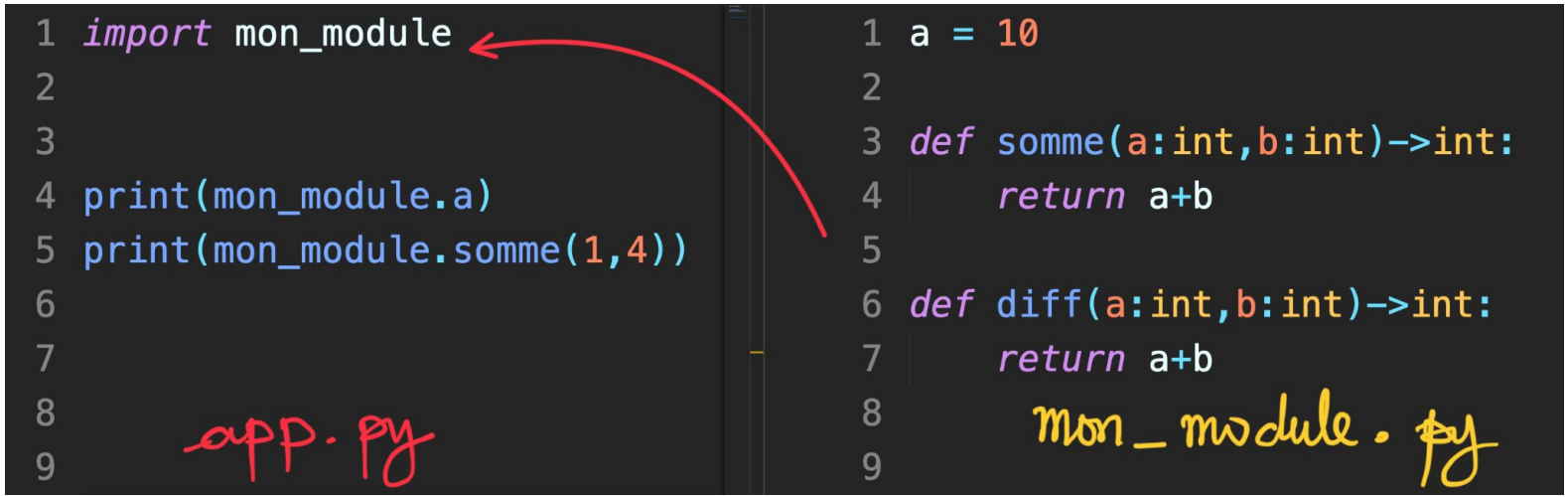
Les Modules

Les modules

- Un module est un fichier contenant du code python, comme le module random, ou le module os,..
- Pour utiliser le code se trouvant dans un module, il faudra l'importer dans notre script comme suit:
  - ***import nom_module***
    - Avec cette importation on doit mettre le nom du module suivi d'un point puis du nom de la fonction à utiliser : ***nom_module.nom_fonction***
  - ***from nom_module import nom_fonction***
    - Avec cette méthode on importe que la fonction que l'on veut utiliser (attention à ne pas avoir une autre fonction ou variable de même nom)
  - ***from nom_module import ****
    - Avec cette méthode, toutes les fonctions du module seront importées. (**syntaxe à éviter**)

exemple :

Les modules



Attention *: ne pas créer un module avec un nom qui existe déjà en python.*

Les modules

- **__name__ (1):**
  - Il se peut qu'on ait un module dans lequel y'a des fonctions qu'on à tester à l'intérieur du module même. Dans ce cas, lors de l'import de ce module, ces fonctions seront aussi exécutées automatiquement.
  - Pour éviter que cela se produise, on fait ces tests uniquement que si on lance directement le script du module : utilisation de la variable ***__name__***

1. ***A éviter :*** 👎

```python
import mon_module

print(__name__)

# Au lancement de ce fichier,
# la console affiche :

# 21
# mon_module
# __main__
```

```python
a = 10

def somme(a:int,b:int)->None:
    print (a+b)

somme(10,11)
print(__name__)
```

Les modules

- **__name__ (2):**
  - Il se peut qu'on ait un module dans lequel y'a des fonctions qu'on à tester à l'intérieur du module même. Dans ce cas, lors de l'import de ce module, ces fonctions seront aussi exécutées automatiquement.
  - Pour éviter que cela se produise, on fait ces tests uniquement que si on lance directement le script du module : utilisation de la variable ***__name__***

2. A faire : 👍

```python
import mon_module

mon_module.somme(10,11)

# Au lancement de ce fichier,
# la console affiche :

# 21

```

```python
a = 10

def somme(a:int,b:int)->None:
    print (a+b)

if __name__ == "__main__":
    somme(10,11)
    print(__name__)

```

Les modules

- **ajouter un module dans PYTHONPATH:**
  - On utilise la variable **path** du module **sys** (***sys.path***) pour voir la liste des chemins ou python va chercher les modules. (*on peut utiliser pprint() du module pprint pour l'affichage de la liste*)
  - on y ajoute un fichier de module, puis on pourra l'utiliser dans nos script

```
1 import sys
2 from pprint import pprint
3 pprint(sys.path)
4
```

```
['/Users/macbookpro/Desktop/python/exemples/modules',
 '/Library/Frameworks/Python.framework/Versions/3.10/lib/python310.zip',
 '/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10',
 '/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/lib-dynload',
 '/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packages']
```

Les modules

- **ajouter nos propres chemins dans PYTHONPATH:**
    - Créer un dossier dans lequel on met nos modules
    - copier le chemin du dossier
    - ajouter ce chemin dans le PYTHONPATH
    - importer nos modules.

```
1 import sys
2 sys.path.append('/Chemin_vers_le_dossier_module')
3
4 import mon_module
```

Remarque *: l'inconvénient c'est qu'on doit réécrire les deux premières lignes dans tous les scripts qui doivent importer nos modules.* 😩 (*Il faut penser à ajouter nos dossier dans la variable d'environnement définitivement*)