# TP 03 – Notion de fonction

Une *fonction* permet de regrouper des instructions dans un « bloc ». Ce bloc d'instructions pourra ensuite être appelé et exécuté à tout moment, sans avoir besoin de réécrire à chaque fois les instructions qu'il contient.

## I Définir et appeler une fonction

En programmation, les **fonctions** sont semblables aux fonctions mathématiques. À une valeur $x$, la fonction va retourner une valeur $y$.

En Mathématiques.

- **Définir** une fonction.

$$\forall x \in \underbrace{\ldots}_{\text{ens. de def}}, \quad f(x) = \underbrace{\ldots}_{\text{expression}}$$

- **Valuation** d'une fonction.

$$f(\underbrace{\ldots}_{\text{pt où on évalue}}) = \underbrace{\ldots}_{\text{résultat}}$$

En Informatique.

- **Définir** une fonction.

```
def f(x):
    return(...)
```

- **Valuation/appel** d'une fonction.

```
Entrée [1]: f(...)

Out [1]: ...
```

La différence principale entre une fonction mathématique et une fonction en informatique est l'absence d'ensemble de définition en informatique. Décortiquons maintenant la syntaxe nécessaire à la bonne définition d'une fonction en Python en prenant appui sur les deux lignes de code ci-dessus.

- La commande `def` permet d'indiquer à Python que l'on souhaite **définir une fonction**.
- La lettre `f` désigne le **nom de la fonction** et sera utilisée pour la valuation de la fonction. En mathématiques, on a l'habitude d'appeler très souvent nos fonctions $f$. En informatique, on essayera de donner des noms plus descriptifs à nos fonctions, comme par exemple `cube` si l'on code la fonction $x \mapsto x^3$.
- La variable `x` désigne l'**argument de la fonction**. Une fonction peut prendre un argument des objets de types différents (`int`, `float`, `list`, `str`) mais peut aussi avoir plusieurs arguments voir aucun (ce qui n'est pas possible en mathématiques).
- Enfin, la commande `return` permet d'indiquer l'**expression de la fonction**. Des instructions intermédiaires peuvent être effectués au sein de la fonction. Les variables définies au cours d'une fonction n'ont pas d'existence en dehors : on parle de variables locales.

En résumé, une fonction en informatique est une séquence d'instructions, dépendant de paramètres d'entrée, appelés arguments, et renvoyant un résultat. La syntaxe est la suivante.

```
def nomfonction(argument1, argument2, ...):
    instruction
    return(result)
# ici, on est hors de la définition de la fonction.
```

Lors de l'exécution d'un tel code, « il ne se passe rien » dans la console interactive. La fonction a simplement été mémorisée. Pour qu'une fonction produise son effet, il faut l'*appeler*, par une instruction dont la syntaxe est la suivante.

```
Entrée [2]: nom_fonction(val1, val2,..., valn)
```

où `val1, val2,..., valn` sont les **valeurs** auxquelles on veut évaluer la fonction.

# II Lecture de programme (sur papier)

**Exercice 1** On considère le programme suivant :

Entrée [3]:
```
def fonction(x):
    return (2*x+1)
```

Suite à l'exécution de ce programme, quel affichage obtient-on ?

Out [3]:

Si on rajoute la ligne de code suivante, quel affichage obtient-on maintenant ?

Entrée [4]:
```
fonction(3)
```

Out [4]:

**Exercice 2** On considère la fonction (mathématique) $f : \mathbb{R} \to \mathbb{R}$ définie par

$$\forall x \in \mathbb{R}, \qquad f(x) = x^2$$

Définir en Python cette fonction et calculer son évaluation en 2.

Entrée [5]:

Entrée [6]:

Out [6]:

**Exercice 3** On considère la fonction (mathématique) $f : \mathbb{R} \to \mathbb{R}$ définie par

$$\forall x \in \mathbb{R}, \qquad f(x) = x^3 + 2x + 1$$

Définir en Python cette fonction et calculer son évaluation en 2.

Entrée [7]:

Entrée [8]:

Out [8]:

**Exercice 4** On considère la fonction (mathématique) $f : \mathbb{R}^3 \to \mathbb{R}^3$ définie par

$$\forall (x, y, z) \in \mathbb{R}^3, \qquad f(x, y, z) = (x + y + z, 2x - y, xy)$$

Définir en Python cette fonction et calculer son évaluation au point $(1, 2, 3)$.

Entrée [9]:

Entrée [10]:

Out [10]:

**Exercice 5** Définir en Python une fonction, qui prend en argument deux nombres réels et qui renvoie le produit de ces deux nombres réels.

