Math93.com

TD - Algorithmique
Débuter en Python - Partie 1

Table des matières

Première partie

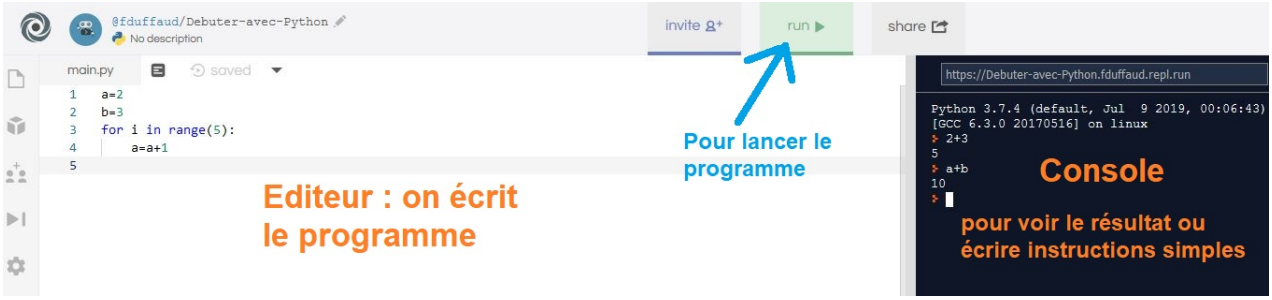
Variables et affichages

I.1. Éditeur et console sous Python

I.1.1 Lancer l'éditeur Python

On ouvre un éditeur Python :

1. avec un éditeur en ligne : www.repl.it
2. avec un éditeur comme Spyder : https ://pypi.org/project/spyder/
3. avec un éditeur comme Edupython : https ://edupython.tuxfamily.org/



I.1.2 Dans la console

La console se reconnaît facilement. C'est elle qui contient le chevron > (ou le triple >>>) qui est l'invite de Python (prompt en anglais) et qui signifie que Python attend une commande.
L'esprit d'utilisation de la console est un peu le même que celui d'une calculatrice.

```
# Dans la console PYTHON
>>>  2+3
5
>>>  a=5
>>>  a-9
-4
```


#

Le symbole # (se lit « croisillon », « hash » en anglais) permet de faire figurer dans le corps du programme un commentaire qui ne sera pas pris en compte lors de son exécution.


=

Le symbole = n'est pas celui de l'égalité mathématique, il n'est d'ailleurs pas symétrique. Il s'agit d'affecter une valeur à une variable : on stocke une valeur numérique ou du texte dans une mémoire.

Exercice 1

Lancer les instructions suivantes dans la console en appuyant sur **Enter** à chaque fin de ligne et regarder le résultat

```
# Dans la console PYTHON
>>> print ( "Hello world !")
>>> x=3
>>> x
>>> 4+5
>>> 5/2  # Division décimale
>>> 7//3 # Quotient de la division entière
>>> print("la valeur de x est", x)
```

I.2. Les types : type(), les variables et les affectations

I.2.1 Les différents types de variables

Voici les différents types de variables que vous devez connaître :

Type	Notation Python	Exemples
Nombres **entiers relatifs**	int( )	> int(−5.5) −5 > type(2) <class 'int'
Nombres **flottants** (décimaux)	float( )	> type(2.0) <class 'float'>
Les **chaînes de caractères (string)**	str( )	> type('a') <class 'str'>
Les **booléens** (True ou False)	bool( )	> type(False) <class 'bool'> > 10 < 2 False > type(2<3) <class 'bool'>
Les **listes**	list( )	> type([1,2]) <class 'list'>

Exercice 2

Donner le type des expressions suivantes

a	type(a)
$a = 2$	
$a = 2.0$	
$a = 2+3$	
$a = 2+3.0$	
$a = 'Bonjour'$	
$a = False$	
$a = 2<3$	
$a =$ "2 < 3"	
$a = [2,3]$	
$a = '2.1'$	

Remarque

On peut afficher ces variables avec la fonction print() par exemple print(a) ou taper seulement a dans la console puis $Enter$

I.2.2 Tester et comparer des variables

Les tests et comparaisons

Une variable booléenne est le résultat True ou False d'une phrase ou d'un test logique.
Exemple :

Le test logique (ou la comparaison) `a<b` peut être **True** ou **False**, tout comme le test `a==b`
Python est capable d'effectuer toute une série de comparaisons entre le contenu de deux variables, telles que :

= =	égal à
! =	différent de
>	supérieur à
> =	supérieur ou égal à
<	inférieur à
< =	inférieur ou égal à

```
# Dans la console PYTHON
>>> 2<3
True

>>> 3 = = 2
False

>>> 3 = 2 # Attention : le symbole égal = est une affectation,
          # cette écriture renvoie une erreur
SyntaxError: cannot assign to literal
```

Exercice 3

Prévoir puis testez les résultats suivants :

```
# Dans la console PYTHON
a = 2
b = 3
c = 5
print( a == b)
print( a+b == c)
print( a < b )
print( a <= c)
print( a==b and a==2)
print(a==b or a==2)
print( type( a == c) )
```

I.2.3 Pour enregistrer vos travaux dans votre éditeur Python

— Sur repl.it : cliquez simplement sur : **+ new repl** et donner un nom à votre fichier. Il sera automatiquement enregistré après chaque **Run**.

— Sinon sur un éditeur hors ligne : Cliquer sur Fichier puis Nouveau puis sélectionner Nouveau Module Python. Enregistrer IMMEDIATEMENT votre programme dans votre répertoire de travail avec le nom Mon1erProgramme.py.

I.3. Variables et affectations simultanées

Exercice 4

On considère l'algorithme ci-dessous écrit sous Python (ne pas le taper).

```
a=2
b=-5
a,b=a+b,a-b
print("Maintenant a= ",a," et b = ",b)
```

1. Compléter le tableau.

Ligne	a	b
L1		
L2		
L3		

2. Vous pourrez ensuite tapez le programme sous Python pour vérifier vos résultats.

On considère l'algorithme ci-dessus écrit sous Python.

```
a=2
b=-5
a=a+b
b=a-b
print("Maintenant a= ",a," et b = ",b)
```

1. Compléter le tableau.

Ligne	a	b
L1		
L2		
L3		
L4		

2. Vous pourrez ensuite tapez le programme sous Python pour vérifier vos résultats.

Remarque

Notez la différence entre les résultats.

