Math93.com

# TD - Algorithmique
# Débuter en Python

## Première partie
# Variables et affichages

### Exercice 1. Éditeur et console sous Python

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

### Exercice 3. Variables

| Ligne | a | b |
|---|---|---|
| L1 | 2 | × |
| L2 | 2 | −5 |
| L3 | −3 | 7 |

| Ligne | a | b |
|---|---|---|
| L1 | 2 | × |
| L2 | 2 | −5 |
| L3 | −3 | −5 |
| L4 | −3 | 2 |

```
# Dans la console PYTHON
>>> Maintenant a = -3 et b = 7
```

```
# Dans la console PYTHON
>>> Maintenant a = -3 et b = 2
```

### Exercice 4. Suite d'affectations

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 | | $N \leftarrow N + 1$ |

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

| Ligne | U | N |
|---|---|---|
| L1 | 500 | × |
| L2 | 500 | 0 |
| L3 | 650 | 0 |
| L4 | 650 | 1 |

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

```
# Dans l'éditeur PYTHON
U,N=500,0
U,N=0.7*U+300,N+1
```

### Exercice 5. Une fonction ... d'Euler (Partie 1)

# Avec le module math

### Exercice 6. Périmètre et aire

```
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 2*pi*r

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

# Quelques problèmes de pourcentages

### Exercice 7. TVA et fonctions

1. La T.V.A. est la taxe sur la Valeur Ajoutée. C'est grossièrement la taxe que l'on paye pour tout objet que l'on achète. Le prix Hors Taxe est le prix sans tenir compte de la T.V.A. et le prix Toutes Taxes Comprises (T.T.C.) est le prix avec la T.V.A..
2. Pour un prix H.T. avec une T.V.A. de 20% le prix T.T.C. est $50+50\times 20\% = 60$.
3. On calcule donc le $prix\ T.T.C. = prix\ H.T. + 20\times(prix\ H.T.)/100$
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   prix_HT+20*prix_HT/100
```

### Exercice 8. TVA, fonction et affichage

$$prix\ T.T.C. = prix\ H.T. + 20\times(prix\ H.T.)/100 = prix\ H.T.(1+20/100) \Longleftrightarrow prix\ H.T. = \frac{prix\ T.T.C.}{(1+20/100)}$$

```
def calc_HT(prix_TTC):
        '''IN : Prix TTC avec une TVA à 20% float
           OUT : prix HT correspondant float'''
        return    prix_TTC/(1+20/100)
```

### Exercice 9. Hausse et Baisse de $x$ %

1. On a $Prix\ final = p + p\times t\% = p + p\times\frac{t}{100} = p\left(1+\frac{t}{100}\right)$

```
def f(p,t):
        '''IN : p prix initial, float
            t correspond à une évolution de t% float
          OUT : prix après evolution de t% float'''
        return p*(1+t/100)
```

2. Modifier le programme pour obtenir un arrondi au centième du résultat.

```
def f(p,t):
        '''IN : p prix initial, float
            t correspond à une évolution de t% float
          OUT : prix après evolution de t% float'''
        return round(p*(1+t/100),2)
```

# Compléments

### Exercice 10. (Optionnel) Affichage avec print()

### Exercice 11. (Optionnel) Input() ou float(input()

## Deuxième partie

# Instructions conditionnelles

### Exercice 12. Si ... alors... sinon

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* »

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

3. On cherche maintenant à résoudre une **équation de type $ax + b = c$**.
   (a) $2x + 3 = 4 \iff x = 1/2$.
   (b) Proposer une fonction nommée soleq2(a,b,c) renvoyant la solution de cette équation.

```
def soleq2(a,b,c):
    '''IN : a ,b et c flotants
      OUT : solution éventuelle de l'équation ax+b=c ou message si a = 0'''
    if a!=0: # si a est différent de 0
        return (c-b)/a
    else:
        if b==c
            return "Infinite de solution"
        else
            return "Pas de solution"
```

(c) Vérifier TOUJOURS votre programme. Un programme non testé est un programme qui a 99,9% de ne pas marcher.

