Définition

Les modules sont des **fichiers python contenant des fonctions**. Certains modules sont inclus de base avec python, d'autres peuvent être téléchargés depuis internet.

Les modules de la librairie standard de Python

Ces modules sont inclus avec Python lors de son installation dans votre système. Pour les utiliser, vous n'avez rien à télécharger mais vous devrez les **inclure** dans votre fichier Python avec le mot clé `import`.

Le module `random`

Permet de générer des données aléatoires.

```
import random as rd

print(rd.randint(1, 5))         # génère un entier aléatoire entre 0 et 5 inclus
print(rd.uniform(0, 1))         # génère un float aléatoire entre 0 et 1 inclus
print(rd.randrange(0, 101, 10)) # génère un entier aléatoire entre 0 et 101 exclus
avec un pas de 10
```

On remarque que l'opérateur `.` signifie dans `random.randint()` que l'on utilise la fonction `randint` du module `random`. On peut se passer de cette syntaxe de la manière suivante.

```
from random import randint

print(randint(0, 5))
```

> ⚠️ Synthaxe déconseillée
>
> Sauf circonstance exceptionnelle, cette dernière synthaxe est à éviter pour deux raisons. Si vous créez une variable portant le même nom que la fonction importée vous allez écraser cette fonction. En second lieu, les bonnes pratiques Python précisées dans The Zen Of Python établissent qu'il faut toujours privilégier l'explicite à l'implicite, quitte à écrire plus de code. Ici en précisant à chaque fois le nom du module quand on utilise une de ses fonctions, on est plus explicite, et notre code Python est de meilleure qualité.

Le module `os`

Permet d'accéder aux ressources du système d'exploitation.

```
import os

chemin = "/home/digi/Documents"

# Création d'un nouveau dossier dans mes Documents
# Documents/nouveau_dossier/os
# La fonction join gére les '\' sous windows ou les '/' sous linux et mac
dossier = os.path.join(chemin, "nouvau_dossier", "os") #
"/home/digi/Documents/nouveau_dossier/os"
if not os.path.exists(dossier):
    os.makedirs(dossier)

# Suppression de l'architecture de dossier nouvellement créée
if os.path.exists(dossier):
    os.removedirs(dossier)
```

Le module `pathlib`

Disponible depuis la 3.6, il propose une modernisation du module `os` avec une meilleure synthaxe.

```
from pathlib import Path

chemin  = "/home/user/Documents/doc.txt"

p = Path(chemin)          # Création d'un objet Path
p.touch(exist_ok=True)    # Création du fichier txt dans le dossier Documents
p.write_text("Coucou !") # Écrit du texte dans le fichier
print(p.read_text())      # Lecture du contenu du fichier

print(p.name)             # Print le nom du fichier
print(p.suffix)           # Print l'extension du fichier
```

La fonction `dir()`

Permet de dresser une liste des fonctions disponibles dans un module. Utile pour savoir quelles fonctionnalités offre un module.

```
import random

print(dir(random)) # Les fonctions avec des underscores _ ne sont pas utilisables
```

> 📌 Je vous renvoi à la fonction help pour obtenir une aide sur la fonction particulière d'un module.

Créer son propre module

Pour créer son propre module il suffit de créer un nouveau fichier python. Dans le fichier où on veut importer ce module, un simple `import nom_du_fichier` suffit à importer tout son contenu.

> ⚠️ Nommage du fichier
>
> Attention à ne pas nommer votre fichier avec le même nom qu'un module standard de python comme `random` ou `math`, sous peine d'écraser tout leur contenu.

Exercice: "12. Les modules Python"