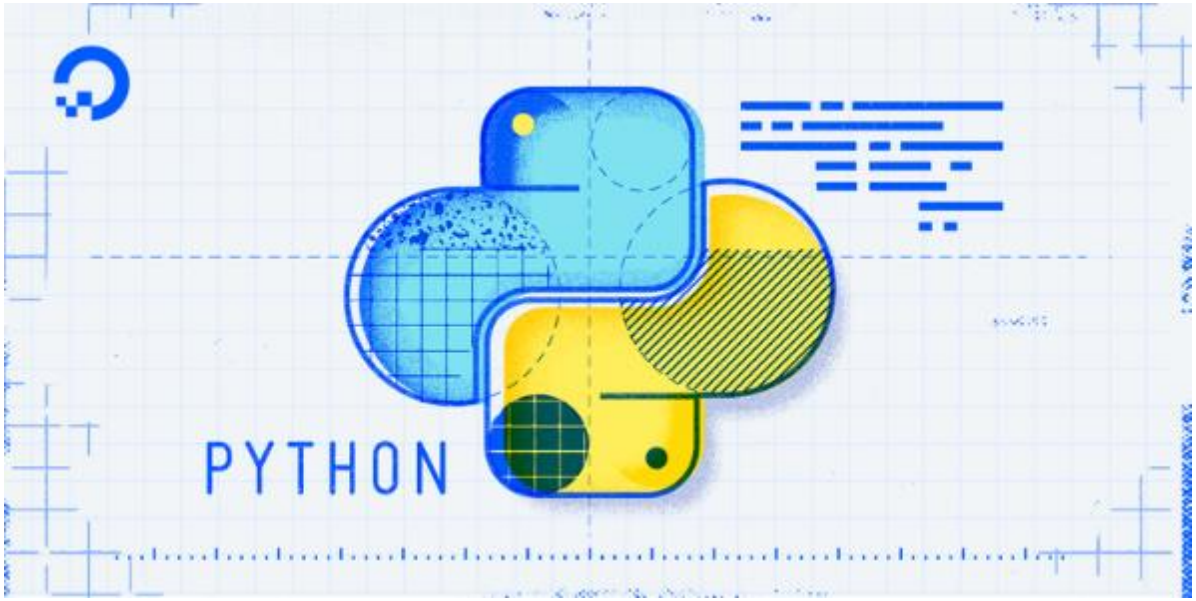


funzioni

python

- ***operatore***, applicato a ***operandi***, per ottenere un ***risultato***
- **`def`** per ***definire*** una funzione
- **`return`** per terminare e restituire un ***risultato***

```
def ipotenusa(a,b):
    c = (a ** 2 + b ** 2) ** 0.5
    return c
```

chiamata di funzione

- **def** definisce una funzione, ma non la esegue!
- per far ***eseguire*** una funzione è necessario «***chiamarla***»
  - la funzione, quando viene eseguita, crea nuovo spazio di nomi
  - i parametri e le variabili hanno ambito locale
  - non sono visibili nel resto del programma
  - nomi uguali, definiti in ambiti diversi, restano distinti

```
c1 = int(input("primo   cateto: "))
c2 = int(input("secondo cateto: "))
ip = ipotenusa(c1,c2)
print(ip)
```

- è spesso preferibile creare una ***funzione principale (main)***
- in questo modo si limitano le variabili globali

```
def ipotenusa(a,b):
    c = (a ** 2 + b ** 2) ** 0.5
    return c

def main():
    c1 = int(input("primo   cateto: "))
    c2 = int(input("secondo cateto: "))
    ip = ipotenusa(c1,c2)
    print(ip)

main()
```

parametri di funzioni

- la definizione della funzione opera sui ***parametri formali***
- al momento della chiamata si definiscono i ***parametri attuali***
- le variabili definite nella funzione rimangono locali a questa

```
def dummy(f1, f2):
    loc = f1 ** f2
    f1 = f1 * 2
    return loc

a1 = float(input("fist value: "))
a2 = float(input("secondt value: "))
print(dummy(a1,a2))
print(loc)     # NameError: name 'loc' is not defined
print(a1)    # print ???
```

