Module

In Python, a module is a file containing Python definitions and statements. It can define functions, classes, and variables, and can also include runnable code. Grouping related code into a module makes the code easier to understand and use, and it also makes the code logically organized.

Common Modules

math	fonctions et constantes mathématiques de base
sys	interaction avec l'interpréteur Python, passage d'arguments
os	dialogue avec le système d'exploitation
random	génération de nombres aléatoires
time	accès à l'heure de l'ordinateur et aux fonctions gérant le temps.
urlib	récupération de données sur internet depuis Python
Tkinter	interface python avec Tk. Création d'objets graphiques
re	gestion des expressions régulières
Numpy	manipulations de vecteurs et de matrices, algèbre linéaire
Biopython	recherche dans les banques de données biologiques, manipulation de séquences ou de structures de biomolécules
matplotlib	eprésentations graphiques : courbes, nuages de points, diagrammes en bâtons...
pandas	analyse de données

Load & usage

```
# Method 1
import module
import module as xx
module.fu nct ion()
# Method 2
from module import function
from module import function1,
function2
function()
# Method 3
from module import *
function()
# to unload a module
del module
# to get help
help(m odule)
/ text # to find the key word
Q # exit
```

To avoid confusion between functions or constants with the same name from different modules, it is advisable to load the module by calling its name while using a function or a constant (Method 1) specially for the module math

module Random

random.randon()	a random float uniformly in the half-open range 0.0 <= X < 1.01
random.randint(a,b)	It returns a random integer from the specified range [a, b], inclusive
random.choice(seq)	returns a random element from the non-empty sequence seq
random.shuffle(x)	randomly shuffles the elements of the list x in place
random.randrange(start, stop, step)	returns a randomly selected element from the specified range

module Random (cont)

random.choices(seq, weight = c() , k =)	It returns a list with the randomly selected element from the specified sequence

choice(seq) & choices (seq, weight, k) The sequence can be a string, a range, a list, a tuple, or any other kind of sequence

common functions

- Le module **sys** pour récupérer les arguments passés à un script Python. **python file.py xx xx** dans un shell **sys.argv** dans le py fircher renvoie une liste des arguments y compris le nom du script lui-même sys.argv[0]. accéder aux arguments avec **sys.argv[1], sys.argv[2]**
- Le module **os** gère l'interface avec le système d'exploitation.

La fonction **os.path.exists()** vérifie la présence d'un fichier sur le disque dur. Il renvoie True ou False

La fonction **sys.exit("message")** est pour quitter le programme. Il est possible de renvoyer une message dans le shell

La fonction **os.getcwd()** renvoie le répertoire (sous forme de chemin complet) depuis le directoire du travail

os.getdir() same to ls in Linux and dir() in R

- time.sleep()
- math.sqrt()

Exemple for chap_7 reading & editing

```
# Iteration of a file using
readline() & while loop
with open(file, " r") as filin:
       line = filin.r ea dline()
          while line != " ":
                  pri nt( line)
                 line =
filin.r ea dline()
# keep a line using readline() &
next()
with open(' fil e.txt', 'r') as
f:
```

Exemple for chap_7 reading & editing (cont)

```
>     line1 = f.readline()
    next(f)
    line3 = f.readline() # return the line after
next
# Iteration of a list resturned by readlines()
with open(file, "r") as filin:
    lines = filin.readlines() # [s:f] to keep line
    for line in lines:
        print(line)
# Iteration directly for a file
with open(file, "r") as filin:
    for line in filin:
        print(line)
# change the position
with open(file, 'r') as f:
    f.seek(20)
    print(f.tell()) # Prints the new position
with open(file, 'r') as f:
    f.seek(-10, 2)
# writing
with open(file, "w") as filout:
     for element in my_list:
         filout.write(f"{element}\n")
# \r
import time
total = 10
for i in range(total + 1):
    progress = i / total * 100
    bar = '#' i + '-' (total - i)
    print(f'Progress: [{bar}] {progress:.1f}%',
end='\r')
    time.sleep(0.5)
print('\nDone!')
```