Entrée [11]:

## III Exercices (sur l'ordinateur)

**Exercice 6** On considère la fonction (mathématique) $f : \mathbb{R} \to \mathbb{R}$ définie par

$$\forall x \in \mathbb{R}, \qquad f(x) = \exp(x) + 1$$

Définir en Python cette fonction et calculer son évaluation en 0. ***Vérification :*** *On vérifiera que l'évaluation de cette fonction en* `0` *donne* `2`*.*

Entrée [12]:

Entrée [13]:

Out [13]:

**Exercice 7** Écrire une fonction qui prend en argument deux nombres réels et qui renvoie le produit et la somme de ces deux nombres réels. Calculer son évaluation pour le couple `(2,4)` ? ***Vérification :*** *On vérifiera que l'évaluation de cette fonction en* `(2,4)` *donne* `(6,8)`*.*

Entrée [14]:

Entrée [15]:

Out [15]:

**Exercice 8** Écrire une fonction `airedisque` qui prend en argument un réel `R` strictement positif (représentant le rayon d'un disque) et qui renvoie l'aire du disque de rayon `R`. Calculer son évaluation en `1`. ***Vérification :*** *On vérifiera que l'évaluation de cette fonction en* `1` *donne environ* `3.14`*.*

Entrée [16]:

Entrée [17]:

Out [17]:

**Exercice 9** Définir en Python une fonction, qui ne prend aucun argument, et qui renvoie le message "Bonjour".

Entrée [18]:

Entrée [19]:

Out [19]:

**Exercice 10** On considère la fonction (mathématique) $f : \mathbb{R} \to \mathbb{R}$ définie par

$$\forall x \in \mathbb{R}, \qquad f(x) = \begin{cases} 2x - 1 & \text{si } x \leqslant 2 \\ \ln(x) & \text{si } x > 2 \end{cases}$$

Définir en Python cette fonction et calculer son évaluation en $-4$ et en e. ***Vérification :*** *On vérifiera que l'évaluation de cette fonction en* `-4` *donne* `-9` *et que l'évaluation de cette fonction en* `e` *donne* `1`.

Entrée [20]:

Entrée [21]:

Out [21]:

Entrée [22]:

Out [22]:

# IV Exercices sur des subtilités de l'environnement def

## IV.1 Sur la différence entre print et return

L'instruction `print` n'a pas de valeur, elle se contente d'afficher un texte dans la console interactive. L'instruction `return` n'affiche pas, mais renvoie une valeur.

**Exercice 11** L'objectif de cet exercice est de mettre en lumière la différence entre les instructions `print` et `return`. Pour cela, on va s'aider de la fonction qui prend en argument un couple et qui renvoie la somme de ces deux nombres.

1. Tester le programme suivant dans la console. Que renvoie-t-il ?

Entrée [23]:
```
def somme(x,y):
    return(x+y)
3*somme(4,5)
```

Out [23]:

2. Tester le programme suivant dans la console. Que renvoie-t-il ?

Entrée [24]:
```
def somme(x,y):
    print(x+y)
3*somme(4,5)
```

Out [24]:

3. Expliquer la différence.

## IV.2 Sur l'importance de l'indentation

**Exercice 12** Pour les trois programmes ci-dessous,

1. Prévoir de tête l'affichage du programme (sans l'ordinateur).
2. Vérifier votre résultat précédent en exécutant le programme dans la console.

*À savoir : la fonction s'arrête dès qu'elle rencontre le premier* `return`. Expliquer la différence d'affichage des trois programmes.

Entrée [25]:
```
def somme(x,y):
    return(x+y)
print(somme(1,2))
print('terminé')
```

Out [25]:

Entrée [26]:
```
def somme(x,y):
    print('terminé')
    return(x+y)
print(somme(1,2))
```

Out [26]:

Entrée [27]:
```
def somme(x,y):
    return(x+y)
    print('terminé')
print(somme(1,2))
```

Out [27]:

## IV.3 Sur la notion de variables locales et globales

En Python, on distingue deux sortes de variables : les variables *globales* et les variables *locales*. Une variable globale est définie pour l'ensemble d'un programme, alors qu'une variable locale n'est définie qu'à l'intérieur d'une fonction.

**Exercice 13** L'objectif est de même en lumière la différence entre variables locales et globales.

1. Tester le programme suivant dans la console. Que renvoie-t-il ?

Entrée [28]:
```
y=-5
def f(x):
    return(x*y)
print(f(3))
print(y)
```

Out [28]:

2. Tester le programme suivant dans la console. Que renvoie-t-il ?

Entrée [29]:
```
def f(x):
    y=-5
    return(x*y)
print(f(3))
print(y)
```

Out [29]:

3. Expliquer la différence.

## V Exercices supplémentaires

**Exercice 14** Écrire une fonction, qui prend en argument deux nombres et qui renvoie la moyenne de ces deux nombres. Afficher son évaluation au point $(11, 16)$. ***Vérification :*** *On vérifiera que l'évaluation de cette fonction en* `(11,16)` *donne* `13.5`.

