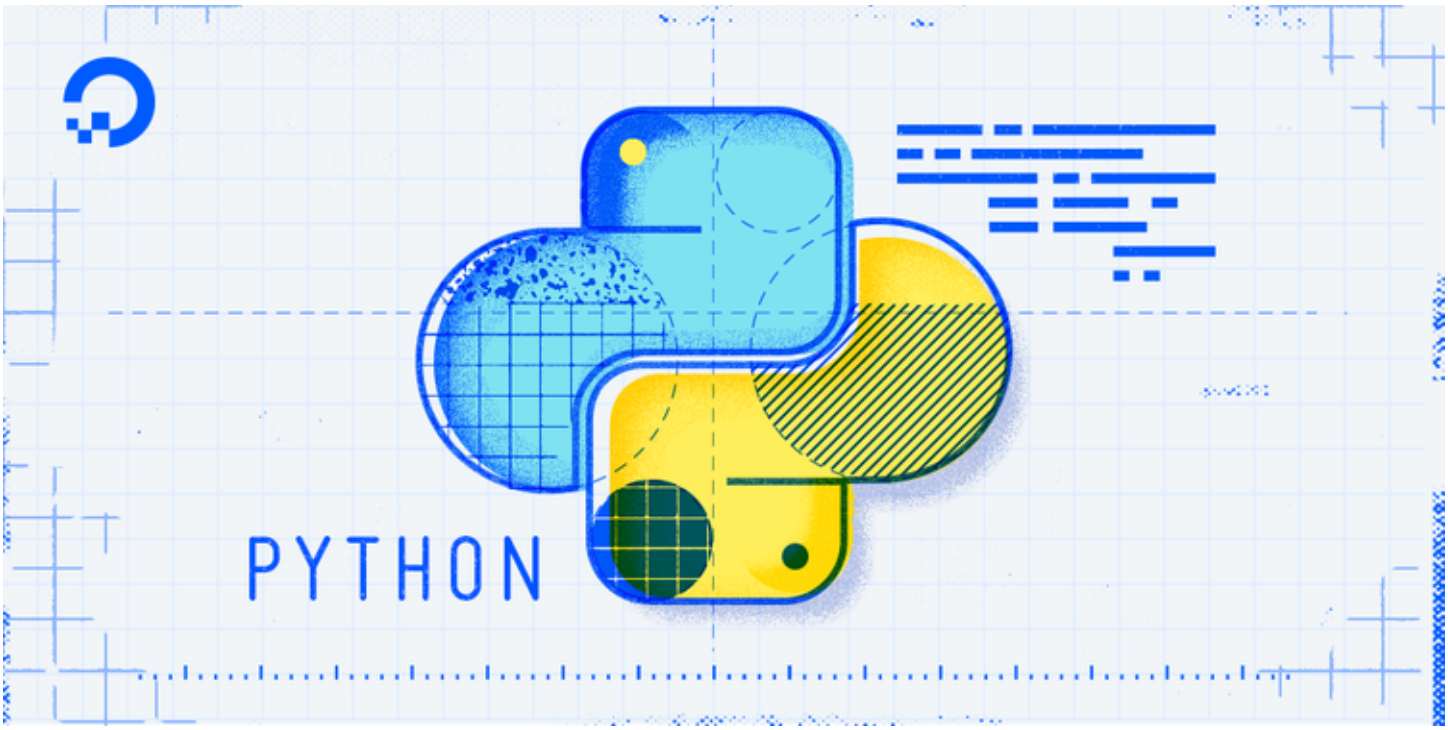
DigitalOcean

How To Code in Python 3 >
How To Use Break, Continue, and P...

Subscribe



How To Use Break, Continue, and Pass Statements when Working with Loops in Python 3

16

Posted January 6, 2017 652.9k PYTHON DEVELOPMENT

By: Lisa Tagliaferri

Introduction

Using **for loops** and **while loops** in Python allow you to automate and repeat tasks in an efficient manner.

But sometimes, an external factor may influence the way your program runs. When this occurs, you may want your program to exit a loop completely, skip part of a loop before continuing, or

ignore that external factor. You can perform these actions with `break`, `continue`, and `pass` statements.

Break Statement

In Python, the `break` statement provides you with the opportunity to exit out of a loop when an external condition is triggered. You'll put the `break` statement within the block of code under your loop statement, usually after a conditional `if` statement.