- ***call-by-object***
- parametri passati «per oggetto»
  - se il parametro è una ***variabile*** le modifiche non si ripercuotono all'esterno
  - se il parametro è una ***lista*** o un ***oggetto*** le modifiche si ripercuotono

```
def inc(f):
    f = f + 1
    print(f) # 11

a = 10
inc(a)
print(a)     # 10
```

```
def inc(f):
    for i in range(0,len(f)):
        f[i] = f[i] + 1
    print(f) # [3,4,6]

a = [2,3,5]
inc(a)
print(a) # [3,4,6]
```

- si possono restituire più valori, come tupla

```
def min_max(f):
    ''' restituisce valore minimo e massimo della lista f '''
    minimo = massimo = f[0]
    for i in range(1,len(f)):
        if f[i] < minimo:
            minimo = f[i]
        if f[i] > massimo:
            massimo = f[i]
    return minimo, massimo

def main():
    a = [2,13,5,-3,8]
    x , y = min_max(a)
    print("minimo: ",x," massimo: ",y)

main()  ## remove if importing the module elsewhere
```

- ***annotazioni***: utili per documentare il tipo dei parametri e il tipo del valore di ritorno *(ma non c'è verifica!)*
- ***docstring***: descrizione testuale di una funzione
- ***help***: funzione per visualizzare la documentazione

```
def hypotenuse(cathetus1: float, cathetus2: float) -> float:
    '''
    Return the hypotenuse of a right triangle,
    given both its legs (catheti).
    '''
    return (cathetus1 ** 2 + cathetus2 ** 2) ** 0.5
```

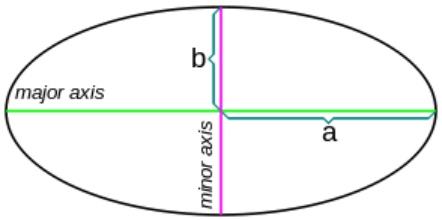
- la stringa di documentazione, posta all'inizio di una funzione, ne ***illustra l'interfaccia***
- per convenzione, la ***docstring*** è racchiusa tra triple virgolette, che le consentono di essere divisibile su più righe
- è breve, ma contiene le informazioni essenziali per usare la funzione
  - spiega in modo conciso ***cosa fa*** la funzione (non come lo fa)
  - spiega il significato di ciascun parametro e il suo tipo
- è una parte importante della progettazione dell'interfaccia
  - un'interfaccia deve essere ***semplice*** da spiegare

funzioni in python 3

esercizi

area di un'ellisse

- definire una funzione ***areaEllise*** che:
  - riceve come parametri i ***semiassi*** di una ellisse: **a, b**
  - restituisce come risultato l'area dell'ellisse: $\pi \cdot a \cdot b$
- definire una funzione ***main*** che:
  - chiede all'utente due valori
  - invoca la funzione areaEllisse con questi parametri
  - stampa il risultato ottenuto



esercizi (2)

A. Ferrari

1. Scrivere la funzione ***volume*** che riceve come parametri le misure degli spigoli di un parallelepipedo rettangolo e restituisce il volume. Scrivere un programma che richiede in input le misure di un parallelepipedo rettangolo e, utilizzando la funzione volume stampa il volume.
2. Scrivere le funzione ***isTriangoloRettangolo*** che riceve come parametri 3 valori e restituisce True se questi possono essere i lati di un triangolo rettangolo. Utilizzando la funzione richiedere in input 3 valori e verificare se possono essere i lati di un triangolo rettangolo
3. Scrivere la funzione ***isPrimo*** che restituisce True se il parametro ricevuto è un numero primo. Visualizzare tutti i numeri primi compresi fra 1 e 100.
4. *In matematica due numeri primi si dicono sexy quando la loro differenza è uguale a sei. Il nome di queste coppie di numeri primi deriva dalla parola latina sex (ovvero sei).* Es. (11,17) è una coppia di primi sexy come pure (67, 73).
   5. Scrivere un programma che visualizza tutte le coppie di primi sexy con valori compresi fra 1 e 100