Aurora Martina
Angelo Raffaele Meo
Clotilde Moro
Mario Scovazzi

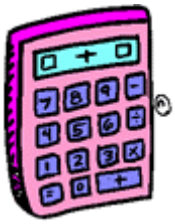
Passo dopo passo impariamo a programmare con PYTHON

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 infatti non si possono usare le parentesi quadre e quelle graffe e si deve fare molta attenzione ad usare le parentesi tonde nel modo giusto e **iniziare 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:

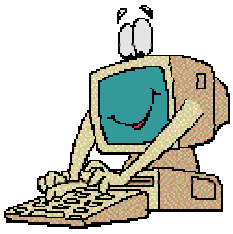
>>>2*(5-2) **6**	>>>(1+1)**(7-4) **8**	>>>2**1+1 **3**
>>>2*5-2 **8**	>>>2/3 **0**	>>>2/3-1 **-1**

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' .

STEP 2

Ovvero passo dopo passo impariamo a programmare

Le Scatole

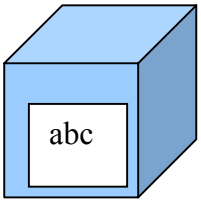


Istruire un calcolatore significa essenzialmente creare e usare degli oggetti.

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:

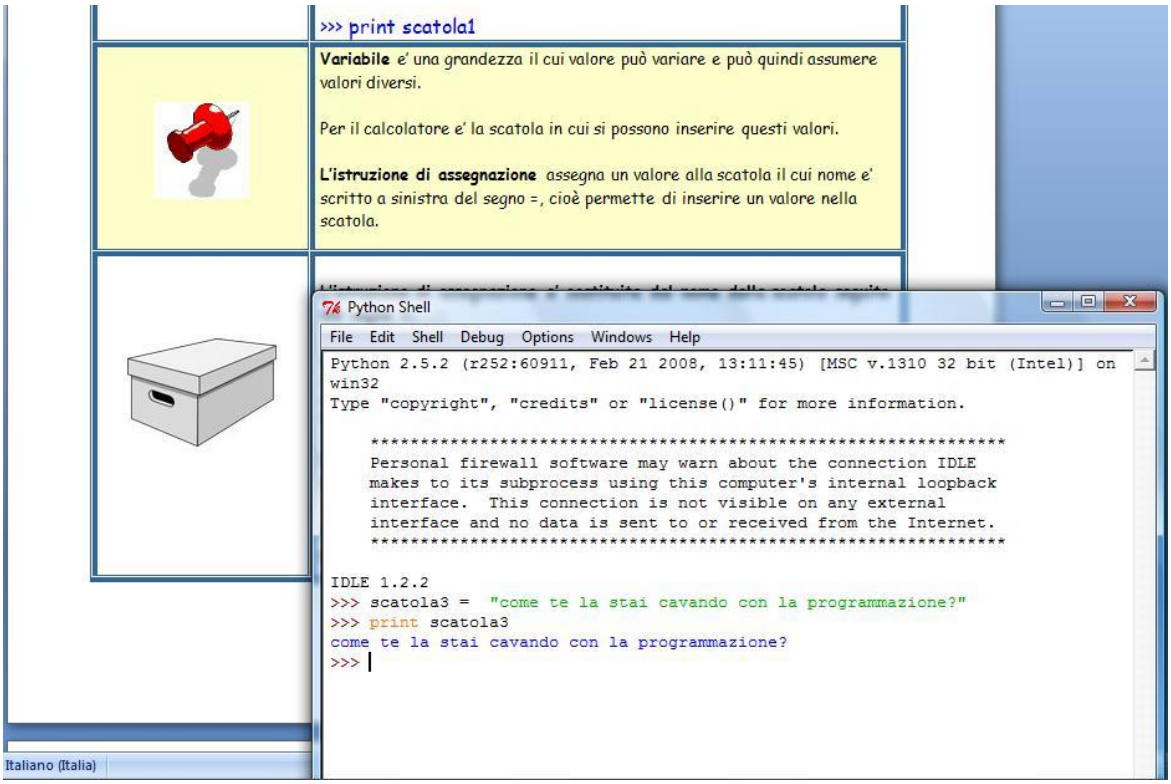
ILNOMEPIULUNGOCHEMIVIENEINMENTEPERILMIOCANE

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

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

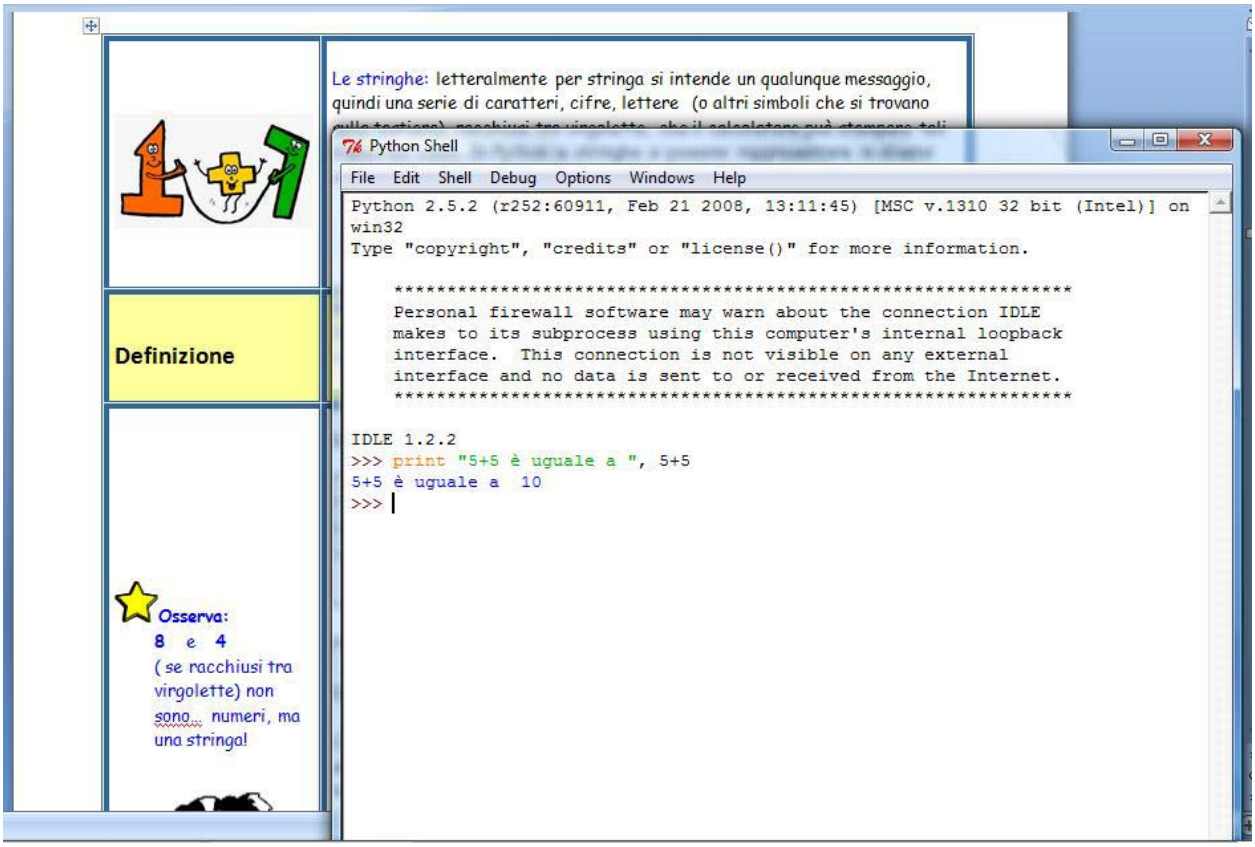


STEP 4
DATI E TIPI DI DATI

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

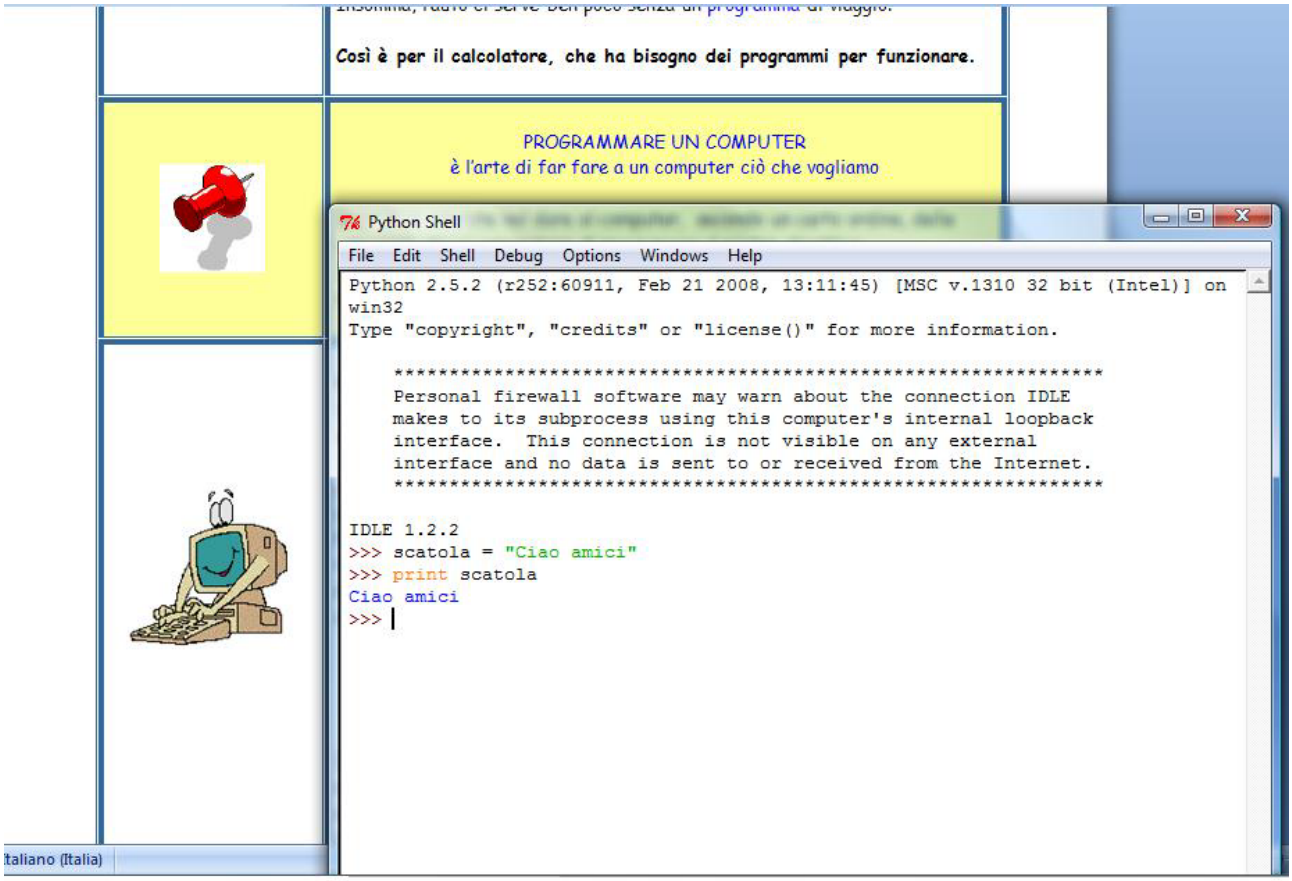
```
>>> print "ciao "*4
ciao ciao ciao ciao
```



STEP 5
IL PROGRAMMA

In questo step impareremo:

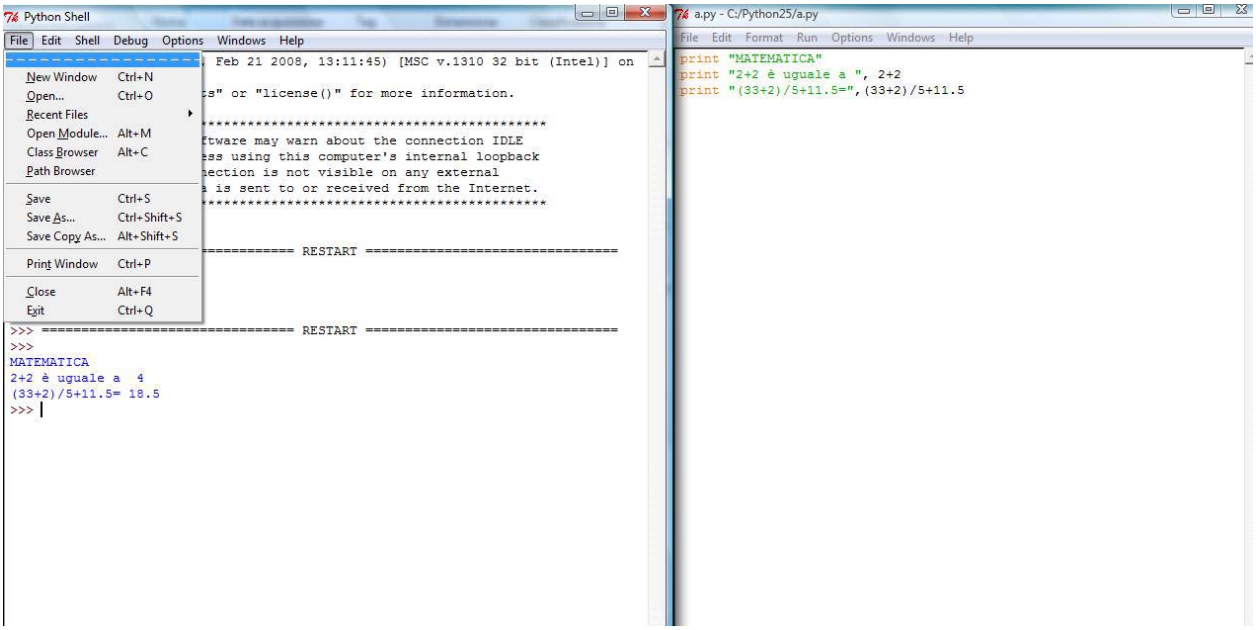
* cosa significa **programmare** e che cos'è un **programma**
* cosa è un **interprete** e un **compilatore**:
  (Python è un linguaggio interpretato)

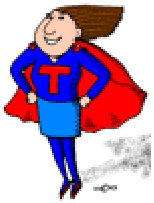


STEP 6
PROGRAMMARE IN PYTHON

In questo step impareremo:

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

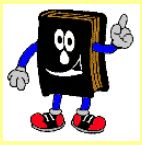




Adesso che hai imparato come fare a salvare il tuo programma in Python, devi imparare a:

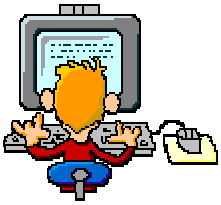
1. Farlo eseguire
2. recuperarlo per poterlo modificare o per utilizzarlo tutte le volte che vuoi

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?



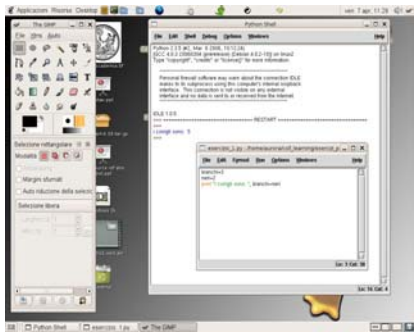
RUN: in italiano significa "corri" o meglio, nel nostro caso, "lancia"

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



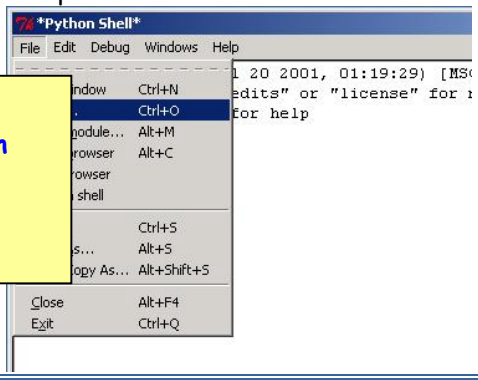
1. Selezionando RUN MODULE dalla finestra programma, Python eseguirà il nostro programma nella finestra interprete. Avremo così, aperte contemporaneamente, una finestra con il testo del programma ed un'altra con il risultato. Questa è una gran comodità! È infatti molto utile vedere contemporaneamente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

