# Feuille de TP n° 2 - Fonctions

En informatique, une fonction désigne un ensemble d'instructions qui peuvent être appelées à n'importe quel moment dans un programme. Une fonction peut nécessiter un ensemble de variables appelé arguments. Une fois le code de la fonction exécuté, une valeur/un objet est renvoyé. De ce point de vue, une fonction en informatique correspond à l'idée d'une fonction en mathématiques : un ensemble de départ, un ensemble d'arrivée, et un processus qui utilise $x$ en entrée pour renvoyer $f(x)$ en sortie.

**Exercice 1.** Voici la syntaxe d'une fonction à un argument :

```
def Celsius2Kelvin(C) :
    K = C + 273.15
    return K

print(Celsius2Kelvin(20))
```

Écrire une fonction `CelsiusFar` qui convertit une température $T$ exprimée en degrés Celsius en degrés Fahrenheit. ( TF = 9TC/5 + 32 ).
Vérifier le type de la fonction $Celsius2Kelvin$ et celui de l'objet retourné par la fonction.

**Exercice 2.**

1. Quel est l'effet du code suivant ?

```
def f(x):
    x*=2
    return 3*x

x=5 ; y=f(x)
print(x,y)
```

2. La variable x passée comme argument est-elle modifiée après l'appel de la fonction ?

On peut créer des fonctions qui utilisent plusieurs arguments, de natures différentes.

**Exercice 3.**

1. Décrire l'effet du code suivant.

```
def CelsiusConvert(C,Unit):  # Unit = True pour Fahrenheit et False pour Kelvin
    if Unit:
        return 9*C/5 + 32
    else:
        return C + 273.15

print(0 ,"C")
print(CelsiusConvert(0,True),"F")
print(CelsiusConvert(0,False),"K")
```

2. Écrire une fonction `Max` qui retourne le plus grand de deux nombres $a, b$.

Une fonction en informatique a plus de libertés qu'une fonction en informatique. On peut construire des fonctions qui ne renvoient rien du tout. La fonction $f$ modifie des données, mais il n'y a pas d'objet $f(x)$ obtenu à la fin.
En Python, le type associé à un objet qui n'existe pas est NoneType.

**Exercice 4.**

1. Décrire la fonction suivante.

```
def CelsiusConvert(C,Far,Kel):
    print(C,"C")
    if Far: print(9*C/5 + 32 ,"F")
    if Kel: print(C + 273.15,"K")
    return

print(type(CelsiusConvert(10,True,True)))
```

2. Écrire une fonction `Pair` sans retour qui indique la parité d'un nombre entier $n$.

Une fonction peut renvoyer plusieurs objets. Pour cela, on regroupe tous les objets en un seul gros objet (ex : on regroupe trois nombres $a$,$b$,$c$ en un triplet de nombres $(a, b, c)$).

**Exercice 5.**

1. Décrire la fonction suivante.

```
def ChuteLibre(t):
    y = 0.5*9.8*t**2
    vy = 9.8*t
    return y,vy

pos, vit = ChuteLibre(0.5)
print(pos,vit)
```

2. Écrire une fonction `MaxMin` qui renvoie le minimum et le maximum d'une liste de nombres $L$.
On utilisera pour cela la commande $L.sort()$ ainsi que $len(L)$.

3. Écrire une fonction `MaxMin2` qui renvoie le minimum et le maximum d'une liste de nombres $L$, sans utiliser la commande $L.sort()$. On utilisera à la place une boucle `for`.

Dans une fonction on peut nommer les arguments pour mieux identifier/se souvenir du rôle de chacun.

**Exercice 6.**

1. Décrire la fonction suivante.

```
def division(numérateur,dénominateur) :
    return numérateur / dénominateur

print(division(dénominateur = 5, numérateur = 45))
```

2. Écrire une fonction `VolCyl` calculant le volume d'un cylindre de révolution, avec comme noms d'arguments $R$ et $h$, désignant respectivement le rayon et la hauteur du cylindre.

Lorsque l'on crée une variable en Python, celle-ci ne peut être utilisée qu'à l'intérieur de l'endroit où elle est créée. Une variable créée dans une fonction ne sera visible qu'au sein de cette fonction : on parle de variable locale. Une variable créee en dehors d'une fonction reste visible au sein de la fonction lorsqu'elle est appelée : on parle de variable globale.

