Stéphane KELLER – Lycée agricole Louis Pasteur

CHAPITRE 4 : LES FONCTIONS

Nous avons déjà vu beaucoup de fonctions : print(), type(), len(), input(), range()... Ce sont des fonctions pré-définies. Les fonctions écrites jusqu'à présent se contentent de réaliser des actions (afficher un message, par exemple). C'est ce l'on appelle des **procédures**.
Une vraie fonction doit, en plus, lorsqu'elle se termine, renvoyer une valeur. Comme une fonction en mathématiques qui renvoie au final l'image d'un nombre : lorsque l'on appelle f avec la valeur 3, elle renvoie le nombre $f(3)$.
Une fonction est une portion de code que l'on peut appeler au besoin (c'est une sorte de sous-programme). L'utilisation des fonctions évite des redondances dans le code : on obtient ainsi des programmes plus courts et plus lisibles.

Par exemple, nous avons besoin de convertir à plusieurs reprises des degrés Celsius en degrés Fahrenheit :

```
>>> print(100.0*9/5+32)
212.0
>>> print(37.0*9/5+32)
98.6
```

La même chose en utilisant une fonction :

```
def f(degrecelsius):
    return degrecelsius*9/5+32

print(f(100))
print(f(37))
x=233
print(f(x))

212.0
98.6
451.4
```

L'instruction def

Syntaxe :

```
def nom_fonction(parametre1, parametre2, parametre3, ...):
    """ Documentation qu'on peut écrire
    sur plusieurs lignes """     # docstring entouré de 3 guillemets (ou apostrophes)
    bloc d'instructions          # attention à l'indentation
    return resultat              # la fonction retourne le contenu de la variable resultat
```

Exemple n°1

```
# script Fonction1.py

def ma_premiere_fonction():          # cette fonction n'a pas de paramètre
    """ Cette fonction affiche 'Bonjour' """
    print('Bonjour')
    return                           # cette fonction ne retourne rien ('None')
                                     # l'instruction return est ici facultative
```

Une fois la fonction définie, nous pouvons l'appeler :

```
>>> ma_premiere_fonction()          # ne pas oublier les parenthèses ()
Bonjour
```

L'accès à la documentation se fait avec la fonction pré-définie help() :

```
>>> help(ma_premiere_fonction)      # affichage de la documentation
Help on function ma_premiere_fonction in module __main__:

ma_premiere_fonction()
    Cette fonction affiche 'Bonjour'
```

Exemple n°2

La fonction suivante simule le comportement d'un dé à 6 faces.
Pour cela, on utilise la fonction randint() du module random.

```
# script Fonction2.py
from random import randint

def tirage_de():
    """ Retourne un nombre entier aléatoire entre 1 et 6 """
    return randint(1, 6)

print(tirage_de()) #Appuyer plusieurs fois sur F5
```

```
>>>
3
>>>
```

Exemple n°3

```
# script Fonction3.py
from random import randint

def tiragede():
    """ Retourne un nombre entier aléatoire entre 1 et 6 """
    return randint(1, 6)

# début du programme
boucle = True
while boucle:
    choix = input('Touche q pour quitter ou Enter pour continuer. ')
    if choix == 'q':
        boucle = False
    else:
        print('Tirage :', tiragede())
>>>
Touche q pour quitter ou Enter pour continuer
Tirage : 5
Touche q pour quitter ou Enter pour continuer
Tirage : 6
q
>>>
```

Exemple n°4

Une fonction avec deux paramètres :

```
# script Fonction4.py
from random import randint
```

```
def tiragede2(valeurmin = 1, valeurmax = 6): #par défaut génère un entier entre 1 et 6
    """ Retourne un nombre entier aléatoire entre valeurmin et valeurmax """
    return randint(valeurmin, valeurmax)

# début du programme
for i in range(5): #génère 5 lancers
    print(tiragede2(1, 10))      # appel de la fonction avec les arguments 1 et 10
>>>
6
7
1
10
2
>>>
```

