## Loop Control Statements

Loop control statements change execution from its normal sequence. When execution leaves a scope, all automatic objects that were created in that scope are destroyed.

Python supports the following control statements.

|  |  |
| --- | --- |
| Sr.No. | Control Statement & Description |
| 1 | BREAK STATEMENT :  Terminates the loop statement and transfers execution to the statement immediately following the loop.  It terminates the current loop and resumes execution at the next statement, just like the traditional break statement 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.  If you are using nested loops, the break statement stops the execution of the innermost loop and start executing the next line of code after the block. Syntax The syntax for a **break** statement in Python is as follows −  break Flow Diagram  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!"  When the above code is executed, it produces 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! |
| 2 | CONTINUE STATEMENT :  Causes the loop to skip the remainder of its body and immediately retest its condition prior to reiterating.  It 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. Syntax continue Flow Diagram  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!"  When the above code is executed, it produces the following result −  Current Letter : P  Current Letter : y  Current Letter : t  Current Letter : o  Current Letter : n  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  Current variable value : 0  Good bye! |
| 3 | 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., in stubs for example) − Syntax pass Example #!/usr/bin/python  for letter in 'Python':  if letter == 'h':  pass  print 'This is pass block'  print 'Current Letter :', letter  print "Good bye!"  When the above code is executed, it produces 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! |