Création d'un module

Nous avons déjà rencontré les modules : `numpy` et `matplotlib` par exemple sont très souvent utilisés en calcul scientifique. Nous allons apprendre maintenant à fabriquer notre propre module.

Rappelons l'intérêt de faire son module :

- cela permet de ranger proprement de nombreuses fonctions que l'on utilise souvent
- de partager son code avec des utilisateurs (`python` a vocation à être partagé)

Pour rendre plus pratique ce cours, nous allons créer un module permettant de gérer et traiter des fonctions : affichage, évaluation vectorielle, tracé, recherche de zéro, intégrale, ...

In [1]:

Premier pas : un fichier `py`

Pour démarrer, nous commençons par créer un répertoire nommé `mon_module` dans lequel nous plaçons un fichier `evaluation.py` .

In [1]:
```
import mon_module_01.evaluation as test
help(test)
```

```
Help on module mon_module_01.evaluation in mon_module_01:

NAME
    mon_module_01.evaluation - Documentation du module évaluation

FUNCTIONS
    evaluation(f, x)
        evaluation de f au point x

FILE
    /Volumes/HD2/Enseignement/depots/l1_mdd153/2020-2021/Cours/mon
_module_01/evaluation.py
```

In [2]:
```
f = lambda x: x*x
print(test.evaluation(f, 2))
#print(test.evaluation(f, [1, 2]))
```

```
4
```

In [3]:
```
help(test.evaluation)
```

```
Help on function evaluation in module mon_module_01.evaluation:

evaluation(f, x)
    evaluation de f au point x
```

Testons notre module comme un script

Il est possible de tester les fonctions définies dans le module. Mais si l'on met du code directement, il est aussi exécuté à l'importation du module.

Pour mettre du code qui n'est exécuté que lorsque le fichier est lu comme un script et pas comme un module, il faut ajouter le test :

```
if __name__ == "__main__":
    ...
```

In [2]:
```
print(test.__name__)
```

```
mon_module_01.evaluation
```

Plusieurs fichiers : `__init__.py`

Ajoutons à présent un autre fichier qui permet d'intégrer une fonction. Nous devons ajouter un fichier `integration.py` .

Nous mettons la documentation dans le fichier `__init__.py` . Nous pourrons également mettre du code dans ce fichier qui sera exécuté au moment de l'importation du module.

In [1]:
```
import mon_module_02 as mm

help(mm)
```

```
Je passe dans le __init__
Help on package mon_module_02:

NAME
    mon_module_02 - Documentation du module complet

DESCRIPTION
    C'est un module pour gérer les fonctions

PACKAGE CONTENTS


SUBMODULES
    temp
    temp2

FILE
    /Volumes/HD2/Enseignement/depots/l1_mdd153/2020-2021/Cours/mon
_module_02/__init__.py
```

In [2]:
```
f = lambda x: x*x
print(mm.evaluation(f, 2))
#print(mm.evaluation(f, [1, 2]))
```

```
4
```

In [3]:
```
print(mm.rect_gauche(f, 0, 1))
print(mm.rect_droite(f, 0, 1))
#print(mm.trapeze(f, 0, 1))
```

```
0.3328335000000004
0.3338335000000004
```

Version du module

Il est possible et recommandé de mettre une version sur son module et également d'utiliser un logiciel de gestion de versions comme `git` .

Cela se fait classiquement dans un fichier `version.py` .

In [1]:
```
import mon_module_02 as mm

help(mm)
```

```
Help on package mon_module_02:

NAME
    mon_module_02 - Documentation du module complet

DESCRIPTION
    C'est un module pour gérer les fonctions

PACKAGE CONTENTS
    version

SUBMODULES
    temp
    temp2

VERSION
    0.0.1

FILE
    /Volumes/HD2/Enseignement/depots/l1_mdd153/2020-2021/Cours/mon
_module_02/__init__.py
```

Pour aller plus loin

On peut évidemment importer d'autres modules quand c'est nécessaire. Il est important de bien structurer son code pour qu'il soit lisible et facilement modifiable.

In [ ]:
```
```