

**1DEV1A – 1DEVL1A – Laboratoires Python****TD 05 – Fonctions**

Nous allons à présent nous initier à l'écriture de nos propres fonctions en Python.

Objectifs :

- ▷ définir et appeler une fonction sans paramètre, et avec un ou plus de paramètres ;
- ▷ utiliser adéquatement le mot-clef `return` ;
- ▷ utiliser l'indentation dans une définition de fonction ;
- ▷ définir et utiliser des modules ;
- ▷ choisir un élément aléatoirement dans une liste (`random.randint`, `random.choice`) ;
- ▷ identifier du code mort après un `return`.

**Table des matières**

<b>1 Les fonctions en Python</b>	<b>2</b>
<b>2 Découpe en fichier</b>	<b>3</b>
<b>3 Fonctions avec un nombre quelconque de paramètres</b>	<b>3</b>
<b>4 Fonctions renvoyant un booléen</b>	<b>4</b>
<b>5 Les fonctions qui ne renvoient aucune valeur</b>	<b>5</b>
<b>6 Code mort</b>	<b>6</b>
<b>7 Exercices récapitulatifs</b>	<b>6</b>
<b>8 En résumé ...</b>	<b>7</b>

## Pré-requis

On suppose ici que vous avez déjà vu la notion de fonction au cours théorique.  
Si ce n'est pas le cas, parlez-en avec votre enseignant.

# 1 Les fonctions en Python

Voici un exemple de script Python définissant et utilisant une fonction `perimetre_cercle` calculant le périmètre d'un cercle de rayon passé en paramètre :

```
1  # *** Les fonctions en Python ***
2
3  # Une fonction calculant le périmètre d'un cercle de rayon donné:
4  def perimetre_cercle(rayon):
5      return 2 * 3.14 * rayon
6
7
8  r = float(input("Entrez la longueur (en cm) du rayon du cercle : "))
9
10 perimetre = perimetre_cercle(r)
11
12 print("Le périmètre du cercle de rayon", r, "cm vaut", perimetre, "cm")
```

À noter :

- ▷ comment la fonction est d'abord *définie* puis *appelée* (on dira aussi *utilisée*) ;
- ▷ l'usage de l'indentation pour délimiter le bloc définissant la fonction ;
- ▷ l'usage du mot-clef `return` pour indiquer la valeur de retour ;
- ▷ l'usage de la variable `perimetre` pour référencer la valeur de retour de la fonction.  
Ceci n'est pas obligatoire dans les cas les plus simples.

## Exercice 1 La notion de fonction

Lancez le script Python fourni ci-dessus plusieurs fois de suite pour calculer le périmètre d'un cercle de rayon 2, puis 5.5 et enfin 345.28 centimètres respectivement.

Mesure du rayon	Périmètre du cercle
2	
5.5	
345.28	

La syntaxe pour la déclaration d'une fonction prenant un paramètre et renvoyant une valeur est donc :

```
1  # Déclaration d'une fonction prenant un paramètre et renvoyant une valeur
2  def nom_de_la_fonction(parametre):
3      #code de la fonction
4      return valeur_de_retour
```

## Exercice 2 La notion de fonction (suite)

Écrivez un script Python définissant la fonction `perimetre_cercle` ci-dessus et affichant à l'écran le résultat des deux appels suivants :

Appel de fonction	Résultat ?
<code>perimetre_cercle(-6)</code>	
<code>perimetre_cercle('bonjour !')</code>	

## 2 Découpe en fichier

L'exemple de la section précédente comportait un seul fichier avec la fonction et le code appelant cette fonction. Une règle de bonne pratique est cependant de placer ses fonctions dans un (des) module(s) et d'inclure ceux-ci dans un fichier contenant votre programme principal qui fera alors appel aux fonctions, comme illustré ci-dessous :

```

1  # Code du module contenant des fonctions
2
3  # Une fonction calculant le périmètre d'un cercle de rayon donné:
4  def perimetre_cercle(rayon):
5      return 2 * 3.14 * rayon
6
7  # Une fonction calculant l'aire d'un cercle de rayon donné:
8  def aire_cercle(rayon):
9      return 3.14 * rayon * rayon
10
11
12 # Éventuelles autres fonction en rapport avec le cercle...

