

TP Python n° 1 : Initiation à l'algorithmique

1. Qu'est-ce qu'un algorithme ?

Un algorithme est l'énoncé d'une suite d'instructions, à effectuer dans un certain ordre, permettant de donner la réponse à un problème donné et ce, en un nombre fini d'étapes.

Etymologie :

Le mot « algorithme » vient du nom d'un mathématicien persan AL-KHWARIZMI dont le nom a été latinisé au Moyen-Age par les Européens en « algorisme » puis « algorithme ».

AL-KHWARIZMI est un mathématicien, géographe, astrologue et astronome arabophone d'origine perse né au VIII^{ème} siècle (783 – 850).



Ce mathématicien écrivit en langue arabe le plus ancien traité d'algèbre sur la résolution des équations. Il y proposait les solutions en décrivant l'enchaînement d'étapes à suivre pas à pas pour résoudre les équations.

Malgré tout, cette notion d'algorithme existait bien avant de porter un nom : 2000 ans avant Thalès et Pythagore, les comptables et les arpenteurs connaissaient déjà des algorithmes permettant d'effectuer des calculs d'intérêts ou de prêts ; des calculs d'aires de surfaces agricoles...

Exercice 1.1

Sans savoir qu'ils portaient ce nom, vous aussi connaissez de nombreux algorithmes utilisés en mathématiques, dont certains depuis l'école primaire. Citez-en trois exemples :

- Programmes de construction
- Programmes de calcul
- Opérations arithmétiques

2. Comment écrire un algorithme ?

Un algorithme peut s'écrire en français mais cela mène vite à des phrases longues, alambiquées et parfois ambiguës. Par souci de clarté et de précision, on utilisera un « pseudo-code » lisible de tous.

Grâce aux calculatrices et ordinateurs, on peut désormais écrire des algorithmes très performants, dont l'exécution « à la main » serait peu envisageable car beaucoup trop longue ou fastidieuse (c'est le cas notamment lorsque les instructions doivent être répétées un très grand nombre de fois). On peut donc, après avoir rédigé l'algorithme, le retranscrire dans « un langage de programmation » (celui d'une calculatrice ou d'un logiciel) afin de le faire exécuter par une machine.

AUGUSTA ADA KING, comtesse de Lovelace est une pionnière de la science informatique. D'origine britannique, elle est née au XIX^{ème} siècle (1815 – 1852).

Elle a réalisé le premier programme informatique, lors de son travail sur un ancêtre de l'ordinateur : la machine analytique de Charles Babbage. Dans ses notes, on trouve en effet le premier algorithme publié, destiné à être exécuté par une machine.



En français

Choisir un nombre
Lui soustraire 1 puis
multiplier le résultat
par 2
Afficher le nouveau
nombre obtenu

En pseudo-code

Variables :
 a est un nombre réel
Début :
Saisir a
 a prend la valeur $2(a-1)$
Afficher a
Fin

En langage de programmation (Python)

```
# a est un nombre
a=float(input("a=?"))
a=2*(a-1)
print(a)
```

Nous allons, dans un premier temps, utiliser exclusivement le pseudo-code.

3. Quelle structure pour un algorithme ?

Un algorithme est composé de deux parties distinctes :

- la déclaration des variables nécessaires à l'élaboration de l'algorithme : une variable joue le rôle d'une « boîte » à laquelle on donne un nom et qui contiendra la plupart du temps un nombre (Attention ! le contenu de cette « boîte » peut changer plusieurs fois au cours d'un même algorithme) ;
- le corps de l'algorithme constitué :
 - des entrées / sorties** : qui permettent d'entrer des données ou d'afficher des résultats (rouge) ;
 - des affectations** : qui modifient par exemple le contenu d'une variable (bleu) ;
 - des instructions conditionnelles** : qui effectuent une tâche sous conditions (vert) ;
 - des boucles** : qui répètent plusieurs fois les mêmes instructions (noir).

Exemple :

<u>Variables :</u> a est un nombre réel	}	Déclaration des variables
<u>Début :</u> Saisir a a prend la valeur $2(a-1)$ Afficher a <u>Fin</u>		

Exercice 1.2

Variables :

a , b et m sont des nombres réels

Début :

Afficher le message « Saisir la 1^{ère} note »

Saisir a

Afficher le message « Saisir la 2^{ème} note »

Saisir b

m prend la valeur $(a + b) \div 2$

Afficher m

Fin

1. Souligner les différentes instructions en respectant les couleurs définies ci-dessus.
2. A quel problème répond cet algorithme ?

Cet algorithme calcule la moyenne de deux notes saisies.

4. Comprendre et exécuter un algorithme

L'algorithme de l'exemple précédent (page 2) ne comporte qu'une seule variable a et se compose de 3 instructions.

- | | |
|--------------------------------|--|
| Saisir a | (1) cette instruction demande à l'utilisateur de rentrer une valeur dans la variable a . |
| a prend la valeur $2(a - 1)$ | (2) cette affectation change la valeur que contenait la variable a . |
| Afficher a | (3) cette instruction affiche le dernier contenu de la variable a . |

Pour exécuter cet algorithme à la main (on parle **d'exécution « pas à pas »**), on peut remplir un tableau permettant de visualiser, à chaque étape de celui-ci, le contenu de chacune des variables.

Si on teste cet algorithme pour $a = 5$, on aura :

Variable a	a
(1)	5
(2)	8
(3)	8

Bien sûr, ce type de présentation est beaucoup plus utile et pertinente lorsqu'il y a plus d'instructions et/ou de variables.

Exercice 1.3

Variables :

a, x sont des nombres réels

Début :

Saisir x

a prend la valeur $x + 4$

x prend la valeur $2 \times a$

a prend la valeur $a - 3$

a prend la valeur $a - x$

Afficher a

Fin

Exécuter cet algorithme pour $x = 3$ en remplissant le tableau suivant :

Variable x, a x a
(1) 3
(2) 3 7
(3) 14 7
(4) 14 4
(5) 14 -10
(6) 14 -10

5. Exercices



Exercice 1.4

Souligner les différentes instructions en respectant les couleurs définies page 2.

Variables :

a, b, c, h et s sont des nombres réels

Début :

Saisir a

Saisir b

Saisir c

h prend la valeur a^2

s prend la valeur $b^2 + c^2$

Si $h = s$ alors :

Afficher « Le triangle ABC est rectangle en

A »

Sinon :

Afficher « Le triangle ABC n'est pas
rectangle en A »

FinSi

Fin



Exercice 1.5

Traduire en pseudo-code l'algorithme correspondant au problème ci-dessous puis souligner les instructions en utilisant le code couleur.

- Choisir deux nombres
- Calculer le carré du premier
- Calculer le double du second
- Ajouter ces deux résultats
- Afficher cette somme

Variables :

a, b, c, d et e sont des nombres réels ...

