(barca, soluzione, string)
string ='Ora chi porti indietro? nessuno, capra, cavolo '
barca = raw_input (string)
soluzione = 'capra'
risposta(barca, soluzione, string)
string ='Ora chi porti al di là del fiume? lupo, capra'
barca = raw_input (string)
soluzione = 'lupo'
risposta(barca, soluzione, string)
string = 'Ora chi porti indietro? nessuno, lupo, cavolo '
barca = raw_input (string)
soluzione = 'nessuno'
risposta(barca, soluzione, string)
string ='Ora chi porti al di là del fiume? capra'
barca = raw_input (string)
soluzione = 'capra'
```

risposta(barca, soluzione, string)

10.18 Scrivi un programma che stampi:

```
bravo
bravo
peperone
bravo
bravo
peperone peperone
bravo
bravo
peperone peperone peperone
```

Soluzione:

```
h=1
while h<4:
   i=0
   while i<2:
         print 'bravo'
         i=i+1
   print 'peperone '*h
   h=h+1
```

10.19 Scrivi un programma che stampi:

```
5
4
3
2
1
```

Soluzione:

```
i=5
while i>0:
      print i
      i=i-1
```

10.20 Scrivi un programma che stampi:

```
5
4
3
2
1
2
3
4
5
```

Soluzione:

```
i=5
while i > 0:
      print i
      i=i-1
while i < 4:
      print i+2
      i=i+1
```

10.21 Scrivi un programma che simuli un semaforo. Ricorda che la sequenza del semaforo è verde – giallo – rosso – verde: verde 6 cicli, giallo 2 cicli, rosso 4 cicli

Soluzione:

```
h=1
while h <= 2:
   i=j=n=1
   while i <= 6:
      print 'verde'
      i=i+1
   while j <= 2:
      print 'giallo'
      j=j+1
   while n <=4 :
      print 'rosso'
```

```
      n=n+1
   h=h+1
```

10.22 Scrivi un programma utilizzando un ciclo while e una funzione per visualizzare la tabellina del 2

Soluzione:

```
def tabellinadel2():
   i=1
   while i < 11:
      print i*2
      i=i+1
tabellinadel2()
```

10..23 Scrivi un programma utilizzando un ciclo while e una funzione che chiede: "Perché il linguaggio di programmazione di chiama Python?" e ci sono 3 possibilità:

- Il nome viene da pitone
- Il nome viene dai comici inglesi "Monty Python"
- Il nome viene da un personaggio di Harry Potter

Soluzione:

```
def pitone(risposta):
   if risposta == '2':
      print 'Bravo, il nome viene dai comici inglesi Monty Python'
   while risposta != '2':
         print 'Hai sbagliato!'
         print '1 Il nome viene da pitone '
         print '2 Il nome viene dai comici inglesi Monty Python '
         print '3Il nome viene da un personaggio di Harry Potter'
         risposta = raw_input('scegli 1,2 o 3 ')
         if risposta == '2':
            print 'Bravo, il nome viene dai comici inglesi Monty Python'
print 'Perché il linguaggio di programmazione di chiama Python?'
print 'Ci sono 3 possibilità:'
print '1 Il nome viene da pitone '
print '2       Il nome viene dai comici inglesi Monty Python '
print '3Il nome viene da un personaggio di Harry Potter'
risposta = raw_input('scegli 1,2 o 3 ')
pitone(risposta)
```

10.24 Con un ciclo while stampa:

```
1              2  3  4  5
2              4  6  8 10
3   6  9 12 15
4   8 12 16 20
5              10 15 20 25
```

Soluzione:

```
i=1
j=i+1
l=i+2
m=i+3
```

```
n=i+4
while i <= 5:
   print i, j*i, l*i, m*i, n*i
   i=i+1
```

10.25 Trova l'errore

```
i=5
while i < 5
print "Che bella partita di calcio"
i=i+1
```

Soluzione:
while i < 5 deve essere seguito da :
while i < 5:

10.26 Trova l'errore

```
def tabellinadel4():
   i=1
   while i <=10:
      print i, i*4
      i=i+1
```

tabellina del 4(4)
Soluzione:
la chiamata alla funzione tabellina del 4(4) deve essere scritta tabellinadel4()

10.27 Indovina un numero! Scrivi un programma per indovinare un numero.
Soluzione:

