

Usare e scrivere moduli Python

Informatica@DSS 2025/2026

Massimo Lauria <massimo.lauria@uniroma1.it>
<https://massimolauria.net/informatica2025/>

Modulo

Un file python è un **modulo**, ovvero un'unità che contiene funzioni e variabili pronte per essere riutilizzate.

```
import math  
print(math.pi * math.sin(0.4))
```

1

2

1.2233938033699718

I moduli python sono documentati

```
import math 1  
help(math) 2  
3
```

Help on module math:

NAME

math

MODULE REFERENCE

<https://docs.python.org/3.6/library/math>

The following documentation is automatically generated from the Python source files. It may be incomplete, incorrect or include features that are considered implementation detail and may vary between Python implementations. When in doubt, consult the module reference at the location listed above.

<.. TANTE ALTRE INFORMAZIONI....>

Anche le funzioni sono documentate

```
import math  
help(math.log)
```

1
2

Help on built-in function log in module math:

```
log(...)  
    log(x[, base])
```

Return the logarithm of x to the given base.
If the base not specified, returns the natural
logarithm (base e) of x.

Spazio dei nomi

In ogni punto e momento del programma esiste uno

spazio dei nomi

- ▶ nomi delle variabili e funzioni definite, moduli importati
- ▶ ad ogni nome corrisponde una sola entità python

```
temp = 4.2 1
def temp(): 2
    return 5 3
4
5
print(type(temp)) 6
```

```
<class 'function'>
```

Usare i moduli inclusi in python

```
import nome_modulo
```

1

- ▶ nome_modulo va nello spazio dei nomi
- ▶ si accede via nome_modulo alle sue funzioni

```
import math  
print(math.pi)  
print(math.cos(0.3))
```

1

2

3

```
3.141592653589793  
0.955336489125606
```

Importare delle funzioni da un modulo

```
from nome_modulo import fun1  
from nome_modulo import fun2
```

1

2

```
from nome_modulo import fun1, fun2
```

1

- ▶ fun1, fun2 va nello spazio dei nomi
- ▶ nome_modulo **non è accessibile** nel programma

```
from math import pi,cos  
print(pi)  
print(cos(0.3))
```

1

2

3

```
3.141592653589793  
0.955336489125606
```

```
>>> print(math.cos(0.4))
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'math' is not defined
```

```
>>> print(cos(0.4))
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'cos' is not defined
```

```
>>> from math import sin,cos
```

```
>>> print(math.cos(0.4))
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'math' is not defined
```

```
>>> print(cos(0.4))
0.9210609940028851
```

```
>>>
```

Riassumendo

```
import nome_modulo
```

1

- ▶ `nome_modulo` va nello spazio dei nomi
- ▶ si accede via `nome_modulo` alle sue funzioni

```
from nome_modulo import fun1, fun2
```

1

- ▶ `fun1`, `fun2` va nello spazio dei nomi
- ▶ `nome_modulo` **non è accessibile** nel programma

Scrivere moduli

Scriviamo il nostro file `primomodulo.py`

```
def quadrato(x): 1
    return x**2 2
3
def cubo(x): 4
    return x**3 5
6
# Un po' di codice di prova per testare 7
print('codice di prova interno al modulo') 8
print(cubo(2)+quadrato(3)) 9
```

```
$ python3 primomodulo.py
codice di prova interno al modulo
17
```

Usiamo `primomodulo.py` come un modulo

```
import primomodulo  
  
print("Codice principale:")  
print(primomodulo.cubo(3))
```

1
2
3
4

```
codice di prova interno al modulo  
17  
Codice principale:  
27
```

Possiamo riutilizzare le funzioni di `primomodulo.py` !

Importare un modulo esegue **tutto** il suo codice. Ma in questo caso il codice di prova ci disturba!

Scriviamo secondomodulo.py

```
def quadrato(x): 1
    return x**2 2
def cubo(x): 3
    return x**3 4

if __name__ == '__main__': 5
    print('codice di prova interno al modulo') 6
    print(cubo(2)+quadrato(3)) 7 8
```

```
$ python3 secondomodulo.py
codice di prova interno al modulo
17
```

```
import secondomodulo 1
print("Codice principale:") 2
print(secondomodulo.cubo(3)) 3
```

```
Codice principale:
27
```