Début :

Saisir a

Saisir b

c prend la valeur a^2

d prend la valeur $2 \times b$

e prend la valeur $c + d$

Afficher e

Fin

Exercice 1.6

Variables :

p et c sont des nombres réels

Début :

Saisir p

c prend la valeur $p - 1$

p prend la valeur $p + 1$

p prend la valeur $p \times p - c \times c$

Afficher p

Fin

1. Compléter la partie variables de cet algorithme.
2. Souligner en couleurs les instructions.
3. Exécuter l'algorithme avec les valeurs suivantes. On pourra s'aider de tableaux.
 $p = 2 : \dots\dots 8 \dots\dots$ $p = -4 : \dots -16 \dots$ $p = 3 : \dots\dots 12 \dots\dots$
4. Est-il indispensable d'utiliser 2 variables ? non, une seule variable suffit
5. Quelle instruction peut remplacer les 3 affectations ?

$$\begin{aligned}(p+1)(p+1) - (p-1)(p-1) &= (p^2 + p + p + 1) - (p^2 - p - p + 1) \dots\dots\dots \\ &= (p^2 + 2p + 1) - (p^2 - 2p + 1) \dots\dots\dots \\ &= p^2 + 2p + 1 - p^2 + 2p - 1 \dots\dots\dots \\ &= 4 \times p \dots\dots\dots\end{aligned}$$

Exercice 1.7

Variables :

a , b et c sont des nombres réels

Début :

a prend la valeur 1

b prend la valeur 2

c prend la valeur b

b prend la valeur $a + c$

a prend la valeur c

Afficher a , b et c

Fin

1. Souligner en couleurs les instructions.
2. Quelles sont les valeurs affichées par l'algorithme ?

..... $a = 2$ $b = 3$ $c = 2$

Ouvrir une interface Python

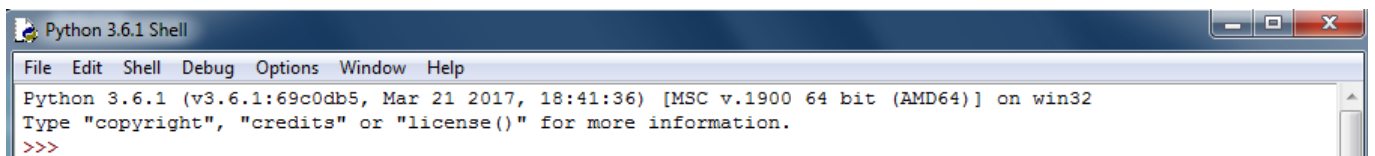
Python est un langage de programmation fonctionnant sur la plupart des plates-formes informatiques.

Au lycée, nous utiliserons la version 3 de Python, dont la syntaxe de certaines commandes diffère légèrement de la version 2.

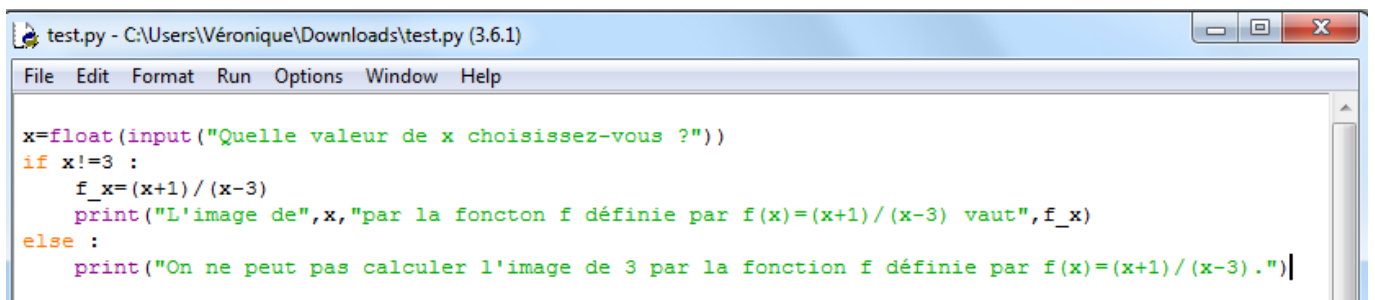
Pour utiliser Python, on utilise souvent un environnement de développement intégré (IDE) tels que IDLE ou EduPython. On peut aussi trouver un interpréteur en ligne sur internet, comme par exemple le site Repl.it.

1. IDLE

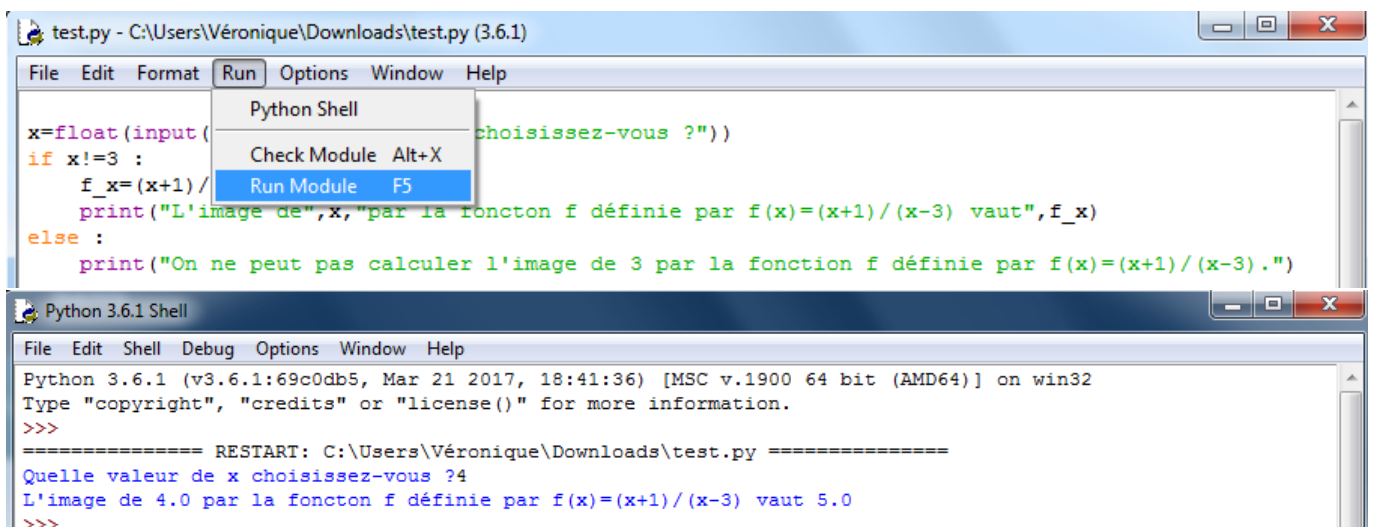
Au lancement d'IDLE, apparaît une seule fenêtre. C'est dans celle-ci que s'afficheront les sorties du programme.



