Python Programming
Unit 2
(KNC-302)

Prepared By

Abhishek Kesharwani

Assistant Professor,UCER Naini,Allahabad

UNIT II

- Conditionals: Conditional statement in Python
- if-else statement, its working and execution
- Nested-if statement and Elif statement in Python
- Expression Evaluation
- Float Representation
- Loops: Purpose and working of loops
- While loop including its working
- For Loop, Nested Loops,
- Break and Continue.

Python Loops

- The flow of the programs written in any programming language is sequential by default.
- Sometimes we may need to alter the flow of the program.
- The execution of a specific code may need to be repeated several numbers of times.

Python for loop

The for **loop in Python** is used to iterate the statements or a part of the program several times. It is frequently used to traverse the data structures like list, tuple, or dictionary.

The syntax of for loop in python is given below.

```
for iterating_var in sequence:
    statement(s)
```

The range() Function

- To loop through a set of code a specified number of times, we can use the range() function,
- 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.

Example

```
for x in range(6):
    print(x)
```

- Note that range(6) is not the values of 0 to 6, but the values 0 to 5.
- 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):

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**):

Example

Increment the sequence with 3 (default is 1):

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

```
i=1
n=int(input("Enter the number up to which you
   want to print the natural numbers?"))
for i in range(0,10):
   print(i,end = ' ')
```

Output:

0 1 2 3 4 5 6 7 8 9

printing the table of the given number

```
i=1
num = int(input("Enter a number:"))
for i in range(1,11):
    print("%d X %d = %d"%(num,i,num*i))
```

Nested for loop in python

- Python allows us to nest any number of for loops inside a for loop.
- The inner loop is executed n number of times for every iteration of the outer loop.

Syntax

```
for iterating_var1 in sequence:
    for iterating_var2 in sequence:
        #block of statements
#Other statements
```

Pattern

```
n=int(input("enter the number of rows"))
for i in range (0,n):
    for j in range (0,i+1):
      print("*",end=" ")
    print()
```

Output

```
D:\python cs\programs>python forloop3.py
enter the number of rows5
*
* *
* * *
* * * *
* * * * *
```

Using else statement with for loop

- Unlike other languages like C, C++, or Java, python allows us to use the else statement with the for loop which can be executed only when all the iterations are exhausted.
- Here, we must notice that if the loop contains any of the break statement then the else statement will not be executed.

Example

```
for i in range(0,5):
    print(i)
else:print("for loop completely exhausted, since
    there is no break.");
```

Output

0

1

2

3

4

for loop completely exhausted, since there is no break.

Example

```
for i in range(0,5):
    print(i)
    break;
else:print("for loop is exhausted");
print("The loop is broken due to break statemen
   t...came out of loop")
```

Output:

0

The loop is broken due to break statement...came out of loop

The break Statement

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

Example

```
fruits = ["apple", "banana", "cherry"]

for x in fruits:
    print(x)
    if x == "banana":
      break
```

```
D:\python cs\programs>python forloop4.py
apple
banana

D:\python cs\programs>
```

The continue Statement

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

Example

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

Output

```
C:\Users\My Name>python demo_for_continue.py
apple
cherry
```

Print each fruit in a fruit list

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

Looping Through a String

```
for x in "banana":
    print(x)
```

Example

Print each adjective for every fruit:

```
adj = ["red", "big", "tasty"]
fruits = ["apple", "banana", "cherry"]
for x in adj:
    for y in fruits:
      print(x, y)
```

```
C:\Users\My Name>python demo_for_nested.py
red apple
red banana
red cherry
big apple
big banana
big cherry
tasty apple
tasty banana
tasty cherry
```

Python while loop

- The while loop is also known as a pre-tested loop. In general, a while loop allows a part of the code to be executed as long as the given condition is true.

while expression:

    statements

Example

```
i=1;
while i<=10:
    print(i);
    i=i+1;
```

Using else with Python while loop

- Python enables us to use the while loop with the while loop also.
- The else block is executed when the condition given in the while statement becomes false. Like for loop, if the while loop is broken using break statement, then the else block will not be executed and the statement present after else block will be executed.

Example

Consider the following example.

```
i=1;
while i<=5:
    print(i)
    i=i+1;
else:print("The while loop exhausted");
```

Output

1

2

3

4

5

The while loop exhausted

Python Pass

- In Python, pass keyword is used to execute nothing; it means, when we don't want to execute code, the pass can be used to execute empty.
- It just makes the control to pass by without executing any code. If we want to bypass any code pass statement can be used.

Example

```
for i in [1,2,3,4,5]:
    if i==3:
        pass
        print "Pass when value is",i
    print i,
```

Example

```
D:\python cs\programs>python pass.py
1
2
Pass when value is 3
3
4
5
```