Les_Bases_5

July 16, 2025

1 La boucle for

La boucle for permet de parcourir des structures de données ou d'itérer sur un objet itérable. Exemple de structures de données:

- listes ([ ])
- tuples (( ))
- dictionnaires ({key: valeur })
- ensembles ({ })
- chaînes de caractères ("str")

[1]:
```
# La structure de la boucle for en Python

for i in range(10):
    print(f"Iteration {i + 1}: Hello, World!")
```

```
Iteration 1: Hello, World!
Iteration 2: Hello, World!
Iteration 3: Hello, World!
Iteration 4: Hello, World!
Iteration 5: Hello, World!
Iteration 6: Hello, World!
Iteration 7: Hello, World!
Iteration 8: Hello, World!
Iteration 9: Hello, World!
Iteration 10: Hello, World!
```

1.1 Exemple 1

[2]:
```
FF12_squad = ["Vann","Balthier", "Fran", "Ashe", "Basch", "Penelo"]
for combattant in FF12_squad :
    print(combattant)
# La boucle for va parcourir la liste FF12_squad et afficher chaque élément de␣
 ↪la liste.
```

```
Vann
Balthier
Fran
Ashe
```

```
Basch
Penelo
```

1.2 Exemple 2

[3]:
```
nombres = [0, 1, 2, 3, 4, 5]
for i in nombres:
    print(i)
```

```
0
1
2
3
4
5
```

1.3 Exemple 3

[4]:
```
for lettre in "Zoléni":
    print(lettre)
```

```
Z
o
l
é
n
i
```

1.4 Exemple 4

[5]:
```
for i in range(1, 11):
    print("PSG")
```

```
PSG
PSG
PSG
PSG
PSG
PSG
PSG
PSG
PSG
PSG
```

La boucle for sert à deux choses :

- Répéter une opération un certain nombre de fois
- Parcourir des structures de données

2 La boucle while

2.1 Principe de la boucle while

La boucle while s'exécutera tant que la condition est vraie

- Ecrivons l'exemple 4 d'une autre façon :

[ ]:
```
# La syntaxe de la boucle while est la suivante :
# while condition:
#     # bloc d'instructions
#     print("PSG")
# La boucle while va continuer à afficher "PSG" tant que la variable i est␣
 ↪inférieure à 10.
# La boucle while est souvent utilisée lorsque le nombre d'itérations n'est pas␣
 ↪connu à l'avance.

i = 0
while i < 10:
    print("PSG")
    i += 1
```

```
PSG
PSG
PSG
PSG
PSG
PSG
PSG
PSG
PSG
PSG
```

2.2 ATTENTION aux boucles infinies

Il faut cependant savoir si la boucle while est mal ulilisée, cela peut aboutir à une boucle infinie. Le script ne va jamais s'arrêter.

[7]:
```
# Mais ATTENTION, la boucle while peut entraîner une boucle infinie si la␣
 ↪condition n'est jamais fausse.
# Par exemple :

# i = 0
# while i < 10:  <---- Ce qui est toujours vrai. Donc boucle infinie
#     print("PSG")
```

Il est donc important de s'assurer que la condition de la boucle while finira par devenir fausse.

Comment limiter les risques de cette boucle infinie ?

[8]:
```
continuer = "y"
# La boucle while va continuer à afficher "PSG" tant que la variable continuer
↪est égale à "y".
while continuer == "y":
    print("PSG")
    continuer = input("Voulez-vous continuer ? (y/n) ")
```

```
PSG
PSG
PSG
PSG
```

Résumé - Une boucle while est exécutée tant qu'une contion est vraie - Attention aux boucles infinies

3 Modifier l'éxecution d'une boucle avec continue et break

L'instrucution continue permet de passer à l'itération suivante de la boucle sans exécuter le reste du bloc d'instructions.