Le programme doit être écrit dans une seconde fenêtre. Pour l'obtenir il faut cliquer sur **File** puis **New File**. Exemple :




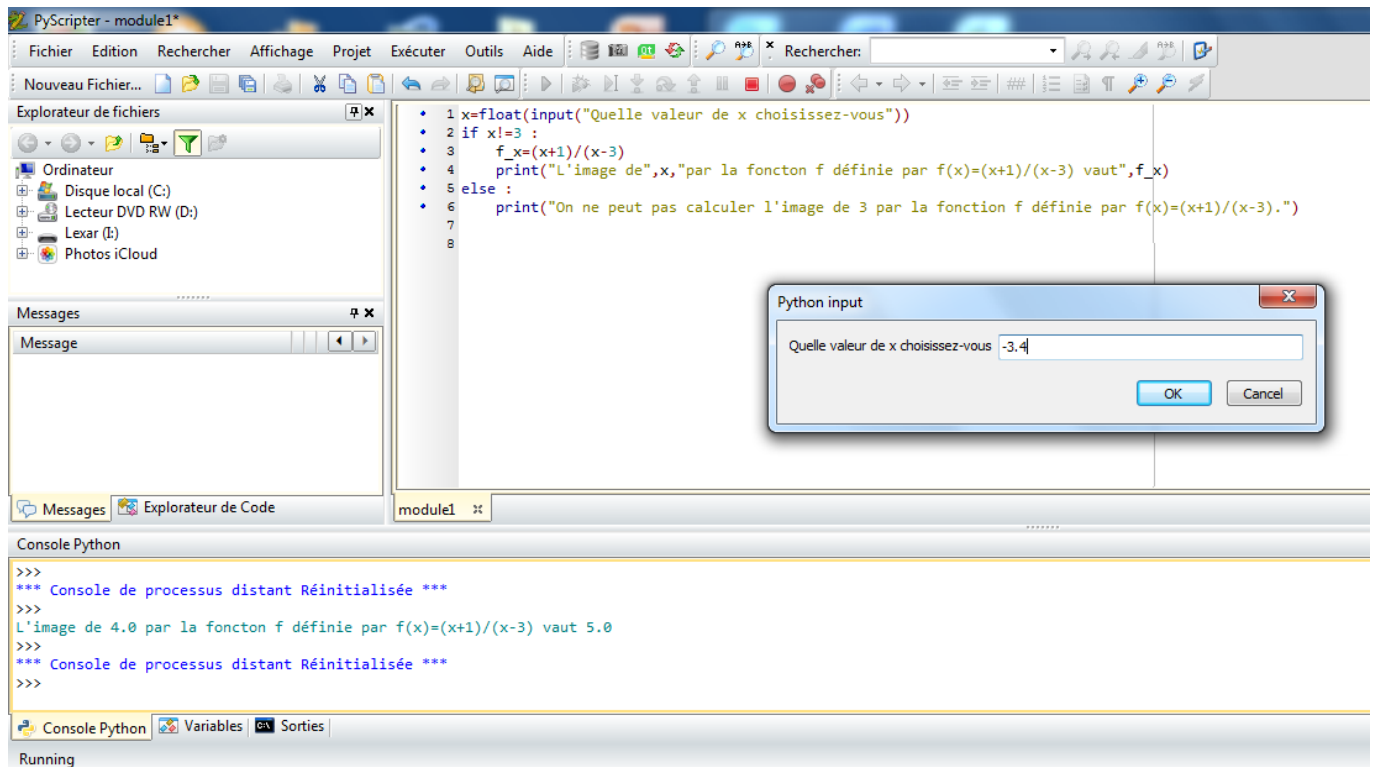
Pour exécuter le programme, il faut cliquer sur **Run** puis **Run Module** et aller voir le résultat dans la première fenêtre :



2. EduPython

EduPython est composé d'une unique fenêtre séparée en plusieurs parties.


Le programme est tapé dans la partie principale. En cliquant sur  les résultats s'afficheront dans la fenêtre en bas d'écran, comme dans l'exemple :

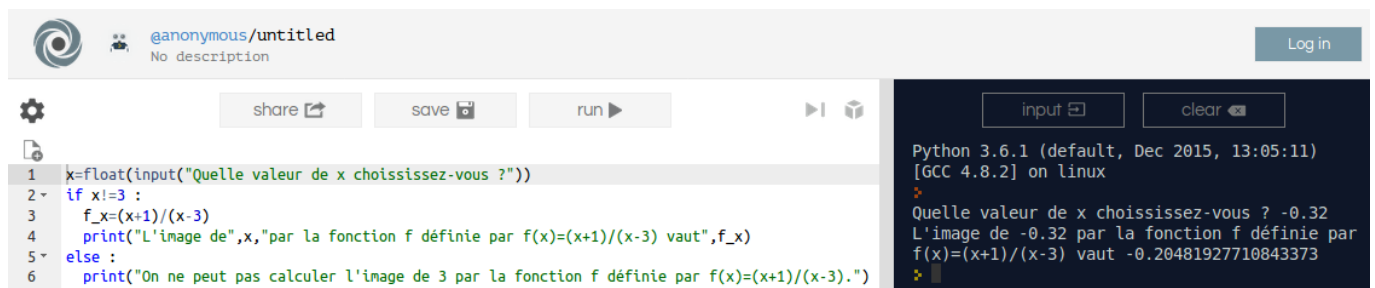


3. Repl.it

Le site internet Repl.it permet d'exécuter des programmes en Python directement en ligne sans aucune installation.

<https://repl.it/languages/python3>

Le programme est tapé dans la partie gauche. En cliquant sur  les résultats s'afficheront sur la partie droite comme dans l'exemple :



TP Python n° 2 : variables, traitement, sorties

Pour attribuer une valeur à une variable, on utilise tout simplement le signe `=`.

Python étant un langage de programmation en anglais, les noms de variables ne doivent pas contenir d'accents. De plus, ils ne doivent pas contenir d'espaces. On choisira un nom de variable explicite pour décrire son contenu.



Exemple

```
age=21
nom="Euler"
```



Exercice 2.1

Si on exécutait le programme ci-dessous, que contiendraient les variables `a`, `b` et `c` ?



```
a=12
b=3
c="orange"
b=a-7
c="pomme"
a=a/b
```

`a` :2,4 `b` : 5 `c` : pomme

Pour afficher un message ou le contenu d'une variable, on utilise la commande `print`.



Exemple

```
nom="Euler"
print("Bonjour, je suis :")
print(nom)
```



Exercice 2.2

1. Taper le programme de l'exercice 2.1.
2. Quelles commandes faut-il ajouter à la fin de ce programme pour vérifier les réponses données dans l'exercice 2.1 ?

..... `print(a)` `print(b)` `print(c)`

Les variables peuvent être de plusieurs types : texte, nombre, ...
Pour connaître le type d'une variable, on utilise la commande `type`.

