**Boucles**

**Boucles**

Les boucles sont un concept de programmation fondamental qui vous permet d'itérer sur les éléments d'une liste ou de répéter une action donnée un nombre défini ou même infini de fois.

Utilisées conjointement avec des listes, les boucles permettent d'accéder successivement à chaque élément de la liste. Ce processus est appelé **« itération »** .

En programmation informatique, une *boucle* est une séquence d'instructions qui est répétée continuellement jusqu'à ce qu'une condition spécifique soit atteinte.

Il existe deux principaux types de boucles : forles boucles et whileles boucles.

Voir la documentation officielle [ici](https://docs.python.org/3/tutorial/controlflow.html#for-statements)

**Boucles for**

Les boucles For sont principalement utilisées avec les listes : vous parcourez chaque élément de la liste dans l'ordre ; elle s'arrête automatiquement une fois qu'elle atteint la fin de la liste.

Les boucles For commencent par le mot-clé for. Vient ensuite ce que l'on appelle un « itérateur », une variable dont la valeur change à chaque tour de boucle.

Nous utiliserons cette variable dans la boucle comme valeur actuelle pour une itération donnée.

Tout d’abord, la syntaxe :

for <variable\_name> in <sequence\_name>:

CODE

CODE

Notez l’utilisation des deux points :et l’indentation du code.

**Exemple**

fruits = ['apple', 'banana', 'kiwi', 'pear']

for fruit in fruits:

print(fruit)

>> apple

>> banana

>> kiwi

>> pear

Dans la forboucle ci-dessus, fruit(singulier) est notre itérateur. Ses valeurs changent à chaque tour de boucle. Il est successivement : 'apple', 'banana', 'kiwi'et enfin 'pear'.

**Un autre exemple**

cities = ["London", "San Francisco", "Paris", "Barcelona"]

for city in cities:

print("I once went to", city)

La première ligne de la boucle est :

for city in cities:

Cette ligne indique à Python de récupérer la première valeur de la liste citieset de la stocker dans la variable city.

Vous pouvez choisir le nom que vous souhaitez pour cette variable ; cependant, il est utile de choisir un nom significatif représentant un seul élément de la liste.

La première valeur est « Londres ». Python lit ensuite la ligne suivante :

print("I once went to", city)

Python affiche la valeur de I once went to London. Comme la liste contient plus de valeurs, il revient à la première ligne.

for city in cities:

Python récupère maintenant l'élément suivant de la liste et le stocke dans la variable city(sa valeur est écrasée).

print("I once went to", city)

Maintenant, Python va imprimer I once went to San Francisco.

Python continue ainsi jusqu'à ce qu'il atteigne la fin de la liste.

**Gammes**

Les plages vous permettent de générer une séquence de nombres d'une valeur de départ à une valeur de fin.

Pour créer une plage, utilisez la rangefonction et transmettez-lui une valeur maximale et une valeur minimale. **La dernière valeur est non inclusive,** ce qui signifie que si vous parcourez une plage, elle ne sera pas incluse.

**Syntaxe**

numbers = range(4, 19)

>> range(4, 19)

**Exemple d'itération**

numbers = range(4, 19)

for number in numbers:

print(number)

>> 4

>> 5

>> 6

>> ...

>> 16

>> 17

>> 18

Pour imprimer des nombres de 1 à 5, utilisezrange(1,6)

Bien qu'il puisse être itéré, le **résultat de range n'est pas unlist** .

Pour le stocker dans un list, vous pouvez convertir les résultats de range()directement en un à listl'aide de la list()fonction. Lorsque vous encapsulez list()un appel à la range()fonction, le résultat sera une liste de nombres.

numbers = list(range(1,6))

print(numbers)

**Certaines fonctions intégrées qui utilisent des boucles for**

Des fonctions comme min, maxou sumutilisent des boucles for en coulisses ; pour vous entraîner, vous pouvez essayer de les programmer vous-même.

**Éviter les erreurs d'indentation**

Python utilise l'indentation pour déterminer quand une ligne de code est connectée à la ligne au-dessus.

Dans les exemples précédents, les lignes qui imprimaient des messages sur chaque ville faisaient partie de la forboucle car elles étaient en retrait.

L’utilisation de l’indentation par Python rend le code très facile à lire.

**Exercice**

Accepter un nombre de l'utilisateur et imprimer sa table de multiplication

**Boucles while**

Bien que les boucles diffèrent des forboucles classiques, elles possèdent ce que l'on appelle un cycle d'itération indéfini. Cela signifie que le nombre d'exécutions n'est pas défini par la longueur d'une liste donnée, mais par un point d'arrêt spécifique qui interrompt l'exécution de la boucle lorsqu'il est atteint.

Une whileboucle est définie de manière similaire à une instruction conditionnelle. Elle s'exécute tant que la condition est remplie Trueet s'arrête lorsque cette condition est évaluée à False.

La whileboucle s'exécute tant qu'une certaine condition est vraie.

Il exécutera toutes les instructions de la boucle, puis testera la condition, si la condition est toujours True, il réexécutera la boucle, et encore jusqu'à ce que la condition devienne fausse.