## Exercice 13. Si ... alors ... sinon ...

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.
- Afficher le résultat.

1. $8 \to 4$ et $11 \to 34$ .
2. Écrire un algorithme correspondant à ce programme de calcul. On pourra compléter le programme ci-dessous :

```
def g(n):
    '''IN : n un int
      OUT : un int'''
    if n%2 ==0 :
        return n/2
    else:
        return n*3+1
```

3. Lorsqu'on entre un decimal, on obtient un résultat alors que l'on ne devrait pas avoir de résultat. Le programme fait un calcul mais qui n'a pas vraiment de sens.
4. On résout le problème avec le code suivant :

```
def g(n):
    '''IN : n un int
      OUT : un int'''
    if n = = int(n):
        return None
    if n%2 ==0 :
        return n/2
    else:
        return n*3+1
```

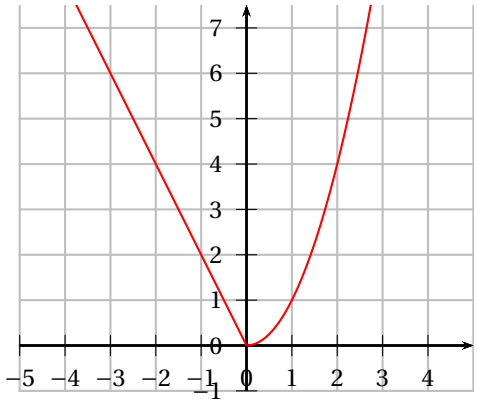
## Exercice 14. Une fonction définie par morceaux et Géogébra

1. On considère l'algorithme suivant écrit en pseudo code :

```
1 Fonction f(x)
2     si x < 0 alors
3         Retourner −2x;
4     sinon
5         Retourner x²;
6     fin
7 fin
```

```
def f(x):
    '''IN : x un float
      OUT : f(x) un float'''
    if x <0 :
        return -2*x
    else:
        return x*x
```

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




| $x$ | $-3$ | $-1$ | $0$ | $0,5$ | $1$ | $1,5$ | $2$ | $2,5$ |
|---|---|---|---|---|---|---|---|---|
| $f(x)$ | $6$ | $2$ | $0$ | $0,25$ | $1$ | $2,25$ | $4$ | $6,25$ |

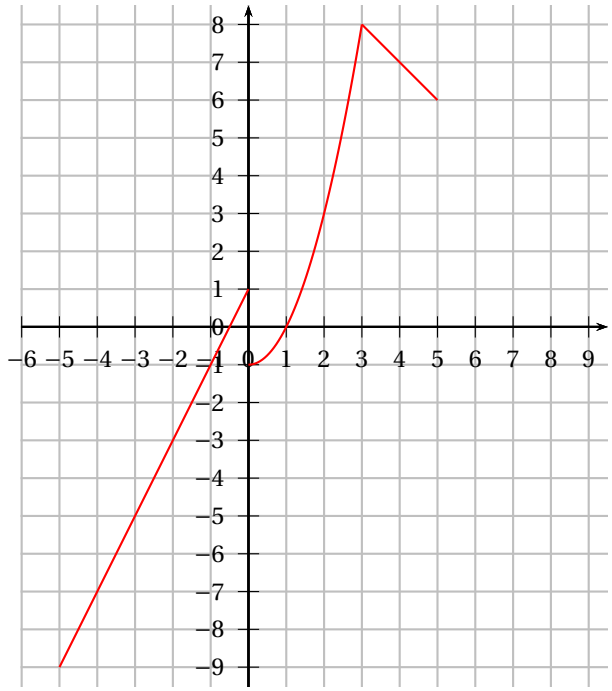
## Exercice 15. Une autre fonction définie par morceaux

1. Écrire un algorithme utilisant une fonction comme dans l'exercice 13 qui permet de calculer l'image d'une valeur demandée, par la fonction $f$ définie sur $\mathbb{R}$ par :

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

```
def g(x):
    '''IN : x un float
      OUT : g(x) un float'''
    if x <0 :
        return 2*x+1
    else:
        if x <= 3:
            return x*x-1
        else:
            return 11-x
```

