INTRODUZIONE[1]

Python è un potente linguaggio di programmazione **interpretato** creato dal ricercatore olandese **Guido van Rossum** durante le vacanze di Natale a cavallo tra il 1989 ed il 1990. Il singolare nome dato a questo a linguaggio non ha nulla a che vedere con i pericolosi rettili. L'autore rimase un'intera notte a pensare ad un nome che fosse corto, unico ed un po' misterioso per battezzare il linguaggio. Mentre pensava a questo ed elaborava script, passava il tempo gustando *The Monty Python's Flying Circus*, una commedia dissacrante prodotta dalla BBC tra il 1969 ed il 1974.

Python è stato progettato in modo da essere altamente leggibile. Visivamente si presenta in modo molto semplice e ha pochi costrutti sintattici rispetto a molti altri linguaggi strutturati come il Pascal o il C. Per esempio, Python ha solo due forme di ciclo (`for` e `while`, con cui possono essere espressi tutti gli altri, tipo l'`until`, il `for`, `do ... while`.

Una cosa inusuale del Python è il metodo che usa per **delimitare i blocchi di programma**, che lo rende unico fra tutti i linguaggi più diffusi.
Nei linguaggi come il Pascal i blocchi di codice sono indicati con `begin` ed `end;` è solo una convenzione degli sviluppatori il fatto di indentare il codice interno ad un blocco, per metterlo in evidenza rispetto al codice circostante. Python, invece di usare parentesi o parole chiavi, usa **l'indentazione** (**una tabulazione** o, preferibilmente, **quattro spazi** bianchi) stessa per indicare i blocchi nidificati.
All'inizio questo modo di indicare i blocchi può confondere le idee a chi viene da altri linguaggi, ma poi si rivela molto vantaggioso, perché risulta molto conciso e obbliga a scrivere sorgenti indentati correttamente, aumentando alquanto la leggibilità del codice quando passa di mano in mano. Lo svantaggio è che la gestione degli spazi e dei caratteri di tabulazione può essere diversa da un editor all'altro, il che costringe a fare attenzione nell'indentare il codice, oppure ad affidarsi alle funzioni di indentazione automatica ormai presenti nella maggior parte degli editor di programmi.

Incominciamo ad usare Python

Apri un terminale (per esempio konsole) e lancia Python; basta scrivere **`python.`**
Dovresti vedere alcune righe di presentazione del programma ed il caratteristico prompt (tre segni di maggiore)
>>>
Siamo in **ambiente interattivo**. Prova a scrivere 2+2 e premi invio; scrivi poi `print "Ciao mondo!"` e premi invio.Siamo pronti a scrivere il nostro primo programma vero e proprio.

Il nostro primo programma in Python

Apriamo kate e dividiamo lo schermo in modo da visualizzare contemporaneamente l'editor ed il terminale; prima segnaliamo a kate che stiamo per scrivere un programma in python:

Strumenti> Indentazione> Stile Python
Strumenti> Evidenziazione> Script> Python

Scriviamo il codice seguente e salviamolo con il nome `fattoriale.py` nella cartella `primopython`.

```
# Primo programma in Python: questo e' un commento.
print "Questo e' il mio primo programma in Python."
print "Calcolo del fattoriale di un numero."
x=input("Inserisci un numero intero: ")
```

1 Questi appunti sono stati realizzati utilizzando i seguenti testi:
1. *Tutorial per principianti in Python* di Josh Cogliati: http://www.python.it/doc/Easytut/easytut-it/index.html
2. *Python - Più di un semplice linguaggio di script* di Marco Buzzo: http://www.python.it/doc/kranio-0.html
3. *Python Istantaneo,* traduzione di un interessante articolo introduttivo di Magnus Lie Hetland, a cura di Alex Martelli: http://www.python.it/doc/articoli/instpy-0.html
4. *Il tutorial di Python*, di Guido van Rossum, traduzione italiana all'indirizzo: http://www.python.it/doc/Python-Docs/html/tut/tut.html

```
# Ora definiamo la funzione fattoriale; nota il doppio uguale
def fattoriale(x):
          if x==0:
                    return 1
          else:
                    return x*fattoriale(x-1)
print "Il fattoriale e': "
print (fattoriale(x))

# Le ultime due istruzioni possono essere sostituite da
# print "Il fattoriale e'", fattoriale(x)
```