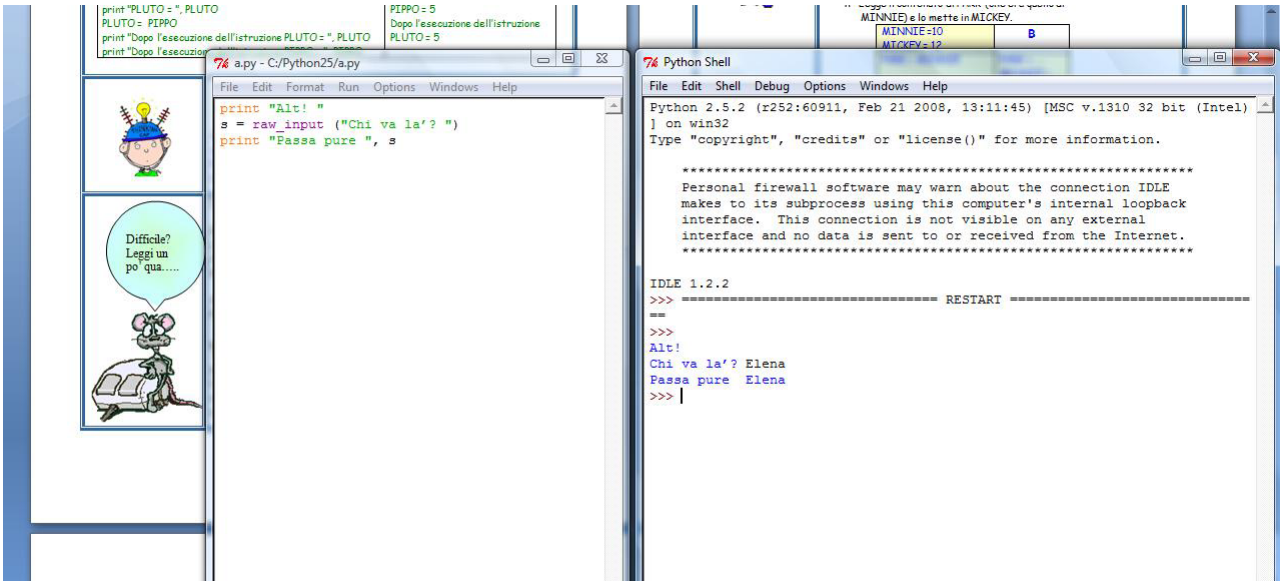


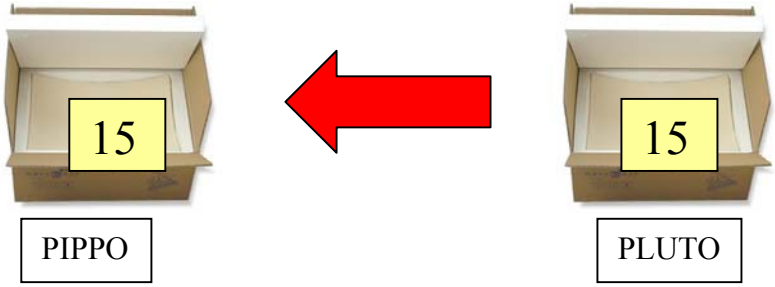
La stessa operazione puoi farla dalla finestra PROGRAMMA: dal menu FILE della finestra interprete seleziona NEW WINDOW e poi dal menù FILE seleziona il comando OPEN. Questa seconda possibilità è utile quando hai un programma già aperto e vuoi aprirne un altro, magari per copiarne un pezzo. Ti consiglio 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.