Exercice 2.3

Quel est le type des variables `a`, `b` et `c` de l'exercice 2.1 ?

`a` : ...décimal (float) ... `b` :entier (int) `c` :texte (str)

Les textes sont vus par Python comme une suite de symboles. En programmation on parle donc de « chaînes de caractères » (string en anglais).

Python différencie les entiers des nombres décimaux. Les calculs sur des nombres entiers (integer en anglais) donnent des résultats toujours exacts, tandis que des calculs sur les nombres décimaux donnent souvent des valeurs approchées. En programmation on ne parle donc pas de décimaux mais de nombres flottants (float en anglais)

Les opérations (+, -, *, /) se comportent différemment selon le type des variables.

Exercice 2.4

Quel est le résultat des instructions suivantes ?

`a+b` :7,4 `c*3` :pommepommepomme
`b*b` :25 `b**2` :25
`c+" d'api"` :pomme d'api `a+c` : ..erreur : unsupported operand ..

Pour demander à l'utilisateur de renseigner la valeur d'une variable, on utilise la commande `input`.



Exemple

```
age=input("Quel est ton âge?")  
print("Tu as",age,"ans.")
```

Exercice 2.5

Quel est le type de la variable `age` dans l'exemple précédent ?texte (str)

Exercice 2.6


Ecrire un programme qui demande un prénom et retourne "Bonjour" suivi du prénom.



```
prenom=input("Comment t'appelles-tu?") .....  
print("Bonjour",prenom,"!") .....
```

Exercice 2.7

Ecrire un programme qui demande une taille (en m) et une masse (en kg) puis affiche l'Indice de Masse Corporelle. L'IMC se calcule grâce à la formule $\frac{\text{masse}}{\text{taille}^2}$.

```
masse=float(input("Masse (en kg)")) .....  
taille=float(input("Taille (en m)")) .....  
IMC=masse/taille**2 .....  
print("L'IMC est",IMC) .....
```

Comme il faut être très prudent avec le type de variable utilisé, on prendra l'habitude d'écrire en commentaires le type des variables utilisé dans le programme. En Python, un commentaire commence par le symbole #.

Exemple

```
# age est un nombre  
age=int(input("Quel est ton âge?"))  
print("Dans trois ans, tu auras",age+3,"ans.")
```

Pour changer le type d'une variable, on utilise les commandes **int**, **float** et **str**.

Exercice 2.8

Compléter le tableau suivant donnant les résultats par les commandes **print(int(a))**, **print(float(a))** et **print(str(a))** pour chaque variable **a**.

a	"2"	"2.1"	"abc"	2	2.1
int2.....	...erreur...	...erreur...2.....2.....
float2.0.....2.1.....	...erreur...2.0.....2.1.....
str"2"....."2.1"..."abc"..."2"....."2.1"...



Exercice 2.9



```
# On importe un outil supplémentaire : la racine carrée
from maths import sqrt
# d, xA, yA, xB, yB sont des nombres
xA=float(input("xA = ?"))
yA=float(input("yA = ?"))
xB=float(input("xB = ?"))
yB=float(input("yB = ?"))
d=sqrt((xB-xA)**2+(yB-yA)**2)
print(d)
```

Que fait ce programme ?

Il retourne la distance entre les points A et B connaissant leurs coordonnées.

.....
Souligner chaque instruction selon le code couleur vu lors du TP Python n° 1.



Exercice 2.10

Ecrire un programme donnant les coordonnées du milieu d'un segment $[AB]$ à partir des coordonnées des points A et B .



```
# xA, yA, xB, yB, xM, yM sont des nombres .....
xA=float(input("xA = ?")) .....
yA=float(input("yA = ?")) .....
xB=float(input("xB = ?")) .....
yB=float(input("yB = ?")) .....
xM=(xA+xB)/2 .....
yM=(yA+yB)/2 .....
print("M(",xM,";",yM,"") .....
```


Souligner chaque instruction selon le code couleur vu lors du TP Python n° 1.

TP Python n° 3 : La boucle « pour »

Lorsque l'on souhaite répéter une ou plusieurs instructions et que l'on sait combien de fois on veut les répéter, on utilise la boucle `for` (Pour).

L'utilisation de cette boucle `for` impose d'écrire en fin de ligne « : » pour annoncer les instructions qui seront répétées dans la boucle. Pour identifier correctement ces dernières, on réalise de plus une indentation (un décalage). Le retour à la ligne sans indentation marque la fin de la boucle (FinPour).

Exercice 3.1


```
 # i est un nombre entier
for i in range(0,10) :
    print(i**2)
```

Remarque : `range` permet de balayer les entiers de 0 inclus à 10 exclu.

Quels nombres sont affichés par ce programme ? .. [les 10 premiers carrés parfaits](#) ..

Exercice 3.2

Écrire un programme affichant la table de multiplication de 7.

```
 # i est un nombre entier .....
for i in range(0,11) : .....
    print(i,"x 7 =",i*7) .....
```


Les boucles sont très utiles pour calculer des sommes.

Exemple

```
# somme et i sont des nombres entiers
somme=0
for i in range(1,100) :
    somme=somme+1
print(somme)
```


Exercice 3.3

Ecrire un programme calculant la somme des 10 premiers carrés parfaits.

```
 # somme et i sont des nombres entiers .....
somme=0 .....
for i in range(0,10) : .....
    somme=somme+i**2 .....
print(somme) .....
```

Exercice 3.4


Une personne souscrit à un plan d'investissement pour 10 ans. Il ouvre un compte en déposant 300 €. En fin d'année, il touche des intérêts s'élevant à 2 % du capital sur ce compte puis au début de l'année suivante il réalise un dépôt de 200 €. Quelle somme est disponible à la fin de ce plan d'investissement ?

```
 # capital est un nombre décimal, i est un nombre entier .
capital=300 .....
for i in range(0,10) : .....
    capital=capital+capital*2/100 # les intérêts .....
    capital=capital+200 # le dépôt .....
print("Il disposera de",capital,"euros.") .....
```

Il disposera de 2555,64 €.

On peut compléter l'instruction **range** par une troisième valeur : le pas.


Exercice 3.5

```
 # i est un nombre entier
for i in range(0,100,2) :
    print(i)
```

Quels nombres sont affichés par ce programme ? **les nombres pairs inférieurs à 100** .


Exercice 3.6

Compléter le programme suivant pour en faire un compte à rebours en partant de 10.

```
 # i est un nombre entier
for i in range( 10 , 0 , -1 ) :
    print(i)
```

Exercice 3.7

Ecrire un programme affichant la somme des multiples de 5 inférieurs ou égaux à 100.


```
 # somme, i sont des nombres entiers .....
somme=0 .....
for i in range(0,101,5) : .....
    somme=somme+i .....
print(i) .....
```

Cette somme est égale à : . 1050 .

