Math93.com

TD - Boucle for et while
Débuter en Python - Partie 4

Table des matières

Quatrième 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.

IV.1. Boucle dont on connaît le nombre d'itérations (FOR)

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)
```

IV.1.1 Des exemples de boucles

Exercice 1

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

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

Exercice 2

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 3

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 :

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

IV.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 4

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 que 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
```

IV.1.3 Un peu de poésie

Exercice 5

1. Ecrire le programme suivant et tester le

```
1
2 A=["parrain","peureux","roi","patraque","punk"]
3 for i in range(5):
4     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",...)
```

IV.1.4 Déterminer les puissances d'un nombre entier

Exercice 6

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

$n = 6$

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**.
   .....

IV.1.5 Somme des entiers

Exercice 7

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:int)->int:
    '''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 $n > 0$.

IV.1.6 Somme de carrés

Exercice 8

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

IV.1.7 Somme des inverses des carrés

Exercice 9

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:int)->float:
    '''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 :

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

IV.2. Boucle dont on ne connaît pas le nombre d'itérations (WHILE)

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 10

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 11

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 12

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:float)->int:
    '''In : x réel positif
      Out : n entier tel que x<= 2^n'''
    n=0
    while ... :
        .....
    return n
```

Exercice 13
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:float)->(int,float):
    '''IN : le seuil
      OUT : l'année et la population qu
      dépasse le seuil'''
    population=7577000000
    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

IV.2.1 La fonction factorielle

On note $n!$ (se lit « factoriel n ») le nombre $1\times2\times3\times\cdots\times n$, pour tout entier naturel $n>0$. Par convention on définit :

$$\begin{cases}0!=1\\ n!=1\times2\times3\times\cdots\times n \quad ,n\in\mathbb{N}^*\end{cases}$$

Remarque historique

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

Exercice 14

1. Calculer 1! , 2! , 3! , 4! et 5!.
2. Écrire une fonction **fact(n)** qui renvoie $n!$, avec $n\geq0$.

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

Avec le module math.

Aide

Factorielle : $n!=1\times2\times3\times\ldots\times n$.
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.

Cinquième partie

Quelques problèmes et exercices supplémentaires

1. Une autre fonction définie par morceaux

Aide

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

Exercice 15

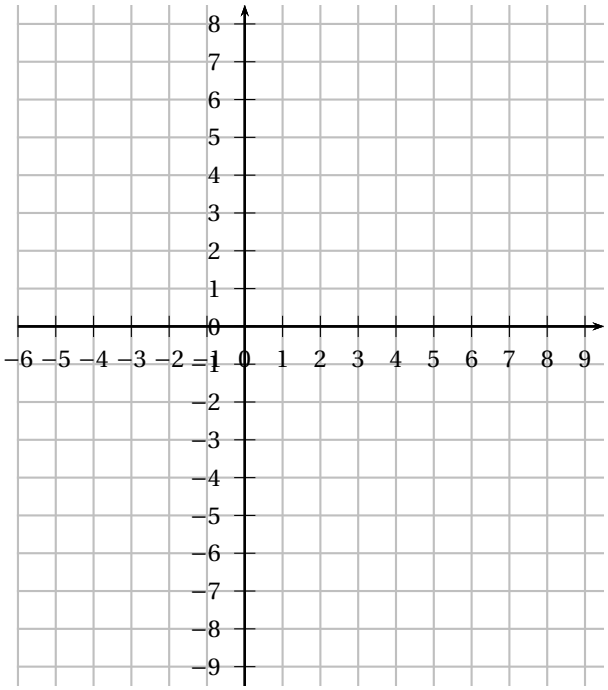
1. Écrire un algorithme utilisant une fonction qui permet de calculer l'image d'une valeur demandée, par la fonction g définie sur $\mathbb{R}$ par :

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

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.

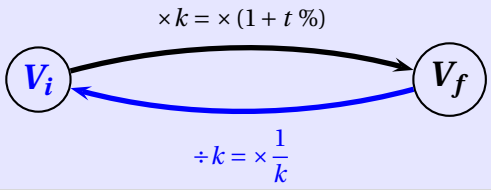
Exemple :

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

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 16

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 : float, prix HT d'un article
           OUT :float, Prix TTC avec une TVA à 20% '''
        return ...
```

TVA, fonction et affichage

Exercice 17

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 V.

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

Hausse et Baisse de x %

Exercice 18

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:float,t:float)->float:
        '''IN : p prix initial
           t correspond à une évolution de t%
         OUT : prix après évolution de t% '''
        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