Let's look at an example that uses the `break` statement in a `for` loop:

```
number = 0

for number in range(10):
   number = number + 1

   if number == 5:
      break    # break here

   print('Number is ' + str(number))

print('Out of loop')
```

In this small program, the variable `number` is initialized at 0. Then a `for` statement constructs the loop as long as the variable `number` is less than 10.

Within the `for` loop, the number increases incrementally by 1 with each pass because of the line `number = number + 1`.

Then, there is an `if` statement that presents the condition that *if* the variable `number` is equivalent to the integer 5, *then* the loop will break.

Within the loop is also a `print()` statement that will execute with each iteration of the `for` loop until the loop breaks, since it is after the `break` statement.

To see when we are out of the loop, we have included a final `print()` statement outside of the `for` loop.

When we run this code, our output will be the following:

```
Output
```

```
Number is 3
Number is 4
Out of loop
```

This shows that once the integer `number` is evaluated as equivalent to 5, the loop breaks, as the program is told to do so with the `break` statement.

The `break` statement causes a program to break out of a loop.

Continue Statement

The `continue` statement gives you the option to skip over the part of a loop where an external condition is triggered, but to go on to complete the rest of the loop. That is, the current iteration of the loop will be disrupted, but the program will return to the top of the loop.

The `continue` statement will be within the block of code under the loop statement, usually after a conditional `if` statement.

Using the same `for` loop program as in the Break Statement section above, we'll use a `continue` statement rather than a `break` statement:

```
number = 0

for number in range(10):
   number = number + 1

   if number == 5:
      continue    # continue here

   print('Number is ' + str(number))

print('Out of loop')
```

The difference in using the `continue` statement rather than a `break` statement is that our code will continue despite the disruption when the variable `number` is evaluated as equivalent to 5. Let's look at our output:

```
Output
Number is 1
Number is 2
Number is 3
```

```
Number is 7
Number is 8
Number is 9
Number is 10
Out of loop
```

Here we see that the line `Number is 5` never occurs in the output, but the loop continues after that point to print lines for the numbers 6-10 before leaving the loop.

You can use the `continue` statement to avoid deeply nested conditional code, or to optimize a loop by eliminating frequently occurring cases that you would like to reject.

The `continue` statement causes a program to skip certain factors that come up within a loop, but then continue through the rest of the loop.

Pass Statement

When an external condition is triggered, the `pass` statement allows you to handle the condition without the loop being impacted in any way; all of the code will continue to be read unless a `break` or other statement occurs.

As with the other statements, the `pass` statement will be within the block of code under the loop statement, typically after a conditional `if` statement.

Using the same code block as above, let's replace the `break` or `continue` statement with a `pass` statement:

```
number = 0

for number in range(10):
   number = number + 1

   if number == 5:
      pass    # pass here

   print('Number is ' + str(number))

print('Out of loop')
```

The `pass` statement occurring after the `if` conditional statement is telling the program to continue to run the loop and ignore the fact that the variable `number` evaluates as equivalent to 5 during one of its iterations.

```
Output
Number is 1
Number is 2
Number is 3
Number is 4
Number is 5
Number is 6
Number is 7
Number is 8
Number is 9
Number is 10
Out of loop
```

By using the `pass` statement in this program, we notice that the program runs exactly as it would if there were no conditional statement in the program. The `pass` statement tells the program to disregard that condition and continue to run the program as usual.

The `pass` statement can create minimal classes, or act as a placeholder when working on new code and thinking on an algorithmic level before hammering out details.

Conclusion

The `break`, `continue`, and `pass` statements in Python will allow you to use `for` loops and `while` loops more effectively in your code.

You can see `break` and `pass` statements in action in our tutorial "How To Create a Twitterbot with Python 3 and the Tweepy Library."

Tutorial Series

How To Code in Python 3

Python is an extremely readable and versatile programming language. Written in a

simplicity and versatility, in terms of extensibility and supported paradigms.

Show Tutorials

Related Tutorials

How To Apply Computer Vision to Build an Emotion-Based Dog Filter in Python 3

How To Detect and Extract Faces from an Image with OpenCV and Python

How To Install and Use TensorFlow on Ubuntu 18.04

Bias-Variance for Deep Reinforcement Learning: How To Build a Bot for Atari with OpenAI Gym

How To Set Up Jupyter Notebook with Python 3 on Ubuntu 18.04

1 Comment

Leave a comment...

Log In to Comment

Topspap *December 1, 2018*

0 Well done, Fully understood.
Thanks for clarification.
Have a nice day