Avec la commande **range**, le pas ne peut être qu'un nombre entier. Pour utiliser un pas décimal, on peut utiliser la commande **arange** du module **numpy**


Exercice 3.8

Compléter le programme suivant pour qu'il affiche les nombres 0 ; 0,2 ; 0,4 ; ... ; 9,8.

```
 # i est un nombre entier
from numpy import arange
for i in arange( 0 , 10 , 0.2 ) :
    print(i)
```

Exercice 3.9

1. Ecrire un programme affichant les images des nombres entiers de 0 à 20 par la fonction $f : x \mapsto 2x^2 - 7x - 1$.

```
 # x et un nombre entier .....  
from numpy import arange .....  
for x in arange(0,21) : .....  
    print("f(",x,")=",2*x**2-7*x-1) .....
```

2. Entre quels entiers semble être située la valeur où f atteint son minimum ?


Ce minimum est compris entre 1 et 2.

3. Modifier l'algorithme et donner un encadrement d'amplitude 10^{-2} de cette valeur.

Ce minimum est compris entre 1,75 et 1,76.

Exercice 3.10


La fonction $f : x \mapsto 7x^2 - 5x - 200$ admet un antécédent de 0 compris entre 0 et 10. Ecrire un programme adapté pour donner un encadrement d'amplitude 10^{-3} de celui-ci.

```
 # x et un nombre entier .....  
from numpy import arange .....  
for x in arange(0,11) : .....  
    print("f(",x,")=",7*x**2-5*x-200) .....
```

Cet antécédent est compris entre 5,714 et 5,175.

Exercice 3.11

On considère deux fonctions f et g définies sur $[0; +\infty[$ par $f(x) = \frac{1}{x}$ et $g(x) = 7 - x$. Ecrire un programme adapté pour donner un encadrement d'amplitude 10^{-3} de la solution de l'équation $f(x) = g(x)$.

```
 # x et un nombre entier .....  
from numpy import arange .....  
for x in arange(0,11) : .....  
    print("f(",x,")=",1/x,"g(",x,")=",7-x) .....
```

Cette solution est comprise entre 6,854 et 6,855.

TP Python n° 4 : La boucle « tant que »

Parfois, on souhaite répéter une ou plusieurs instructions si une condition est respectée, sans connaître à l'avance le nombre d'itérations (de répétitions). L'utilisation de cette boucle **while** impose d'écrire en fin de ligne « : » pour annoncer les instructions qui seront répétées dans la boucle. Pour identifier correctement ces dernières, on réalise de plus une indentation (un décalage). Le retour à la ligne sans indentation marque la fin de la boucle (FinTantQue).



Exemple : reste de la division euclidienne de a par b

```
# a, b, r sont des nombres entiers positifs
a=int(input("a=?"))
b=int(input("b=?"))
r=a
while r>b :
    r=r-b
print("Le reste de la division de a par b est",r)
```



Exercice 4.1

Ecrire un programme affichant la première puissance de 3 dépassant 3333.



```
# a est un nombre entier .....
a=1 .....
while a<3333 : .....
    a=a*3 .....
print(a) .....
```

Pour utiliser le nombre d'itérations déjà effectuées, on utilise un compteur. Il faut lui attribuer une valeur (initialisation) avant la boucle **while** et le mettre à jour (incréméntation) à chaque itération.




Exemple : division euclidienne de a par b

```
# a, b, q et r sont des nombres entiers positifs
a=int(input("a=?"))
b=int(input("b=?"))
r=a
q=0 # On initialise le compteur q
while r>b :
    r=r-b
    q=q+1 # On incrémente le compteur
print(a,"=",b,"*",q,"+",r)
```


Dans certains cas, si on oublie d'incrémenter le compteur, la boucle ne peut pas s'arrêter. On force alors l'arrêt du programme grâce à la combinaison des touches **Ctrl** + **C**.

Exercice 4.2


Décrire le résultat des programmes ci-dessous :

```
 # i est un nombre entier
i=0
while i<10 :
    print(i)
```

Ce programme ne s'arrête jamais ! Le compteur n'est pas incrémenté.

```
 # i est un nombre entier
while i<10 :
    print(i)
    i=i+1
```

Une erreur s'affiche : `name 'i' is not defined`. Le compteur n'est pas initialisé.

```
 # i est un nombre entier
i=0
while i<10 :
    print(i)
    i=i+1
```

Ce programme affiche les nombres entiers de 0 à 9



Exercice 4.3

On considère l'algorithme en pseudo-code suivant :

Variables :

a , x et pas sont des nombres décimaux

Début :

x prend la valeur 0

pas prend la valeur 0,001

a prend la valeur $7x^2 - 5x - 200$

Tant que $a < 0$:

x prend la valeur $x + pas$

a prend la valeur $7x^2 - 5x - 200$

Fin Tant que

Afficher $x - pas$

Afficher x

Fin

1. Traduire en python cet algorithme.



```
# a, b, x et pas sont des nombres décimaux
# a est l'image de x,
x=0 .....
pas=0.001 .....
a=7*x**2-5*x-200 .....
while a<0 : .....
    x=x+pas .....
    a=7*x**2-5*x-200 .....
print(x-pas) .....
print(x) .....
```

2. A quoi correspondent les valeurs retournées par cet algorithme ?

Cet algorithme donne un encadrement d'amplitude 10^{-3} de l'antécédent de 0 par la fonction $f : x \mapsto 7x^2 - 5x - 200$.



Exercice 4.4

On considère l'algorithme en pseudo-code suivant :

Variables :

a, b, x et pas sont des nombres décimaux

Début :

x prend la valeur 0

pas prend la valeur 0,005

a prend la valeur $7x^2 - 2x - 1$

b prend la valeur $7(x + pas)^2 - 2(x + pas) - 1$

Tant que $a > b$:

x prend la valeur $x + pas$

a prend la valeur $7x^2 - 2x - 1$

b prend la valeur $7(x + pas)^2 - 2(x + pas) - 1$

Fin Tant que

Afficher $x - pas$

Afficher $x + pas$

Fin

1. Traduire en python cet algorithme.



```
# a, b, x et pas sont des nombres décimaux
# a est l'image de x, b est l'image de x+pas
x=0 .....
pas=0.005 .....
a=7*x**2-2*x-1 .....
b=7*(x+pas)**2-2*(x+pas)-1 .....
while a>b : .....
    x=x+pas .....
    a=7*x**2-2*x-1 .....
    b=7*(x+pas)**2-2*(x+pas)-1 .....
print(x-pas) .....
print(x+pas) .....
```

2. A quoi correspondent les valeurs retournées par cet algorithme ?

Cet algorithme donne un encadrement d'amplitude 10^{-2} du minimum de la
fonction $f : x \mapsto 7x^2 - 2x - 1$

TP Python n° 5 : Tests

Pour exécuter des commandes sous certaines conditions, on utilise le test `if`.