Entrée [30]:

Entrée [31]:

Out [31]:

**Exercice 15** Écrire une fonction, qui prend en argument deux nombres et qui renvoie le plus grand de ces deux nombres. Afficher son évaluation au point $(101, 100)$. ***Vérification :*** *On vérifiera que l'évaluation de cette fonction en* `(101,100)` *donne* `101`.

Entrée [32]:

Entrée [33]:

Out [33]:

**Exercice 16** Définir en Python une fonction, qui prend en argument un prénom, et qui renvoie le message "Bonjour [Prénom]".

Entrée [34]:

Entrée [35]:

Out [35]:

**Exercice 17** Écrire une fonction, qui prend en argument trois nombres réels $a$, $b$ et $c$ et qui renvoie le(s) racines du polynôme $x \mapsto ax^2 + bx + c$ (et qui renvoie un message 'Pas de racine' s'il en a pas). Afficher la valuation de cette fonction au point $(2, -6, 4)$.

Entrée [36]:

Entrée [37]:

Out [37]:

**Exercice 18**

1. Définir une fonction $f$, qui prend en argument un nombre réel $x$ et qui renvoie la valeur de $1 + \ln(x)$.

Entrée [38]:

2. On considère la suite définie par $u_0 = 2$ et pour tout $n \in \mathbb{N}$, $u_{n+1} = f(u_n)$ où $f$ est la fonction définie à la question précédente. Écrire un programme qui calcule et affiche les 5 premiers termes de la suite.

Entrée [39]:

Out [39]:

**Exercice 19** Écrire une fonction, qui prend en argument un entier `n` et qui affiche la table de multiplication de cet entier `n`.

Entrée [40]:

**Exercice 20** Écrire une fonction, qui prend en argument un nombre réel (représentant une somme d'argent en euros) et une chaine de caractère (représentant une devise, soit le dollar, soit la livre, soit le yens) et qui renvoie la somme d'argent dans la devise souhaitée. *On donne les conversions suivantes : 1 euro = 1,15 dollars ; 1 euro = 0,81 livre et 1 euro = 130 yens).*

Entrée [41]:

Entrée [42]:

Out [42]:

**Exercice 21** Voici la réduction pour le prix d'un billet de train en fonction de l'âge du voyageur.

- Réduction de 50% pour les moins (strictement) de 10 ans
- Réduction de 30% pour les 10 à 18 ans.
- Réduction de 20% pour les 60 ans et plus.

Écrire une fonction `reduction`, qui prend en argument un entier correspondant à l'âge du voyageur et un réel correspondant au prix initial du billet et qui renvoie un réel correspondant au montant à payer après réduction. Quel montant doit débourser un voyageur de 17 ans pour un billet qui coute initialement 100 euros ? Et pour un voyageur de 23 ans pour un billet qui coute initialement 63 euros ?

Entrée [43]:

Entrée [44]:
```
#Test pour 17 ans et billet de 100 euros
```

Out [44]:

**Exercice 22** Écrire une fonction prenant en argument trois nombres réels et qui renvoie le plus petit de ces trois nombres. Que renvoie la fonction pour le triplet `(1,2,3)` ? pour le triplet `(101,99,100)` ? pour le triplet `(101,99,1)` ?

Entrée [45]:

Entrée [46]:
```
#Test pour le triplet (1,2,3)
```

Out [46]:

Entrée [47]:
```
#Test pour le triplet (101,99,100)
```

Out [47]:

Entrée [48]:
```
#Test pour le triplet (101,99,1)
```

Out [48]:

**Exercice 23** Écrire une fonction `renverse` prenant en argument une chaîne de caractères et renvoyant la chaîne de caractères inversée. Par exemple, l'évaluation de cette fonction en `'PTSI'` doit renvoyer `'ISTP'`.

Entrée [49]:

Entrée [50]:

Out [50]:

**Exercice 24** Ecrire une fonction `palindrome` prenant en argument une chaîne de caractères et renvoyant `True` si celle-ci est un palindrome et `False` sinon. Un palindrome est une chaîne de caractères se lisant de la même manière dans un sens et dans l'autre. Par exemple `'abcba'` est un palindrome.

Entrée [51]:

**Exercice 25** Sans utiliser la fonction `factorial` de `Python`, écrire une fonction `somme` prenant en argument un entier `n` et renvoyant la valeur de la somme :

$$S_n = \sum_{k=0}^{n} (-1)^k k!$$

On s'attachera à minimiser le nombre de calculs. En particulier, on remarquera que lorsque l'on connaît $k!$, il est possible d'obtenir $(k+1)!$ en effectuant une seule opération. ***Vérification :*** *On vérifiera que l'évaluation de cette fonction en* `10` *donne* `3301820`.

Entrée [52]:

Entrée [53]:

Out [53]: