



fonction se termine au premier `return` exécuté qui désigne la valeur à retourner. L'exécution revient alors à l'endroit où l'on a effectué l'appel à la fonction, et c'est la valeur retournée par la fonction qui y est utilisée.

```
# definition de la fonction g contenant du code mort
def g(x):
    a = x+1
    return a*a + x + 1
    # code mort - erreur de programmation a eviter
    b = a + 1
```

La définition de fonction ci-dessus contient du code mort : les instructions qui suivent une instruction `return` ne sont jamais exécutées, c'est une erreur de programmation.

## EXERCICE 2

Après avoir défini la fonction `f` comme ci-dessous,

```
# definition de la fonction f
def f(x):
    a = x+1
    return a*a + x + 1
```

taper les instructions suivantes qui appellent cette fonction en lui passant différents arguments et prenez le temps d'expliquer en détail les résultats obtenus.

```
x = 0
y = f(2)
t = 4
y = f(t)           # on passe la valeur d'une variable
y = f(1) + f(2)   # on effectue deux appels
z = x+1
y = f(z)
y = f(x+1)        # on passe directement la valeur d'une expression
z = f(x-t)
t = f(t)          # on peut meme passer la variable qui servira a
                  # stocker le resultat
x = f(f(1))       # on peut combiner deux appels, le resultat de
                  # l'un est passe en parametre a l'autre
```

## EXERCICE 3

Parmi les codes suivants, quels sont les programmes Python qui ne comportent pas d'erreur? Rayer les codes erronés.

```
Def fun(x):
    return x + 1
```

```
y=fun(3)
```

```
def fun(y):
    return x + 1
```

```
y=fun(3)
```

```
def fun(x):
    return x + 1
```

```
y=fun(3)
```

```
def fun(x):
    return x + 1
```

```
y=fun(3)
```

```
def fun(x)
    return x + 1
```

```
y=fun(3)
```

```
def fun():
    return 2
```

```
y=fun(3)
```

## EXERCICE 4

1. Écrire une fonction `moyenne(a,b)` qui retourne la moyenne des deux nombres `a` et `b`. Testez-la avec les arguments 42 et 23.
2. Écrire une fonction `moyennePonderee(a, coef_a, b, coef_b)` qui retourne la moyennne pondérée par le coefficient `coef_a` pour la note `a` et par le coefficient `coef_b` pour la note `b`. Testez-la en appelant `moyennePonderee(5, 2, 12, 3)`.

## EXERCICE 5

Taper les instructions suivantes et prenez le temps d'expliquer précisément l'évolution des variables pendant l'exécution pas à pas de ces instructions.

```
i = 9
j = 0
b = i < j      # b ne contient donc pas un entier, mais True ou False
b = i != 9
b = i == 9
bi = i % 2 == 0 or i % 3 == 0
bj = j % 2 == 0 or j % 3 == 0
b = not (bi and bj)
```

## EXERCICE 6

Parmi les expressions suivantes, quelles sont celles qui valent True *si et seulement si*<sup>a</sup> les trois variables a, b, c ont des valeurs toutes différentes? Rayer les expressions incorrectes.

- a != b and b != c
- a != b and b != c and a != c
- a != b or b != c
- a != b or b != c or a != c

a. *si et seulement si* signifie que l'un est vrai si l'autre est vrai, et que l'un est faux si l'autre est faux.

## EXERCICE 7

Écrire une expression qui vaut True si x appartient à [0, 5[ et False dans le cas contraire.

## EXERCICE 8

Parmi les expressions suivantes, quelles sont celles qui valent True si l'entier n est pair et False dans le cas contraire? Rayer les mauvaises réponses.

- n == 0 % 2
- 1 != 2 % n
- n // 2 == 0
- n % 2 != 1
- 0 != n % 2
- n % 2 == 0

Écrire une fonction pair(n) qui teste si n est pair; la valeur calculée doit être booléenne, c'est-à-dire égale à True ou False. Testez-la.

## EXERCICE 9

Donner les valeurs des variables x et y après exécution de l'exemple suivant pour x valant 1, puis pour x valant 8.

```
if x % 2 == 0:
    y = x // 2
else:
    y = x + 1
x = x + 1
```

Même question pour chacun des codes suivants :

```
if x % 2 == 0:
    y = x // 2
    y = x + 1
x = x + 1
```

```
if x % 2 == 0:
    y = x // 2
else:
    y = x + 1
    x = x + 1
```

```
if x % 2 == 0:
    y = x // 2
else:
    y = x + 1
x = x + 1
```

### EXERCICE 10

Écrire une fonction `compare(a, b)` qui retourne  $-1$  si  $a < b$ ,  $0$  si  $a = b$ , et  $1$  si  $a > b$ . Testez-la. Donner plusieurs versions de cette fonction.

### EXERCICE 11

Écrire une fonction `max2(x, y)` qui calcule et retourne le maximum de deux nombres  $x$  et  $y$ .  
**Attention** : bien appeler cette fonction `max2`, car la fonction `max` est prédéfinie en python.

### EXERCICE 12

Écrire une fonction `max3(x, y, z)` qui calcule et retourne le maximum de trois nombres  $x, y, z$ .

### EXERCICE 13

Écrire une fonction `uneMinuteEnPlus` qui calcule et retourne l'heure une minute après celle passée en paramètre sous forme de deux entiers que l'on suppose cohérents. Exemples :

- `uneMinuteEnPlus(14, 32)` retourne `(14, 33)`.
- `uneMinuteEnPlus(14, 59)` retourne `(15, 0)`.

Ne pas oublier le cas de minuit.

