usiamo-le-funzioni-di-libreria

June 5, 2023

0.1 Funzioni di libreria

Di sicuro non è divertente "reinventare la ruota". A meno che non abbia uno scopo didattico, non ha senso riscrivere un programma che faccia qualcosa che è già stato risolto da qualcun'altro.

Rivediamo l'esempio dell'area del cerchio:

[1]:
```
raggio = float(input())

area = raggio * raggio * 3.14

print("L'area del cerchio di raggio",raggio,"è",area)
```

```
 23
```

```
L'area del cerchio di raggio 23.0 è 1661.0600000000002
```

Non sarebbe meglio usare un'approssimazione migliore per il pi greco? Potremmo calcolarne una e inserirla nel nostro programma. Tuttavia python ne include una. Per usarla è necessario **importare** il modulo `math`.

[2]:
```
import math

print(math.pi)
```

```
3.141592653589793
```

Esercizio: siete in grado di integrare i due programmi precedenti e scrivere una versione più precisa del primo, utilizzando l'approssimazione del pi greco nel secondo?

0.1.1 Uso di moduli

I *moduli* python sono dei "contenitori" di funzionalità. Python ne contiene moltissimi e tanti altri possono essere installati. Uno dei principali è in modulo `math` che contiene tra le alte cose `pi` (approssimazione del pi greco), `sin` e `cos` (funzioni teigonometriche seno e coseno). Per utilizzale basta

- Importare il modulo all'inizio del programma, ad esempio `import math`
- Usarne le funzionalità, ad esempio `math.pi` o `math.cos`

[3]:
```
import math
```

```
print(math.pi)

x = math.pi * math.sin(0.4)
print(x)
```

```
3.141592653589793
1.2233938033699718
```

Come dice il nome stesso, il modulo `math` contiene molte funzioni e costanti matematiche. Dal python interattivo potere vedere l'help in linea. Sia il modulo che le funzioni contenute hanno un help.

[5]:
```
help(math.cos)
```

```
Help on built-in function cos in module math:

cos(x, /)
    Return the cosine of x (measured in radians).
```

[ ]:
```
help(math)
```

Se volete usare un modulo ricordatevi di importarlo prima di riferirvi ad esso, altrimenti il programma produrrà un `NameError`

[7]:
```
print(sys.version)
```

[8]:
```
import sys
print(sys.version)
```

```
3.10.3 (main, Mar 17 2022, 00:25:45) [GCC 9.4.0]
```

0.1.2 Altre sintassi

Se importate in questo modo

[ ]:
```
import math
```

allora potrete aggedere a `math` e alle sue funzioni e constanti utilizzando il prefisso `math.` così come abbiamo visto prima. Tuttavia è possibile usare la sintassi

[ ]:
```
from math import cos
```

che va messa sempre all'inizio del programma. Questa sintassi permette di usare la funzione `cos` senza dover usare il prefisso `math.`. Fate attenzione che in questo caso non avete accesso al modulo `math`, ma solo alla funzione `cos`.

Le due sintassi possono essere mescolare

[ ]:
```
import math
from math import cos,sin

print ( math.cos(0.0) * cos(math.pi) + sin(0.5 * math.pi) )
```

Esercizio: Esplorate il modulo `math` e scrivere un programma che usi una delle sue funzioni diversa da `sin` e `cos`.

0.1.3 Installazione di librerie

Ci sono moltissimi moduli e librerie aggiuntive che non sono incluse con l'installazione di base di Python, e.g., `matplotlib` per fare i grafici, o altre librerie per il calcolo scientifico. Una lista la si può trovare su Python Package Index.

Se si utilizza Python a linea di comando è possibile installare qualunque di queste librerie con il comando

`python3 -m pip install <nomedellalibreria>`

Ad esempio si può scrivere `python3 -m pip install matplotlib`, e se l'installazione va a buon fine sarà possibile usare `import matplotlib` nei propri programmi.

Le funzionalità di queste librerie e il loro uso è solitamente descritto nella rispettiva documentazione.

0.1.4 Riassunto

Abbiamo visto - come **importare** un modulo - come **utilizzare** le sue funzionalità - come **installare** moduli aggiuntivi