```

cercle.py

```

1  # On importe le module désiré ...
2  # Pour que l'import ci-dessous fonctionne bien, le fichier
3  # cercle.py doit se trouver dans le même répertoire que main.py
4  import cercle
5
6  r = float(input("Entrez la longueur (en cm) du rayon du cercle : "))
7
8  # ... et on fait appel aux fonctions nécessaires
9  perimetre = cercle.perimetre_cercle(r)
10
11 print("Le périmètre du cercle de rayon", r, "cm vaut", perimetre, "cm")

```

main.py

### Exercice 3 L'aire d'un carré

Écrivez :

1. un module `carre.py` contenant une fonction `aire_carre` qui prend en paramètre un nombre  $x$  et qui renvoie l'aire du carré de côté  $x$  cm ;
2. un programme principal `main.py` utilisant cette fonction de façon à demander un nombre positif  $x$  à l'utilisateur et afficher l'aire du carré associé à l'écran. Si le nombre entré par l'utilisateur n'est pas positif, le script affiche un message d'erreur.

### Découpe en modules

Soyez attentif dans la suite des exercices (de ce TD ou des suivants) à *toujours* découper votre code en modules et programme principal. Nous ne le précisons pas à chaque exercice, mais la demande est implicite.

## 3 Fonctions avec un nombre quelconque de paramètres

Une fonction peut admettre un nombre quelconque de paramètres en entrée (zéro, un, deux...) et peut aussi faire appel à d'autres fonctions. Le script suivant utilise deux

fonctions pour tirer trois lettres de l'alphabet au hasard. Ces lettres peuvent alors par exemple être utilisées pour développer un jeu, comme le Scrabble <sup>1</sup>.

Remarquez l'utilisation de la fonction `randint` dans la fonction `hasard` et des fonctions `len` et `sorted` dans la fonction `lettres`.

```
1 import random
2
3 # Retourne un nombre aléatoire entre a et b (appelée par la fonction lettres())
4 def hasard(a, b):
5     return random.randint(a, b)
6
7 # Tire 3 lettres de l'alphabet au hasard (répétitions possibles) et les renvoie
8 def lettres():
9     alphabet = 'abcdefghijklmnopqrstuvwxyz'
10    nb_lettres = len(alphabet)
11
12    lettre1 = alphabet[hasard(0, nb_lettres-1)]
13    lettre2 = alphabet[hasard(0, nb_lettres-1)]
14    lettre3 = alphabet[hasard(0, nb_lettres-1)]
15
16    return sorted([lettre1, lettre2, lettre3])
```

```
1 import lettres
2
3 mes_lettres = lettres.lettres()
4
5 print("Voici trois lettres de l'alphabet tirées au hasard :")
6 print(mes_lettres[0], mes_lettres[1], mes_lettres[2])
```

#### Exercice 4 Les couleurs aléatoires

Inspirez-vous du code précédent pour écrire une fonction `couleur` qui renvoie une couleur au hasard parmi les chaînes de caractères :

```
les_couleurs = ['rouge', 'vert', 'jaune', 'bleu', 'blanc', 'noir']
```

Mettez cette fonction en application pour écrire un script qui affiche à l'écran une couleur tirée au hasard.

#### Exercice 5 Les cartes aléatoires

Écrivez une fonction `tirer_carte` qui permet de tirer une carte au hasard dans un jeu de 52 cartes. Une carte d'un tel jeu contient une valeur choisie parmi :

```
les_valeurs = [2, 3, 4, 5, 6, 7, 8, 9, 10, 'valet', 'dame', 'roi', 'as']
```

ainsi qu'une couleur, choisie parmi :

```
les_couleurs = ['pique', 'trèfle', 'cœur', 'carreau']
```

La fonction `tirer_carte` doit renvoyer une liste de deux éléments contenant la valeur et la couleur de la carte.

Voici un exemple de valeur de retour possible : `[9, 'carreau']`.

## 4 Fonctions renvoyant un booléen

Une fonction peut aussi renvoyer un booléen, comme illustré ci-dessous :

---

1. voir : <https://fr.wikipedia.org/wiki/Scrabble>

```

1 # Fonction qui détermine la parité de son paramètre
2 def est_pair(x):
3     return x%2 == 0

```

nombres.py

```

1 import nombres
2
3 x = int(input('Entrez un nombre entier : '))
4
5 if nombres.est_pair(x):
6     print('Ce nombre est pair')
7 else:
8     print('Ce nombre est impair')

```

main2.py

La fonction `est_pair` renvoie `True` si le nombre est pair, et `False` sinon. La valeur de retour de cette fonction peut alors par exemple être utilisée au sein d'une instruction `if` (comme ci-dessus).

### Exercice 6 Divisibilité

Écrivez une fonction `est_divisible(a, b)` qui renvoie `True` si un des deux nombres divise l'autre, et `False` sinon.

Par exemple :

- ▷ `est_divisible(2, 3)` renvoie `False` ;
- ▷ `est_divisible(4, 8)`, `est_divisible(8, 4)` et `est_divisible(15, 3)` renvoient `True`.

