007 Python for loop

for loop

Like most other languages, Python has for loops, but it differs a bit from other like C or Pascal. In Python for loop is used to iterate over the items of any sequence including the Python list, string, tuple etc. The for loop is also used to access elements from a container (for example list, string, tuple) using built-in function range().

Syntax:

```
for variable_name in sequence :
    statement_1
    statement_2
    ....
```

Parameter:

Name	Description
variable_name	It indicates target variable which will set a new value for each iteration of the loop.
sequence	A sequence of values that will be assigned to the target variable variable_name. Values are provided using a list or a string or from the built-in function range().
statement_1, statement_2	Block of program statements.

Example: Python for loop

```
>>> #The list has four elements, indices start at 0 and end at 3

>>> color_list = ["Red", "Blue", "Green", "Black"]

>>> for c in color_list:
        print(c)


  Red

  Blue

  Green

  Black

>>>
```

In the above example color_list is a sequence contains a list of various color names. When the for loop executed the first item (i.e. Red) is assigned to the variable c. After this, the print statement will execute and the process will continue until we reach the end of the list.

Python for loop and range() function

The range() function returns a list of consecutive integers. The function has one, two or three parameters where last two parameters are optional. It is widely used in for loops. Here is the syntax.

```
range(a)
range(a,b)
range(a,b,c)
```

range(a) : Generates a sequence of numbers from 0 to a, excluding a, incrementing by 1.

Syntax:

```
for <variable> in range(<number>):
```

Example:

```
>>> for a in range(4):

  print(a)
  0

  1

  2

  3

>>>
```

range(a,b): Generates a sequence of numbers from a to b excluding b, incrementing by 1.

Syntax:

```
for "variable" in range("start_number", "end_number"):
```

Example:

```
>>> for a in range(2,7):

 print(a)



  2

  3

  4

  5

  6

>>>
```

range(a,b,c): Generates a sequence of numbers from a to b excluding b, incrementing by c.

Example:

```
>>> for a in range(2,19,5):

  print(a)
```

```
 2

 7

 12

 17

>>>
```

Python for loop: Iterating over tuple, list, dictionary

Example: Iterating over tuple

The following example counts the number of even and odd numbers from a series of numbers.

```
numbers = (1, 2, 3, 4, 5, 6, 7, 8, 9) # Declaring the tuple

count_odd = 0

count_even = 0

for x in numbers:

        if x % 2:

            count_odd+=1

        else:

            count_even+=1

print("Number of even numbers :",count_even)

print("Number of odd numbers :",count_odd)
```

Output:

```
Number of even numbers:4
Number of odd numbers:  5
```

In the above example a tuple named numbers is declared which holds the integers 1 to 9.

The best way to check if a given number is even or odd is to use the modulus operator (%).
The operator returns the remainder when dividing two numbers.
Modulus of 8 % 2 returns 0 as 8 is divided by 2, therefore 8 is even and modulus of 5 % 2 returns 1 therefore 5 is odd.

The for loop iterates through the tuple and we test modulus of x % 2 is true or not, for every item in the tuple and the process will continue until we rich the end of the tuple.
When it is true count_even increase by one otherwise count_odd is increased by one.
Finally, we print the number of even and odd numbers through print statements.

Example: Iterating over list

In the following example for loop iterates through the list "datalist" and prints each item and its corresponding Python type.

```
datalist = [1452, 11.23, 1+2j, True,  'google', (0, -1), [5, 12],

{"class":'V', "section":'A'}]

for item in datalist:

   print ("Type of ",item, " is ", type(item))
```

Output:

```
Type of  1452  is  <class 'int'>
Type of  11.23  is  <class 'float'>
Type of  (1+2j)  is  <class 'complex'>
Type of  True  is  <class 'bool'>
Type of   google  is  <class 'str'>
Type of  (0, -1)  is  <class 'tuple'>
Type of  [5, 12]  is  <class 'list'>
Type of  {'section': 'A', 'class': 'V'}  is  <class 'dict'>
```

Example: Iterating over dictionary

In the following example for loop iterates through the dictionary "color" through its keys and prints each key.

```
>>> color = {"c1": "Red", "c2": "Green", "c3": "Orange"}

>>> for key in color:

   print(key)



c2

c1

c3

>>>
```

Following for loop iterates through its values :

```
>>> color = {"c1": "Red", "c2": "Green", "c3": "Orange"}

>>> for value in color.values():

   print(value)



Green

Red

Orange

>>>
```

You can attach an optional else clause with for statement, in this case, syntax will be -

```
for variable_name in sequence :
    statement_1
    statement_2
    ....
else :
    statement_3
    statement_4
    ....
```

The else clause is only executed after completing the for loop. If a break statement executes in first program block and terminates the loop then the else clause does not execute.