```
def indovina(x,y):
   i=1
   while i <= 3:
      if x == y:
         print "Bravo hai indovinato!"
         i = 3
      elif x!= y and i < 3:
         y = input ("Hai sbagliato, ritenta ancora\n")
      else:
         print "Ritenta un altro giorno!"
      i=i+1
x = 8
y = input ("prova ad indovinare un numero compreso tra 1 e 10. Hai 3 tentativi\n")
indovina(x,y)
```

10.28 Nell'esercizio 10.22 dovevi stampare i multipli di 2; ora prova a generalizzare, scrivendo un programma che calcoli i multipli di un qualsiasi numero intero (ricordati delle funzioni)
Soluzione:

```
def tabellina(x):
   i=1
   while i < 11:
      print i*x
      i=i+1
```

```
x = input ("inserisci il numero di cui vuoi visualizzare la tabellina \n")
tabellina(x)
```

Rifletti ancora:
Cosa ottieni se chiami la funzione con il parametro 2?
Soluzione:
la tabellina del 2

Come farai a stampare la Tavola Pitagorica?
Soluzione:

```
def tavolapitagorica():
    i=1
    while i <= 10:
        print i, 2*i, 3*i, 4*i, 5*i, 6*i, 7*i, 8*i, 9*i, 10*i
        i=i+1

tavolapitagorica()
```

STEP11 Esercitiamoci un po' pagina 101

1. Scrivi un programma che analizzi le seguenti stringhe e per ognuna stampi le lettere una ad una: banana, cipolla, bambino, finestra, girotondo.

Soluzione:

```
stringa = "banana"
for Lettera in stringa:
   print Lettera
```

2. Scrivi un programma che sommi tutti gli elementi di una lista data.

Soluzione:

```
numeri_pari = [2,4,6,8,10,12,14,16,18,20]
somma = 0
for numero in numeri_pari:
      somma = somma + numero
print "La somma e` : ",somma
```

3. Scrivi un programma che usi il concatenamento e un ciclo for per generare una lista di valori nei quali gli elementi appaiono in ordine alfabetico. Per esempio:

Soluzione:

```
Prefissi = "BCDRST"
Suffisso = "OMA"
for Lettera in Prefissi:
  print Lettera + Suffisso
```

4. Scrivi un programma che stampi la "serie di Fibonacci". Cioè, la somma di due elementi definisce l'elemento successivo (1,2,3,5,8,13…..)

Soluzione:

```
a = 0
b = 1
indice = 0
while indice < 20:
   indice = indice + 1
   old_a = a
   old_b = b
   a = old_b
   b = old_a + old_b
   print old_a,
```

5. Scrivi un programma per trovare i numeri primi in una lista di numeri che va da 2 a 20.

Soluzione:

```
for n in range(2, 20):
    for x in range(2, n):
       if n % x == 0:
          print n, 'è uguale a', x, '*', n/x
          break
    else:
```

```
    print n, 'è un numero primo'
```

L'istruzione break causa l'uscita immediata dal ciclo for o while.

6. Scrivi un programma che conta il numero di volte in cui la lettera 'a' compare in una stringa, usando un contatore.

Soluzione:

```
parola = "anabattista"
Conteggio = 0
for Carattere in parola:
  if Carattere == 'a':
    Conteggio = Conteggio + 1
print Conteggio
```

7. Scrivi un programma che esamini una lista di numeri e conti il numero di volte in cui uno di questi numeri cade in un determinato intervallo, ossia è superiore a un valore indicato ed è inferiore ad un altro valore assegnato. Non è molto diverso dal programma precedente che conta le lettere.

Soluzione:

```
lista_numeri = [5, 3, 7, 6, 8, 12, 4, 9, 11, 23, 16]
Conteggio = 0
inferiore = 5
superiore = 20
for numero in lista_numeri:
    if inferiore < numero and numero < superiore:
        Conteggio = Conteggio + 1
print Conteggio
```

8. Prova a incapsulare il codice del programma in una funzione chiamata "intervallo" che ha come parametri la lista da controllare ed i valori Inferiore e Superiore.

Soluzione:

```
def intervallo(numeri, inferiore, superiore):
  Conteggio = 0
  for numero in numeri:
    if inferiore <numero and numero < superiore:
      Conteggio = Conteggio + 1
  return Conteggio
```

9. Prova a svolgere l'esercizio 11.4 usando il ciclo FOR.

Soluzione:

```
a = 1
b = 1
for c in range(1,10):
        print a,
        n = a + b
        a = b
        b = n
```

fare la verifica finale pagina 124

PRIMO ESERCIZIO

I numeri di Martina. Ordina un elenco di numeri in ordine decrescente

Soluzione:

```
# questo programma ordina in ordine decrescente un elenco di numeri inseriti dall'utente.

