Chapter -4 : Operators, Expressions and Python Statements

Nested loops

Python programming language allows using one loop inside another **while** or **for** loop.

Syntax nested for loop

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

Syntax nested while loop

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

Program uses a nested for loop to find the prime numbers from 2 to 100

```
i = 2
while(i < 100):
  j = 2
  while(j <= (i/j)):
    if not(i%j): break
    j = j + 1
  if (j > i/j) : print (i, " is prime")
  i = i + 1
```

Output

It gives the list of prime number from 2 to 100 .

Program uses a nested-for loop to display multiplication tables from 1-10.

```
for i in range(1,11):
  for j in range(1,11):
    k = i*j
    print (k, end=' ')
  print()
```

Output

```
1 2 3 4 5 6 7 8 9 10
2 4 6 8 10 12 14 16 18 20
3 6 9 12 15 18 21 24 27 30
4 8 12 16 20 24 28 32 36 40
5 10 15 20 25 30 35 40 45 50
6 12 18 24 30 36 42 48 54 60
7 14 21 28 35 42 49 56 63 70
8 16 24 32 40 48 56 64 72 80
9 18 27 36 45 54 63 72 81 90
10 20 30 40 50 60 70 80 90 100
```

Program to print the pattern

```
for i in range(1,6):
    for j in range(i):
        print("*",end=' ')
    print()
```

Output

```
*
* *
* * *
* * * *
* * * * *
```

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.

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.
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.

break statement

- 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.

Example

```
for letter in 'Python':     # First Example
   if letter == 'h':
      break
   print ('Current Letter :', letter)
```

Output

```
Current Letter : P
Current Letter : y
Current Letter : t
```

Continue statement

- 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.

Example

```
for letter in 'Python':     # First Example
   if letter == 'h':
      continue
   print 'Current Letter :', letter
```

Output

```
Current Letter : P
Current Letter : y
Current Letter : t
Current Letter : o
Current Letter : n
```

pass statement

- It 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

Example

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

Output

```
Current Letter : P
Current Letter : y
Current Letter : t
This is pass block
Current Letter : h
Current Letter : o
Current Letter : n
```