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 ...)




| $x$ | $-5$ | $-3$ | $-2$ | $-1$ | $-0,5$ | $-0.1$ | $0$ |
|---|---|---|---|---|---|---|---|
| $g(x)$ | $-9$ | $-5$ | $-3$ | $-1$ | $0$ | $0,8$ | $-1$ |

| $x$ | $0.5$ | $1$ | $2$ | $3$ | $4$ | $5$ | $6$ |
|---|---|---|---|---|---|---|---|
| $g(x)$ | $-0,75$ | $0$ | $3$ | $8$ | $7$ | $6$ | $5$ |

## Exercice 16. Des photocopies

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

1. 15 photocopies coûtent $15 \times 10 = 150$ centimes $= 1,5$ Euros. 30 photocopies coûtent $20 \times 10 + 10 \times 8 = 200 + 80 = 280$ centimes $= 2,8$ Euros.

2.
```
def f(n):
    '''IN : n un int, le nombre de photocopie
      OUT : f(n) un int prix en centimes'''
    if n <=20 :
        return 10*n
    else:
        return 200+8*(n-20)
```

3. Écrire l'expression de cette fonction en fonction des valeurs de $x$.

$$f : n \longmapsto f(n) = \begin{cases} 10 \times n & \text{si } n \leq 20 \\ 200 + 8 \times (n-20) & \text{si } n > 20 \end{cases}$$

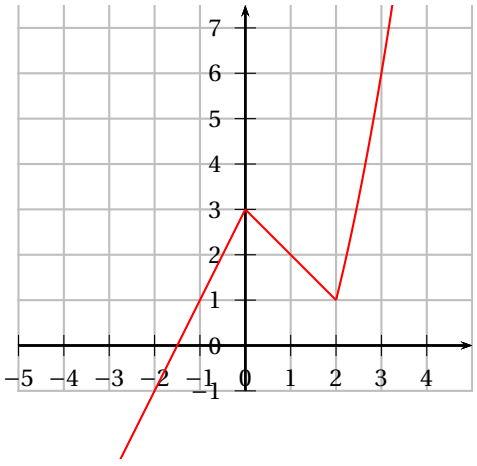
## Exercice 17. if test1 : instructions1 elif test2 : instructions2

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 $x$.

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

```
def h(x):
    '''IN : ...
      OUT : ...'''
    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
```

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.*



Troisième partie

# Structures itératives

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

### Exercice 18. Boucle dans des listes

1. On considère le programme suivant. Exécutez-le en écrivant simplement ex1(L1) 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 ex1(liste):
     for i in liste :
             print (i)
```

```
# Dans la console PYTHON
>>> ex1(L5)
```

```
# Dans l'éditeur
# On définit une liste L5
L5 = ['M' , 'o' , 'r' , 'n' , 'i' , 'n' , 'g']
def ex1(liste):
     for i in liste :
             print (i,end="")
```

2. On considère le programme suivant. Exécutez-le en écrivant simplement ex2() dans la console. Modifiez-le pour qu'il affiche les entiers de 0 à 15 .

```
def ex2():
     for n in range(10) :
             print (n)
```

```
def ex2():
     for n in range(16) :
             print (n)
```

3. On considère le programme suivant. Exécutez-le en écrivant simplement ex3() dans la console. Modifiez-le pour qu'il affiche les entiers de 5 à 10 .

```
def ex3():
    for n in range(7,13) :
        print (n)
```

```
def ex3():
    for n in range(5,11) :
        print (n)
```

4. Modifiez le programme suivant pour qu'il affiche les entiers impairs de 5 à 17 puis créez une fonction ex5(A,B) qui renvoie la liste des entiers de A à B de deux en deux.