- Dans l'exemple de gauche ci-dessus, les valeurs de a et b sont affectées simultanément en utilisant les valeurs des lignes précédentes.
- En revanche dans celui de droite, les affectations sont successives, ce qui explique les résultats différents.

Ainsi $a,b = b,a$ échange les valeurs des deux variables a et b (sans utilisation d'une variable tampon).

Exercice 5

On considère l'algorithme suivant écrit en pseudo code :

L1	**Traitement :**	$U \leftarrow 500$
L2		$N \leftarrow 0$
L3		$U \leftarrow 0.7 \times U + 300$
L4		$N \leftarrow N + 1$
L5		Afficher U , N

1. Compléter le tableau suivant afin de déterminer les valeurs affichées en sortie.

Ligne	U	N
L1		
L2		
L3		
L4		

2. Écrire sous Python ce programme en utilisant le moins de lignes possible.

I.4. Les fonctions mathématiques de bases

Opérations	Interprétation	Exemples de syntaxe	Remarque
+ , - , * , /	addition, soustraction, multiplication et division		
a//b	Partie entière de la division de a par b	> 12//11 1	$12 \div 11 \approx 1,0909$
a % b	Reste de la division euclidienne de a par b	> 17 % 3 2	$17 = 3 \times 5 + 2$
int(a)	partie entière	> int(12.123) 12	
divmod(a,b)	Quotient et Reste de la division euclidienne de a par b	> divmod(20,3) (6,2)	$20 = 3 \times 6 + 2$
a**b ou pow(a,b)	a Puissance b	> 2**3 ou pow(2,3) 8	$2^3 = 8$
a**(1/2)	Racine carrée $\sqrt{a}$	> 9**(1/2) 3	$\sqrt{9} = 3$
a**(1/n)	Racine $n^{\text{ième}}$ de a : $\sqrt[n]{a}$	> 27**(1/3) 3	$\sqrt[3]{27} = 3$
abs(x)	Valeur absolue de x : $\lvert x \rvert$	> abs(-5.2) 5.2	$\lvert -5.2 \rvert = 5.2$
round(a,n)	Arrondie de a à 10^{-n} près	> round(2.2563,2) 2.26	
+ pour les str	concatène deux chaînes de caractères	> "bon"+"jour" bonjour	Pas vraiment mathématique mais je ne savait pas où le mettre

Exercice 6

Compléter le tableau suivant en anticipant le résultat, sans utiliser votre éditeur Python!

Exemples de syntaxe	Résultat obtenu	Exemples de syntaxe	Résultat obtenu
> 25/10		> 5*2	
> 25//10		> 5**2	
> 25%10		> abs(-5.7)	
> int(25/10)		> 'mama'+'mia'	
> round(3.1416,2)		> '2'+'3'+'4'	

Deuxième partie

Les fonctions sous Python

II.1. Présentation des fonctions sous Python

def nom_ fonction(paramètres) :

def nom_ fonction(paramètres) :
instruction 1
instruction 2
...
return valeur

Cela définit une nouvelle fonction, les deux points entraînent une indentation délimitant la déclaration de la fonction.
Le bloc sert à effectuer une série d'actions. Le plus souvent il se termine par *return* pour renvoyer une ou plusieurs valeurs.

Exercice 7

On veut coder une fonction :

$$f(x) = x^2 - x + 41$$

f est le nom de la fonction et x le paramètre de cette fonction
x**2=x*x (notation puissance)

1. Dans l'éditeur PYTHON tapez

```
def f(x):
    valeur=x**2-x+41
    return valeur
```

2. Pour calculer l'image de 5 par f, tapez dans la console f(5). Même chose pour l'image de 8 tapez f(8). Vous devez obtenir l'affichage suivant dans la console python.

```
>>> f(5)
61
>>> f(8)
97
```

   Tester le programme avec des entiers naturels en écrivant dans la console (*à droite sur ripl.it ou votre logiciel IDLE*) directement $f(5)$ ou $f(8)$ par exemple.
3. Dans la fonction ci-dessus, la variable x se nomme paramètre de la fonction. On peut changer ce paramètre à notre guise, vérifiez-le :

```
# Dans l'éditeur PYTHON
def f(a):
    valeur=a**2-a+41
    return valeur
```

4. On peut même renvoyer directement l'image en gagnant une ligne (et une variable) :

```
# Dans l'éditeur PYTHON
def f2(x):
    return x**2-x+41
```

5. Il est possible d'effectuer des calculs directement avec les valeurs renvoyées par la fonction. Essayez :

```
# Dans la console PYTHON
>>> f(5)
61
>>> 2*f(5)+1 # doit renvoyer 2x61 + 1 = 123
123
```

6. Renvoyer un print(), c'est le mal!

ATTENTION

DANS LA MESURE DU POSSIBLE ON PROSCRIRA LE RENVOI D'UNE VALEUR AVEC print

```
# Dans l'éditeur PYTHON
def f3(x):
    valeur=x**2-x+41
    print( valeur) #C'EST TRES MAUVAIS
```

Essayer d'effectuer le calcul suivant comme dans la question 4°) :

```
# Dans la console PYTHON
>>> f3(5)
61
>>> 2*f3(5)+1
61
TypeError: unsupported operand type(s) for *: 'int' and 'NoneType'
```

II.2. Calculer le résultat d'une somme, produit, différence ..

II.2.1 Calculer le résultat d'une somme

Exercice 8

1. Écrire en python la fonction **somme** de paramètres *a* et *b*, qui renvoie la somme de deux nombres *a* et *b*.
2. TESTEZ votre fonction.

```
# Dans l'éditeur PYTHON
def somme(a,b): # cette fonction a 2 paramètres, a et b
        ...
        return ...
```

```
# Les tests dans la console PYTHON
>>> somme(5,3)
8
>>> somme(8,24)
32
```

On peut aussi tester en lançant le programme suivant :

```
# Dans l'éditeur PYTHON
def somme(a,b):
        ...
        return ...
print(somme(5,3)) #L'instruction est en dehors de la fonction.
print( 'somme(8,24)=', somme(8,24) )
```

II.2.2 Calculer le résultat d'une différence

Exercice 9

1. Écrire en python la fonction **difference** de paramètres *a* et *b*, qui renvoie la différence de deux nombres a et b. Remarquez que j'évite les accents dans la définition de ma fonction, c'est important pour la suite.
2. TESTEZ votre fonction.

```
# Dans l'éditeur PYTHON
def difference(a,b):
        ...
        return ...
```

```
# les tests dans la console PYTHON
>>> difference(5,3)
2
>>> difference(8,24)
-16
```