Exemple n°5
Une fonction qui retourne une liste :

```
# script Fonction5.py
from random import randint

def tiragemultiplede(nbtirage):
    """ Retourne une liste de nombres entiers aléatoires entre 1 et 6 """
    return [randint(1, 6) for i in range(nbtirage)]

# début du programme
print(tiragemultiplede(10))
>>>
[4, 1, 3, 3, 2, 1, 6, 6, 2, 5]
>>> help(tiragemultiplede)
Help on function tiragemultiplede in module __main__:

tiragemultiplede(nbtirage)
    Retourne une liste de nombres entiers aléatoires entre 1 et 6
```

Exemple n°6
Une fonction qui affiche la parité d'un nombre entier.
Il peut y avoir plusieurs instructions return dans une fonction.
L'instruction return provoque le retour immédiat de la fonction.

```
# script Fonction6.py

def parite(nombre):
    """ Affiche la parité d'un nombre entier """
    if  nombre % 2 == 0:   # L'opérateur % donne le reste d'une division
        return 'est pair'
    else:
        return 'est impair'

# début du programme
nombre = int(input('Entrer un nombre : '))
print(nombre, parite(nombre))
>>>
```

```
13 est impair
24 est pair
```

Un programme doit avoir la structure suivante :
1°) Importation des librairies et fonctions prédéfinies nécessaires au programme.
2°) Définition de nos fonctions personnelles.
3°) Le corps principal du programme (Main).

LES FONCTIONS TESTS

I. 1°) Écrire une fonction carre() qui retourne le carré d'un nombre :

```
>>> print(carre(11.11111))
123.4567654321
```

2°) Avec une boucle while et la fonction carre(), écrire un script qui affiche le carré des nombres entiers de 1 à 100 :

```
>>>
1² = 1
2² = 4
3² = 9
...
99² = 9801
100² = 10000
Fin du programme
```

II. Écrire une fonction qui retourne l'aire de la surface d'un disque de rayon r.
Exemple :

```
>>> print(aire_disque(2.5))
19.634954084936208
```

III. 1°) Écrire, avec la boucle while, une fonction qui retourne la factorielle d'un nombre entier n positif ou nul. On rappelle que : n! = 1×2×...×(n-1)×n et 0!=1.
Comparez avec le résultat de la fonction **factorial**() du module math.
Exemple :

```
>>> print(factorielle(50))
30414093201713378043612608166064768844377641568960512000000000000
```

2°) Même programme avec une boucle for.
IV. 1°) À l'aide de la fonction randint() du module random, écrire une fonction qui retourne un mot de passe de longueur N (chiffres, lettres minuscules ou majuscules).
On donne :

```
ch =
'0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz'
>>> print(Password(10))
mHVeC5rs8P
>>> print(Password(6))
PYthoN
```

2°) Reprendre la question 1°) avec la fonction choice() du module random.
Pour obtenir de l'aide sur cette fonction :

```
>>> import random
>>> help(random.choice)
```

3°) Quel est le nombre de combinaisons possibles ?
4°) Quelle durée faut-il pour casser le mot de passe avec un logiciel capable de générer 1 million de combinaisons par seconde ?
Lien utile : www.exhaustif.com/Generateur-de-mot-de-passe-en.html

V. Écrire une fonction qui retourne une carte (au hasard) d'un jeu de Poker à 32 cartes.
On n'utilisera pas la fonction choice() du module random.
On donne :

```
liste_carte
=['7s','7h','7d','7c','8s','8h','8d','8c','9s','9h','9d','9c','Ts','Th','Td','Tc','Js','Jh','Jd','Jc','Qs','Qh','Qd',
'Qc','Ks','Kh','Kd','Kc','As','Ah','Ad','Ac'].
```

Pour information : cœur = heart ; carreau = diamond ; pique = spade et trèfle = club.

```
>>> print(tirage_carte())
7s
>>> print(tirage_carte())
Kd
```

VI. 1°) Écrire une fonction qui retourne une liste de n cartes **différentes** d'un jeu de Poker à 32 cartes.
Noter qu'une fonction peut appeler une fonction : on peut donc réutiliser la fonction tirage_carte() de l'exercice précédent.
Exemple :

```
>>> print(tirage_n_carte(2))
['As', 'Ah']
>>> print(tirage_n_carte(25))
['Jc', 'Jh', 'Tc', 'Ad', '9h', 'Qc', '8d', '7c', 'As', 'Td', '8h', '9c', 'Ad', 'Qh',
'Kc', '8s', '9h', 'Qd', 'Kh', '9h', 'Rd', 'Js', 'Ks', '7c', 'Th']
```

