THE PYTHON BASICS

COS'È PYTHON?

Python è un linguaggio di programmazione che permette di creare applicazioni e automatizzare i processi in modo facile e veloce.

Python è un lunguaggio interpretato e non compilato!

Si distingue dagli altri per la sua estrema semplicità nella sintassi e nella gestione della memoria!

ANCORA UN PAIO DI INFO UTILI

È stato sviluppato da Guido Van Rossum nel 1991.

È un linguaggio di «alto livello» ovvero molto automatizzato.

È un linguaggio open source.

LA FUNZIONE ‘print’

La prima funzione che andremo ad analizzare è la funzione ‘print’!

```
print('Hello World')
```

Mostra sulla riga di comando quello sta scritto tra parentesi

VARIABILI

In python è possibile rappresentare diversi tipi di valori, quelli basici sono:
numeri interi (`int`), numeri decimali (`float`),
stringhe (`str`) e valori booleani come vero o falso (`bool`).

5 => `int`; 4,66 => `float`; 'ciao' => `str`; `True` => `bool`

Questi valori possono essere contenuti in delle variabili che ne assumono il tipo e il valore!

```
x = 5
print(x)

word = 'Hello World'
print(word)
```

```
y = 7
print(y)

ciao = 'Hello World'
print(ciao)
```

Per creare una variabile vuota si può assegnare come valore `None`: `x = None`

LE OPERAZIONI CON I TIPI

Con questi valori si possono svolgere diverse operazioni:

Con i numeri si possono svolgere le 4 operazioni (+, -, *, /)

```
x = 5
print(x + 5)
print(x - 2)
```

```
print(x * 2)
print(x / 5)
print(x ** 2)
```

Con le stringhe si possono svolgere 2 operazioni (+, *)

```
x = 'ba'
print(x + 'na'*2)
```

Operatore	Esempio	Significato
+	A + b	Somma
-	A – b	Sottrazione
*	A * B	Moltiplicazione
/	A / b	Divisione
**	A ** b	Elevamento a potenza
//	A // b	Parte intera della divisione
%	A % b	Resto della divisione

Operatore	Esempio	Significato
==	A == b	È uguale a
!=	A != b	È diverso da
<	A < b	È minore di
<=	A <= b	È minore o uguale a
>	A > b	È maggiore di
>=	A >= b	È maggiore o uguale a

Operatore	Esempio	Significato
Not	Not p	Non vero, quindi falso
And	P and q	Entrambi veri
Or	P or q	Almeno uno dei due è vero

CONDIZIONI

Le condizioni si basano sul valori booleani (`True` or `False`) e valutano se una determinata espressione è vera o falsa.

```
x = 3
if x + 2 == 5:
    print('x + 2 è uguale a 5')
```

In questo caso la condizione `x + 2 == 5` è vera, quindi vengono le eseguite le operazioni che hai definito

Se fosse stato diverso da 3 non sarebbe stata eseguita alcuna azione...

`else` serve ad eseguire delle azioni in caso la condizione non fosse vera.

Nell'esempio infatti la condizione `x + 2 == 5` non è vera e quindi vengono eseguite le azioni definite dopo `else`.

```
x = 5
if x + 2 == 5:
    print('x + 2 è uguale a 5')
else:
    print('x + 2 è diverso da 5')
```

LE FUNZIONI (`function`)

Una funzione è un'insieme di operazioni definite dal programmatore che accettano dei parametri che devi fornire alla chiamata.

In python le funzioni si definiscono con la parola chiave '`def`':

```
def saluta():
    print('ciao!')
```

Le funzioni possono anche accettare dei parametri per operazioni più complesse:

```
def somma(n1, n2):
    return n1 + n2
```

In questo caso sono `n1` e `n2` parametri della funzione '`somma`' che ne fa una addizzione e ne ritorna il valore (`return`)

USARE LE FUNZIONI

Per utilizzare le funzioni definite in precedenza dobbiamo effettuare una chiamata

```
# chiamiamo la funzione saluta
saluta() # ora dovremmo vedere
# che la funzione è stata eseguita!
```

Per chiamare la funzione somma dobbiamo specificare anche i due parametri richiesti ovvero n1 e n2

```
# chiamo la funzione somma con parametri 5 e 3
s = somma(5, 3) # memorizzo l'output
# in una variabile
print(s) # s dovrebbe valere 8!
```

LE LISTE (`list`)

Una lista è un insieme ordinato di elementi.
In python le liste si definiscono con le parentesi quadre: `l = [1,2,3]`

Si possono scegliere elemanti da una lista attaraverso l'indice dell'elemento nella lista:

```
# elemento nella lista al posto 0
print(l[0]) # cioè il primo elemento
```

Per aggiungere un'elemento ad una lista si può utilizzare la funzione '`append`':

```
l.append(5)
print(l)
```

Invece per rimuovere un'elemento si può usare la funzione '`remove`':

```
l.remove(5)
print(l)
```

IL CICLO `while`

Un ciclo è una ripetizione di una serie determinate operazioni.

In python esistono 2 tipi di cicli (loop): il ciclo `for` e il ciclo `while`.

Il ciclo while si definisce come:

```
while expr:
    do_something
```

Se expr è falso allora il ciclo si interrompe sennò il ciclo continua.

Alcuni esempi:

```
x = 0
while x != 5:
    x = x + 1
    print(x)
```

```
word = input('Inserisci una parola: ')
# finchè la parola non è 'stop'
while word != 'stop': # continua a chiedere
    print('Hai scritto: ' + word)
    word = input('Inserisci una parola: ')
```

IL CICLO `for`

Un ciclo for ripete la stessa azione (itera) tra una determinata serie di elementi come le liste.

Il ciclo `for` è definito come:

```
for i in x:
```

Dove `x` è la lista nella quale iterare e `i` è la variabile nella quale viene memorizzato l'elemento di iterazione.

Alcuni Esempi:

```
for i in [1,2,3,4,5]:
    print(i)
```

```
# range() itera per n volte,
# i assume il valore del numero
# della ripetizione partendo da 0
for i in range(5):
    print(i)
```

```python
def fibonacci(n):
    numeri = [0,1]

    for i in range(n):
        if i > 1:
            numeri.append(numeri[i-2]+numeri[i-1])

    return numeri

fibonacci(10)
```