II.2.3 Calculer le résultat d'un produit

Exercice 10

1. Écrire en python la fonction **produit** de paramètres *a* et *b*, qui renvoie le produit de deux nombres a et b.
2. TESTEZ votre fonction.

```
# Dans l'éditeur PYTHON
def produit(a,b):
        ...
        return ...
```

```
# Les tests dans la console PYTHON
>>> produit(5,3)
15
>>> produit(8,24)
192
```

II.3. La division euclidienne

II.3.1 Calculer le quotient de la division euclidienne

Exercice 11

1. Écrire en python la fonction **EuclideQuotient** qui renvoie le quotient de la division euclidienne de deux entiers a et b. Si vous ne connaissez pas la syntaxe de l'opération, consultez le tableau page 6 ou allez chercher dans L'essentiel de python
2. TESTEZ votre fonction.

```
# Dans l'éditeur PYTHON
def EuclideQuotient(a,b):
        ...
        return ...
```

```
# les test dans la console PYTHON
>>> EuclideQuotient(5,3)
1
>>> EuclideQuotient(30,7)
4
```

II.3.2 Calculer le reste de la division euclidienne

Exercice 12

1. Écrire en python la fonction **EuclideReste** qui renvoie le reste de la division euclidienne de deux entiers a et b. Si vous ne connaissez pas la syntaxe de l'opération, consultez le tableau page 6 ou allez chercher dans L'essentiel de python
2. TESTEZ votre fonction.

```
# Dans l'éditeur PYTHON
def EuclideReste(a,b):
        ...
        return ...
```

```
# Les tests dans la console PYTHON
>>> EuclideReste(5,3)
2
>>> EuclideReste(55,7)
6
```

II.4. Des fonctions

II.4.1 Fonction de test

Exercice 13

Voici une fonction **mystere(a,b)** écrite en python.

```
# Dans l'éditeur PYTHON
def mystere(a,b):
        valeur = a>b
        return valeur
```

1. Testez cette fonction avec les valeurs suivantes :

```
# Dans la console PYTHON
>>> mystere(5,3)

>>> mystere(5,5)

>>> mystere(3,5)

>>> mystere(5*5,25)

>>> mystere(3.4-3.3,1)
```

   Que fait cette fonction si on utilise des entiers (int) où des flottants (float) comme paramètres? Quelle type de variable renvoie-t-elle?

2. Essayez avec les valeurs suivantes :

```
# Dans la console PYTHON
>>> mystere("a","b")

>>> mystere("b","a")

>>> mystere("ab","ac")

>>> mystere("ac","ab")

>>> mystere("manger","mangez")
```

   Que fait cette fonction quand les paramètres sont des chaînes de caractères (str) minuscules sans accent?

II.4.2 Fonction Échange

Exercice 14

1. On donne le programme suivant.

```
# Dans l'éditeur PYTHON
x=1
y=5
print(x,y)
tmp=x
x=y
y=tmp
print (x,y)
```

   Avant d'essayer le programme, estimer les variables x et y après l'exécution de ce programme . Puis VÉRIFIEZ.

2. On donne le programme suivant.

```
# Dans l'éditeur PYTHON
def echange(x,y):
        tmp=x
        x=y
        y=tmp
a=1
b=5
echange(a,b)
print (a,b)
```

   Avant d'essayer le programme, que vaut la variable a après l'exécution de ce programme? Puis VÉRIFIEZ, vous risquez d'avoir une surprise.

3. Que va affichez le programme précédent si on rajoute à la fin l'instruction

   ```
   print( echange(a,b) )
   ```

   Vérifiez. Pourquoi cette affichage?

4. Dans la fonction, rajoutez l'instruction

   ```
   return x,y
   ```

```
# Dans l'éditeur PYTHON
def echange(x,y):
        tmp=x
        x=y
        y=tmp
        return x,y
```

   et testez le programme. Est ce que les valeurs de a et b ont été échangées? Vérifiez en rajoutant

   ```
   print( a,b )
   ```

   à la fin du programme.

5. Que peut on en déduire sur les variables utilisées dans une fonction?

II.4.3 Faire d'autres fonctions

Exercice 15

Coder en python les fonctions suivantes :

1. Le carré d'un nombre
2. La fonction inverse
3. La fonction qui concatène trois mots.

Par exemple

```
>>>tripleConcatenation("do","mi","nation")
domination
```

Troisième partie

Avec le module math

III.1. Périmètre et aire

Remarque

Les fonction mathématiques de base sont présentes dans le *module math* qu'il faut appeler au début du programme sous la forme **import math**.
Pour le nombre π écrire tout simplement : **math.pi**

Astuce : En écrivant **from math import *** au début on importe directement toutes les fonctions et **from math import pi** on importe le nombre pi (enfin une valeur approchée!) ce qui permet d'écrire directement pi (au lieu de math.pi).
Les puristes ont tendances à proscrire l'utilisation de **from math import *** ...

Docstring

Les **docstrings** sont des chaines de caractères qui doivent être placées juste en dessous des définitions de fonction et entre 3 apostrophes ' ' '. On définit ainsi les variables de la fonction en entrée (IN) et celles en sortie de la fontion (OUT). Les docstrings sont récupérables dynamiquement avec la fonction **help( )**.

```
def f(x):
    ''' In : x décimal (un flottant)
        Out : image de x par f definie par f(x) = x^2 + 1'''
    return x**2 + 1
```

Exercice 16

Écrire une fonction de paramètre r qui renvoie le périmètre d'un cercle de rayon r puis une autre (aussi de paramètre r) qui renvoie l'aire du disque de rayon r.

```
from math import pi

def perimetre(r):
        '''IN : r le rayon du cercle float
          OUT : une valeur approchée du périmètre du cercle de rayon r float'''
        return ...

def aire(r):
        '''IN : r le rayon du disque float
          OUT : une valeur approchée de l'aire du disque de rayon r float'''
        return ...
```

III.2. (Optionnel) Input() ou float(input()

nom=input("question") et a=float(input("question"))

— *var=input("question")* :
La fonction *input*, dont la syntaxe est : *nom=input(question)* affiche une fenêtre où figure le texte *question* et un cadre blanc dans lequel on entrera ce qui est demandé.
La réponse est alors affectée à la variable *var* qui est alors considérée comme une chaine de caractère (un mot).

