Theory: While loop

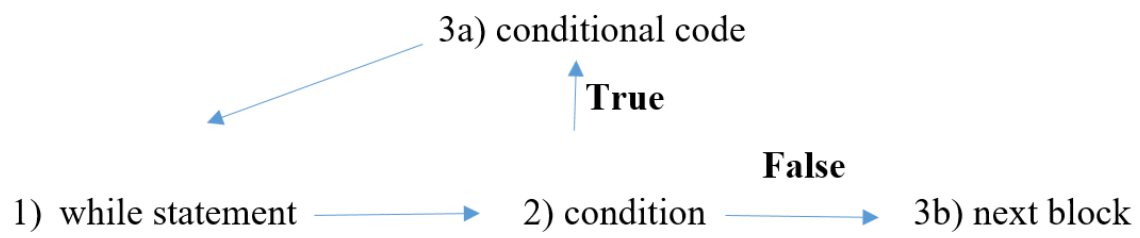
24 minutes 15 / 16 problems solved

Start practicing

Sometimes one **iteration** (=execution) of a statement is not enough to get the result you need. That is why Python offers a special statement that will execute a block of code several times. Meet the **loop** command and one of the universal loops — the **while** loop.

People generally don't choose Python to write **fast** code. The main advantages of Python are readability and simplicity. As the **while loop** requires the introduction of extra variables, iteration takes up more time. Thus, the while loop is quite slow and not that popular. It resembles a conditional operator: using the while loop, we can execute a set of statements as long as the condition is true.

The condition itself (2) is written before the **body** of the loop (some call it the conditional code) and is checked before the body is executed. If the condition is true (3a), the iterations continue. If the condition is false (3b), the loop execution is terminated and the program control moves further to the next operation.



Current topic:
While loop 11
Topic depends on:
If statement 16 Stage 1
Topic is required for:
Loop control statements 11
Recursion in Python
Custom generators
Exception handling 3

Table of contents:
↑ While loop
§1. Visualization
§2. The infinite loop
Feedback & Comments

§1. Visualization

If we visualize the while loop, it'll look like this:

```
number = 0
while number < 5:
    print(number)
    number += 1
print('Now, the number is equal to 5')
```

The variable `number` plays here the role of a **counter** – a variable that changes its value after each iteration. In this case, the iterations continue until the number is equal to 5 (note that the program outputs the value of the number *before* increasing it). When the value of a counter reaches 5, the program control moves to the next operation and prints the message. Here you can see the output of this code:

```
0
1
2
3
4
Now, the number is equal to 5
```

§2. The infinite loop

If you delete a part of the conditional code where you increase the value of a counter, you will bump into the **infinite loop.** What does it mean? Since you don't increase your variable, a condition never becomes false and can work forever. Usually, it is a logical fallacy, and you'll have to stop the loop using special statements or finishing the loop manually.

Sometimes the infinite loop can be useful, e.g. in querying a client when the loop works continuously to provide the constant exchange of information with a user. You can implement it by writing `True` as a condition after the `while` header.

```
while True:
    ...
```

Now you are familiar with the **while** loop and its usage. Don't forget about the role of a counter, otherwise, you'll have to deal with the infinite loop. After you've written the code, try to "run" it as if you were a Python program. That'll help you understand how the loop works.

Programming is all about simplification, so the code should be readable, short, and clear. Don't forget about comments and syntax. In the beginning, it may seem that the while loop is not that easy to implement, but after a couple of times, you'll see that it's a very useful tool.

Report a typo

994 users liked this theory. **14** didn't like it. **What about you?**

Start practicing

Comments (16) Hints (0) Useful links (1) Show discussion