Notes sur Python

Overview du langage

- Langage interprété
- Paradigme : multiparadigme (orienté objet, fonctionnel (lambda calcul, etc.) entre autres)
- Typage :
  - dynamique + idéologie de duck-typing ;
  - typage fort (erreurs de type identifiées et signalées en runtime, peu de conversion de types implicite)
    - `print('The number is ' + 5) # erreur`
    - `print('The number is ' + str(5))`
- Syntaxe : basée sur l'indentation
- Dernière version du langage : Python 3 (beaucoup de changements par rapport à Python2.7, pas de back-compatibility)

Importer des librairies

- ```
  import library
  a = library.multiply(b,c)
  z = library.subtract(x,y)
  ```
- ```
  import library as lb
  a = lb.multiply(b,c)
  z = lb.subtract(x,y)
  ```
- ```
  from library import multiply
  a = multiply(b,c)
  ```
- ```
  from library import *
  a = multiply(b,c)
  z = subtract(x,y)
  ```

Comment créer et importer son propre module python ?

- Créer un script python (*myscript.py*)
- Dans un autre script :
  ```
  import sys
  sys.append('/dossier/contenant/votre/script')
  import myscript
  ```

NOTE : au premier import un fichier *myscript.pyc* va être créé dans le même dossier.

Chaînes de caractères

- Guillemets simples (`'Paul\nMarie\tAnnick\n"Yu"'`)
- Guillemets doubles (`"Don't panick! \n Calm down."`)

- `Raw string (r"Don't panick! \n Calm down.")`

Détails, conseils

- La section du code à être exécutée seulement si le script est lancé directement :
```
if __name__ == '__main__':
      print ‘This is a test/sandbox section’
```
- Warnings :
```
import warning
warn('message')
```
- Parser les arguments de la ligne de commande :
```
import argparse
parser = argparse.ArgumentParser()
## préciser un argument de ligne de commande
## --input, -i : les alias du nom de l’argument
## dest : le nom de la variable dans laquelle la valeur de cet argument
##          va être stockée
parser.add_argument('-i', '--input', metavar='INPUT', default = None,
                        dest='input_file',
                        help="""Path to the input file""")
## parser les arguments de ligne de commande
args = parser.parse_args()
## récupérer un argument de ligne de commande
path2inputfile = args.input_file
```
- Manipulation de fichiers
  - Méthodes standards
```
INPUT = open(input_file,'r')
```
    - ```
      for line in INPUT:
      if line:
         ## effacer les retours de ligne
         line = line.rstrip('\n\r')
      ```
    - `file_contents = INPUT.read()`
    - `file_lines = INPUT.readlines() ## lire dans une liste python`
    - ```
      ## lire ligne par ligne
      first_line = INPUT.readline() ; second_line = INPUT.readline() ;
      ```
  - NumPy
```
import numpy as np
X = numpy.fromfile(file, dtype=float, count=-1, sep='')
```