PYTHON *BREAK, CONTINUE AND PASS* STATEMENTS

You might face a situation in which you need to exit a loop completely when an external condition is triggered or there may also be a situation when you want to skip a part of the loop and start next execution.

Python provides **break** and **continue** statements to handle such situations and to have good control on your loop.

This tutorial will discuss the *break, continue* and *pass* statements available in Python.

THE *BREAK* STATEMENT:

The **break** statement in Python terminates the current loop and resumes execution at the next statement, just like the traditional break found in C.

The most common use for break is when some external condition is triggered requiring a hasty exit from a loop. The **break** statement can be used in both *while* and *for* loops.

Example:

```
#!/usr/bin/python

for letter in 'Python':     # First Example
   if letter == 'h':
      break
   print 'Current Letter :', letter

var = 10                    # Second Example
while var > 0:
   print 'Current variable value :', var
   var = var -1
   if var == 5:
      break

print "Good bye!"
```

This will produce the following result:

```
Current Letter : P
Current Letter : y
Current Letter : t
Current variable value : 10
Current variable value : 9
Current variable value : 8
Current variable value : 7
Current variable value : 6
Good bye!
```

THE *CONTINUE* STATEMENT:

The **continue** statement in Python returns the control to the beginning of the while loop. The **continue** statement rejects all the remaining statements in the current iteration of the loop and moves the control back to the top of the loop.

The **continue** statement can be used in both *while* and *for* loops.

Example:

```
#!/usr/bin/python

for letter in 'Python':     # First Example
```

```
   if letter == 'h':
      continue
   print 'Current Letter :', letter

var = 10                    # Second Example
while var > 0:
   var = var -1
   if var == 5:
      continue
   print 'Current variable value :', var
print "Good bye!"
```

This will produce following result:

```
Current Letter : P
Current Letter : y
Current Letter : t
Current Letter : o
Current Letter : n
Current variable value : 10
Current variable value : 9
Current variable value : 8
Current variable value : 7
Current variable value : 6
Current variable value : 4
Current variable value : 3
Current variable value : 2
Current variable value : 1
Good bye!
```

THE *ELSE* STATEMENT USED WITH LOOPS

Python supports to have an **else** statement associated with a loop statements.

- If the **else** statement is used with a **for** loop, the **else** statement is executed when the loop has exhausted iterating the list.
- If the **else** statement is used with a **while** loop, the **else** statement is executed when the condition becomes false.

Example:

The following example illustrates the combination of an else statement with a for statement that searches for prime numbers from 10 through 20.

```
#!/usr/bin/python

for num in range(10,20):  #to iterate between 10 to 20
   for i in range(2,num): #to iterate on the factors of the number
      if num%i == 0:      #to determine the first factor
         j=num/i #to calculate the second factor
         print '%d equals %d * %d' % (num,i,j)
         break #to move to the next number, the #first FOR
   else:        # else part of the loop
      print num, 'is a prime number'
```

This will produce following result:

```
10 equals 2 * 5
11 is a prime number
12 equals 2 * 6
13 is a prime number
14 equals 2 * 7
15 equals 3 * 5
16 equals 2 * 8
17 is a prime number
18 equals 2 * 9
```

```
19 is a prime number
```

Similar way you can use **else** statement with **while** loop.

THE *PASS* STATEMENT:

The **pass** statement in Python is used when a statement is required syntactically but you do not want any command or code to execute.

The **pass** statement is a *null* operation; nothing happens when it executes. The **pass** is also useful in places where your code will eventually go, but has not been written yet $e.g., instubsforexample$:

Example:

```
#!/usr/bin/python

for letter in 'Python':
   if letter == 'h':
      pass
      print 'This is pass block'
   print 'Current Letter :', letter

print "Good bye!"
```

This will produce following result:

```
Current Letter : P
Current Letter : y
Current Letter : t
This is pass block
Current Letter : h
Current Letter : o
Current Letter : n
Good bye!
```

The preceding code does not execute any statement or code if the value of *letter* is 'h'. The *pass* statement is helpful when you have created a code block but it is no longer required.

You can then remove the statements inside the block but let the block remain with a pass statement so that it doesn't interfere with other parts of the code.