— *a=float(input("question"))* : si la valeur de a demandée est considérée comme un nombre flottant,

— *a=int(input("question"))* : pour un entier.

Par exemple :

```
# Dans l'éditeur PYTHON
# On peut définir une fonction sans paramètre,
# juste pour éviter de l'appeler à chaque RUN
def questions():
    nom=input("Donnez votre nom svp ")
    age=int (input("Donnez votre age svp ")   )
    quizz=float (input("Diviser 1 par 4 svp ")   )
    print("Bonjour",nom, "vous avez" ,age, "ans et avez dit que 1/4 donnait" ,quizz)
```

```
# Dans la console PYTHON on obtient :
>>> questions()
Donnez votre nom svp Evariste
Donnez votre age svp 31
Diviser 1 par 4 svp 0.25
Bonjour Evariste, vous avez 31 ans et avez dit que 1/4 donnait 0.25
```

Exercice 17

Faire un quizz. Construire la fonction pythonQuizz() qui pose les questions suivantes :

1. De quelle type est la variable a=3 ?
2. De quelle type est la variable b=25.6 ?
3. De quelle type est la variable c=25>2 ?
4. De quelle type est la variable d="Bonjour" ?
5. Quelle est l'instruction qui permet d'afficher un message ?

Et qui affiche les réponses de l'utilisateur. Comme par exemple :

```
Vous pensez que en python
a est du type nombre
b est du type décimal
c est du type inégalité
d est du type phrase
un message s'affiche avec l'instruction return
```

Toutes ces réponses sont fausses, appelez le professeur quand vous pensez avoir réussi votre programme et savoir répondre à ces questions.

Quatrième partie

Instructions conditionnelles

IV.1. Si ... alors... sinon

if *test*

effectue (une fois) les instructions indentées qui suivent lorsque le test est vérifié.

Indentation pour les blocs

— Dans le langage Python, on peut passer des lignes pour plus de clarté, ce qui n'est pas pris en compte lors de l'exécution du programme. Par contre, vous ne pouvez pas ajouter un espace en début de ligne comme bon vous semble, car cela a une signification.

— Indentation
On appelle *indentation* ce décalage d'un cran d'une ou de plusieurs lignes d'un programme. Elle permet de délimiter un bloc d'instructions dans une boucle ou lors d'une exécution conditionnelle.

— Deux points " :"
La ligne précédent l'indentation se finit toujours par deux points. Quand vous appuyez sur la touche après avoir tapé « : », l'indentation est automatiquement effectuée en même temps que le passage à la ligne.

— On peut aussi appuyer sur la touche de tabulation *Tab*, à gauche de la touche "A" pour gérer l'indentation, mais les deux points : sont toujours nécessaires.

1. Le programme ci-dessous cherche à résoudre l'équation : $a\,x = b$.

```
def soleq(a,b):
    '''IN : a et b flotants
      OUT : solution éventuelle de l'équation ax=b'''
    if a!=0: # si a est différent de 0
        return b/a
```

Exécutez-le dans la console de droite avec deux valeurs de votre choix. Il permet la résolution de l'équation $ax = b$ avec un test pour savoir si a est bien différent de zéro.
La condition « a différent de zéro » s'écrit « *a!=0* » . Aucun affichage n'est proposé ici.

2. Si on entre *soleq(0,2)* observer le résultat produit. Pour compléter, on va utiliser la structure « *if ... else* »

if test : ... else : ...

if test : ... else : ... : Effectue les instructions indentées lorsque le test est vérifié, sinon effectue les instructions alternatives indentées.
Remarques :

— *Le "else" est aligné avec le "if" qui lui correspond.*

— *Taper " :" puis appuyer sur la touche "entrée" pour passer à la ligne, provoque l'indentation automatique.*

```
def soleq(a,b):
    '''IN : ...
      OUT : ...'''
    if a!=0: # si a est différent de 0
        return b/a
    else:
        if b==0:
            return "R"
        else:
            return "pas de solution"
```

Exercice 18

On cherche maintenant à résoudre une **équation de type $ax+b=c$**.

1. Commencer par résoudre, à la main, l'équation $2x+3=4$.
2. Proposer une fonction nommée soleq2(a,b,c) renvoyant la solution de cette équation.
3. Vérifier votre programme avec l'équation de la question 1.

Les Conditions du test

- **a!= 0** : La condition « *a* différent de zéro » s'écrit « *a != 0* ».
- **a = = 0** : La condition « *a* égal à zéro » s'écrit « *a = = 0* », (avec deux fois le symbole =).
- **< et >** : Ces symboles désignent les inégalités strictes habituelles.
- **<= et >=** : Ces combinaisons de symboles désignent les inégalités larges ≤ et ≥ habituelles.
- **and et or**
  - **and** : Permet d'effectuer une instruction si deux tests sont vérifiés simultanément.
  - **or** : Permet d'effectuer une instruction si au moins un test sur deux est vérifié.

IV.1.1 Un programme de calcul

Exercice 19

Voici un programme de calcul :

- Choisir un nombre entier ;
- Si il est pair, le diviser par 2 ;
- Sinon le multiplier par 3 et ajouter 1.
- Renvoyer le résultat.

1. Appliquer ce programme de calcul à 8 et à 11 (à la main).
2. Écrire un algorithme sous forme d'une fonction de paramètre *n* correspondant à ce programme de calcul. On pourra compléter le programme ci-après.
3. Que se passe-t-il lorsqu'on entre un décimal ? Pourquoi ?
4. Élaborer une stratégie pour éviter cette erreur et modifier le programme en conséquence.