[9]:
```
# Par exemple, si on veut afficher les nombres de 0 à 9 sauf le nombre 5 : b
for i in range(10):
    if i == 5:
        continue
    print(i)
```

```
0
1
2
3
4
6
7
8
9
```

[10]:
```
villes = ["Paris", "Lyon", "Marseille", "Toulouse", "Nice"]
villes.extend(["77100", "Torcy", "Chessy"])
for ville in villes:
    if ville == "Marseille": # <----- On ne veut pas afficher Marseille
        continue
    if ville.isdigit():
        print("C'est le code postal de la ville de Meaux")
        continue
    print(ville)
```

```
Paris
Lyon
Toulouse
```

```
Nice
C'est le code postal de la ville de Meaux
Torcy
Chessy
```

L'instruction break permet de sortir d'une boucle, quelle qu'elle soit (for ou while).

[11]:
```python
# L'instruction break permet de sortir d'une boucle, quelle qu'elle soit (for␣
 ↪ou while).

villes = ["Paris", "Lyon", "Meaux", "77100", "Toulouse", "Nice", "Toulon",␣
 ↪"Saint-Nazaire"]
for ville in villes:
    if ville.isdigit():
        print("C'est un code postal")
        break
    print(ville)
```

```
Paris
Lyon
Meaux
C'est un code postal
```

Résumé : - continue fait passer la boucle directement à la prochaine itération - break arrête l'éxécution de la boucle au complet

4 Les compréhensions de liste

[12]:
```python
liste = [-5,-4,-3,-2,-1,0,1,2,3,4,5]
# On va créer une liste qui va contenir tous les nombres négatifs de la liste␣
 ↪ci-dessus.
nb_negatifs = []
for i in liste:
    if i < 0:
        nb_negatifs.append(i)
        print("Nombre négatif trouvé : ", i)
print("Liste des nombres négatifs : ", nb_negatifs)
```

```
Nombre négatif trouvé :  -5
Nombre négatif trouvé :  -4
Nombre négatif trouvé :  -3
Nombre négatif trouvé :  -2
Nombre négatif trouvé :  -1
Liste des nombres négatifs :  [-5, -4, -3, -2, -1]
```

[13]:
```python
# La compréhension de liste est une façon concise de créer une liste en une␣
 ↪seule ligne de code.
```

```
# Elle permet de créer une nouvelle liste en appliquant une expression à chaque␣
 ↪élément d'une séquence ou d'un itérable.
# Par exemple, on peut créer une liste qui va contenir tous les nombres␣
 ↪négatifs de la liste ci-dessus
# en utilisant la compréhension de liste :

nb_negatifs.clear()
nb_negatifs = [i*2 for i in liste if i < 0]
print("Liste des nombres négatifs : ", nb_negatifs)
```

```
Liste des nombres négatifs :  [-10, -8, -6, -4, -2]
```

5 Exercices

5.1 Exercice n°1 : Remplacer des boucles par des compréhensions de liste

[14]:
```
# La compréhension de liste est souvent plus rapide et plus lisible que␣
 ↪d'utiliser une boucle for classique.
# On peut également utiliser la compréhension de liste pour créer une liste qui␣
 ↪va contenir tous les nombres pairs
# de la liste ci-dessus.

nombres = [1, 21, 5 , 44, 4, 9, 5, 83, 29, 31, 25, 38]
nombres_pairs  = []
for i in nombres:
    if i % 2 == 0:
        nombres_pairs.append(i)
print("Liste des nombres pairs : ", nombres_pairs)
```

```
Liste des nombres pairs :  [44, 4, 38]
```

[15]:
```
# On peut aussi utiliser la compréhension de liste pour créer une liste qui va␣
 ↪contenir tous les nombres pairs de la liste ci-dessus :
nombres_pairs.clear()
nombres_pairs = [i for i in nombres if i % 2 == 0]
print("Liste des nombres pairs : ", nombres_pairs)
```

```
Liste des nombres pairs :  [44, 4, 38]
```

[16]:
```
# On va créer une liste qui va contenir tous les nombres positifs de la liste␣
 ↪ci-dessus.
nombres = range(-10, 10)
nombres_positifs = []
for i in nombres:
    if i >= 0:
        nombres_positifs.append(i)
print("Liste des nombres positifs : ", nombres_positifs)
```

```
Liste des nombres positifs :  [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

[17]:
```
# On peut aussi utiliser la compréhension de liste pour créer une liste qui va␣
↪contenir tous les nombres positifs de la liste ci-dessus :
nombres_positifs.clear() # type: ignore
nombres_positifs = [i for i in nombres if i >= 0] # type: ignore
print("Liste des nombres positifs : ", nombres_positifs)
```

