LOOPS (contd...)

Topics:

- While Loop
- Break
- Continue

Syntax for "*for loop*" can be written in 2 ways:

⇨ for i in range (0, 10)
    print (i)
⇨ for i in "*ABCDEFG*"
    print(i)

Ex: for ch in "*ABCDEFGHIJ*"

print (ch)

ch will have values as ("A", "B" ... ... "J")

INPUT:

```
for ch in "ABCDEFGHIJ":
  print ("The character is ", ch)
```

OUTPUT:

```
The character is A
The character is B
The character is C
The character is D
The character is E
The character is F
The character is G
The character is H
The character is I
The character is J
```

WHILE LOOP

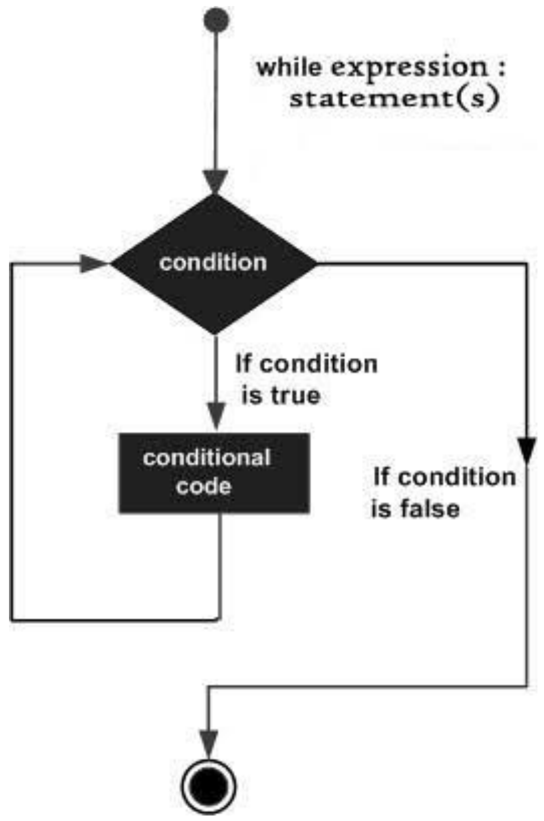
Syntax:

while (boolean_condition):

So, in a while loop also you write a Boolean condition and after colon, from the next line after every tab, it will be the body of the while loop.

while (condition):

print (“something”)



So, if the condition in true, then the program will enter the body of the loop and once the condition becomes false, the program will come out of the loop.

Ex:

While (True):

Print (“hello”)

This loop will run infinite times. But, if I want to repeat it only for a set of 4 times, then we can create variable with its initial value as 0 and count < 4

INPUT:

```
cnt = 0
while cnt < 4:
    print("hello!")
```

The program will take cnt value as 0 and will come to the next line. In the second line it will enter the while loop and check if cnt < 4 is true or false. Since it is true, the program will enter the while loop and print “hello”.

After executing the print statement, the program will go back to the first line and take value of cnt as 0. It will do run again and will print “hello”. The program will print hello infinite times.

Now, to stop the infinite loop, use Cntrl+C.

To get the hello word to be printed only 4 times, we have to use cnt += 1. This will increase the value of cnt after every cycle. So once the cnt value become 4, 4 < 4 is false, so the program will exit the loop.

Can we get an increment via multiplication and not by addition?

Ex: 2, 8, 32, 128 …

In for loop, we can take a step by adding the step and not by multiplication. But we can do this in while loop.

Ex:

```
i = 2

while (i < 2):

    print (i)

    i = i * i
```

BREAK

```
for x in range (0, 10):
    if x == 5:
        print("found")
    else:
        print("not found")
```

OUTPUT:

```
not found          not found          not found
not found          found              not found
not found          not found
not found          not found
```

When we use the break command, once the value is found, the program will break and will come out.

```
for x in range (0, 10):
    if x == 5:
        print("found")
        break
    else:
        print("not found")
```

OUTPUT

```
not found
not found
not found
not found
not found
found
```

If we want to search for a particular value in range (1, 10) then once you come to I = 5, there is no need to check the remaining values. In such scenarios, we can use the ***break*** command.

BREAK is used inside a loop and it will break the nearest loop it is part of in other words, it terminates the current loop and resumes execution at the next statement

The first program will check for all values of I and even after the program finds the value, it will check for the remaining values of I. but when we use the break command, once the value is found, the program will come out of the loop.

