**Study on Loop Control Structures in Python**

**while loop**

Repeats a statement or group of statements while a given condition is TRUE. It tests the condition before executing the loop body.

**for loop**

Executes a sequence of statements multiple times and abbreviates the code that manages the loop variable.

**nested loops**

You can use one or more loop inside any another while, for or do..while loop.

**break statement**

Terminates the loop statement and transfers execution to the statement immediately following the loop.

**continue statement**

Causes the loop to skip the remainder of its body and immediately retest its condition prior to reiterating.

**Examples**

With the **while loop** we can execute a set of statements as long as a condition is true.

i = 1  
while i < 6:  
  print(i)  
  i += 1

With the **break statement** we can stop the loop even if the while condition is true:

Exit the loop when i is 3:

i = 1  
while i < 6:  
  print(i)  
  if i == 3:  
    break  
  i += 1

With the **continue statement** we can stop the current iteration, and continue with the next:

Continue to the next iteration if i is 3:

i = 0  
while i < 6:  
  i += 1  
  if i == 3:  
    continue  
  print(i)

With the **else statement** we can run a block of code once when the condition no longer is true:

i = 1  
while i < 6:  
  print(i)  
  i += 1  
else:  
  print("i is no longer less than 6")

A **for loop** is used for iterating over a sequence (that is either a list, a tuple, a dictionary, a set, or a string).

The **range() function** returns a sequence of numbers, starting from 0 by default, and increments by 1 (by default), and ends at a specified number.

for x in range(6):  
  print(x)

The **range() function defaults** to 0 as a starting value, however it is possible to specify the starting value by adding a parameter: range(2, 6), which means values from 2 to 6 (but not including 6):

for x in range(2, 6):  
  print(x)

**The range() function defaults to increment the sequence by 1**, however it is possible to specify the increment value by adding a third parameter: range(2, 30, **3**):

Increment the sequence with 3 (default is 1):

for x in range(2, 30, 3):  
  print(x)

The **else keyword in a for loop** specifies a block of code to be executed when the loop is finished:

Print all numbers from 0 to 5, and print a message when the loop has ended:

for x in range(6):  
  print(x)  
else:  
  print("Finally finished!")

[Try it Yourself »](https://www.w3schools.com/python/trypython.asp?filename=demo_for_else)

**With the for loop we can execute a set of statements, once for each item in a list, tuple, set**

Print each fruit in a fruit list:

fruits = ["apple", "banana", "cherry"]  
for x in fruits:  
  print(x)

**Even strings are iterable objects, they contain a sequence of characters:**

for x in "banana":  
  print(x)

With the **break statement** we can stop the loop before it has looped through all the items:

Exit the loop when x is "banana":

fruits = ["apple", "banana", "cherry"]  
for x in fruits:  
  print(x)  
  if x == "banana":  
    break

Exit the loop when x is "banana", but this time the break comes before the print:

fruits = ["apple", "banana", "cherry"]  
for x in fruits:  
  if x == "banana":  
    break  
  print(x)

With the **continue statement** we can stop the current iteration of the loop, and continue with the next:

Do not print banana:

fruits = ["apple", "banana", "cherry"]  
for x in fruits:  
  if x == "banana":  
    continue  
  print(x)