Aide

a % b : Renvoie le reste de la division euclidienne de *a* par *b*. Et si un nombre est pair, le reste de sa division euclidienne par 2 est nul ...
Par exemple $17 = 7\times 2+3$ *donc le reste de la division euclidienne de 17 par 7 est 3. Donc :*

```
>>>17%7
 3
```

```
def programme(n):
    '''IN : n un entier (int)
      OUT : ...'''
    if ...
        return ...
    else:
        return ...
```

IV.1.2 Une fonction définie par morceaux et Géogébra

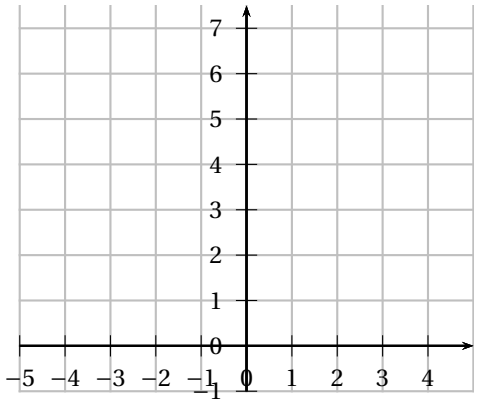
Exercice 20

1. Écrire un programme utilisant une fonction qui permet de calculer l'image d'une valeur demandée, par la fonction f définie sur $\mathbb{R}$ par :

$$f_{20} : x \longmapsto f_{20}(x) = \begin{cases} -2x & \text{si } x < 0 \\ x^2 & \text{si } x \geq 0 \end{cases}$$

```
def f20(x):
    if ...
        return ...
    else:
        return ...
```

2. Compléter alors la tableau de valeurs suivant et tracer $\mathscr{C}_f$ dans le repère ci-contre.



x	-3	-1	0	$0,5$	1	$1,5$	2	$2,5$
$f_{20}(x)$								

IV.1.3 Des photocopies

Exercice 21

Un magasin de reprographie propose un tarif dégressif. Les 20 premières photocopies sont facturées à 10 centimes et les suivantes à 8 centimes.

1. Calculer à la main le coût de 15 puis de 30 photocopies.
2. Écrire un algorithme utilisant une fonction qui renvoie le montant de la facture en euros pour un nombre de photocopies donné.
3. Écrire l'expression de cette fonction en fonction des valeurs de x.

$$facture : n \longmapsto facture(n) = \begin{cases} \cdots\cdots\cdots & \text{si } \cdots\cdots \\ \cdots\cdots\cdots & \text{si } \cdots\cdots \end{cases}$$

IV.2. if test1 : instructions1 elif test2 : instructions2

if test1 : instructions1 elif test2 : instructions2

if test1 : instructions1 elif test2 : instructions2 : Effectue les instructions1 indentées lorsque le test1 est vérifié, sinon effectue le test2 et, si celui-ci est vérifié, effectue les instructions2 indentées.
Remarques :

- *On peut enchaîner autant de "elif" que nécessaire.*
- *il faut terminer une série de "elif" par un "else" afin d'être sûr de traiter tous les cas.*

Exercice 22

1. On considère le programme ci-dessous qui calcule l'image d'un nombre x par une fonction h définie par morceaux sur $\mathbb{R}$. Écrire l'expression de cette fonction en fonction des valeurs de x.

$$h : x \longmapsto h(x) = \begin{cases} \cdots\cdots\cdots & \text{si } \cdots\cdots \\ \cdots\cdots\cdots & \text{si } \cdots\cdots \\ \cdots\cdots\cdots & \text{si } \cdots\cdots \end{cases}$$

2. Exécuter ce programme (dans la console) pour différentes valeurs de x puis tracer la courbe représentative de cette fonction sur l'intervalle $[-5;5]$ dans un repère de votre choix. *Vous pourrez utiliser le logiciel Geogébra.*

```
def h(x):
    '''IN : x un flottant
      OUT : image de x par f'''
    if x<0:
       return 2*x+3
    elif x<2: # Donc x >=0 et x<2 soit 0 <= x < 2
       return 3-x
    else: # cas où x >= 2
       return x**2-3
```

IV.2.1 Résoudre $x^2 = a$

Exercice 23

Compléter le programme ci-dessous pour résoudre l'équation $x^2 = a$.

```
from math import sqrt

def resolxcarre(a):
    '''IN : ...
      OUT : ...
    '''
    if a<0:
       return ...
    elif ....
      return ...
    else:
       return ...
```

Cinquième partie

Structures itératives

Comme dans la plupart des langages, il existe en Python principalement deux manières de réaliser une boucle, c'est à dire une répétition d'un bloc d'instructions. Comme pour l'instruction « *if* » , la partie à répéter sera indentée vers la droite, ce qui permet en plus une bonne visibilité de l'algorithme.

V.1. Boucle dont on connaît le nombre d'itérations

Boucle for

for *var* ***in*** *L* :
    instructions

Réalise une boucle en faisant parcourir à la variable *var* toutes les valeurs de la liste L.

Boucle for

for *var* ***in*** *range(n)* :
    instructions

Réalise une boucle en faisant parcourir à la variable *var* toutes les valeurs de 0 à $n-1$. C'est donc équivalent à écrire en pseudo-code :

Pour *var* ***allant*** *de 0 à n-1* ***Faire***
    instructions

range(début , fin , pas)

range(début , fin , pas) : Le type range représente une séquence immuable de nombres et est couramment utilisé pour itérer un certain nombre de fois dans les boucles for. Les paramètres *début* et *pas* sont optionnels.
L'avantage du type range sur une list classique est qu'un objet range prendra toujours la même (petite) quantité de mémoire, car elle ne stocke que les valeurs start, stop et step.

- Dans l'intervalle [0 , fin[si un seul paramètre est renseigné.
  L = list(range(4)) va créer la liste [0 , 1 , 2 , 3] de 4 termes, le premier sera L[0] = 0,
  le dernier L[3] = 3.
- Dans l'intervalle [début ; fin[si 2 paramètres sont renseignés.
  L = list(range(1 , 5)) va créer la liste [1 , 2 , 3 , 4] le premier terme sera L[0] = 1 *et le dernier L*[3] = 4.
- Dans l'intervalle [début ; fin[mais de *pas* en *pas*, si les 3 paramètres sont renseignés.
  L = list(range(2 , 9 , 2)) va créer la liste [2 , 4 , 6 , 8] .