```
def ex4():
    Liste = [ i for i in range( 10 , 20 , 2 ) ]
    return Liste
```

```
def ex4():
    Liste = [ i for i in range( 5 , 18 , 2 ) ]
    return Liste
```

```
def ex5(A,B):
    Liste = [ i for i in range(A, B+1 , 2 ) ]
    return Liste
```

## Exercice 19. Somme des entiers

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

$$S(10) = 0+1+\cdots+10 = 55\cdots \quad \text{et} \quad S(15) = 0+1+\cdots+15 = 120$$

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 (n+1):
        s= s+i
    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**

**$n = 10$**

| $i$ | X | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $s$ | 0 | 0 | 1 | 3 | 6 | 10 | 15 | 21 | 28 | 36 | 45 | 55 |

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

$$I(10) = 1+3+5+7+9 = 25$$

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

```
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 (1,n+1,2):
        s=s+i
        print(s)
    return s
```

## Exercice 20. Somme de carrés

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.

$$C(5) = 0^2 + 1^2 + 2^2 + \cdots + 5^2 = 55 \quad \text{et} \quad C(10) = 0^2 + 1^2 + 2^2 + \cdots + 10^2 = 385$$

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 $n$.

```
def sommecarres(n):
    '''IN : entier n >=0
      OUT : somme des carrés des entiers de 0 à n'''
    s = 0 # on initialise s à 0
        s = 0 # on initialise s à 0
    for i in range (n+1):
        s=s+i*i
        print(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**

$n = 10$ .

| $i$ | X | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $s$ | 0 | 0 | 1 | 5 | 14 | 30 | 55 | 91 | 140 | 204 | 285 | 385 |

## Exercice 21. Somme des inverses des carrés

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

$$I(4) = \frac{1}{1^2} + \frac{1}{2^2} + \frac{1}{3^2} + \frac{1}{4^2} \approx 1,423611$$

2. Écrire un programme qui calcule la somme des inverses des carrés de 1 à $n$. Tester votre programme avec le calcul précédent.

```
def inv(n):
    '''IN : entier n >=0
      OUT : somme des inverses des carrés des entiers de 1 à n'''
    s = 0 # on initialise s à 0
    for i in range (1,n+1):
        s=s+1/(i*i)
        print(s)
    return s
```

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.*

**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 :

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

## Exercice 22. Moyenne d'une liste de valeurs

### Une liste : L

Une liste est une suite d'éléments numérotés de même type dont le premier indice est 0. En Python, une liste s'écrit entre crochets [... , ..., ..., ...] avec les éléments séparés par des virgules.

- Le premier élément de la liste est L[0], le 2e est L[1], ...
- Une liste peut être écrite de manière explicite : L = ["Lundi", "Mardi","Mercredi"]
- Sa longueur est donnée par $len(L)$.
- Si les éléments de la liste sont comparables, le max. est donné par *max(L)*, le min. par *min(L)*
- L=[] permet de définir une liste vide.
- Si L est une liste, l'instruction *L.append(x)* va ajouter l'élémentt [x] à la liste L.

1. Définissez deux listes de nombres L1 et L2 ainsi :

```
# Dans l'éditeur PYTHON
L1 =  [-1,3,5,7,10]  # Liste 1 pour les exemples
L2 =  [5,12,15,7,10,19]  # Liste 2
```

2. Calculer la moyenne de ces deux séries de valeurs.

```
# Dans l'éditeur PYTHON
# moyenne L1 = 4,8
# moyenne L2 = 11,3333...
```

3. On veut écrire une fonction qui renvoie la la moyenne des termes d'une liste. Complétez-la et testez-la sur les listes L1 et L2 .

```
# Dans l'éditeur PYTHON
def moyenne(L):
    '''IN : L une liste de nombres flottants (décimaux)
      OUT : la moyenne des nombres de la liste'''
    s = 0 # on initialise s à 0
    for i in L:
        s=s+i
    return s/len(L)
```

4. Modifiez-la pour que la fonction renvoie aussi la longueur de L, la valeur maximale et la minimale de la liste.

### max(L) et min(L) :

max(L) et min(L) renvoie les valeurs maximale et minimale des éléments de la liste L.

```
# Dans l'éditeur PYTHON
def moyenneMaxMin1(L):
    '''IN : L une liste de nombres flottants (décimaux)
      OUT : la moyenne des nombres de la liste, la valeur minimale
      et la valeur maximale de la liste'''
    s = 0 # on initialise s à 0
    for i in L:
        s=s+i
    return s/len(L),min(L),max(L)
```

### Question Bonus

(a) Écrire une fonction **maxi(liste)** qui donne le maximum de la liste, sans utiliser la fonction max.