L'utilisation de ce test `if` impose d'écrire en fin de ligne « : » pour annoncer les instructions qui seront exécutées seulement si la condition est vérifiée. Pour identifier correctement ces dernières, on réalise de plus une indentation (un décalage). Le retour à la ligne sans indentation marque la fin du test (FinSi).

Pour les tests, on peut utiliser :

- `>` ou `<` strictement plus grand, strictement plus petit que ;
- `>=` ou `<=` plus grand ou égal à, plus petit ou égal à ;
- `==` égal à ;
- `!=` différent de.



Exemple

```
# age est un nombre entier
age=int(input("Quel âge as-tu?"))
if age<18 :
    print("Tu es mineur.")
```



Exercice 5.1

Lorsqu'il est 19h à Paris, il est 14h à Rio.

Compléter ce programme pour qu'il retourne l'heure à Rio lorsqu'on lui donne l'heure de Paris.



```
# Paris et Rio sont des nombres entiers
Paris=int(input("Quelle heure est-il à Paris?"))
Rio=Paris-5 .....
if Rio<0 : .....
    Rio=Rio+24 .....
print(Rio)
```

La « boucle » `if` peut être complétée par un sinon (`else`) afin d'exécuter d'autres instructions si la condition n'est pas vérifiée. La syntaxe pour la commande `else` est la même que pour l'instruction `if` (utilisation des « : » en fin de ligne et indentations).




Exemple

```
# age est un nombre entier
age=int(input("Quel âge as-tu?"))
if age<18 :
    print("Tu es mineur.")
else :
    print("Tu es majeur.")
```

Exercice 5.2

À l'université, un élève est admis si sa moyenne est supérieure ou égale à 10.

Ecrire un programme qui demande la moyenne d'un élève puis affiche « Admis » ou « Refusé ».

```
 # moyenne est un nombre décimal .....
moyenne=float(input("Quelle est ta moyenne?")) .....
if moyenne < 10 : .....
    print("Refusé") .....
else : .....
    print("Admis") .....
```


On peut aussi détailler le « sinon » en réalisant un autre test grâce à la commande **elif** (même syntaxe que pour les commandes **if** et **else**).

Exemple

```
# age est un nombre entier
age=int(input("Quel âge as-tu?"))
if age < 23 :
    print("Tu es plus jeune que moi.")
elif age > 23 :
    print("Tu es plus vieux que moi.")
else :
    print("Tu as le même âge que moi.")
```

Exercice 5.3

Au rugby, les enfants sont classés en différentes catégories selon leur âge : Poussins (moins de 11 ans), Benjamins (moins de 14 ans) et Minimes (moins de 15 ans). Ecrire un programme demandant l'âge d'un enfant et affichant sa classe d'âge.

```
 # age est un nombre entier .....
age=int(input("Quel est l'âge de l'enfant?")) .....
if age <= 11 : .....
    print("Poussin") .....
elif age <= 14 : .....
    print("Benjamin") .....
elif age <= 15 : .....
    print("Minime") .....
else : .....
    print("Ce n'est plus un enfant!") .....
```

Si deux conditions (ou plus) doivent être vérifiées, on utilise le mot clé **and** (et).

Exemple

```
# age est un nombre entier
age=int(input("Quel est l'âge de l'enfant?"))
if age>11 and age <= 14 :
    print("Cet enfant est un Benjamin")
```



Exercice 5.4

Compléter ce programme pour qu'il indique si $ABCD$ est un parallélogramme.



```
# xA, yA, xB, yB, xC, yC, xD, yD sont des nombres
décimaux
xA=float(input("xA=?"))
yA=float(input("yA=?"))
xB=float(input("xB=?"))
yB=float(input("yB=?"))
xC=float(input("xC=?"))
yC=float(input("yC=?"))
xD=float(input("xD=?"))
yD=float(input("yD=?"))
if xB-xA == xD-xC and yB-yA == yD-yC : .....
    print("ABCD est un parallélogramme") .....
else : .....
    print("ABCD n'est pas un parallélogramme") .....
```

Si une seule condition parmi deux doit être vérifiée, on utilise le mot clé **or** (ou).



Exemple

```
# age est un nombre entier
age=int(input("Quel âge as-tu?"))
if age<0 or age > 120 :
    print("Age non valide")
```




Exercice 5.5

Ecrire un programme qui détermine si deux vecteurs \vec{u} et \vec{v} sont colinéaires.



```
# xu, yu, xv, yv sont des nombres décimaux
xu=float(input("xu=?"))
yu=float(input("yu=?"))
xv=float(input("xv=?"))
yv=float(input("yv=?"))
if xu==0 : .....
    if xv==0 or yu==0 : .....
        print("Ces vecteurs sont colinéaires") .....
    else : .....
        print("Ces vecteurs ne sont pas colinéaires") .....
elif yu*xv/xu==yv : .....
    print("Ces vecteurs sont colinéaires") .....
else : .....
    print("Ces vecteurs ne sont pas colinéaires") .....
```



Exercice 5.6

Écrire un programme qui détermine si trois points A , B et C sont alignés.




```
# xA, yA, xB, yB, xC, yC, xAB, yAB, xAC, yAC sont des
nombres décimaux
xA=float(input("xA=?")) .....
yA=float(input("yA=?")) .....
xB=float(input("xB=?")) .....
yB=float(input("yB=?")) .....
xC=float(input("xC=?")) .....
yC=float(input("yC=?")) .....
xAB=xB-xA .....
yAB=yB-yA .....
xAC=xC-xA .....
yAC=yC-yA .....
if xAB==0 : .....
    if xAC==0 or yAB==0 : .....
        print("Ces points sont alignés") .....
    else : .....
        print("Ces points ne sont pas alignés") .....
elif yAB*xAC/xAB==yAC : .....
    print("Ces points sont alignés") .....
else : .....
    print("Ces points ne sont pas alignés") .....
```

TP Python n° 6 : Listes

Pour pouvoir réaliser des calculs statistiques, un type de variable est particulièrement pratique : les listes.

Exercice 6.1


```
 # L est une liste
L=[1,2,3,4,5,6,7,8,9,10]
print(L[5])
```

Compléter les phrases suivantes :


- a. L[5] donne le 6^{ème} élément.
- b. L[2] donne le 3^{ème} élément.
- c. L[0] donne le 1^{er} élément.
- d. L[-1] donne le ...dernier ... élément.

Pour construire une liste on peut :


- écrire tous les éléments de la liste

```
 Exemple
# L est une liste
L=[1,2,3,4,5,6,7,8,9,10]
print(L)
```


- utiliser **range**

```
 Exemple
# L est une liste
L=range(1,11)
print(L)
```

- utiliser l'opération * qui d'affecte autant de fois qu'on le veut un même élément à une liste

```
 Exemple
# L est une liste
L=[0]*10
print(L)
```

- utiliser une boucle et la commande **append** (ajouter) qui permet d'affecter des éléments à la liste

```
 Exemple
# L est une liste, i est un nombre entier
L=[] # La liste L est vide
for i in range(1,13) :
    L.append(i**2)
print(L)
```

On peut afficher le nombre d'éléments d'une liste L grâce à la commande `len(L)`.