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**

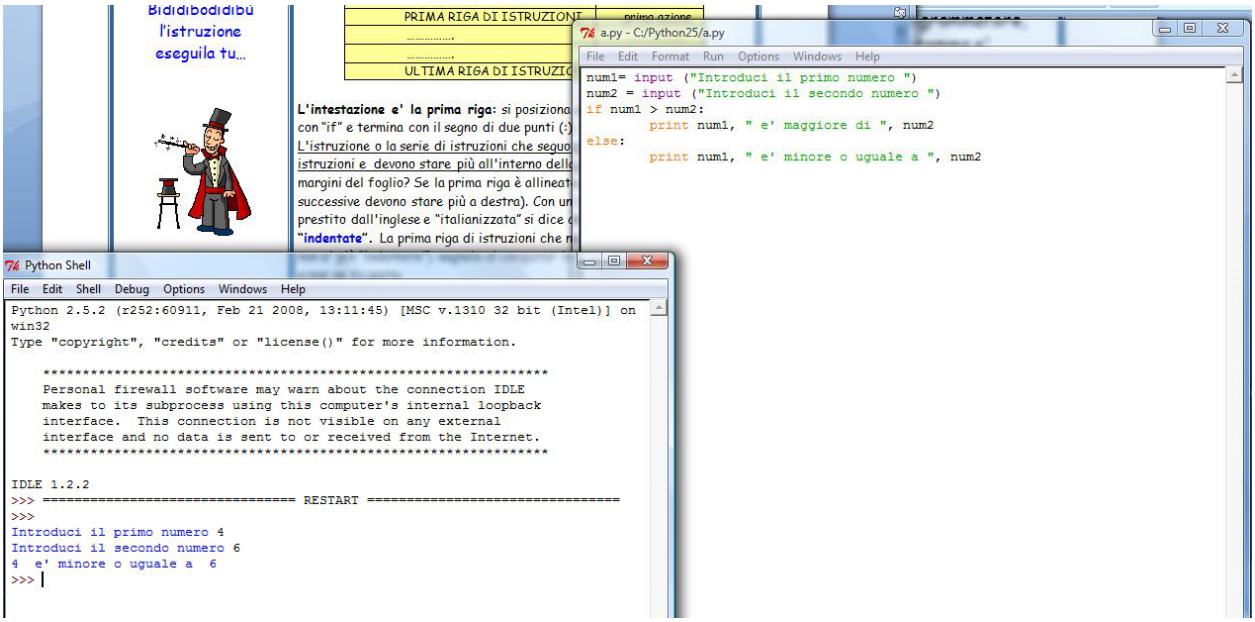


STEP 7 Ovvero, un passo dopo l'altro impariamo a programmare	**Da' una scatola all'altra**
	**COPIATURA** Supponiamo di avere a disposizione due scatole di nome PIPPO e PLUTO, con PIPPO = 5 e PLUTO = 15. Vediamo che cosa succede quando diamo al computer un'istruzione come questa: PIPPO = PLUTO Il computer 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.***

STEP 8

In questo step impareremo:

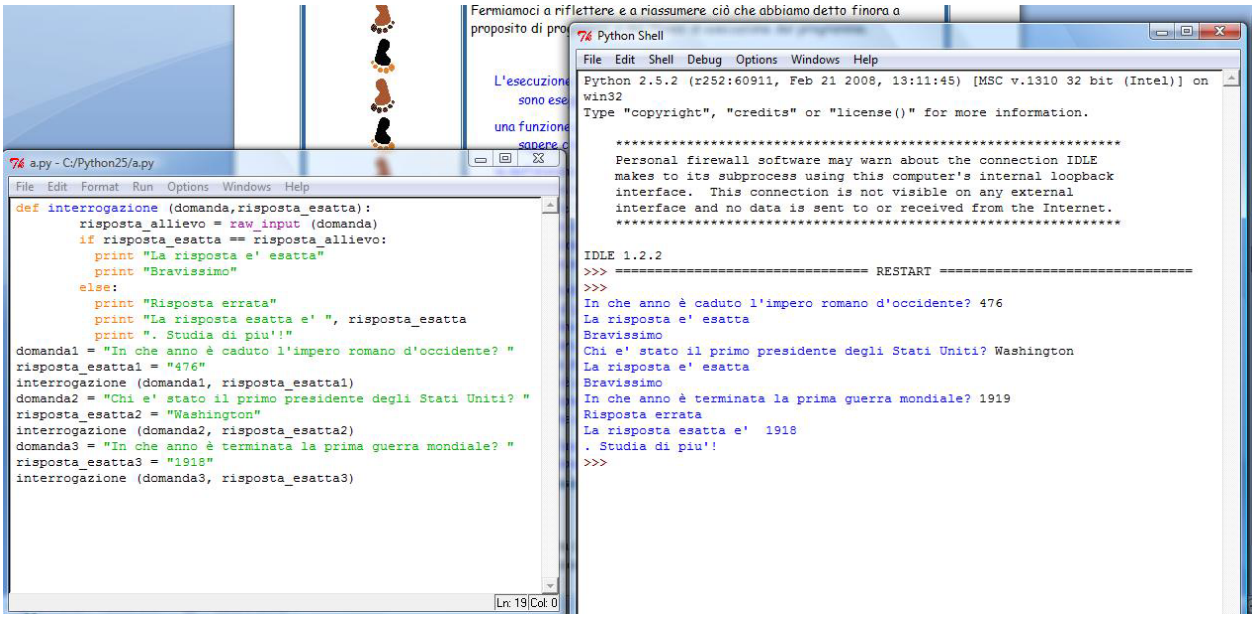
* come utilizzare le istruzioni condizionali **if** e **if else**
* come utilizzare gli operatori logici **or, and, not**
* cos'è **un'istruzione indentata**