**Exemple**

current\_number = 1

while current\_number <= 5:

print(current\_number)

current\_number += 1

print("Finished")

>> 1

>> 2

>> 3

>> 4

>> 5

Dans la première ligne, nous commençons à compter à partir de 1 en définissant la valeur de current\_numberà 1.

La boucle while est configurée pour continuer à fonctionner tant que la valeur de current\_numberest inférieure ou égale à 5.

Le code à l'intérieur de la boucle imprime la valeur de current\_numberpuis ajoute 1à cette valeur avec current\_number += 1. (L' +=opérateur est l'abréviation de current\_number = current\_number + 1.)

Python répète la boucle tant que la condition current\_number <= 5est True. Parce 1que est inférieur à 5, Python imprime 1puis ajoute 1, ce qui donne current\_number2. Parce que 2est inférieur à 5, Python imprime 2et ajoute 1à nouveau, ce qui donne current\_number3, et ainsi de suite.

Une fois que la valeur de current\_numberest supérieure à 5, la boucle s'arrête et le programme se termine.

**Exemple**

password = ''

while password != 'hello-world-123':

password = input('What is the top secret password? ')

print('You guessed the right password!')

**Exercice**

Imprimez les nombres de 1 à 10 en utilisant la boucle while

**Boucle infinie**

Certaines conditions seront toujours remplies True, par exemple si nous exécutons ce code :

while 1 == 1:

print("Looping...")

Cela ne s'arrêtera jamais car 1 est toujours égal à 1. On peut aussi utiliser while 1:ou while True:.

**Utiliser un drapeau**

Qu'en est-il des programmes plus complexes dans lesquels de nombreux événements différents pourraient entraîner l'arrêt de l'exécution du programme ?

Pour un programme qui ne doit s'exécuter que tant que de nombreuses conditions sont remplies True, vous pouvez définir une variable qui détermine si le programme entier est actif. Cette variable, appelée *indicateur* , agit comme un signal envoyé au programme.

Nous pouvons écrire nos programmes de manière à ce qu'ils s'exécutent lorsque l'indicateur est défini sur , Trueet s'arrêtent lorsqu'un événement parmi plusieurs définit la valeur de l'indicateur sur False. Par conséquent, notre whileinstruction globale ne doit vérifier qu'une seule condition : si l'indicateur flagest actuellement défini sur True.

Ensuite, tous nos autres tests (pour voir si un événement s'est produit qui devrait définir le flagà False) peuvent être organisés proprement dans le reste du programme.

**Exemple**

active = True

while active:

city = input("Please enter the name of a city you have visited (enter 'quit' when you are finished): ")

if city == 'quit':

active = False

elif city == 'leave me alone':

active = False

elif city == 'stop':

active = False

else:

print("I'd love to go to", city)

print("Goodbye !")

**Pause et Continuer**

breaket continuesont deux instructions qui nous aident à gérer le flux de la boucle.

**casser**

breakquittez immédiatement la boucle while sans exécuter le code restant dans la boucle, quels que soient les résultats de toute condition.

Prenons par exemple un programme qui demande à l’utilisateur quels lieux il a visités.

Nous pouvons arrêter la boucle while dans ce programme en appelant break dès que l'utilisateur entre la valeur « quit » :

while True:

city = input("Please enter the name of a city you have visited (enter 'quit' when you are finished): ")

if city == 'quit':

break

else:

print("I'd love to go to", city)

print("Goodbye !")

Une boucle qui commence par while Trues'exécutera indéfiniment à moins qu'elle n'atteigne une instruction break.

La boucle de ce programme demande à l'utilisateur d'entrer les noms des villes qu'il a visitées jusqu'à ce qu'il entre « quit ».

Lorsqu'ils entrent « quit », l'instruction break s'exécute, obligeant Python à quitter la boucle :

Please enter the name of a city you have visited: (Enter 'quit' when you are finished.) New York

I'd love to go to New York!

Please enter the name of a city you have visited: (Enter 'quit' when you are finished.) San Francisco

I'd love to go to San Francisco!

Please enter the name of a city you have visited: (Enter 'quit' when you are finished.) quit

**Exemple**

secret\_number = 4

while True:

user\_number = input('Guess the secret number: ')

if int(user\_number) == secret\_number:

print('Congrats! You win!')

break

else:

print('Wrong guess...')

**continuer:**

Plutôt que de sortir complètement d'une boucle sans exécuter le reste de son code, vous pouvez utiliser l'instruction continue pour revenir au début de la boucle.

Par exemple, considérons une boucle qui compte de 1 à 10 mais imprime uniquement les nombres qui ne sont pas compris entre 3 et 7 :

current\_number = 0

while current\_number <= 10:

current\_number += 1

if 3 < current\_number < 7: # If the number is between 3 and 7

continue # Go back to the beginning of the loop

print(current\_number)

>> 1

>> 2

>> 3

>> 7

>> 8

>> 9

>> 10

Soyez prudent, cela peut finir en boucle infinie.

current = 0

while current <= 10:

print(current)

if 3 < current < 7:

continue

current += 1

// Here the incrementation is after the if statement, therefore as soon as the variable `current` reachers the number 4, the program enters into an infinite loop.