(b) Écrire une fonction **mini(liste)** qui donne le minimum de la liste, sans utiliser la fonction min.

```
# Dans l'éditeur PYTHON
def MiniMaxi(L):
    '''IN : L une liste de nombres flottants (décimaux)
      OUT : la valeur minimale
      et la valeur maximale de la liste'''
    mini=L[0]#remarquer qu'il ne faut pas utiliser
    maxi=L[0]#les mots max et min qui sont reservés
    for i in L:
        if i<mini:
            mini=i
        if i>maxi:
            maxi=i
    return mini,maxi
```

# 2. Boucle dont on ne connait 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).

## Exercice 23. 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 f(seuil):
    '''IN : le seuil
      OUT : l'année et la population qu
      dépasse le seuil'''
    population=7 577 000 000
    annee=2018
    while population<seuil:
     annee=anne+1
     population=population +1.2*populat
    return (annee,population)
```

**Pseudo Code**

Fonction f(seuil)
  population,annee ← 7577000000, 2018
  Tant que population < seuil Faire
    annee ←annee+1
    population ← population+$\frac{1,2}{100}$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)=f(20*10**9)
b=b/10**9 # pour avoir le résultat en milliards
print("La population sera de ",b," milliards en  ",a)
```

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*

```
def f(seuil):
    '''IN : le seuil
      OUT : l'année et la population qui
      dépasse le seuil'''
    population=7 577 000 000
    annee=2018
    while population<seuil:
            annee=anne+1
            population=population +1.2*population/100
    return (annee,round(population,2))
```

# 3. Applications

## Exercice 24. 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).

1. Calculer $1! = 1$ , $2! = 2$ , $3! = 6$ , $4! = 24$ et $5! = 120$.
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
    i=1
    P=1
    while i<n:
        i=i+1
        P=P*i
    return P
```

3. Avec le module math.

**Aide**

**Factorielle : $n! = 1 \times 2 \times 3 \times \ldots \times n$ .**
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 $n$.

## Quatrième partie

# Les Listes

### Exercice 25. TD sur les listes

Faire le TD : - Recherche des occurrences sur des valeurs de type quelconque.

### Un premier bilan sur les listes

| | | |
|---|---|---|
| Créer une liste vide | $L = []$ ou $L = list()$ | |
| Créér une valeur | Pour récupérer une valeur dans une liste python : nom_liste[index]<br>*Attention le 1er élément de la liste L est L[0], le 2e est L[1] ...* | L[0]=25<br>*le 1er item est 25* |
| Remplacer item | On peut ajouter ou remplacer un élément dans une liste :<br>$liste[index] = valeur$ | $L[1] = 76$<br>$L[2] = 20$<br>Donc $L = [25, 76, 20]$ |
| Ajouter item :<br>— .append() | On peut ajouter un élément dans une liste, à la fin :<br>$liste.append(valeur)$ | L.append(15)<br>*on ajoute 15 à la fin de la liste L*<br>$L = [25, 76, 20, 15]$ |
| Supprimer item :<br>— del<br>— .remove()<br>— .pop() | On peut supprimer un valeur d'une liste<br>— Avec l'index : **del ou pop**<br>— La fonction **del liste[i]** va supprimer l'item à l'index i spécifié (pas besoin de connaître l'item).<br>— La méthode **liste.pop(i)** va supprimer l'item à l'index i spécifié et renvoyer sa valeur.<br>— Avec sa valeur : la méthode **.remove()**<br>liste.remove(x) : supprime de la liste le premier élément dont la valeur est égale à x | $L = [25, 76, 20, 15]$<br>> del L[0]<br>$L = [76, 20, 15]$<br>> L.pop(1)<br>$L = [76, 15]$<br>> L.remove(15)<br>$L = [76]$ |
| Parcourir | On peut parcourir une liste :<br>— par les index : **for index in range(len(liste))**<br>— directement : **for élément in liste** | |
| index() | Trouver l'index d'une valeur $v1$<br>La méthode : liste.index($v1$) | > L2=['a' , 'c' , 'f']<br>> L2.index('f')<br>2 |
| sort() : trier | liste.sort() va trier la liste par ordre croissant | L.sort() |
| count() | liste.count(x) : renvoie le nombre d'apparitions de x dans la liste | |
| liste[$i : j$] | Liste[$i : j$] :<br>renvoie une sous liste composée des éléments Liste[$i$] à Liste[$j - 1$] | $L[2 : 4] = \left[L[2], L[3]\right]$ |
| insert() | Insère un élément à la position indiquée.<br>Le premier argument est la position de l'élément courant avant lequel l'insertion doit s'effectuer.<br><br>Donc **liste.insert(0, x)** insère l'élément x en tête de la liste et **liste.insert(len(a), x)** est équivalent à **liste.append(x)**. | |
| Copier une liste<br>$L2 = list(L1)$ | Attention, l'instruction $L1 = L2$ ne peut pas être utilisée car dans ce cas les deux listes seront modifiée simultanément. | L2 = list(L1)<br>ou L2 = L1[ :] |

## Cinquième partie

# Les Dictionnaires (facultatifs)

Cette section peut être omise ne première lecture. Les dictionnaires ne sont au programme que de la fameuse spécialité NSI de première.

**$\{clé_1 : val_1, clé_2 : val_2, ..., clé_n : val_n\}$ :**

$\{clé_1 : val_1, clé_2 : val_2, ..., clé_n : val_n\}$ est une liste dont la clé n'est pas l'indice de position mais qui une valeur qui peut être de n'importe quel type.
- Un dictionnaire en python est donc une sorte de liste mais au lieu d'utiliser des index , on utilise des clés alphanumériques.
- Dans un dictionnaire, les informations ne sont pas stockées dans un ordre précis. Pour accéder aux valeurs, on utilise les clés.

**Exemples** :