2°) Simplifier le script avec la fonction shuffle() ou sample() du module random.
VII. Écrire une fonction qui retourne une grille de numéros du jeu Euro Millions.
On utilisera la fonction sample() du module random.

```
>>> print(euromillions())
[37, 23, 9, 11, 49, 2, 11]
>>> print(euromillions())
[16, 32, 8, 30, 40, 6, 4]
```

VIII. 1°) Écrire une fonction qui retourne la valeur de la fonction mathématique $f(x)=27x^3 -27x^2 -18x +8$:

```
>>> print(f(0),f(1),f(0.5),f(0.25),f(0.375))
8.0 -10.0 -4.375 2.234375 -1.123046875
```

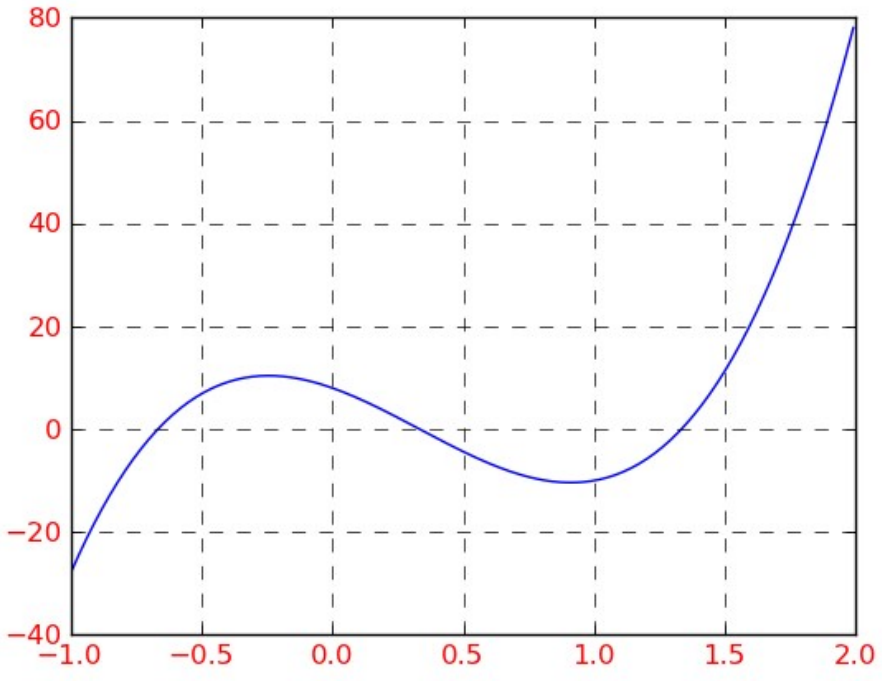
2°) On se propose de chercher les zéros de cette fonction par la méthode de dichotomie.
Écrire le script correspondant.

```
>>>
Recherche d'un zéro dans l'intervalle [a,b]
a? 0
b? 1
Précision ? 1e-12
0.5
0.25
0.375
0.3125
0.34375
0.328125
0.3359375
0.33203125
0.333984375
0.3330078125
0.33349609375
0.333251953125
```

```
...
...

>>>
```

3°) Chercher tous les zéros de cette fonction.
Annexe : représentation graphique de la fonction $f(x)=27x^3 -27x^2 -18x +8$ (graphique réalisé avec la librairie matplotlib de Python)



IX. Suite de Fibonacci.
1°) Concevoir un programme avec les contraintes suivantes.
- En entrée : le nombre n de termes de Fibonacci.
- En sortie : la liste des n termes de la suite.

2°) Concevoir un programme avec les contraintes suivantes.
- En entrée : la liste précédente des n premiers termes de Fibonacci.
- En sortie : la liste des quotients de deux termes consécutifs.

Afficher la valeur du quotient $\frac{1+\sqrt{5}}{2}$.