**Exercice 7.**

1. Identifier dans le code suivant les variables locales et globales.

```
a = 2
def f(c):
    b=a*c*3
    return b

print(f(1),b)
```

2. L'argument `global` est utilisé afin d'informer Python qu'une variable a une portée globale. Prédire le comportement des codes suivants.

```
a=3
def f(c):
    global a
    a=2
    return a*c

print(f(2),a)
```

```
a=3
def f(c):
    a=2
    return a*c

print(f(2),a)
```

A l'intérieur de fonctions, on utilisera des variables locales, et on ne cherchera pas à utiliser `global` cette année.

**Exercice 8.**

1. Ecrire une fonction `Envers` qui prend une chaîne de caractères $c$ et renvoie la chaîne écrite à l'envers (`Envers('chat')` renvoie `'tahc'`).
   On utilisera une boucle `for` et la chaîne de caractères vide `e=''`.
2. Ecrire une fonction `EnversN` qui prend en entrée un entier $n$ et renvoie l'entier obtenu en inversant l'ordre des chiffres de $n$.
   On utilisera la fonction `Envers` et des conversions.
3. Ecrire une fonction `Palindrome` qui teste si une chaîne de caractères $c$ est un palindrome (qui se lit de la même façon dans les deux sens).
4. On part du nombre 47, on le retourne (74) et on somme ces deux nombres. On trouve 121 qui est un palindrome. Il faut parfois itérer le processus avant d'obtenir un palindrome :
   - 349+943 = 1292
   - 1292+2921 = 4213
   - 4213+3124 = 7337

   Il semblerait (non prouvé) que certains nombres comme 196 ne donnent jamais de palindrome. De tels nombres sont appelés nombres de Lychrel.
   On admet que pour un nombre $n$ inférieur à 10000 :
   - soit $n$ fournit un palindrome en moins de 50 itérations,
   - soit $n$ est un nombre de Lychrel.

   Ecrire une fonction `Lychrel` qui prend en entrée un nombre entier $n$ et vérifie si en au plus 50 itérations du processus on obtient un palindrome (renvoyer `True` si oui, `False` si non). On utilisera une boucle `while`, un test `if`, ainsi que la fonction `EnversN`.
5. Ecrire une fonction qui renvoie la liste de tous les nombres de Lychrel inférieurs à 10000.

**Exercice 9** (Bonus)**.**

1. Ecrire une fonction `Moy` qui prend en entrée une liste de nombres $L$ et qui renvoie la moyenne de tous ces nombres.
2. Ecrire une fonction `Var` qui prend en entrée une liste de nombres $L$ et qui renvoie la variance de tous ces nombres (moyenne des carrés des différences entre les nombres et leur moyenne).
3. Erire une fonction `Var2` qui prend en entrée une liste de nombres $L$ et qui renvoie la différence entre la moyenne des carrés de ces nombres et entre le carré de la moyenne de ces nombres.
4. Tester `Var` et `Var2`. Que remarque-t-on ?

**Exercice 10** (Bonus)**.**

1. Ecrire une fonction `g` qui correspond à la fonction polynômiale $x \mapsto x^3 - 6x + 7$.
2. Pour $f$ une fonction continue sur $[a, b]$, la somme de Riemann associée à $f$ (ou méthode des rectangles) correspond à la suite $(I_n)_n$ avec

$$I_n = \frac{b-a}{n} \sum_{k=0}^{n-1} f\left(a + k\frac{b-a}{n}\right).$$

   Cette suite est convergente, et on a $I_n \to_n \int_a^b f(t)dt$. Cela fournit des valeurs approchées de l'intégrale de $f$ sur un intervalle.
   Ecrire une fonction `Riemann` qui prend en entrée $f, a, b, n$ et qui renvoie la somme de Riemann $I_n$.
3. Tester la fonction `Riemann` pour `g` sur $[0, 1]$ et pour `cos` sur $[0, 2\pi]$ (on utilisera le module `math`).
   Vérifier que lorsque $n$ est grand les nombres $I_n$ sont proches des intégrales des fonctions considérées.
4. Pour $n = 100, 1000, 10000$, quel est l'ordre de grandeur de l'écart entre $I_n$ et l'intégrale ?