Remarque

On fait une boucle quand on veut éviter d'écrire une répétition d'instructions du même genre. Par exemples :

```
for i in range(5):
                print ("Chocolat")
```

peut être remplacé, si on **développe** la boucle par

```
print ("Chocolat")
print ("Chocolat")
print ("Chocolat")
print ("Chocolat")
print ("Chocolat")
```

La variable i, dans l'exemple suivant, varie en fonction de chaque étape :

```
for i in range(5):
            print (i)
```

peut être remplacé, si on **développe** la boucle par

```
print (0)
print (1)
print (2)
print (3)
print (4)
```

V.1.1 Des exemples de boucles

Exercice 24

Ecrire le développement de la boucle suivante. Que vaut s à la fin de ce programme?

```
for i in range(5):
            s=i
```

Exercice 25

Écrire la deuxième partie de ce code avec une boucle **for**. Que vaut s à la fin de ce code?

```
#Premiere partie
s=0
#Deuxieme partie
s=s+0
s=s+1
s=s+2
s=s+3
s=s+4
s=s+5
```

Exercice 26

1. On considère le programme suivant. Exécutez-le en écrivant simplement exoA(L5) dans la console. Modifiez-le pour qu'il affiche « *Morning!* »

```
# Dans l'éditeur
# On définit une liste L5
L5 = ['B' , 'o' , 'n' , 'j' , 'o' , 'u' , 'r' , '!']
def exoA(liste):
     for i in liste :
             print (i)
```

```
# Dans la console PYTHON
>>> exoA(L5)
```

Remarque

end = "" permet de ne pas renvoyer à la ligne automatiquement après une instruction print().
Tester donc la même fonction avec print (i,end = "").

2. On considère le programme suivant. Exécutez-le en écrivant simplement exoB() dans la console. Modifiez-le pour qu'il affiche les entiers de 0 à 15 .

```
def exoB():
    for n in range(10) :
        print (n)
```

3. On considère le programme suivant. Exécutez-le en écrivant simplement exoC() dans la console. Modifiez-le pour qu'il affiche les entiers de 5 à 10 .

```
def exoC():
    for n in range(7,13) :
        print (n)
```

4. Modifiez le programme suivant pour qu'il affiche les entiers impairs de 5 à 17 puis créez une fonction exoD(A,B) qui renvoie la liste des entiers de A à B de deux en deux.

```
def exoD():
    Liste = [ i for i in range( 10 , 20 , 2 ) ]
    return Liste
```

Remarque

S'inspirant de l'écriture mathématique d'un ensemble en compréhension, par exemple

$$\left\{x \mid x \in [\![0\,;\,19]\!]\right\}$$

Python propose une syntaxe utile pour la création d'une liste en compréhension :

$$[\ x \text{ for } x \text{ in range}(20)\ ]$$

V.1.2 Table de multiplications

Voici un programme qui donne la table d'additions de l'entier n (de 0 à 9).

```
def table_addition(n):
    for i in range(10):
        print(i,'+',n,'=',i+n)
```

Exercice 27

1. Testez la fonction **table_addition(n)**.
2. Écrire une fonction **table_multiplication(n)** qui affiche la table de multiplication de n, (de 1 à 10). Voici ce qui vous devrez obtenir :

```
# Dans la console PYTHON on obtient :
>>> table_multiplication(9)
1 * 9 = 9
2 * 9 = 18
3 * 9 = 27
4 * 9 = 36
5 * 9 = 45
6 * 9 = 54
7 * 9 = 63
8 * 9 = 72
9 * 9 = 81
10 * 9 = 90
None
```

V.1.3 Un peu de poésie

Exercice 28

1. Ecrire le programme suivant et tester le

```
1 A=["parrain","peureux","roi","patraque","punk"]
2 for i in range(5):
3     print("A l'étage",i+1,"je suis",A[i])
```

2. Compléter la liste A et changer le 5 de la ligne 2 pour aller jusqu'à l'étage 10.
3. Voici les deux premiers vers du poème *Voyelles* d'Arthur Rimbaud

A noir, E blanc, I rouge, U vert, O bleu : voyelles,
Je dirai quelque jour vos naissances latentes :

Il faut faire apparaître les lignes suivantes

```
A est noir
E est blanc
I est rouge
O est bleu
U est vert
Y est indeterminé
```

Ecrire et compléter le programme

```
couleurs=["noir",......]
voyelles=["A",......]
longueur=len(voyelles)
for i in range(...):
    print(...,"est",...)
```

V.1.4 Déterminer les puissances d'un nombre entier

Exercice 29

Voici un algorithme en pseudo code d'une fonction qui pour un nombre entier $n \geq 1$ en entrée, calcule un nombre x et l'affiche en sortie

```
fonction puiss(n):
    x prend la valeur 1
    Pour i allant de 0 à n-1
        x prend la valeur 5x
    Fin Pour
    Renvoyer x
```

1. Appliquer cet algorithme avec $n = 6$ et compléter le tableau dans lequel on suit les valeurs successives prises par les variables i et x.

A compléter sur cette feuille

$\boldsymbol{n = 6}$

i	\	0	1	2	3	4	5	6
x	0							

2. Quelle est alors pour $n = 6$ la valeur renvoyée en sortie?
   .....
3. On applique maintenant l'algorithme avec n=10. Quelle est alors la valeur renvoyée en sortie?
   .....
4. Traduire le pseudo code en python et le tester, en particulier pour n compris entre 0 et 10.
5. De façon plus générale exprimer en fonction de n la valeur renvoyée par la fonction **puiss**.
   .....

V.1.5 Somme des entiers

Exercice 30

1. Calculer à la mains la somme $S(10)$ des entiers de 0 à 10 puis la somme $S(15)$ des entiers de 0 à 15.

$$S(10) = 0+1+\cdots+10 = \cdots \quad \text{et} \quad S(15) = 0+1+\cdots+15 = \cdots$$

2. On cherche une fonction qui renvoie la somme des entiers de 0 à n, où n est le paramètre. Compléter le programme et vérifier que la valeur en sortie est correcte pour plusieurs valeurs de n.

```
def somme(n):
    '''IN : entier n >=0
      OUT : somme des entiers de 0 à n'''
    s = 0 # on initialise s à 0
    for i in range (...):
            s= ...
    return s
```

3. Il est souvent utile de compléter un tableau avec les valeurs des variables pour chaque itération . Faites-le ici pour :

A compléter sur cette feuille

$\boldsymbol{n = 10}$

i	X	0	1	2	3	4	5	6	7	8	9	10
s	0											

4. Somme des impairs.
   (a) Calculer la somme des entiers impairs inférieurs à 10 :

$$I(10) = 1+3+5+7+9 = \cdots$$

   (b) Écrire un programme qui renvoie la somme des entiers impairs de 1 à n entier , où $n > 0$.

