Boucles et comparaisons

En programmation, on est souvent amené à répéter plusieurs fois une instruction.
les boucles vont nous aider à réaliser cette tâche de manière compacte.

Si vous souhaitiez afficher les éléments d'une liste les uns après les autres….il faudra utiliser la fonction print autant de fois que nous avons d'éléments!!!!!

```
animaux = ['girafe','tigre','singe','souris']

print (animaux[0])
print (animaux[1])
print (animaux[2])
print (animaux[3])
```

Et pour une liste à 120 éléments ????

Solution


```
>>> animaux = ['girafe','tigre','singe','souris']
>>> for animal in animaux:
... print(animal)
...
girafe
tigre
singe
souris
```

Imprimer chaque valeur de la variable animal prise dans la liste animaux

La boucle for

Syntaxe générale

```
for x in ensemble:
    instruction 1
    ...
    instruction n
```

Bloc d'instructions exécuté pour chaque valeur de x

Le bloc d'instruction est reconnu par une indentation produite automatiquement lors de l'écriture de la boucle (tabulation).

Cours réalisé par le professeur Khalid +212 622176552

Utilisation avec range()

La boucle la plus répandue est celle qui parcourt des indices entiers compris entre 0 et n-1. On utilise pour cela la boucle for et la fonction range

```
Somme = 0                    # initialisation de la variable somme à 0
N = 10
for n in range(0, N):        # n va parcourir la gamme (0,N-1)
   somme += n                # incrémenter somme
```

somme += n est équivalente à somme= somme +n

Beaucoup plus simple encore

```
>>> for i in range(4):
... print(i)
...
0123
```

la fonction range() peut être utilisée telle quelle dans une boucle. Il n'est pas nécessaire de taper for i in list(range(4)):, même si cela fonctionnerait également.

range() est une fonction qui a été spécialement conçue que l'on peut itérer directement dessus. Pour Python, il s'agit d'un nouveau type, par exemple dans x = range(3) la variable x est de type *range (tout comme on avait les types int, float, str ou list) à utiliser spécialement avec les boucles.*

L'instruction list(range(4)) se contente de transformer un objet de type range en un objet de type list.

C'est même contre-productif. En effet, range() se contente de stocker l'entier actuel, le pas pour passer à l'entier suivant, et le dernier entier à parcourir, ce qui revient à stocker seulement 3 entiers et ce quel que soit la longueur de la séquence, même avec un range(1000000).

Si on utilisait list(range(1000000)), Python construirait d'abord une liste de 1 million d'éléments dans la mémoire puis itérerait dessus, d'où une énorme perte de temps !

Itération sur l'indice d'une liste

```
>>> x=0                              #initialisation de x
>>> li=[1,2,-1,3]                    #definition de la liste
>>> for i in range(0,len(li)):       #i parcours la gamme (0,4)
        x=x+li[i]                    #cumule dans x
>>> print (x)
5
>>>
```

L'itération sur une liste peut se faire soit directement avec for i in liste ou bien en utilisant l'indice de position des éléments de la liste

La comparaison

```
>>> x = 5
>>> x == 5
True
>>> x > 10
False
```

Syntaxe Python	Signification
==	égal à
!=	différent de
>	supérieur à
>=	supérieur ou égal à
<	inférieur à
<=	inférieur ou égal à

Python renvoie la valeur True si la comparaison est vraie et False si elle est fausse. True et False sont des booléens.

important

ne pas confondre l'**opérateur d'affectation = qui affecte une valeur à une** variable et l'**opérateur de comparaison == qui compare les valeurs de deux variables.**

La boucle while

Syntaxe générale

```
while cond :
    instruction 1
    ...
    instruction n
```

Exemple

```
>>> i = 1
>>> while i <= 4:
... print(i)
... i = i + 1
...
1234
```

Une boucle while nécessite généralement trois éléments pour fonctionner correctement :

- l'initialisation de la variable de test avant la boucle ;
- le test de la variable associé à l'instruction while ;
- la mise à jour de la variable de test dans le corps de la boucle.

En cas d'erreur qui mène à une boucle infinie on utilise la **Ctrl-C**

Instructions break et continue

=> L'instruction break stoppe la boucle.

=> L'instruction continue saute à l'itération suivante.

```
for i in range(5):
... ….if i > 2:
... ….Break
….print(i)
012
```

```
>>> for i in range(5):
... if i == 2:
... continue
... print(i)
...
0134
```

Questionnaire

1- Donnez la syntaxe générale de la boucle **for**

2- Donnez la signification de la fonction range(0,n)

3- Donnez la syntaxe générale de la boucle **while**

4- Donnez la signification des instructions break et continue

5- Expliquez le role du module random et la fonction randint

Travaux pratiques *(solution page suivante)*

1. Proposez un script qui permet de faire la somme des n termes et de renvoyer le résultat à l'aide de
    a. la boucle **for** et de la fonction **range**
    b. la boucle **while**

2. Proposez un script qui permet d'imprimer la table de multiplication d'un nombre choisi par l'utilisateur à l'aide de la fonction input() et la boucle for
    a. **verticalement**
    b. **dans une liste**

3. Tirez au hasard 5 nombres entiers entre 0 et 100 et placer les dans une liste à l'aide du module **random** et la fonction **randint**, afficher le minimum et le maximum de la liste en utilisant la boucle for et les fonctions min et max


```
n=input('rentrer un nombre entier')
n=int(n)
s=0
for i in range(0,n+1):
    s=s+i
print ('la somme est ' + str(s))
```

```
n=input('rentrer un nombre entier')
n=int(n)
s=0
i=0
while i < n+1:
    s=s+i
    i=i+1
print (str(s))
```

```
n=input('rentrer un nombre entier' +"\n")
n=int(n)
for i in range(1,11):
    print (n*i)
```