Python Loops

For Loop and While Loop

Loops in Python

There are two types of loops in Python :

1. For Loop
2. While Loop

For Loop

The for loop in Python is used to iterate over a sequence (list, string, tuple) or other iterable objects.

- To get a sequence of numbers use range() function. range(10) will generate numbers from 0 to 9(10 numbers)
- It also contains start, stop and step size as **range**(start, stop, step_size). Step_size defaults to 1 if not provided.

Syntax:

```
for iterator_var in sequence:
    statements(s)
```

Example

```
# Python program to illustrate
# Iterating over range 0 to n-1

n = 4
for i in range(0, n):
    print(i)
```

Break statement

- In Python, break and continue statements can alter the flow of a normal loop.
- Loops iterate over a block of code until the test expression is false, but sometimes we wish to terminate the current iteration or even the whole loop without checking the test expression.
- The break and continue statements are used in these cases.

Example: Python break

```
# Use of break statement inside the loop

for val in "string":
    if val == "i":
        break
    print(val)

print("The end")
```

Continue statement

The continue statement is used to skip the rest of the code inside a loop for the current iteration only. Loop does not terminate but continues on with the next iteration.

Example: Python continue

```
# Program to show the use of continue statement inside loops

for val in "string":
    if val == "i":
        continue
    print(val)

print("The end")
```

While Loop

The while loop in python is used to iterate over a block of code as long as the test expression(Condition) is True.

Syntax:

```
while expression:
    statement(s)
```

Example

Example: Python while Loop

```
# Program to add natural
# numbers up to
# sum = 1+2+3+...+n

# To take input from the user,
# n = int(input("Enter n: "))

n = 10

# initialize sum and counter
sum = 0
i = 1

while i <= n:
    sum = sum + i
    i = i+1    # update counter

# print the sum
print("The sum is", sum)
```

While-Else

- While loops can also have an optional **ELSE** block.
-
- The while loop can be terminated with a break statement. In such cases, the else part is ignored.

```
while condition:
    # execute these statements
else:
    # execute these statements
```

Example

```
'''Example to illustrate
the use of else statement
with the while loop'''

counter = 0

while counter < 3:
    print("Inside loop")
    counter = counter + 1
else:
    print("Inside else")
```

Output

```
Inside loop
Inside loop
Inside loop
Inside else
```

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.

Example:

```
#!/usr/bin/python

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

print "Good bye!"
```