STEP 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

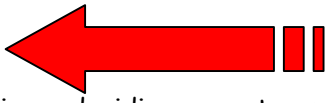


I Cicli annidati

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.

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

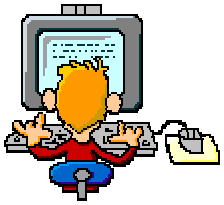
```
["pane","latte","zucchero",1,15,230,"bicchieri",1.5,2.5]
```

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.
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.

Ad esempio:

```
["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 dobbiamo dare un nome.

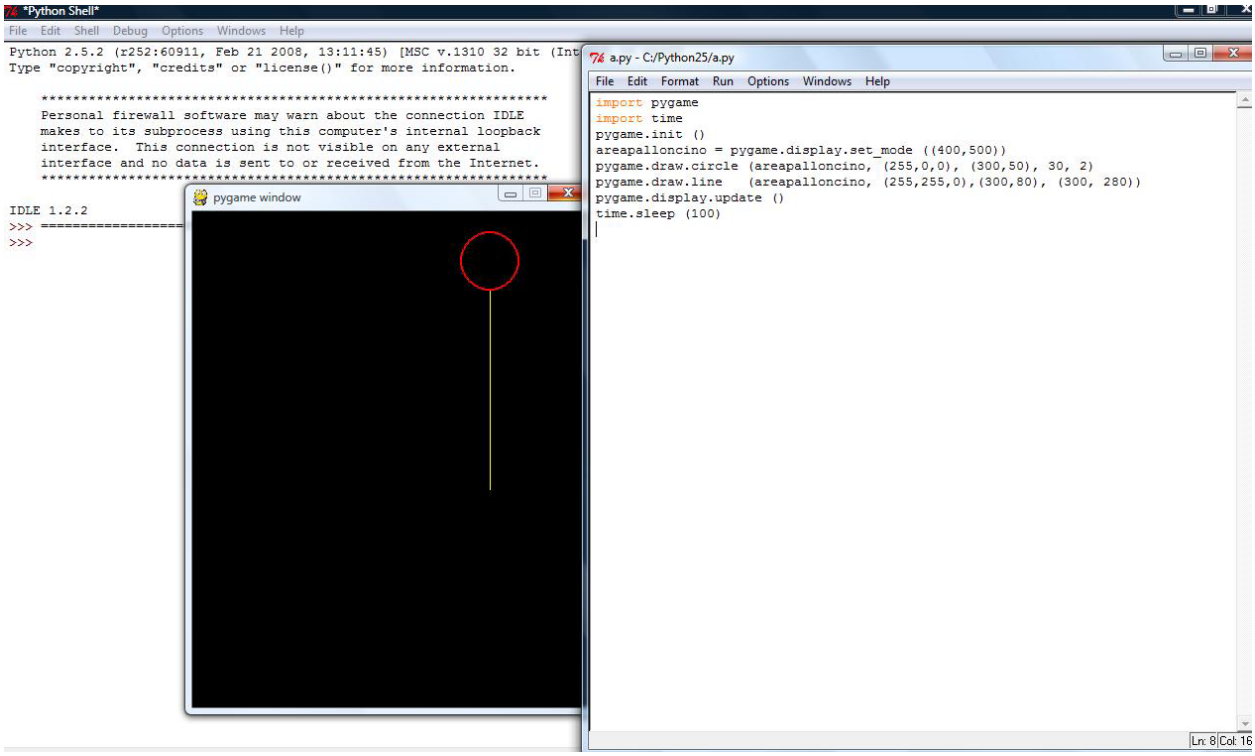
```
spesa
=["pane","latte","zucchero",1,15,230,"bicchieri",1.5,2.5]
vocabolario = ["bicicletta","casa","scuola"]
dati_di_delpiero = ["Del Piero", "Alessandro", 1974]
lista_vuota = []
```

Prova a visualizzare sul video il contenuto delle liste precedenti.

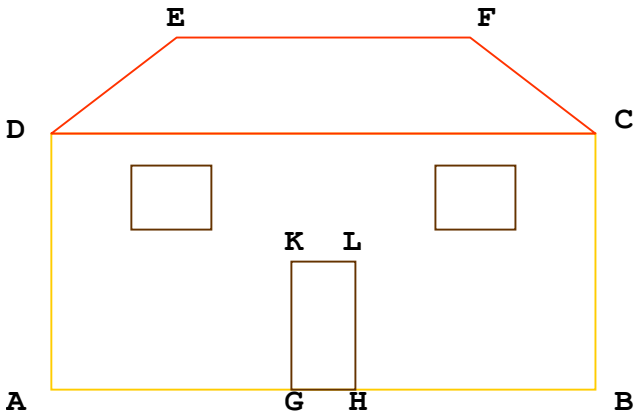
STEP 12

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**



