Langage Python

Les bases du langage 4 : Les boucles

Formation POEC Cybersécurité

Théo Hubert

Syntaxe

- *Boucles*

En programmation, on est souvent amené à **répéter plusieurs fois une ou plusieurs instruction(s)**. Incontournables à tout langage de programmation, **les boucles vont nous aider à réaliser cette tâche de manière compacte et efficace**.

Il existe 2 types de **répétitions** :

- Les répétitions dont le **nombre est connu à l'avance**.
- Les répétitions dont le **nombre n'est pas connu à l'avance**, car dépendantes d'une certaine condition.

=> Quand on sait combien de fois doit avoir lieu la répétition, on utilise généralement une boucle **for**.

=> Quand on ne connaît pas à l'avance le nombre de répétitions, on utilise une boucle **while**.

Syntaxe

- *Boucles*

Boucles for *(boucle borné)* :

```
marques = ["Renault", "Peugeot", "Tesla"]
for i in range(len(marques)):
  print(marques[i])
```

```
Renault
Peugeot
Tesla
```

```
marques = ["Renault", "Peugeot", "Tesla"]
for constructeur in marques:
  print(constructeur)
```

```
Renault
Peugeot
Tesla
```

- L'en-tête de la boucle for se présente sous la forme : **for compteur in sequence**
  Cette séquence peut-être un objet **range**, ou une **liste**, ou une **chaîne de caractères**, ou plus généralement tout objet itérable.

- Le corps de la boucle :
  Il est **indenté** par rapport à l'en-tête.

- Quand le corps de la boucle a été effectué pour tous les i de l'objet range, la boucle est terminée et le programme continue avec les instructions qui suivent la boucle for, c'est à dire celles qui ne sont plus indentées par rapport à l'en-tête.

Syntaxe

- *Boucles*

Fonctions range() :

La fonction range( ) permet de construire des suites (arithmétiques) d'entiers :

- **range(fin)** => **range(4)** similaire à **[0, 1, 2, 3]**
- **range(début, fin)** => **range(2, 5)** similaire à **[2, 3, 4]**
- **range(début, fin, pas)** => **range(1, 10, 2)** similaire à **[1, 3, 5, 7, 9]**

```
test = range(4)
print(type(test), " ", test)
print(test[0])
print(test[1])
print(test[2])
print(test[3])
```

```
<class 'range'>   range(0, 4)
0
1
2
3
```

```
marques = ["Renault", "Peugeot", "Tesla"]
print(len(marques))
print(range(len(marques)))
```

```
3
range(0, 3)
```

Syntaxe

- *Boucles*

Boucles for *(boucle borné)* :

```
for i in range(4):
  print("i a pour valeur :", i)
```

```
i a pour valeur : 0
i a pour valeur : 1
i a pour valeur : 2
i a pour valeur : 3
```

Ici, on répète 4 fois une instruction à l'aide d'une boucle **for** et une séquence **range(4)**.

```
animaux = [" girafe ", " tigre ", " singe ", " souris "]
for i in range(len(animaux)):
  print("L'animaux à la position", i, " =>", animaux[i])
```

```
L'animaux à la position 0  =>  girafe
L'animaux à la position 1  =>  tigre
L'animaux à la position 2  =>  singe
L'animaux à la position 3  =>  souris
```

Ici, on affiche l'ensemble des éléments d'une listes "animaux" à l'aide d'une boucle **for** et une séquence **range(len(animaux))** qui une liste d'entier itérable correspondant à la taille de la liste.

Syntaxe

- *Boucles*

Boucles for *(boucle borné)* :

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

```
girafe
tigre
singe
souris
```

Ici, on affiche l'ensemble des éléments d'une listes "animaux" à l'aide d'une boucle **for** et une séquence **animaux** qui est une **liste d'animaux itérable**.

```
mot = "test"
for lettre in mot:
  print(lettre)
```

```
t
e
s
t
```

Ici, on affiche l'ensemble des lettres d'un str "mot" à l'aide d'une boucle **for** et une séquence mot qui est une **liste de caractères itérables**.

Syntaxe

- *Boucles*

Boucles while *(boucle non borné)* :

Si on ne connaît pas à l'avance le nombre de répétitions, on choisit une boucle **while**.

```
x = 1
while x < 20:
    print("x a pour valeur", x)
    x = x * 2
print("Fin")
```

```
x a pour valeur 1
x a pour valeur 2
x a pour valeur 4
x a pour valeur 8
x a pour valeur 16
Fin
```

Une boucle while nécessite généralement trois éléments pour fonctionner correctement :

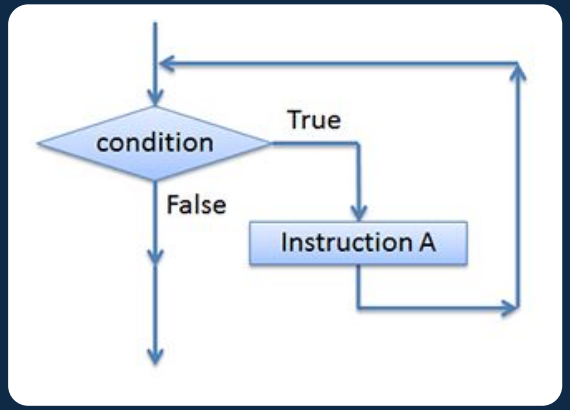
- **Initialisation** de la **variable d'itération** avant la boucle.
- **Test de la variable d'itération** associée à l'instruction while.
- **Mise à jour de la variable d'itération** dans le corps de la boucle.

Syntaxe

- *Boucles*

Boucles while *(boucle borné)* :

- L'en-tête de la boucle while se présente sous la forme : **while** **condition**
  La boucle est effectuée **tant que la condition est vraie**.
- Le corps de la boucle :
  Il est **indenté** par rapport à l'en-tête.
- Quand la condition est fausse, la boucle n'est plus effectuée et le programme continue avec les instructions qui suivent la boucle while, c'est à dire celles qui ne sont plus indentées par rapport à l'en-tête. Dans ce cas précis, on sort de la boucle quand i vaut 26



Syntaxe

- *Boucles*

Boucles while *(boucle borné)* :

```
note = 0
while note < 20:
  note += 1

print("J'ai réussi")
print(note)
```

```
J'ai réussi
20
```

Ici, tant que la variable note est inférieur à 20 la note augmente de 1 (l'élève continue de travailler jusqu'à ce que sa note soit égal à 20).

```
k = 0
while k < 6 :
    k += 2
    print(k)
```

```
2
4
6
```

Ici, tant que la variable k est inférieur à 6 on augmente cette variable de 2 puis on l'affiche.

Syntaxe

Travaux Pratique

Références

https://courspython.com/boucles.html

https://python.iutsf.org/lecon-4-les-boucles/

https://www.bdrp.ch/system/files/docs/2021-02-01/1erensi_python_cours_niv2.pdf

Fin

La suite :
Partie 5 : Les fonctions