# Fondamenti di Python

## Parte 1

- **In Python la tipizzazione delle variabili è dinamica, non è necessario dichiarare le variabili prima di utilizzarle ne dichiarare il loro tipo**

In [11]:

```
a = 1
type(a)
```

Out[11]:

```
int
```

In [9]:

```
a = 1.34567
type(a)
```

Out[9]:

```
float
```

- **Le stringhe si definiscono con ' ' 0 " " sono seguenze IMMUTABILI di caratteri sulle quali è possibile itarare (es. for, while)**

In [12]:

```
a = 'abc'
a(2) = c
```

```
  File "<ipython-input-12-dfcb56d73929>", line 2
    a(2)=c
    ^
SyntaxError: cannot assign to function call
```

In [15]:

```
a = 'abc'
for letter in a :
    print(letter)
```

```
a
b
c
```

In [17]:

```
for letter in 'abc' :
    print(letter)
```

```
a
b
c
```

- **La funzione print() permette di stampare oggetti in output, gli ogetti passati alla funzione vengono convertiti in stringhe**

In [18]:

```
print (2+8)
```

```
10
```

In [19]:

```
print ('fanti')
```

```
fanti
```

In [20]:

```
print('a')
print('b')
```

```
a
b
```

- **La formattazione di print(), di default la funzione va a capo da sola, mentre per mettere due output nell stessa riga separati da uno spazio si fa nel seguente modo:**

In [23]:

```
print('a', end='')
print('\tb')
```

```
a b
```

- **Slicing-> notazione che permette di accedere ad una serie di elementi di una sequenza ordinata attraverso il loro indice**

In [36]:

```
a = 'babbimorti'
a[:10]
```

Out[36]:

```
'babbimorti'
```

In [37]:

```
a = 'babbimorti'
a[2:8:2]
```

Out[37]:

```
'bio'
```

In [38]:

```
a = 'babbimorti'
a[6:]
```

Out[38]:

```
'orti'
```

In [39]:

```
a = 'babbimorti'
a[::-1]
```

Out[39]:

```
'itromibbab'
```

- **Gli operatori sono quelli di default(+ * - ecc...) tranne l'elevamento a potenza(**) e la floor division(//) che ritorna la parte intera della divisione**

- **Possiamo perfino concatenare gli operatori di confronto**

- **Possiamo perfino concatenare gli operatori di confronto**

In [48]:

```
a = 3
b = 4
c = 5
a < b < c
```

Out[48]:

```
True
```

- **Gli operatori logici sono quelli che conosciamo, più interessanti invece sono gli operatori di *identità* -> ulizzati per confrontare gli oggetti, non per verificare se i valori sono uguali ma se sono lo stesso oggetto(is ------- is not), operatori di *appartenenza*-> (in ------- not in)**

In [49]:

```
a = 'babbimorti'
'babbi' in a
```

Out[49]:

```
True
```

In [52]:

```
a = 'babbimorti'
b = a
b is a
```

Out[52]:

```
True
```

In [57]:

```
a = 'babbimorti'
'c' not in a
```

Out[57]:

```
True
```