

# PYTHON – Modules

---

## Introduction

Un module est un bout de code que l'on a enfermé dans un fichier.

On **emprisonne** ainsi des **fonctions et des variables ayant toutes un rapport entre elles**.

Ainsi, si l'on veut travailler avec les fonctionnalités prévues par le module (celles qui ont été enfermées dans le module), il n'y a qu'à **importer le module** et utiliser ensuite toutes les fonctions et variables prévues.

Il existe un grand nombre de modules disponibles avec Python sans qu'il soit nécessaire d'installer des bibliothèques supplémentaires.

Par exemple, le **module math contient**, comme son nom l'indique, des **fonctions mathématiques**.

## La méthode import

Lorsque vous ouvrez l'interpréteur Python, les fonctionnalités du module math ne sont pas incluses. Il s'agit d'un module, il vous appartient donc de l'importer si vous avez besoin de fonctions mathématiques.

première syntaxe d'importation.

```
import math
```

- le mot-clé import, qui signifie « importer » en anglais,
- suivi du nom du module, ici math.

Après l'exécution de cette instruction, Python importe le module math. Toutes les fonctions mathématiques contenues dans ce module sont maintenant accessibles.

## help

Afin de connaître les fonctions disponibles dans le module, il faut utiliser help.

help prend en argument la fonction ou le module sur lequel vous demandez de l'aide

L'aide est fournie en anglais.

Vous voyez la description du module.

Ensuite se trouve une liste des fonctions, chacune étant accompagnée d'une courte description.

Tapez

- Q pour revenir à la fenêtre d'interpréteur,
- Espace pour avancer d'une page,
- Entrée pour avancer d'une ligne.

```
import math
help(math)
```

Résultat dans la console :

```
Help on built-in module math:
NAME
    math
DESCRIPTION
    This module is always available. It provides access to the
    mathematical functions defined by the C standard.
FUNCTIONS
    acos(x, /)
        Return the arc cosine (measured in radians) of x.

    acosh(x, /)
        Return the inverse hyperbolic cosine of x.

    asin(x, /)
        Return the arc sine (measured in radians) of x.
    ...
```

Vous pouvez également passer un nom de fonction en paramètre de la fonction help.

Ne mettez pas les parenthèses habituelles après le nom de la fonction.

```
import math
help("math.sqrt")
```

Résultat dans la console :

```
Help on built-in function sqrt in math:
math.sqrt = sqrt(x, /)
    Return the square root of x.
```

sqrt renvoie la racine carrée de x

## Appeler une fonction du module

Pour appeler une fonction du module, il faut taper

- le nom du module
- suivi d'un point « . »
- le nom de la fonction.

```
import math
print(math.sqrt(16))
```

Résultat dans la console :

```
4.0
```

la fonction sqrt du module math renvoie la racine carrée du nombre passé en paramètre.

## Une autre méthode d'importation : from ... import ...

Il existe une autre méthode d'importation, qui nous permet **d'importer que la fonction dont nous avons besoin**, au lieu d'importer tout le module.

Dans le module math , si nous avons uniquement besoin, dans notre programme, de la fonction renvoyant la valeur absolue d'une variable.

```
from math import fabs
print(fabs(-5))
```

Résultat dans la console :

```
5.0
```

on ne met plus le préfixe math. devant le nom de la fonction : nous l'avons importée avec la méthode from qui charge la fonction depuis le module indiqué et la place dans l'interpréteur au même plan que les fonctions existantes.

fabs est donc au même niveau que les fonctions principales

Vous pouvez appeler toutes les variables et fonctions d'un module en tapant « \* » à la place du nom de la fonction à importer

```
from math import *
print(sqrt(4))
print(fabs(5))
```

Résultat dans la console :

```
2.0
5.0
```


À la ligne 1 de notre programme, l'interpréteur a parcouru toutes les fonctions et variables du module math et les a importées directement

Avantage : on économise la saisie systématique du nom du module en préfixe de chaque fonction.

Inconvénient : si on utilise plusieurs modules de cette manière et qu'il existe dans deux modules différents deux fonctions portant le même nom, l'interpréteur ne conservera que la dernière fonction appelée (il ne peut y avoir deux variables ou fonctions portant le même nom)

## Modules connus

Les modules standards les plus importants sont :

- **cgi** : assure la connexion du script Python avec l'interface **CGI** , laquelle permet de dialoguer avec le serveur web
- **math** : fournit l'accès aux fonctions mathématiques
- **os** : contient tous les appels système habituels que vous utilisez dans les programmes C et les scripts Shell. Ses appels traitent les répertoires, les processus, les variables Shell, etc.
- **pickle** : permet de sauvegarder une ou plusieurs variables dans un fichier et de récupérer leurs valeurs ultérieurement
- **random** : regroupe des fonctions permettant de simuler le hasard
- **re** : permet de manipuler des expressions régulières en Python
- **socket** : contient toutes les fonctions nécessaires à l'utilisation de **socket** 
- **sys** : contient des éléments se rapportant à l'exécution en cours
- **time** : permet, entre autres, d'obtenir la date et l'heure de votre système

## Créer son propre module

Vous pouvez créer votre propre module, le mettre dans un fichier.

**monmodule.py** : fichier qui contient le module :

```
def mafonction(nb):
    print(nb)
```

Vous pourrez alors l'importer depuis un autre fichier *contenu dans le même répertoire* en précisant le nom du fichier (sans l'extension.py)

**appelmodule.py** : fichier qui teste le module :

```
from monmodule import *
mafonction(4)
```

- Au moment d'importer votre module, Python va lire (ou créer si il n'existe pas) un fichier .pyc.
- À partir de la version 3.2, ce fichier se trouve dans un dossier \_\_pycache\_\_.
- Ce fichier est généré par Python et contient le code compilé de votre module.
- Il ne s'agit pas réellement de langage machine mais d'un format que Python décode un peu plus vite que le code que vous pouvez écrire.
- Python se charge lui-même de générer ce fichier.