CONTINUE

```
for i in range (0,100):
    if (condition):
            ------------------------      ———>      this is called the “if block”
            ------------------------
print (“hello”)
```

Whenever in your code, if we use continue, it will skip the remaining code and will go to the start of the loop. In the above mentioned, example, irrespective of the if condition being true or false, the print statement would run because it is not part of the **if block**.

Ex:

```
for i in range (0, 10):
    if (i < 5):
        continue
    print (i)
```

For i = 0, 1, 2, 3, 4 the value of i is true. Since it is true, it will enter the if block and will execute the continue command. The continue command will take you back to the nearest loop it is in, i.e., the if (i < 5). When the value of i becomes 5, then the if condition is false and the program will print 5. The remaining vales of i (6, 7, 8, 9) are greater than 5 the program will print them. So, the output would be

OUTPUT:

```
5
6
7
8
9
```

It will not print 10, because the program will only check till n-1. Since n is 10, it will only check till 9.

The continue statement in Python 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.

MCQ 1:

```python
for x in range(0, 4):
    if x % 2 == 0:
        continue
    print(x)
```

ANS: 1 3

MCQ 2:

```python
for x in range (0, 4):
    if x % 2 == 0:
        print (x)
        break
    print (x)
```

ANS: 0

MCQ 3:

```python
for x in range (0, 4):
    if x < 2:
        continue
    if x == 4:
        break
    print(x)
```

ANS: 2 3

NOTE:

Break → it comes out of the loop
continue → skip the loop iteration

Ex:

```
cnt = 0
while cnt < 10:
    if cnt < 5:
        continue
    if cnt == 7:
        break
    print(cnt)
    cnt += 1
```

OUTPUT

It will print nothing,

Initially, the cnt will be zero. The program will come to the second line. It wil check the while condition. Since cnt < 10 is true, it will enter the while loop. It will check the if statement (cnt < 5). Since it is true, it will enter the if block and execute continue command.

Continue will take the program back to the while statement and will the run the program again with the value of cnt as zero. The program will be in an infinite loop.

If we add an increment after cnt < 5, then,

```
cnt = 0
while cnt < 10:
    if cnt < 5:
        cnt += 1
        continue
    if cnt == 7:
        break
    print(cnt)
    cnt += 1
```

<u>OUTPUT</u>: 5 6

NESTING LOOPS

<u>INPUT:</u>

```python
for i in range (10):
    cnt = 0
    while cnt < 10:
        print(cnt)
        cnt += 1
```

<u>OUTPUT</u>

```
0       0       0       0       0       0       0       0       0       0
1       1       1       1       1       1       1       1       1       1
2       2       2       2       2       2       2       2       2       2
3       3       3       3       3       3       3       3       3       3
4       4       4       4       4       4       4       4       4       4
5       5       5       5       5       5       5       5       5       5
6       6       6       6       6       6       6       6       6       6
7       7       7       7       7       7       7       7       7       7
8       8       8       8       8       8       8       8       8       8
9       9       9       9       9       9       9       9       9       9
```

The value of i is from 0 – 9. When i is 0, cnt will be 0. Since cnt < 10, it will print cnt. Then in the next line, the value of cnt is increased with 1. So, the value of cnt is 1. Since 1 < 10, it will print cnt again. This will continue until the value of cnt = 9.

Once it prints for the value of cnt = 9, the cnt will be increased by 1. Since 10 < 10 is false, the program will go back to the "***for statement***". And then the value of i in the for loop will become 1. The whole process in the while is repeated and this will go on unitl the value of i in the "***for statement***" is 9.

NOTE:

Break & Continue will always be executed for the nearest loop (check for indentations).

Ex: 01

```
cnt = 0
while cnt < 10:
    if cnt < 5:
        cnt += 1
        continue
    if cnt == 7:
        break
    print(cnt)
    cnt += 1
```

OUTPUT: **5 6**

Ex:

Given n print the sum of cube of all the number 1 – n

n = 3

(1 ^ 3) + (2 ^ 3) + (3 ^ 3) = 1 + 8 + 27 = 36

INPUT:

```
sum = 0
for i in range (1, n + 1):
        sum = i**3
print (sum)
```