Exemple : nombre de carrés parfait inférieurs à 1000

```
# carres est une liste
carres=[] # La liste carres est vide
i=0
while i**2 <1000 :
    carres.append(i**2)
    i=i+1
print(len(carres))
```

On peut réaliser une boucle faisant appel à chaque élément de la liste.



Exemple

```
# L est une liste, e est un nombre entier
L=[1,1,2,3,5,8,13,21,34,55]
for e in L :
    print(e)
```



Exercice 6.2

Écrire un programme calculant la moyenne de la liste [5,6,8,5,7,7,8,8,8,4].



```
# L est une liste, m, e sont des nombres .....
L=[5,6,8,5,7,7,8,8,8,4] .....
m=0 .....
for e in L : .....
    m=m+e .....
m=m/len(L) .....
print(m) .....
```

Les chaînes de caractères se comporte comme des listes. C'est très pratique pour calculer la somme des chiffres d'un nombre.



Exemple

```
# n, S sont nombres entiers
# txt, e sont des chaînes de caractères
n=1234567890
S=0
txt=str(n) # Conversion en chaîne de caractères
for e in txt :
    S=S+int(e)
print(S)
```

Générer des échantillons aléatoires

Il existe un module en Python pour générer des nombres aléatoires. Celui-ci fournit (entre autres) trois commandes très utiles pour générer des nombres aléatoires :

- **random** génère un nombre aléatoire de l'intervalle $[0; 1[$;
- **randint** génère un nombre entier aléatoire compris entre deux bornes (incluses);
- **choice** permet de choisir un élément au hasard dans une liste.



Exemple

```
# a, b sont des nombres entiers, L est une liste
from random import random, randint, choice
a=0
b=9
L=[0,2,4,6,8]
print(random())
print(randint(a,b))
print(choice(L))
```



Exercice 6.3


On souhaite simuler des parties de piles ou faces avec une pièce bien équilibrée. Compléter le programme suivant pour qu'il génère un échantillon de taille n . On associera 0 au côté pile et 1 au côté face.



```
# i, n sont des nombres entiers, L est une liste
from random import randint
n=int(input("Taille de l'échantillon : "))
L=[] .....
for i in range(0,n) : .....
    L.append(randint(0,1)) .....
print(L) .....
```

Exercice 6.4


On souhaite simuler un lancer de dé équilibré à six faces. Ecrire un programme qui donne un échantillon de n lancers sous forme dépouillée (liste des effectifs).

```
 # a, i, n sont des nombres entiers, L est une liste .....
from random import randint .....
n=int(input("Taille de l'échantillon : ")) .....
L=[0]*7 .....
for i in range(0,n) : .....
    a=randint(1,6) .....
    L[a]=L[a]+1 .....
print(L) .....
```

Exercice 6.5

Ecrire un programme affichant la répartition des fréquences de l'échantillon suivant, obtenu grâce à l'exercice 6.3.

[0,1,0,1,1,0,1,1,0,1,0,0,1,0,1]

```
 # i, somme sont des nombres entiers, L est une liste .....
somme=0 .....
L=[0,1,0,1,1,0,1,1,0,1,0,0,1,0,1] .....
for i in L : .....
    somme=somme+i .....
print("Fréquence de faces :",somme/len(L)) .....
print("Fréquence de piles :",1-somme/len(L)) .....
```

Exercice 6.6

1. Ecrire un programme qui donne la répartition des fréquences d'un échantillon de n lancers d'un dé équilibré à six faces.

```

# a, i, n sont des nombres entiers, E, F sont des listes
from random import randint .....
n=int(input("Taille de l'échantillon : ")) .....
E=[0]*6 .....
F=[0]*6 .....
for i in range(0,n) : .....
    a=randint(1,6) .....
    E[a-1]=E[a-1]+1 .....
for i in range(0,6) : .....
    F[i]=E[i]/n .....
print(F) .....
```

2. Exécuter plusieurs fois ce programme et remplir le tableau suivant avec les fréquences obtenues :

n	1	2	3	4	5	6
10
10
10
1 000
1 000
1 000
100 000
100 000
100 000

3. Que remarque-t-on ?

Sur un petit nombre de lancers, la répartition des fréquences fluctue beaucoup.

Sur un grand nombre de lancers, la répartition des fréquences se stabilise.

Les fréquences sont alors proches des probabilités de ces issues.

TP Python n° 7 : Fonctions

Définition d'une fonction (mathématiques) : Une fonction est un procédé ... qui permet d'associer à un nombre, un unique autre nombre : son image.

En programmation, la notion de fonction est plus large. On distingue donc :

- les fonctions au sens des mathématiques ;
- les fonctions à une ou plusieurs variables dont le type n'est pas toujours un nombre ;
- les fonctions sans variables que l'on appelle **procédures**.

1. Les fonctions au sens des mathématiques

La déclaration d'une fonction se réalise grâce à la commande `def`. Comme pour les boucles, il faut indenter toutes les lignes permettant de calculer l'image du nombre souhaité. L'image à retourner se déclare grâce à la commande `return`.



Exemple

```
# f est une fonction, x est nombre décimal
def f(x) :
    return(7*x-5)
print(f(0))
print(f(1))
print(f(-3))
```



Exercice 7.1

Terminer le programme qui permet de compléter le tableau de valeurs de la fonction $f : t \mapsto t^2 - 3t + 5$.



```
# f est une fonction, t est nombre .....
def f(t) :
    return(t**2-3*t+5) .....
for t in range(-4,5) :
    print(f(t)) .....
```

t	-4	-3	-2	-1	0	1	2	3	4
$f(t)$. 33 .	. 23 .	. 15 .	.. 9 5 3 3 5 9 ..

Une fonction est dite définie par morceaux lorsque l'expression algébrique de cette fonction dépend de l'intervalle dans lequel se situe l'antécédent.

Exercice 7.2

Ecrire un programme qui permet de compléter le tableau de valeur de la fonction définie par morceaux :

$$f : t \mapsto \begin{cases} 0 & \text{si } t \in]-\infty; 0] \\ t^2 & \text{si } t \in]0; 2] \\ 4t - 4 & \text{si } t \in]2; +\infty[\end{cases}$$

```
# f est une fonction, t est nombre .....
def f(t) : .....
    if t<=0 : .....
        return(0) .....
    elif t<=2 .....
        return(t**2) .....
    else : .....
        return(4*t-4) .....
for t in range(-1,3.5,0.5) : .....
    print(f(t)) .....
```

t	-1	-0,5	0	0,5	1	1,5	2	2,5	3
$f(t)$..000 ..	0,25	..1 ..	2,25	..468 ..

