Python

Cours 4 - Boucles Itératives

Table des matières

Boucles Itératives - Définition

Les structures itératives, également appelées boucles, permettent de répéter des tâches sans avoir à dupliquer le même code.

Parmi les boucles les plus courantes, on trouve la boucle “for” et la boucle “while”.

Boucles Itératives - Différence entre “for” et “while”

La boucle “for” est généralement utilisée lorsque le nombre d’itérations est défini à l’avance, tandis que la boucle “while” exécute des instructions tant qu’une condition reste vraie, ce qui la rend particulièrement adaptée aux situations où le nombre d’itérations n’est pas déterminé à l’avance.

Boucles “for”

Une boucle “for” est utilisée pour itérer les valeurs d’une structure de données (liste, dictionnaire, etc.) et les traiter séparément.

Contrairement aux autres langages de programmation, le “for” en Python est plus exploité pour sa fonctionnalité d’itérateur.

Boucles “for” - Sur une liste

Lorsqu’une boucle “for” est utilisée sur une liste, les éléments sont itérés un après l’autre.

La gestion d’arrêt de la boucle est gérée automatiquement par le language de programmation.

Boucles “for” - Sur une liste

Listing: Boucle “for” sur une liste

```
1  # Steve Levesque, All rights reserved
2
3  courses = ["Python", "NoSQL", "Artificial Intelligence"]
4  for course in courses:
5      print("Course name: " + course)
6
7  """
8  Course name: Python
9  Course name: NoSQL
10 Course name: Artificial Intelligence
11 """
```

Boucles “for” - Sur une chaîne de caractères

Une chaîne de caractères peut être itérée à même titre qu’une liste.

Le résultat de cette opération est chaque caractère séparé pour chaque itération.

Boucles “for” - Sur une chaîne de caractère

Listing: Boucle “for” sur une chaîne de caractères

```
1  # Steve Levesque, All rights reserved
2
3  for i in "ABC":
4      print(i)
5
6  """
7  A
8  B
9  C
10 """
```

Boucles “for” - Sur une étendue (range)

Une étendue (“range”) est utile pour programmer spécifiquement le nombre d’itération,
et le saut (au besoin) entre chaque itération qu’une boucle doit avoir.

Il est souvent le cas où il n’y a pas de listes ou structure de données disponible au préalable. Donc, le programmeur doit pouvoir gérer le contexte lui-même.

Boucles “for” - Sur une étendue (range)

Listing: Boucle “for” avec une étendue

```
1  # Steve Levesque, All rights reserved
2
3  for i in range(3):
4      print(i)
5  print("---")
6  for i in range(1, 3):
7      print(i)
8  print("---")
9  for i in range(4, 10, 2):
10     print(i)
11
12 """
13 0  -  1  -  4
14 1  -  2  -  6
15 2  -     -  8
16 """
```

Boucles “for” - Sur un dictionnaire

Les dictionnaires sont importants puisqu'ils permettent d'associer une clé avec une valeur.

Souvent utilisés dans le contexte d'Intelligence Artificielle (AI).

Il est possible de l'itérer à même titre qu'une liste. La **différence** est qu'on a le choix d'avoir **soit la clé, la valeur, ou les deux**.

Boucles “while”

Listing: Boucle “for” sur un dictionnaire

```
1  # Steve Levesque, All rights reserved
2  ex_dict =      {
3      "course_name": "Artificial Intelligence",
4      "short": "AI",
5      "level": 1
6  }
7
8  for key in ex_dict.keys():
9      print(key)
10
11 for value in ex_dict.values():
12     print(value)
13
14 for key, value in ex_dict.items():
15     print(key, value)
```

Boucles “while”

Surtout en Python, contrairement au boucles “for”, les boucles “while” demandent une initialisation plus manuelle.

Quand utiliser une boucle “while” ?

Boucles “while”

On utilise généralement une boucle “while” quand on ne sait pas le nombre de valeurs qu'on va avoir à traiter (et dans le cas de Python, quand on ne sait pas les valeurs à la compilation dans le cas d'une liste par exemple).

Sinon, il suffit de faire une boucle “for” bornée comme on le souhaite.

Boucles “while”

Listing: Boucle “while” avec évaluation d’arrêt

```
1  # Steve Levesque, All rights reserved
2
3  i = 1
4  while i < 6:
5      print(i)
6      i += 1
7
8  """
9  1
10 2
11 3
12 4
13 5
14 """
```

Opérande “continue”

L’opérande “continue” permet d’esquiver un tour de boucle et continuer directement à l’élément suivant.

Utile, par exemple, pour ne pas faire une opération respectant une certaine règle, sans autant briser le rythme du programme.

Opérande “continue”

Listing: Opérande “continue” avec une boucle “for”

```
1  # Steve Levesque, All rights reserved
2
3  courses = ["Python", "NoSQL", "Artificial Intelligence"]
4  for course in courses:
5      if course == "NoSQL":
6          continue
7      print("Course name: " + course)
8
9  """
10 Course name: Python
11 Course name: Artificial Intelligence
12 """
```

Opérande “break”

L’opérande “break” permet d’arrêter la
boucle complètement et de continuer les instructions suivant celle-ci.

Utile, par exemple, si une règle doit faire arrêter le processus
cyclique sans autant arrêter le programme en entier.

Opérande “break”

Listing: Opérande “break” avec une boucle “for”

```
1  # Steve Levesque, All rights reserved
2
3  courses = ["Python", "NoSQL", "Artificial Intelligence"]
4  for course in courses:
5      if course == "NoSQL":
6          break
7      print("Course name: " + course)
8
9  """
10 Course name: Python
11 """
```

Bibliographie

- https://www.w3schools.com/python/