# Chapitre 4 : Modularité en python

Les modules et packages permettent de découper un programme sur plusieurs fichiers et de regrouper ceux-ci de manière logique.

## 1 Python en dehors des notebooks

Il est possible d'exécuter du code Python en dehors des notebooks.

**Python interactif** Dans un terminal, la commande `python` transforme ce terminal en une console python interactive. Il est possible de saisir des instructions python. À chaque instruction saisie, celle-ci est exécutée.

**Fichiers de code python** Il est possible d'écrire du code Python dans des fichiers que l'on peut exécuter ensuite dans un terminal. Pour écrire du code python dans un fichier, il faut utiliser un éditeur de texte (**pas open office**) tel que `atom`, `gedit` ou `vscode`. Pour exécuter le fichier, il faut alors dans le terminal taper la commande :

```
python nom_du_fichier_python
```

**Remarques :**

- les fichiers de code python ont généralement l'extension .py ;
- pour qu'il n'y ait pas de problème d'accents, il faut que le fichier soit encodé en utf-8 ;
- si votre fichier commence par la ligne

  ```
  #!/usr/bin/env python
  ```

  et qu'il est exécutable, vous pouvez alors directement lancer le script python en tapant dans le terminal :

  ```
  ./nom_du_fichier_python
  ```

## 2 Création de modules

Un module est un fichier contenant du code python, généralement des définitions de fonctions, qui peut par la suite être inclus dans un autre script python (qui pourra appeler les fonctions définies dans le module). Les scripts[1] qui appeleront les fonctions définies dans le module devront alors inclure (avant tout appel de fonction) l'instruction :

```
from nom_du_module import *
```

[1] On suppose pour l'instant que les scripts sont dans le même répertoire que les modules.

**Remarque :** on ne met pas l'extension .py dans le nom du module lors de l'import.

**Remarque :** on peut importer seulement une partie des fonctions du module en spécifant après le mot clé `import` la liste des fonctions que l'on veut importer. Par exemple, si l'on souhaite importer uniquement la fonction `f` alors qu'il y a plusieurs fonction dans le module `mod`, on écrit l'import de la manière suivante : `from mod import f`.

## 3 Utiliser des modules comme des scripts

Un module est utilisé dans d'autres scripts python. Cependant, il est possible de mettre du code qui sera exécuté uniquement si le module est le fichier exécuté (autrement dit, si l'on a fait `python nom_du_module`). Ceci se fait en utilisant :

```
if __name__ == '__main__':
    # Code qui sera exécuté uniquement si le module est exécuté
    # directement (pas inclus dans un autre script)
```

Ceci permet notamment d'insérer des tests pour des fonctions définies au sein du module. Ces tests seront exécutés lorsque le script sera exécuté (et non inclus).

## 4 Packages en Python

Les packages en Python permettent de regrouper logiquement des modules ensemble. La création de packages est simple puisqu'un package correspond à un répertoire : les fichiers (et répertoires) à l'intérieur correspondent aux modules (et sous-packages) contenus dans le package. Dans un script, pour utiliser un module du package, il faut importer le module selon[2] :

```
from package.module import *
```

## 5 Utiliser des modules et packages qui ne sont pas dans le répertoire courant

Si le module ou le package n'est pas dans le même répertoire que le script (ou notebook) qui les utilise, alors python ne le trouve pas et cela génère une erreur. Pour corriger ce problème, il faut ajouter le répertoire contenant le module (ou package) dans le chemin de Python. Pour cela on utilise le package `sys` qui permet d'ajouter des nouveaux chemins :

```
#chemin doit être une chaîne de caractères correspondant au chemin
#(relatif ou absolu) du répertoire
#contenant le module ou package que l'on souhaite utiliser
import sys
sys.path.append(chemin)
#On importe ensuite le module ou package qui est dans le répertoire chemin
import ...
```

[2] On suppose pour l'instant que le script est dans le même répertoire que le package.