Funzioni

Le funzioni permettono di definire blocchi di codice che viene eseguito solamente quando viene chiamato.

Quando si programma due cose sono fondamentali:

1. Non riscrivere il codice che hanno già scritto altri
2. Non duplicare il codice che hai già scritto tu

Tutto cio è reso possible dalle **funzioni**.

Abbiamo già usato molte funzioni sino ad ora come ad esempio **input**, **print**, **len**.

Le funzioni devono essere definite *prima* del loro utilizzo, ed hanno questa sintassi:

```
def nome(parametro1, parametro2, ...):
    return qualcosa
```

Una funzione può prendere dei valori come parametri, quindi dei valori che può utilizzare all'interno del suo blocco di codice, e può anche ritornare dei valori.

In [4]:
```
def ciao_mondo():
    print("Ciao mondo!")
```

In [6]:
```
ciao_mondo()
```

```
Ciao mondo!
```

Come abbiamo detto la cosa interessante è che ci permette di non riscrivere del codice uguale e compattare tutto in un unico blocco. Immaginiamo di voler stampare spesso una formattazione particolare del testo ad esempio:

---------- Nuovo Titolo ----------

con ovviamente un titolo sempre diverso per ogni argomento.

In [18]:
```
def bel_titolo(argomento):
    print(f"----------{argomento}----------")
```

In [24]:
```
# Run per vedere l'errore
# bel_titolo()
```

In [26]:
```
bel_titolo("Programmazione")
```

```
----------Programmazione----------
```

In [30]:
```
arg1 = "Matematica"
arg2 = "Storia"
arg3 = "Filosofia"
bel_titolo(arg1)
bel_titolo(arg2)
bel_titolo(arg3)
```

```
----------Matematica----------
----------Storia----------
----------Filosofia----------
```

Nota: Questa funzione non ritorna alcun valore.

Esempio

Adesso proviamo a creare la nostra funzione che ritorni il massimo di una lista. Il codice sappiamo farlo benissimo, però proviamo a trasformarlo in una funzione da chiamare ogni volta se vogliamo sapere il massimo di una lista.

In [33]:
```
lista = [3, 9, 13, 27, 89, 56, 42, 98, 64, 6, 12]

def massimo(listone):
    numero_max = 0
    for num in listone:
        if num > numero_max:
            numero_max = num
    return numero_max

print(massimo(lista))
```

```
98
```

Adesso che ci siamo scritti la nostra funzione per calcolare il massimo numero all'interno di una lista, si può utilizzare ogni volta che si vuole invece di scriverci ogni volta tutto il codice.

Questa è la figata!

Import

Come abbiamo già detto, è utile e figo non riscrivere codice già scritto da altri. Prendiamo ad esempio il calcolo del fattoriale di un numero, esercizio che abbiamo già svolto con i cicli.

In [45]:
```
# Calcolo del fattoriale di 5
n = 5
fattoriale = 1
if n < 0:
    print("Il fattoriale non esiste per numeri minori di 0")
elif n == 0:
    print("Il fattoriale di 0 è 1")
else:
    for i in range(1, n + 1):
        fattoriale = fattoriale * i
    print("Il fattoriale di", n, "è", fattoriale)
```

```
Il fattoriale di 5 è 120
```

In [12]:
```
# Rendiamo il codice precedente una funzione
def fattoriale(numero):
    fattoriale = 1
    if numero < 0:
        return 0
    elif numero == 0:
        return 1
    else:
        for i in range(1, numero + 1):
            fattoriale = fattoriale * i
        return fattoriale
```

In [14]:
```
print(fattoriale(5))
```

```
120
```

Quanta fatica scrivere codice inutile! Si può importare direttamente la funzione tramite il modulo **math**, un modulo base di python.

In [50]:
```
from math import factorial
factorial(5)
```

Out[50]:
```
120
```

Esempi di moduli utili

Modulo	Descrizione	Esempio
`math`	Funzioni matematiche avanzate, come radici e fattoriali.	`math.sqrt(25) # → 5.0`
`random`	Generazione di numeri casuali o scelta casuale da una lista.	`random.randint(1, 10) # → Numero tra 1 e 10`
`datetime`	Lavorare con date e orari.	`datetime.date.today() # → Data odierna`
`time`	Funzioni per gestire il tempo e creare pause.	`time.sleep(3) # Pausa di 3 secondi`
`os`	Interagire con file, cartelle e sistema operativo.	`os.getcwd() # → Cartella attuale`
`sys`	Informazioni su Python e il sistema.	`sys.version # → Versione di Python`

Esempio / Esercizio

Tempi di rezione (moduli time e random)

In [26]:
```
import time
import random

def test_tempo_reazione():
    print("Preparati... quando vedi 'VIA!', premi INVIO il più velocemente possibile!")

    # Attendi un tempo casuale tra 2 e 5 secondi
    attesa = random.uniform(2, 5)
    time.sleep(attesa)

    # Mostra il segnale di partenza
    print("VIA! (premi INVIO ora!) ")

    # Registra il tempo di inizio
    inizio = time.time()

    # Aspetta l'input dell'utente
    input()

    # Registra il tempo di fine
    fine = time.time()

    # Calcola il tempo di reazione
    tempo_reazione = fine - inizio
    print(f"Hai impiegato {tempo_reazione:.3f} secondi!")

# Avvia il test
test_tempo_reazione()
```

```
Preparati... quando vedi 'VIA!', premi INVIO il più velocemente possibile!
VIA! (premi INVIO ora!)
Hai impiegato 0.290 secondi!
```