```
Liste des nombres positifs :  [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

[18]:
```
nombres = range(5)
nombres_doubles = []
for i in nombres:
    nombres_doubles.append(i*2)
print(nombres_doubles)
```

```
[0, 2, 4, 6, 8]
```

[19]:
```
# On peut aussi utiliser la compréhension de liste pour créer une liste qui va␣
↪contenir tous les nombres doubles de la liste ci-dessus :
nombres_doubles.clear() # type: ignore
nombres_doubles = [i*2 for i in nombres] # type: ignore
print(nombres_doubles)
```

```
[0, 2, 4, 6, 8]
```

[20]:
```
nombres = range(10)
nombres_inverses = []
for i in nombres:
    if i % 2 == 0:
        nombres_inverses.append(i)
    else:
        nombres_inverses.append(-i)
print(nombres_inverses)
```

```
[0, -1, 2, -3, 4, -5, 6, -7, 8, -9]
```

[21]:
```
# On peut aussi utiliser la compréhension de liste pour créer une liste qui va␣
↪contenir tous les nombres inversés de la liste ci-dessus :
nombres_inverses.clear() # type: ignore
nombres_inverses = [i if i % 2 == 0 else -i for i in nombres]
print(nombres_inverses)
```

```
[0, -1, 2, -3, 4, -5, 6, -7, 8, -9]
```

5.2 Exercice n°2: Afficher dix utilisateurs

[22]:
```
# Ma solution :
i = 0
while i < 10:
    print(f"Utilisateur {i+1}")
    i += 1
```

```
Utilisateur 1
Utilisateur 2
Utilisateur 3
Utilisateur 4
Utilisateur 5
Utilisateur 6
Utilisateur 7
Utilisateur 8
Utilisateur 9
Utilisateur 10
```

[23]:
```
# Autre solution :
for i in range(10):
    print(f"Utilisateur {i+1}")
```

```
Utilisateur 1
Utilisateur 2
Utilisateur 3
Utilisateur 4
Utilisateur 5
Utilisateur 6
Utilisateur 7
Utilisateur 8
Utilisateur 9
Utilisateur 10
```

5.3 Exercice n°3: Afficher un mot à l'envers

[24]:
```
mot = "Python"
liste_mot = (list(reversed(mot))) # ----> On inverse la chaîne de caractères
                                  # ----> On transforme la chaîne de caractères␣
 ↪en liste de caractères
print(liste_mot)

for i in liste_mot:
    print(i)
```

```
['n', 'o', 'h', 't', 'y', 'P']
n
o
h
t
y
```

```
P
```

[25]:
```python
# Autre solution (La plus simple) :
for lettre in reversed(mot):
    print(lettre)
```

```
n
o
h
t
y
P
```

[26]:
```python
# Très simple :
continuer = "o"
while continuer == "o":
    print("On continue !")
    continuer = input("Voulez-vous continuer ? (o/n) ")
```

```
On continue !
On continue !
On continue !
```

[27]:
```python
# Autre solution :
continuer = "o"
while continuer == "o":
    print("On continue !")
    resulat = input("Voulez-vous continuer ? (o/n) ")
    if resulat != "o":
        break
```

```
On continue !
```

6 Projet : La calculatrice et gestion d'erreurs

[ ]:
```python
# Objectif : Demander à l'utilisateur de saisir deux nombres et afficher la␣
 ↪somme de ces deux nombres.
# Tout en gérant les erreurs de saisie.
# On va utiliser une boucle while pour demander à l'utilisateur de saisir deux␣
 ↪nombres

a = b = ""
while not (a.isdigit() and b.isdigit()):
    a = input("Entrez un premier nombre : ")
    b = input("Entrez un deuxième nombre : ")
    if not (a.isdigit() and b.isdigit()):
        print("Erreur de saisie, veuillez entrer deux nombres entiers.")
```

```python
print(f"La somme de {a} et {b} est : {int(a) + int(b)}")
```

```
Erreur de saisie, veuillez entrer deux nombres entiers.
Erreur de saisie, veuillez entrer deux nombres entiers.
La somme de 25 et 77 est : 102
```


12 mai 2025, mis à jour le 15 juillet 2025