numeri = input('Quanti numeri vuoi ordinare?\n')
a = [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
i=0
print 'inserisci i numeri che vuoi ordinare decrescente (max 15)\n'
while i < numeri:
    print 'inserisci il numero', i+1
    a[i] = input('')
    i=i+1
i=0
while i < numeri:
    j=0
    while j < numeri-1 :
        if a[j] < a[j+1]:
            scambio = a[j+1]
            a[j+1] = a[j]
            a[j] = scambio
        j=j+1
    i=i+1
print 'i numeri ordinati in ordine decrescente sono'
i=0
while i < numeri:
    print a[i]
    i=i+1
```

SECONDO ESERCIZIO

Il ciao di Clotilde: utilizzando la funzione time.sleep di pygame visualizza la lettera C poi la I, a seguire la A e la O, facendo in modo che appaiano una dopo l'altra a distanza di alcuni secondi.

Soluzione:

```
# il ciao di Clotilde
import pygame
import math
import time
pygame.init()

def ciao():
        surface.fill ((0,0,0))
        pygame.draw.line(surface, (255,255,0),(60,110), (120, 110))
        pygame.draw.line(surface, (255,255,0),(60,110), (60,210))
        pygame.draw.line(surface, (255,255,0),(60,210), (120, 210))
```

```
        pygame.display.update()
        time.sleep(1)
        surface.fill ((0,0,0))
        pygame.draw.line(surface, (255,255,0),(140,110), (140,210))
        pygame.display.update()
        time.sleep(1)
        surface.fill ((0,0,0))
        pygame.draw.line(surface, (255,255,0),(160,110), (220, 110))
        pygame.draw.line(surface, (255,255,0),(160,160), (220, 160))
        pygame.draw.line(surface, (255,255,0),(160,110), (160,210))
        pygame.draw.line(surface, (255,255,0),(220,110), (220,210))
        pygame.display.update()
        time.sleep(1)
        surface.fill ((0,0,0))
        pygame.draw.line(surface, (255,255,0),(240,110), (300,110))
        pygame.draw.line(surface, (255,255,0),(240,210), (300,210))
        pygame.draw.line(surface, (255,255,0),(240,110), (240,210))
        pygame.draw.line(surface, (255,255,0),(300,110), (300,210))
        pygame.display.update()
        time.sleep(1)
surface=pygame.display.set_mode((600,600))
i=0
while i < 2:
        ciao()
        i=i+1
```

TERZO ESERCIZIO

L'orologio di Angelo: utilizzando pygame, cerchi e segmenti, disegna un orologio rotondo con una lancetta che cambi posizione di 90° ogni 15''. Ad esempio, l'orologio sarà un cerchio e la lancetta un segmento che cambierà posizione di 90° ogni 15 secondi.
(Ricordati di misurare il tempo corretto per la funzione time.sleep)

Soluzione:

```
# L'orologio di Angelo

import pygame
import math
import time

pygame.init()

def orologio():
        surface.fill ((0,0,0))
        pygame.draw.circle(surface,(200,0,0),(95,100),60,1)
        pygame.draw.line(surface, (255,255,0),(95,100), (95, 50))
        pygame.display.update()
        time.sleep(2)
        surface.fill ((0,0,0))
        pygame.draw.circle(surface,(200,0,0),(95,100),60,1)
        pygame.draw.line(surface, (255,255,0),(95,100), (145, 100))
```

```
        pygame.display.update()
        time.sleep(2)
        surface.fill ((0,0,0))
        pygame.draw.circle(surface,(200,0,0),(95,100),60,1)
        pygame.draw.line(surface, (255,255,0),(95,100), (95, 150))
        pygame.display.update()
        time.sleep(2)
        surface.fill ((0,0,0))
        pygame.draw.circle(surface,(200,0,0),(95,100),60,1)
        pygame.draw.line(surface, (255,255,0),(95,100), (45, 100))
        pygame.display.update()
        time.sleep(2)
surface=pygame.display.set_mode((600,600))
pygame.draw.circle(surface,(200,0,0),(95,100),60,1)
i=0
while i < 5:
        orologio()
        i=i+1
```

Alla fine dei 12 passi.... pagina 126

Qui di seguito troverai il listato del programma per realizzare il gioco del TRIS. Non è che uno dei tanti modi possibili per risolvere il problema perché, come ormai avrai capito, la soluzione dei problemi nella programmazione non è mai univoca. Quindi, se vuoi diventare un "vero" programmatore, non ti rimane che trovare le altre soluzioni!