V.1.6 Somme de carrés

Exercice 31

1. Calculer à la mains la somme C_1 des carrés entiers de 0 à 5 puis la somme C_2 des carrés des entiers de 0 à 10.

$$C(5) = 0^2 + 1^2 + 2^2 + \cdots + 5^2 = \cdots \quad \text{et} \quad C(10) = 0^2 + 1^2 + 2^2 + \cdots + 10^2 = \cdots$$

2. On cherche une fonction qui renvoie la somme des carrés entiers de 0 à n, où n est le paramètre. Compléter le programme et vérifier que la valeur en sortie est correcte pour plusieurs valeurs de n.
3. Il est souvent utile de compléter un tableau avec les valeurs des variables pour chaque itération . Faites-le ici pour :

A compléter sur cette feuille

$n = 10$.

i	X	0	1	2	3	4	5	6	7	8	9	10
s	0											

```
def sommecarres(n):
    '''IN : entier n >=0
      OUT : somme des carrés des entiers de 0 à n'''
    s = 0 # on initialise s à 0
    ...
```

V.1.7 Somme des inverses des carrés

Exercice 32

1. Calculer la somme des inverses des carrés de 1 à 4 :

$$I(4) = \frac{1}{1^2} + \frac{1}{2^2} + \frac{1}{3^2} + \frac{1}{4^2} = \cdots$$

2. Écrire une fonction de paramètre n (avec n entier), qui renvoie la somme des inverses des carrés de 1 à n. Tester votre programme avec le calcul précédent.
3. Calculer des valeurs pour n très grand et conjecturer la limite de cette somme.

   Comparer votre résultat au nombre $\frac{\pi^2}{6} \approx 1,64493416$.

   Pour calculer une valeur approchée de ce nombre, n'oubliez-pas d'importer le module math.

```
def inv(n):
    '''IN : entier n >=0
      OUT : somme des inverses des carrés des entiers de 1 à n'''
    ...
```

Remarque historique

C'est le génial mathématicien suisse Leonhard Euler (1707-1783) qui démontre le premier que cette somme converge (on parle de série convergente). Il prouve ce merveilleux résultat :

$$\frac{1}{1^2} + \frac{1}{2^2} + \frac{1}{3^2} + \frac{1}{4^2} + \cdots = \frac{\pi^2}{6}$$

V.2. Boucle dont on ne connaît pas le nombre d'itérations

Dans la pratique, on ne connaît que rarement le nombre d'itérations pour arriver au résultat (d'où l'intérêt d'un programme). On peut alors utiliser des boucles de type TANT QUE .... FAIRE : ...

while condition :

***while** condition* : Exécute une instruction ou un bloc d'instructions tant que la condition est vérifiée. (La boucle peut donc ne jamais être exécutée si, d'entrée la condition n'est pas remplie).

Un exemple :

```
# Dans l'éditeur PYTHON
n=1
while n<5:
    print(n)
    n=n+1
```

On obtient :

```
# Dans la console PYTHON
1
2
3
4
```

Remarque

En supposant que l'on peut anticiper le nombre d'instruction qui seront lancées. Remarquez l'intérêt (c'est trés long) de la boucle *while* si on **développe** le code :

```
n=1

print(n)
n=n+1
print(n)
n=n+1
print(n)
n=n+1
print(n)
n=n+1#On s'arrete parce que n>= 5
```

Exercice 33

Ecrire le développement de la boucle suivante. Que vaut n à la fin de ce programme ?

```
n=5
while n>1:
    if n%2==0:
        n=n//2
    else:
        n=3*n+1
```

Exercice 34

Écrire la deuxième partie de ce code avec une boucle **while**. Que vaut s à la fin de ce code ?

```
#Premiere partie
s=0
#Deuxieme partie
s=s+0
s=s+1
s=s+2
s=s+3
s=s+4
```

Exercice 35

Écrire une fonction **dp(x)** de paramètre x positif qui renvoie le plus petit entier n tel que $x \leq 2^n$.

— Par exemple le premier exposant de 2 qui dépasse $x = 10$ est $n = 4$ car $2^3 = 8 < x = 10 < 2^4 = 16$.

— Par exemple le premier exposant de 2 qui dépasse ou égal $x = 128$ est $n = 7$ car $x = 128 = 2^7$.

Donc on devra obtenir :

```
# Dans la console PYTHON
>>> dp(10)
4
>>> dp(128)
7
```

```
# Dans l'éditeur PYTHON
def dp(x):
    '''In : x réel positif
      Out : n entier tel que x<= 2^n'''
    n=0
    while ... :
        .....
    return n
```

Exercice 36
La population mondiale

En 2018 la population mondiale est estimée à 7 577 millions (environ 7,6 milliards) . Le taux annuel de la croissance démographique de la population mondiale est d'environ 1,2 %.

Aide

Un milliard se note : $1\,000\,000\,000 = 10^9$, et s'écrit sous Python : $10 * *9$.
Dix milliards se note : $10 \times 10^9 = 10^{10}$, et s'écrit sous Python : $10 * 10 * *9$ ou $10 * *10$.

1. Le programme suivant cherche à déterminer en quelle année, si cette évolution se poursuit, la population mondiale dépassera 10 milliards et quelle sera cette population . Compléter puis exécuter cet algorithme dans la console afin d'obtenir la réponse cherchée .

```
def recherche(seuil):
    '''IN : le seuil
      OUT : l'année et la population qu
      dépasse le seuil'''
    population=7 577 000 000
    annee=2018
    while .... :
            annee=...
            population=...
    return (annee,population)
```

Pseudo Code

Fonction recherche(seuil)
    population,annee ← 7577000000, 2018
    Tant que ..... Faire
        annee ← .....
        population ← .....
    Fin Tant que
    Renvoyer (annee , population)

2. On cherche un affichage différent. Compléter le programme ci-dessous et lancez-le pour déterminer quand la population mondiale dépassera les 20 milliards et quelle sera cette population.

```
(a,b)=recherche(...)
print("La population sera de ",....," milliards en  ",...)
```

3. Modifier l'algorithme afin de renvoyer une valeurs arrondie au centième de la population, exprimée en milliards.

round(b , n)

round(b , n) va renvoyer l'arrondie de b à 10^{-n} près.
Par exemple : round(2.2563 , 2) => 2.26

V.2.1 La fonction factorielle

On note $n!$ (se lit « factoriel n ») le nombre $1 \times 2 \times 3 \times \cdots \times n$, pour tout entier naturel $n > 0$. Par convention on définit :

$$\begin{cases} 0! = 1 \\ n! = 1 \times 2 \times 3 \times \cdots \times n \quad , n \in \mathbb{N}^* \end{cases}$$

Remarque historique

La **notation factorielle** est introduite par le mathématicien Christian KRAMP (1760-1826) en 1808 dans *Éléments d'arithmétique universelle* (1808).

