Studiamo la condizione if

Con l'istruzione if possiamo eseguire soltanto porzioni di codice in base al verificarsi di una determinata condizione.

Ecco un esempio: le istruzioni print("zero") e print("numero negativo") non saranno eseguite

In [1]:
```
x = 1
if x>0:
    print("numero positivo")
elif x == 0:
    print("zero")
else:
    print("numero negativo")
```

```
numero positivo
```

In questo caso. invece, l'istruzione print("numero positivo") non sarà eseguita

In [2]:
```
x = -1
if x>0:
    print("numero positivo")
else:
    print("numero negativo")
```

```
numero negativo
```

Attenzione a questi tre codici simili. Studiare il risultato nei vari casi

In [3]:
```
x = 10
if x > 5:
    print("numero maggiore di 5")
elif x >= 0 :
    print("numero maggiore o uguale a 0")
else:
    print("numero negativo")
```

```
numero maggiore di 5
```

In [4]:
```
x = 10
if x >= 0:
    print("numero maggiore o uguale a 0")
elif x > 5 :
    print("numero maggiore di 5")
else:
    print("numero negativo")
```

```
numero maggiore o uguale a 0
```

In [5]:
```
x = 10
if x > 5:
    print("numero maggiore di 5")
if x >= 0:
    print("numero maggiore o uguale a 0")
else:
    print("numero negativo")
```

```
numero maggiore di 5
numero maggiore o uguale a 0
```

Studiamo la condizione while

Con l'istruzione while posso ripetere delle porzioni di codice più volte, fin quando resta verificata una determinata condizione espressa all'inizio del ciclo while.

Ad esempio questo codice ripeterà più volte l'istruzione x = input("Inserisci un numero"), finquando non inseriremo un numero negativo

In [7]:
```
x = input("Inserisci un numero")

while int(x) >= 0:
    x = input("Inserisci un numero")

print("hai interrotto il ciclo perché hai inserito un numero negativo")
```

```
hai interrotto il ciclo perché hai inserito un numero negativo
```

Break e continue all'interno del while

L'istruzione break provoca **immediatamente** l'uscita dal ciclo While, interrompando l'esecuzione corrente e annullando anche le eventuali successive.

Ad esempio il codice seguente stampa soltanto i valori 1, 2. Quando x è uguale a 3, la presenza del break provoca sia la fine dell'iterazione corrente e sia la fine dell'intero ciclo while, anche se la condizione x<=5 è ancora vera

In [8]:
```
x = 1
while x<5:
    if(x==3):
        break
    print(x)
    x = x+1
```

```
1
2
```

L'istruzione continue provoca soltanto l'interruzione dell'iterazione corrente. La condizione del while sarà nuovamente rivalutata.

Ad esempio nel codice seguente saltiamo l'istruzione print(x) quando x è uguale a 3

In [9]:
```
x = 0
while x<5:
    x=x+1
    if(x==3):
        continue
    print(x)
```

```
1
2
4
5
```

Facciamo attenzione! **Potremmo incappare facilmente in codici di durata infinita**, come il seguente:

```
x = 0

while x<5:

    if(x==3):
        continue

    x=x+1
```

Una volta arriviti a 3, entriamo in un loop infinito: la x non si incrementa per via del continue. D'altra parte il ciclo non finisce perché la x continua ad essere sempre minore di 5

Ciclo For

In Python il ciclo `for` serve per iterare su **sequenze**:

In [10]:
```
# range(n) genera i numeri da 0 a n-1
for i in range(5):
    print("Iterazione numero:", i)
```

```
Iterazione numero: 0
Iterazione numero: 1
Iterazione numero: 2
Iterazione numero: 3
Iterazione numero: 4
```

In [11]:
```
# range(start, end) genera numeri da start incluso a end escluso.
for i in range(2, 6):
    print(i)   # Stampa 2, 3, 4, 5
```

```
2
3
4
5
```

In [12]:
```
#range(start, end, step) permette di specificare un "passo".
for i in range(0, 10, 2):
    print(i)
```

```
0
2
4
6
8
```

In [13]:
```
#Iterare su una stringa
s = "Python"
```

```
for char in s:
    print(char)   # Stampa ogni carattere
```

```
P
y
t
h
o
n
```

In [14]:
```
#Iterare su una lista
frutta = ["a", "b", "c"]
for f in frutta:
    print("elemento lista:", f)
```

```
elemento lista: a
elemento lista: b
elemento lista: c
```

In [15]:
```
# enumerate() restituisce sia l'indice che l'elemento
animali = ["gatto", "cane", "coniglio"]
for indice, animale in enumerate(animali):
    print(indice, animale)
```

```
0 gatto
1 cane
2 coniglio
```