NSI	Modules et bibliothèques	Cours 3

1. Créer un module :

Commençons par constituer un fichier contenant diverses fonctions utiles. Une fonction `maximum` qui renvoie le maximum de deux nombres et une fonction `minimum` qui renvoie le minimum de deux nombres.

```
def maximum (a,b):
    """a et b sont des nombres de type int ou float la fonction
renvoie le
    maximum des deux nombres"""
    if a > b:
        return a
    else:
        return b

def minimum (a,b):
    """a et b sont des nombres de type int ou float la fonction
       renvoie le
    minimum des deux nombres"""
    if a < b:
        return a
    else:
        return b
```

Ce fichier est enregistré sous le nom **mes_fonctions.py**.

Afin de l'utiliser dans un autre fichier pour écrire une fonction qui renvoie le maximum de quatre nombres, nous commençons par l'importer.

```
from mes_fonctions import maximum

def maxi(a,b,c,d):
    """a,b,c et d sont des nombres de type int ou float
     la fonction renvoie le maximum des quatre nombres"""
    m1 = maximum(a,b)
    m2 = maximum(c,d)
    m = maximum (m1,m2)
    return m
```

Attention, ce fichier et le fichier **mes_fonctions.py** doivent être placés dans le même répertoire.

2. Import de modules :

On peut importer une seule fonction :

```
>>> from math import cos
>>> cos(35)
-0.9036922050915067
>>> sin(45)
Traceback (most recent call last):
  File "<pyshell#77>", line 1, in <module>
    sin(45)
NameError: name 'sin' is not defined
```

On peut importer toutes les fonctions du module :

```
>>> from math import *
>>> cos(56)
0.853220107722584
>>> sin(34)
0.5290826861200238
```

Attention : L'importation de toutes les fonctions avec * est fortement déconseillée. En effet, elle ne permet pas d'avoir une vision claire des fonctions qui ont été importées. Ceci est donc une source potentielle d'erreurs.
En effet, si deux modules ont une fonction du même nom, on ne saura pas de quel module la fonction appelée est extraire.

On peut importer tout le module, dans ce cas il préciser le nom du module devant les fonctions :

```
>>> import math
>>> math.cos(45)
0.5253219888177297
```

Pour faire plus simple, on peut importer le module avec un alias :

```
>>> import math as m
>>> m.cos(56)
0.853220107722584
```

3. Quelques modules utiles :

Le langage Python possède de nombreux modules.

3.1. Module math :

La fonction **dir** permet d'explorer le contenu du module **math.**
L'instruction **import math** est obligatoire.

```
>>>import math
>>> dir (math)
['__doc__', '__loader__', '__name__', '__package__', '__spec__',
'acos', 'acosh', 'asin', 'asinh', 'atan', 'atan2', 'atanh', 'ceil',
'copysign', 'cos', 'cosh', 'degrees', 'e', 'erf', 'erfc', 'exp',
'expm1', 'fabs', 'factorial', 'floor', 'fmod', 'frexp', 'fsum',
'gamma', 'gcd', 'hypot', 'inf', 'isclose', 'isfinite', 'isinf',
'isnan', 'ldexp', 'lgamma', 'log', 'log10', 'log1p', 'log2',
'modf', 'nan', 'pi', 'pow', 'radians', 'sin', 'sinh', 'sqrt',
'tan', 'tanh', 'tau', 'trunc']
```

3.2. Module random :

Pour plus d'informations sur ce module : **https://docs.python.org/3/library/random.html**

```
>>> import random
>>> dir(random)
['BPF', 'LOG4', 'NV_MAGICCONST', 'RECIP_BPF', 'Random', 'SG_MAGICCONST',
'SystemRandom', 'TWOPI', '_BuiltinMethodType', '_MethodType', '_Sequence',
'_Set', '__all__', '__builtins__', '__cached__', '__doc__', '__file__',
'__loader__', '__name__', '__package__', '__spec__', '_acos', '_bisect',
'_ceil', '_cos', '_e', '_exp', '_inst', '_itertools', '_log', '_pi',
'_random', '_sha512', '_sin', '_sqrt', '_test', '_test_generator',
'_urandom', '_warn', 'betavariate', 'choice', 'choices', 'expovariate',
'gammavariate', 'gauss', 'getrandbits', 'getstate', 'lognormvariate',
'normalvariate', 'paretovariate', 'randint', 'random', 'randrange', 'sample',
'seed', 'setstate', 'shuffle', 'triangular', 'uniform', 'vonmisesvariate',
'weibullvariate']
```

```
>>> from random import randint
>>> randint(1,20)
19
>>> randint(1,20)
8
>>> randint(1,20)
12
>>> randint(1,20)
5
>>> randint(1,20)
2
```

3.3. Module Turtle :

Le module turtle permet de dessiner en programmant.
On commence par tout importer puis on utilise des instructions simples comme :

- forward(distance) , backward(distance), pour avancer ou reculer ;
- left(angle), right(angle), pour tourner à gauche ou à droite ;
- reset( ) , pour effacer ;
- goto(x,y), up( ), down( ), pour aller à un point donné, lever ou baisser le crayon ;
- color('couleur'), width(épaisseur), pour la couleur et l'épaisseur du trait ;
- write ('texte'), pour afficher du texte ;
- begin_fill(), end_fill(), pour remplir une figure fermée avec une couleur.

Le point central a pour coordonnées (0 ;0).

Observons ce que donnent les instructions suivantes

```
>>> import turtle as tt
>>> tt.forward(100)
>>> tt.goto(-80,50)
```

4. Bibliothèque Matplotlib :

Pour plus d'informations : **https://matplotlib.org/gallery/index.html**

Une bibliothèque est un ensemble de fonctions, de constantes, de types et de modules.
La bibliothèque standard de Python contient les modules **math** et **random** parmi beaucoup d'autres.
La bibliothèque **matplotlib** contient le module **pyplot** utilisé pour le tracé des courbes.

Principe pour tracer une courbe :
On précise les coordonnées des points dans 2 listes.
La première contient les abscisses des points et la seconde les coordonnées [1,2,3] [0.5,3,5]

```
import matplotlib.pyplot as plt

a=[1,2,3]
b=[0.5,45,-2.4]

plt.plot(a,b)
plt.show() #affichage de la courbe
```

```
import matplotlib.pyplot as plt

def f(x):
    return x**2-5*x+1

x=[i*0.01 for i in range(-500,1001)]
y=[f(u) for u in x]

plt.plot(x,y) #construction de la courbe
plt.grid() #quadrillage facultatif
plt.show() #affichage de la courbe
```