Con le istruzioni che abbiamo visto siamo già in grado di disegnare una casetta come questa. Non è bellissima ma è un disegno!

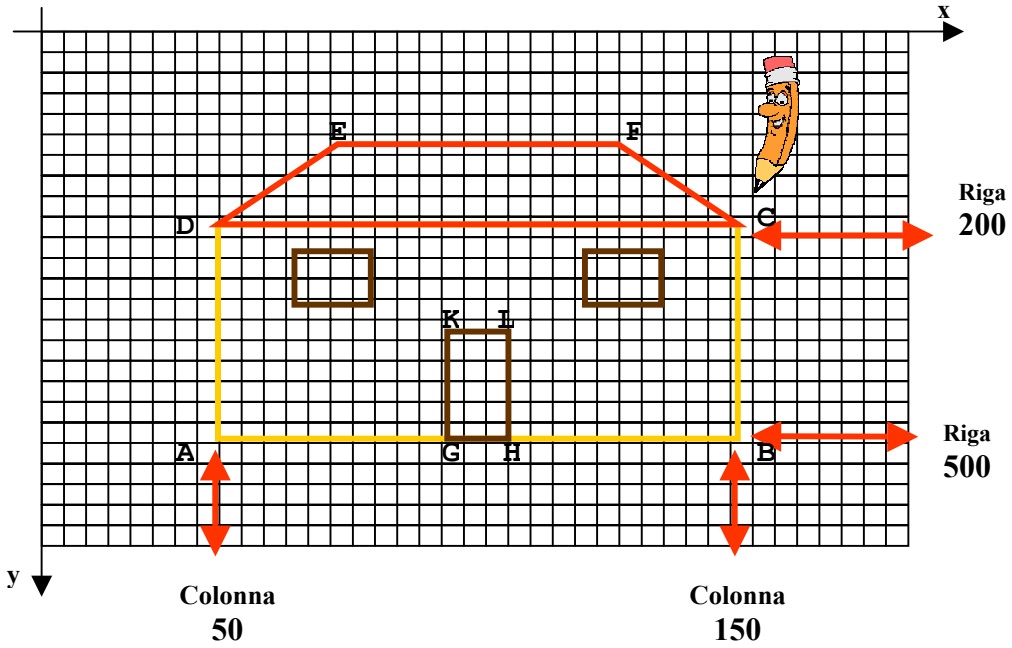


Cominciamo con disegnare il segmento AB partendo dall'ipotesi che A stia nella colonna 50 a partire da sinistra e nella riga 500 a partire dall'alto e che B sia nella colonna 150 e nella riga 500.
Con parole più eleganti, anche se più difficili, si dovrebbe dire che:
il punto A ha ascissa = 50 e ordinata = 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))
```



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

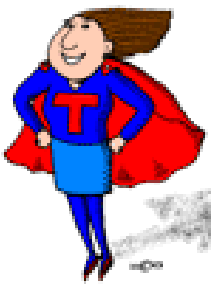
La settima istruzione:
`pygame.display.update ()`
equivale al seguente ordine " proietta sul video i due disegni del palloncino e del cordino che hai preparato".

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….

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.