Per eseguire il programma passa alla finestra del terminale, assicurati di essere nella cartella dove hai salvato il programma e scrivi:

```
python fattoriale.py
```

Esercizio 1

Il seguente frammento di codice:

```
for value in range(100):
          print value
```

scrive i numeri da 0 a 99 (N.B. value è il nome della variabile che puoi modificare a tuo piacimento; se vuoi scrivere i numeri da 1 a 100 devi usare range(1,101)).

Scrivi un programma per generare i numeri da 1 a n, con n intero inserito dall'utente.
Ricordati di inserire nella prima riga un commento che indichi sinteticamente lo scopo del programma.

Esercizio 2

Osserva il seguente codice

```
x=input("Immettere un numero: ")

  print "Il quadrato del numero è", x*x
```

Scrivi un programma che generi i quadrati dei numeri da 1 a n, con n inserito da tastiera.

Esercizio 3

Osserva il seguente codice

```
if x < 5 or 10 < x < 20:

     print "Il valore è OK."
```

Scrivi un programma che permetta di stabilire se tre numeri inseriti da tastiera possono essere le misure dei lati di un triangolo.

Osservazione

Osserva il seguente codice, che serve per definire una funzione

```
def square(x):
        return x*x
print square(2) # Stampa 4
```

Una cosa che potrebbe esserti utile sapere è che in Python le funzioni sono valori. Così, se hai una funzione come square, potresti fare qualcosa come:

```
quadrato = square

quadrato(2)

Stampa 4
```

E per quelli di voi che vogliono rendere eseguibile uno script, usare la seguente prima riga per farlo funzionare da solo:

```
#!/usr/bin/env python
```

Occorre anche modificare i permessi del file in modo da renderlo eseguibile. Il comando da utilizzare è il seguente:
`chmod a+x nomefile.py`
Per provare, collocati nella directory dove si trova il file eseguibile e scrivi:
`./nomefile.py`

Esercizio 4

Scrivi un programma che permetta di stabilire se un triangolo di dati lati è rettangolo.

Esercizio 5

La successione di Fibonacci: copia ed esegui il programma.

```
print ""
print "Sequenza di Fibonacci fino a N"
try:
    qnt = int(raw_input("Introdurre il valore di N: "))
    print ""
    a, b = 0, 1
    while b < qnt:
        print b
        a, b = b, a+b
except ValueError:
    print "Il valore di N deve essere un intero!"
print ""
```

Nota
Se un programma si blocca a causa di un errore viene creata un'**eccezione**: l'interprete si ferma e viene creato un messaggio di errore. Ad esempio

```
>>> print 55/0
ZeroDivisionError: integer division or modulo
```

Molte operazioni possono generare errori in esecuzione ma in genere desideriamo che il programma non si blocchi quando questo avviene. La soluzione è quella di **gestire** l'eccezione usando le istruzioni **try** ed **except**.
Esempio:

```
def InputNumero():
    x = input ('Dimmi un numero: ')
    if x > 16:
        raise 'ErroreNumero', 'mi aspetto numeri minori di 17!'
```

```
    return x
```

Se provi ad eseguire il programma inserendo 17 dovresti avere com output

```
ErroreNumero: mi aspetto numeri minori di 17!
```

Concludiamo questa lezione ricordando il motto per imparare il Python:

"Use the source, Luke"

(Traduzione: Leggi tutto il codice su cui puoi mettere le mani :))