### EXERCICE 14

Le service de reprographie propose les photocopies avec le tarif suivant : les 10 premières coûtent 20 centimes l'unité, les 20 suivantes coûtent 15 centimes l'unité et au-delà de 30 le coût est de 10 centimes. Écrire une fonction `coutPhotocopies(n)` qui calcule et retourne le prix à payer pour  $n$  photocopies.

### EXERCICE 15

On considère la fonction suivante :

```
def mystere(n):
    s = 0
    i = 0
    while i <= n :
        s = s + i**2
        i=i+1
    return s
```

1. Quelle est la valeur de `mystere(20)` ?
2. De façon générale, que calcule la fonction `mystere` ?

### EXERCICE 16

Pour simuler le lancé d'un dé à 6 faces on peut utiliser l'appel `randrange(1, 7)`. Pour utiliser cette fonction il faudra que votre code commence par la phrase magique :

```
from random import *
```

Écrire le code des fonctions suivantes :

1. `obtenirUn6()` qui retourne le nombre de lancers effectués avant d'obtenir le nombre 6.
2. `obtenirUnDouble()` qui retourne le nombre de lancers effectués pour obtenir deux fois consécutivement le même nombre.

### EXERCICE 17

Pour tout entier  $n \geq 0$ , le nombre de Cullen d'indice  $n$  est le nombre  $n \times 2^n + 1$ .

1. Écrire une fonction `cullen(n)` prenant en paramètre un entier  $n$  et retournant le nombre de Cullen d'indice  $n$ . On rappelle que Python peut calculer une puissance  $a^k$  avec la notation `a**k`.
2. Écrire une fonction `indiceCullen(x)` prenant en paramètre un entier  $x$ , et retournant le plus grand entier  $n$  tel que  $x$  soit supérieur au nombre de Cullen d'indice  $n$ .

### EXERCICE 18

On considère la fonction suivante :

```
def mystere3(n):  
    s = 0  
    while n > 0 :  
        s = s + n % 10  
        n = n // 10  
    return s
```

1. Quelle est la valeur de `mystere3(2705)` ? De façon générale, que calcule la fonction `mystere3` ?
2. Écrire une fonction `nombreDeChiffres(n)` qui retourne le nombre de chiffres contenus dans le nombre entier  $n$ . Par exemple, la valeur de `nombreDeChiffres(2705)` est 4.
3. Écrire une fonction `plusGrandChiffre(n)` qui retourne le plus grand chiffre contenu dans le nombre entier  $n$ . Par exemple, la valeur de `plusGrandChiffre(2705)` est 7.

### EXERCICE 19

On considère la définition suivante :

```
def mystere4(n):  
    s = 0  
    for i in range(1, n+1):  
        s = s + i  
    return s
```

Que retournent les appels `mystere4(2)`, `mystere4(3)`, et `mystere4(n)` en général ?  
Connaissez-vous une formule qui permet de calculer le même résultat ?

### EXERCICE 20

Entrer les instructions suivantes et analyser les réponses de python :

```
for j in range(10):  
    print(j, j * j)                                     for k in range(3, 9, 2):  
                                                         print(k)  
  
for k in range(3, 8):  
    print(k, 2 * k + 1)
```

### EXERCICE 21

Écrire une fonction `sommeCarres(n)` qui calcule et retourne  $\sum_{i=1}^n i^2$ .

### EXERCICE 22

Écrire une fonction `liste3chiffres()` qui permet d'afficher par ordre croissant tous les nombres à 3 chiffres dont la somme des chiffres est multiple de 5.

### EXERCICE 23

1. Écrire une fonction `nombreDiviseurs(n)` qui affiche le nombre de diviseurs d'un entier naturel  $n$ .
2. Écrire une fonction `estPremier(n)` qui renvoie `True` si le nombre est premier `False` sinon.  
Rappel : un nombre entier est premier s'il est strictement supérieur à 1 et si le nombre de ses diviseurs est égal à deux, 1 et lui-même.

### EXERCICE 24

Écrire une fonction `approxPi(nbEtapes)` qui calcule en `nbEtapes` une approximation de  $\pi$  à partir de la formule de Leibniz :

$$\pi = 4 \left( 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \dots \right)$$

### EXERCICE 25

1. La fonction `factorielle(n)` (notée mathématiquement «  $n!$  ») peut être définie de la manière suivante (on suppose que  $n \geq 1$ ) :

$$n! = 1 \times 2 \times \dots \times n$$

Écrire une fonction `factorielle(n)` qui utilise cette définition pour calculer et retourner  $n!$ . Tester votre fonction en affichant les factorielles des nombres de 0 à 100.

2. Écrire une fonction `coeff_binomial(n, k)` qui prend en argument deux entiers naturels  $n$  et  $k$  (avec  $k \leq n$ ) et qui renvoie la valeur du coefficient binomial correspondant "k parmi n" :  $\binom{n}{k} = \frac{n!}{(n-k)!k!}$ .

### EXERCICE 26

Écrire une fonction `triangle_pascal(nb_lignes)` qui prend en argument un entier `nb_lignes` et qui affiche le triangle de Pascal (voir définition ci-dessous) en s'arrêtant au bout du nombre de lignes indiqué par l'argument. Voici un exemple du triangle de Pascal avec 6 lignes :

```
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1
```

En numérotant les lignes et les colonnes à partir de zéro, le nombre sur la ligne numéro  $n$  et la colonne numéro  $k$  est le coefficient binomial  $\binom{n}{k}$ . Par exemple, pour la colonne numéro 0, on a toujours  $\binom{n}{0} = 1$  et pour la colonne 1 on a toujours  $\binom{n}{1} = n$ . De même, lorsque  $k = n$ , on a  $\binom{n}{n} = 1$  donc chaque ligne se termine par un 1.