### Exercice 7 Inclusion de chaînes de caractères

Écrivez une fonction `contient(texte1, texte2)` qui renvoie `True` si `texte1` est contenu dans `texte2` et `False` sinon.

Par exemple :

- ▷ `contient('chat', 'Le chat gris')` renvoie `True` ;
- ▷ `contient('chien', 'Le chat gris')` renvoie `False`.

Aide : utilisez simplement l'opérateur `in`.

## 5 Les fonctions qui ne renvoient aucune valeur

Une fonction en Python ne renvoie pas obligatoirement une valeur : il suffit d'omettre le mot clé `return` dans le code de celle-ci. Voici un exemple de scripts utilisant de telles fonctions pour afficher une ligne d'étoile de longueur donnée à l'écran :

```

1 # Fonction qui affiche n étoiles à l'écran
2 def afficher_etoiles(n):
3     print(n * '*')
4     # Aucun return: la fonction se contente d'afficher des étoiles

```

affichage.py

```

1 import affichage
2
3 # Affiche 5 étoiles
4 affichage.afficher_etoiles(5)
5
6 # Affiche 10 étoiles
7 affichage.afficher_etoiles(10)

```

main3.py

### Exercice 8 La ligne de caractères

Écrivez une fonction `affichage(n, car)` qui affiche `n` fois le caractère `car` à l'écran.

Par exemple, l'appel de fonction `affichage(10, '-')` affiche :

-----

à l'écran.

## 6 Code mort

Faites attention à l'endroit où se trouve votre instruction `return`. Par exemple dans le code suivant :

```
1  # *** L'instruction return en Python ***
2
3  def afficher(x):
4      print(x)
5      return 0
6      print("Les fonctions en Python")
7
8  afficher(5)
9  afficher('salut')
10 afficher([1, 2, 3])
```

### Exercice 9 Code mort

Exécutez le script ci-dessus. Qu'affiche ce script à l'écran ? Le script affiche :

La chaîne de caractères `"Les fonctions en Python"` n'est jamais affichée. Pourquoi ? Votre réponse :

## 7 Exercices récapitulatifs

### Exercice 10 Le volume de la sphère

Écrivez une fonction `volume_sphere(r)` qui renvoie le volume d'une sphère de rayon `r` passé en paramètre.

Faites une recherche sur internet si vous ne connaissez plus la formule permettant de calculer le volume d'une sphère.

### Exercice 11 La moyenne

Écrivez une fonction `moyenne(a, b, c)` qui renvoie la moyenne des trois nombres `a`, `b` et `c`.

Écrivez un script utilisant cette fonction pour calculer la moyenne de trois nombres entrés au clavier par l'utilisateur.

### Exercice 12 L'adresse e-mail

Écrivez une fonction `adresse_valide(adresse)` qui vérifie la validité de l'adresse e-mail donnée en paramètre.

Pour simplifier très fort, nous considérons ici qu'une adresse e-mail est valide si :

1. elle contient une fois le symbole arobase '@' ;
2. elle contient au moins un point '.' ;
3. le premier caractère et le dernier caractère ne sont ni '@' ni '.'.

Par exemple :

▷ `adresse_valide(charlot@gmail.be)` renvoie `True` ;

▷ `adresse_valide(charlot)` et `adresse_valide(@toto.be)` renvoient `False`.

Écrivez un script utilisant cette fonction pour vérifier la validité d'une adresse e-mail entrée au clavier par l'utilisateur : si l'utilisateur entre une mauvaise adresse e-mail, le script affiche un message d'erreur.

### Exercice 13 La boîte

Écrivez une fonction `afficher_boite(n)` qui affiche à l'écran une boîte vide, c'est-à-dire un rectangle avec un bord mais d'intérieur vide (voir l'illustration ci-dessous), constitué de  $n$  étoiles de largeur et de 4 étoiles de hauteur.

Par exemple, l'appel de fonction `afficher_boite(10)` affiche à l'écran :

```
*****
*       *
*       *
*****
```

Remarque : n'oubliez pas de bien traiter les cas où  $n$  vaut zéro (0), un (1) et deux (2) ou est négatif ( $-2$ , par exemple).

## 8 En résumé ...

### Principaux points de matière du TD

Voici les principaux points abordés lors de ce TD. Vous devez absolument être à l'aise avec ceux-ci avant d'aborder la prochaine séance d'exercice.

1. Écrire une fonction en Python en utilisant le mot clé `def`.
2. Découper son script en plusieurs fichiers : modules et programme principal.
3. Utiliser le mot clé `return` pour renvoyer une valeur au programme appelant.
4. Faire appel à une fonction en lui passant des paramètres, utiliser l'éventuelle valeur de retour de la fonction.
5. Écrire une fonction Python résolvant un problème donné.