```
#Programma per giocare a TRIS

#Funzione che visualizza la scacchiera

def scacchiera ():
     print '\t 1 \t 2 \t 3 \n'
     print 'a \t', a1, '\t', a2, '\t', a3, '\n'
     print 'b \t', b1, '\t', b2, '\t', b3, '\n'
     print 'c \t', c1, '\t', c2, '\t', c3, '\n'

#Funzione che effettua il controllo della vittoria

def vittoria (n):
     if ((a1 == n and a2 == n and a3 == n) or (b1 == n and b2 == n and b3 == n) or
        (c1 == n and c2 == n and c3 == n) or
       (a1 == n and b1 == n and c1 == n) or (a2 == n and b2 == n and c2 == n) or
       (a3 == n and b3 == n and c3 == n) or
       (a1 == n and b2 == n and c3 == n) or (a3 == n and b2 == n and c1 == n)):
          EsisteVincitore = 'si'
     else:
          EsisteVincitore = 'no'
     return EsisteVincitore

# Inizializzo la scacchiera mettendo  x nelle 9 caselle
a1 = 'x'
a2 = 'x'
a3 = 'x'
b1 = 'x'
b2 = 'x'
b3 = 'x'
c1 = 'x'
c2 = 'x'
c3 = 'x'

scacchiera ()            # visualizzo la scacchiera

partitafinita = 'no'     # inizializzo la variabile che indica quando la partita Ã¨
finita
nmosse = 0             # inizializzo la variabile nmosse che conta il numero delle
mosse fatte dai giocatori

print 'Il primo giocatore sarÃ  N \n'

while nmosse < 9 :
               print 'Muova N \n'
               mossafatta = 'no'
                  x = raw_input ('Dove vuoi scrivere N: a1, a2, a3, b1, b2, b3, c1,
                                                                    c2, c3? \n')
               if x == 'a1' and a1 == 'x':
                    a1 = 'N'
                    mossafatta = 'si'
               elif x == 'a2' and a2 == 'x':
```

```
     a2 = 'N'
     mossafatta ='si'
elif x == 'a3' and a3 == 'x':
     a3 = 'N'
     mossafatta ='si'
elif x == 'b1' and b1 =='x':
     b1 = 'N'
     mossafatta ='si'
elif x == 'b2' and b2 == 'x':
     b2 = 'N'
     mossafatta ='si'
elif x == 'b3' and b3 == 'x':
     b3 = 'N'
     mossafatta ='si'
elif x == 'c1' and c1 == 'x':
     c1 = 'N'
     mossafatta ='si'
elif x == 'c2' and c2 == 'x':
     c2 = 'N'
     mossafatta ='si'
elif x == 'c3' and c3 == 'x':
     c3 = 'N'
     mossafatta ='si'

if mossafatta == 'no':
     print 'SQUALIFICATO!'
     partitafinita = 'si'
     nmosse = 10

scacchiera ()
nmosse = nmosse + 1
partitafinita = vittoria ('N')

if partitafinita == 'si':
     print 'Ha vinto N \n'
     nmosse = 10
elif   nmosse == 9:
     print 'PAREGGIO'
     nmosse = 10

if partitafinita == 'no' and nmosse < 9:
     nmosse = nmosse +1
     mossafatta = 'no'
     print 'La mossa tocca a R \n'
       x = raw_input ('Dove vuoi scrivere R: a1, a2, a3, b1, b2, b3,
                                                    c1, c2, c3? \n')
     if x == 'a1' and a1 == 'x':
          a1 = 'R'
          mossafatta = 'si'
     elif x == 'a2'and a2 == 'x':
          a2 = 'R'
          mossafatta = 'si'
     elif x == 'a3'and a3 == 'x':
          a3 = 'R'
          mossafatta = 'si'
     elif x == 'b1'and b1 == 'x':
          b1 = 'R'
          mossafatta = 'si'
     elif x == 'b2'and b2 == 'x':
          b2 = 'R'
```

```
     mossafatta = 'si'
elif x == 'b3'and b3 == 'x':
     b3 = 'R'
     mossafatta = 'si'
elif x == 'c1'and c1 == 'x':
     c1 = 'R'
     mossafatta = 'si'
elif x == 'c2'and c2 == 'x':
     c2 = 'R'
     mossafatta = 'si'
elif x == 'c3'and c3 == 'x':
     c3 = 'R'
     mossafatta = 'si'

if mossafatta == 'no':
     print 'SQUALIFICATO!'
     partitafinita = 'si'
     nmosse = 10

scacchiera ()
partitafinita = vittoria ('R')
if partitafinita == 'si':
     print 'Ha vinto R \n'
     nmosse = 10
```

INDICE