```
# Dans la console PYTHON
# Pour initialiser un dictionnaire
>>>dico = {} # ou dico=dict{}

# Pour ajouter des valeurs à un dictionnaire il faut indiquer
# une clé ainsi qu'une valeur :
>>> dico["nom"] = "Turing" # clé1 = "nom" et valeur associée "Turing"
>>> dico["prenom"] = "Alan"
>>> dico["annee_naissance"] = 1912

>>> dico
{'nom': 'Turing', 'prenom': 'Alan', 'annee_naissance': 1912}

# On peut vérifier que la variable dico est bien de type dictionnaire
>>>type(dico)
<class 'dict'>
```

## 25.1 .get() : Comment récupérer une valeur dans un dictionnaire python ? : .

La méthode get vous permet de récupérer une valeur dans un dictionnaire et si la clé est introuvable, vous pouvez donner une valeur à retourner par défaut :

```
>>> dico
{'nom': 'Turing', 'prenom': 'Alan', 'annee_naissance': 1912}

>>> dico
{'nom': 'Turing', 'prenom': 'Alan', 'annee_naissance': 1912}
>>>dico.get("nom")
'Turing'
>>>dico.get("annee_naissance")
1912
>>>dico.get("poids") # si la clé n'est pas trouvée, rien ne s'affiche
```

## 25.2 .key() : Comment récupérer les clés d'un dictionnaire python par une boucle ?

Pour récupérer les clés on utilise la méthode keys .

```
fiche = {"nom":"Wayne", "prenom":"Bruce"}
for cle in fiche.keys():
    print cle
```

## 25.3 .value() : Comment récupérer les valeurs d'un dictionnaire python par une boucle ?

Pour cela on utilise la méthode values .

```
fiche = {"nom":"Wayne", "prenom":"Bruce"}
for valeur in fiche.values():
    print valeur
```

## 25.4 .item() : Comment récupérer les clés et les valeurs d'un dictionnaire python par une boucle ?

Pour récupérer les clés et les valeurs en même temps, on utilise la méthode items qui retourne un tuple .

```
fiche = {"nom":"Wayne","prenom":"Bruce"}
for cle,valeur in fiche.items():
    print cle, valeur
```