2. Les fonctions à une ou plusieurs variables



Exercice 7.3

Écrire une fonction qui, à partir d'un texte, donne le nombre de «e» dans ce texte.



```
# compte_e est une fonction, nb_e est un nombre entier
# texte et lettre sont des chaînes de caractères
def compte_e(texte) :
    nb_e=0 .....
    for lettre in texte : .....
        if lettre=="e" : .....
            nb_e=nb_e+1 .....
    return(nb_e) .....
```



Exercice 7.4

Compléter cette fonction pour qu'elle affiche les images par la fonction f définie dans l'exercice 7.2 de tous les nombres allant de a à b avec un pas p sous forme d'une liste.



```
# tabuler est une fonction, L est une liste
# x, a, b, p sont des nombres décimaux
def tabuler(f,a,b,p) :
    L=[] .....
    for x in range(a,b,p) : .....
        L.append(f(x)) .....
    return(L) .....
```

3. Les procédures



Exercice 7.5

Écrire une procédure générant un échantillon de 100 nombres entiers tous compris entre 0 et 9 sous forme d'une série brute.



```
# echantillon est une procédure, L est une liste .....
from random import randint .....
def echantillon() : .....
    L=[] .....
    for x in range(0,100) : .....
        L.append(randint(0,9)) .....
    return(L) .....
```

TP Python n° 8 : Dichotomie

La méthode de dichotomie est un algorithme de recherche d'un antécédent de zéro par une fonction. Il consiste à répéter des partages d'un intervalle en deux parties puis à sélectionner le sous-intervalle dans lequel appartient l'antécédent recherché.

En langage naturel, on peut l'écrire sous la forme :

- On suppose qu'une fonction f est monotone sur un intervalle $[a; b]$.
- On suppose que $f(a)$ et $f(b)$ sont de signes contraires.
- Répéter :
 - $c = (a + b) \div 2$;
 - si $f(a)$ et $f(c)$ sont de signes contraires, alors $b = c$;
 - sinon $a = c$.

Exercice 8.1

On considère la fonction $f : x \mapsto 3x^2 + x - 2$. On admet que cette fonction est croissante sur l'intervalle $[0; 1]$.

1. Calculer $f(0)$ et $f(1)$ afin de vérifier que ces images sont bien de signes contraires.

$f(0) = \dots\dots\dots -2 \dots\dots\dots$ $f(1) = \dots\dots\dots 2 \dots\dots\dots$

2. Appliquer l'algorithme suivant à cette fonction et remplir le tableau ci-dessous.

Variables :

f est la fonction $x \mapsto 3x^2 + x - 2$

i, a, b et c sont des nombres réels

Début :

a prend la valeur 0

b prend la valeur 1

Pour i allant de 1 à 5 faire :

c prend la valeur $(a + b) \div 2$

 Si $f(c) \geq 0$ alors :

b prend la valeur c

 Sinon :

a prend la valeur c

 FinSi

FinPour

Afficher a et b

Fin

Etape	a	b	c	signe de $f(c)$
Initialisation 0 1 0,5 -
Boucle n° 1 0,5 1 0,75 +
Boucle n° 2 0,5 0,75 0,625 -
Boucle n° 3	... 0,625 0,75 0,6875 +
Boucle n° 4	... 0,625 0,6875 0,65625 -
Boucle n° 5	.. 0,65625 0,6875 0,671875 +

3. Donner un encadrement précis de l'antécédent de zéro par la fonction f .

Cet antécédent est compris entre 0,65625 et 0,671875.


4. Quelle est l'amplitude de cet encadrement ?

$0,671875 - 0,65625 = 0,015625$

Cet encadrement est donné à 0,016 près.

Exercice 8.2

1. Ecrire programme cherchant l'antécédent de 0 compris entre 5 et 10 par la fonction $f : x \mapsto 0,6x^2 - 4x - 4,6$ par la méthode de dichotomie avec 10 itérations.

```
 # x, a, b, c sont des nombres décimaux .....
# f est une fonction, i est un nombre entier .....
def f(x) : .....
    return(0.6*x**2-4*x-4.6) .....
a=5 .....
b=10 .....
for i in range(0,10) : .....
    c=(a+b)/2 .....
    if f(c)>=0 : .....
        b=c .....
    else : .....
        a=c .....
print("[",a,";",b,""])
```

2. Donner un encadrement de l'antécédent de zéro par la fonction f .
Cet antécédent est compris entre 7,66 et 7,67.
3. Quelle est l'amplitude de cet encadrement ? 10^{-2}

Exercice 8.3

1. Quelle ligne faut-il remplacer pour que le programme donne automatiquement un encadrement d'amplitude 10^{-6} ?
L'instruction `for i in range(0,10) :` devient `while b-a>10**(-6) :` .
2. Modifier le programme de l'exercice précédent et donner un encadrement d'amplitude 10^{-6} de la valeur recherchée.
Cet antécédent est compris entre 7,666666 et 7,666667.

Erreurs courantes

❌ Exercice 2.4

— Code tapé :

```
a=2.4 .....  
c="pomme" .....  
print(a+c) .....
```

— Le message d'erreur est :

```
unsupported operand type(s) for +: 'float' and 'str' .....
```

— Il apparaît car :

```
On essaye d'ajouter un nombre à du texte .....
```

❌ Exercice 2.7

— Code tapé :

```
taille=input("taille (en m) :") .....  
masse=input("masse (en kg) :") .....  
IMC=masse/taille**2 .....
```

— Le message d'erreur est :

```
Can't convert 'int' object to str implicitly .....
```

— Il apparaît car :

```
On essaye de diviser deux textes entre eux .....
```

— Solution :

```
Il faut convertir la taille et la masse en nombre en utilisant les commandes : ..
```

```
taille=float(input("Taille (en m) :")) .....
```

```
masse=int(input("Masse (en kg) :")) .....
```

```
.....
```

❌ Exercice 4.2

— Code tapé :

```
while i<10 : .....  
    print(i) .....  
    i=i+1 .....
```

— Le message d'erreur est :

```
NameError: name 'i' is not defined .....
```

— Il apparaît car :

```
On n'a pas initialisé le compteur de la boucle while .....
```


Erreurs courantes



— Code tapé :

.....

.....

.....

— Le message d'erreur est :

.....

— Il apparaît car :

.....

— Solution :

.....

.....

.....



— Code tapé :

.....

.....

.....

— Le message d'erreur est :

.....

— Il apparaît car :

.....

— Solution :

.....

.....

.....

Erreurs courantes



— Code tapé :

.....

.....

.....

— Le message d'erreur est :

.....

— Il apparaît car :

.....

— Solution :

.....

.....

.....



— Code tapé :

.....

.....

.....

— Le message d'erreur est :

.....

— Il apparaît car :

.....

— Solution :

.....

.....

.....