Theory: For loop

35 minutes 15 / 16 problems solved

Start practicing

10541 users solved this topic. Latest completion was about 1 hour ago.

§1. What is iteration?

Computers are known for their ability to do things which people consider to be boring and energy-consuming. For example, repeating identical tasks without any errors is one of these things. In Python, the process of repetitive execution of the same block of code is called an **iteration**.

There are two types of iteration:

Definite iteration, where the number of repetitions is stated in advance.

Indefinite iteration, where the code block executes as long as the condition stated in advance is true.

After the first iteration, the program comes back to the beginning of the code's body and repeats it, making the so-called **loop**. The most used one is the `for` **loop,** named after the `for` **operator** that provides the code's execution.

§2. For loop syntax

Here is the scheme of the loop:

```
for variable in iterable:
    statement
```

where **statement** is a block of operations executed for each item in **iterable,** an object used in iteration (e.g. string or list). Variable takes the value of the next iterable after each iteration.

Now try to guess which output we'll get if we execute the following piece of code:

```
oceans = ['Atlantic', 'Pacific', 'Indian', 'Southern', 'Arctic']
for ocean in oceans:
    print(ocean)
```

During each iteration the program will take the items from the list and execute the statements with them, so the output will be:

```
Atlantic
Pacific
Indian
Southern
Arctic
```

Even strings are iterable, so you can spell the word, for example:

```
for char in 'magic':
    print(char)
```

Like this:

```
m
a
g
i
c
```

§3. Range function

Current topic:

- For loop 12★ (Stage 1)

Topic depends on:

- List 13★ (Stage 1)
- If statement 16★ (Stage 1)

Topic is required for:

- Loop control statements 11★
- List comprehension 5★
- Dictionary methods (Stage 1)
- Args
- Split and join 8★ (Stage 1)
- Iterators
- Reading files (Stage 2)
- BeautifulSoup
- How to read a traceback
- Django template language (Stage 2)
- Operations with tuple
- POS tagging

Table of contents:

The `range()` **function** is used to specify the number of iterations. It returns a sequence of numbers from 0 (by default) and ends at a specified number. Be careful: the last number won't be in the output.

Let's look at the example below:

```
for i in range(5):
    print(i)
```

What we'll get is this:

```
0
1
2
3
4
```

You can change the **starting value** if you're not satisfied with 0, moreover, you can configure the **increment (step)** value by adding a third parameter:

```
for i in range(5, 45, 10):
    print(i)
```

According to the parameters included, we've asked to print the numbers from 5 to 45 with an increment value of 10. Be careful again, the last value is not included in the output:

```
5
15
25
35
```

If you're not going to use the counter variable in your loop, you can show it by replacing its name with an underscore symbol:

```
for _ in range(100):
    do_smth()
```

In the example above, we don't need the counter variable in any way, we simply use the loop to repeat `do_smth()` function a given amount of times.

§4. Input data processing

You can also use the `input()` **function** that helps a user to pass a value to some variable and work with it. Thus, you can get the same output as with the previous piece of code:

```
word = input()
for char in word:
    print(char)
```

Oh, look, you can write a piece of code with a practical purpose:

```
times = int(input('How many times should I say "Hello"?'))
for i in range(times):
    print('Hello!')
```

You can, therefore, ask a user to specify the number of iterations to be performed.

§5. Nested loop

In Python, it is easy to put one loop inside another one – a **nested loop**. The type of inner and outer loops doesn't matter, the first to execute is the outer loop, then the inner one executes:

```
names = ['Rose', 'Daniel']
surnames = ['Miller']
for name in names:
    for surname in surnames:
        print(name, surname)
```

The output is shown below:

```
Rose Miller
Daniel Miller
```

In this example, we use the two `for` loops to create fictional people's names. Obviously, you can deal with iterable objects of different sizes without too much fuss.

§6. To sum up

All in all, `for` loops are an efficient way to automatize some repetitive actions. You can add variables and operations to make a nested loop. Moreover, you can control the number of iterations with the help of the `range()` function. Be careful with the syntax: an extra indent or lack of colon can cause a mistake!

Report a typo

820 users liked this theory. **16** didn't like it. **What about you?**

😍 🙂 😐 ☹️ 😡

Start practicing

Comments (10) Hints (2) Useful links (0) Show discussion