Exercice 37

1. Calculer 1! , 2! , 3! , 4! et 5!.
2. Écrire une fonction **fact(n)** qui renvoie $n!$, avec $n \geq 0$.

```
# Dans l'éditeur PYTHON
def fact(n):
    '''In : indice n, entier naturel (int)
      Out : n!'''
    assert n>=0
    ...
    return ..
```

Avec le module math.

Aide

Factorielle : $n!$ = 1 × 2 × 3 × ... × n .
Ce produit ce nomme factoriel n et se note $n!$.
Avec le *module math*, il existe un fonction en python qui le calcul directement : *math.factorial(n)*.
Pour l'appeler, il faut charger le *module math* au début du programme (ligne 1),
la syntaxe est *import math*.

Chargez le module math en ligne 1 en écrivant *import math* puis vérifiez que *math.factorial(n)* donne bien le même résultat que votre fonction pour quelques valeurs de n.

Sixième partie

Quelques problèmes et exercices supplémentaires

1. Une autre fonction définie par morceaux

Aide

Indication : Utilisez le test : *condition 1* ***and*** *condition 2*

Exercice 38

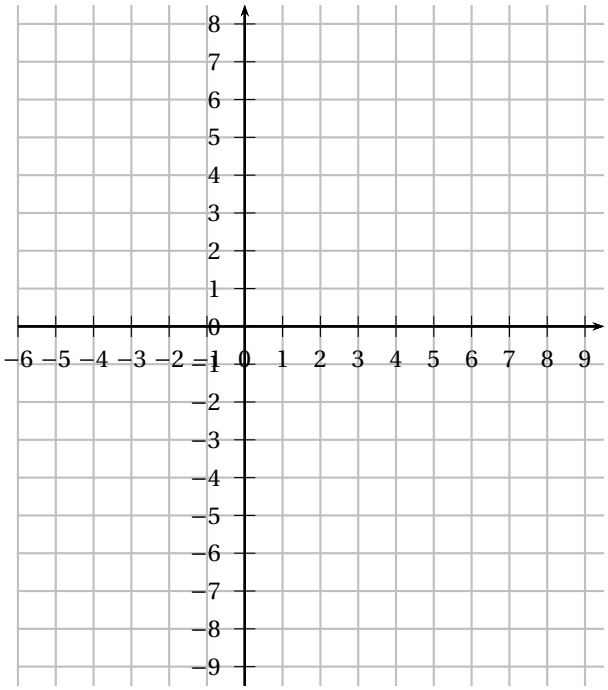
1. Écrire un algorithme utilisant une fonction qui permet de calculer l'image d'une valeur demandée, par la fonction g définie sur $\mathbb{R}$ par :

$$g : x \longmapsto g(x) = \begin{cases} 2x+1 & \text{si } x < 0 \\ x^2 - 1 & \text{si } 0 \leq x < 3 \\ 11 - x & \text{si } x \geq 3 \end{cases}$$

2. Compléter alors les tableaux de valeurs suivants et tracer $\mathscr{C}_g$ dans le repère ci-contre. Attention, bien identifier les fonctions de référence (fonctions affines, fonctions polynôme du second degré ...)

x	-5	-3	-2	-1	$-0,5$	-0.1
$g(x)$						

x	0	0.5	1	2	3	4	5	6
$g(x)$								



Facultatif : Vérifier votre graphique à l'aide du logiciel géogébra en utilisant l'instruction Si(condition,expression,sinon condition,expression)

Aide

Des exemples sur la page d'aide de Geogebra https://wiki.geogebra.org/fr/Fonctions.

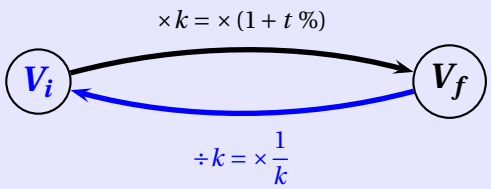
Exemple :

$$f(x) = Si(x > 5\ ,\ x+1\ ,\ 0 < x <= 5\ ,\ x \wedge 3\ ,\ x <= 0\ ,\ 1-5x)$$

définit la fonction $f : x \longmapsto f(x) = \begin{cases} x+1 & : \ x > 5 \\ x^3 & : \ 0 < x \leq 5 \\ 1-5x & : \ x \leq 0 \end{cases}$

2. Avec des pourcentages

Rappels pourcentages



$k = 1 + t\,\%$	$t\,\% = k - 1$
$k = \frac{V_f}{V_i}$	$t\,\% = \frac{V_f - V_i}{V_i}$

TVA et fonctions

Exercice 39

1. Chercher sur internet ce qu'est la T.V.A., le prix Hors Taxes (HT) et le prix Toutes Taxes Comprises (T.T.C) d'un article.
2. Calculer le prix TTC (avec une TVA à 20%) d'un article qui coûte 50 euros HT.
3. Écrire un algorithme utilisant une fonction qui calcule directement le prix TTC avec une T.V.A. à 20 %.
4. Vérifier votre fonction avec le calcul de la question 2.

```
def calc_TTC(prix_HT):
        '''IN : prix HT d'un article float
           OUT : Prix TTC avec une TVA à 20% float'''
        return ...
```

TVA, fonction et affichage

Exercice 40

Écrire un algorithme utilisant une fonction qui demande le prix d'un article TTC et qui calcule directement le prix HT avec une T.V.A. à 20 %. Vérifier votre fonction avec le calcul de la question 2 de l'exercice VI.

```
def calc_HT(prix_TTC):
        '''IN : Prix TTC avec une TVA à 20% float
           OUT : prix HT correspondant float'''
        return ...
```

Hausse et Baisse de x %

Exercice 41

1. Écrire un algorithme avec une fonction de paramètres p et t qui calcul le prix final, après une évolution de t% d'un prix initial de p euros.
2. Modifier le programme pour obtenir un arrondi au centième du résultat.

```
def hausse_baisse(p,t):
        '''IN : p prix initial, float
            t correspond à une évolution de t% float
          OUT : prix après évolution de t% float'''
        return ...
```

round(*b* , n)

round(b , n) va renvoyer l'arrondie de b à 10^{-n} près.
Par exemple : round(2.2563 , 2) => 2.26