Python Flow Control(contd.)

Iterative Statement(loop):
If we want to repeat some lines of code multiple times, then we use iterative statements or loops. In python, there are two types of loops: **for loop** and **while loop**.

- **for loop:**
  Sequence is compulsory in for loop and is used only when we know how many iterations are needed.
  Syntax:
  for x in sequence:
  body
  Here, x is a temporary variable as it is used only inside the for loop. It represents the elements in the sequence.
  The sequence is compulsory in for loop and can be list, set, tuple, dict, string and range. for loop iterates as many times as the length of sequence.

  Example 1:

```
s='python is fun'
for x in s:
 print(x)
```

  Output:

```
p
y
t
h
o
n

i
s

f
u
n
```

  Example 2: Odd numbers from 0 to 20

```
for x in range(20):
```

```
    if x%2!=0:
      print(x)
```

Output:

```
1
3
5
7
9
11
13
15
17
19
```

Example 3: Odd numbers from 0 to 20 printed in reverse

```
for x in range(20,0,-1):
  if x%2!=0:
    print(x)
```

Output:

```
19
17
15
13
11
9
7
5
3
1
```

Example 4: Sum of list elements

```
list= eval(input("Enter list: "))
sum=0
for x in list:
 sum+= x
print("The sum is ",sum)
```

Output:

```
Enter list: [10,20,30,40]
The sum is  100
```

- **while loop:**

Incase of a while loop, as long as the condition is fulfilled, the loop goes on iterating.

Syntax:

while condition
    body

Example 1: Natural numbers upto 5

```
x=1
while x<=5:
 print(x)
 x+=1
```

Output:

```
1
2
3
4
5
```

Example 2: Multiple of 3

```
x=1
while x<=20:
 if x%3==0:
  print(x)
 x+=1
```

Output:

```
3
6
9
12
15
18
```

Example 3: Sum of n natural numbers

```
n=int(input("Enter number: "))
sum =0
i=1
while i<=n:
 sum+=i
 i+=1
print(f"The sum of first {n} natural numbers is ",sum)
```

Output:

```
Enter number: 5
```

```
The sum of first 5 natural numbers is  15
```

Example 4: Entering value unless it's the one we want

```
name= input("Enter name: ")
while name!='Ram':
            name = input("Try another name: ")
print("Thanks for confirmation")
```

Output:

```
Enter name: shyam
Try another name: ram
Try another name: Ram
Thanks for confirmation
```

- **Infinite loops:**

  The loop that doesn't terminate is known as the infinite loop. Incase of while, if the condition is always true, then the loops turns to be infinite loop.

  Example:

```
i=0
while True:
 i+=1
 print("Hello ",i)
```

  In this program, the while loop goes on indefinitely as it is an infinite loop.

- **Nested Loops:**

  If a loop consists of another loop, then it is called nested loops.

  Example:

```
for i in range (3):
  for j in range(2):
     print(i,j)
```

  Output:

```
0 0
0 1
1 0
1 1
2 0
2 1
```

  Pattern problem:

```
n= int(input("Enter number of rows and columns: "))
for i in range(n):
```

```
    for j in range(n):
     print("*",end=' ')
    print(" ")
```

Output:

```
Enter number of rows and columns: 5
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
```

Transfer Statement:

It consists of break and continue statements. These are known as transfer statements because they transfer the flow of execution of the program. A prerequisite for these transfer statements is that it needs to be used only inside a loop.

- **break statement:**
  If we want to stop the execution of a loop even if the condition of the loop has not been met, we use a break statement to come out of the loop.
  Example 1:

```
for i in range(100):
 if i==10:
  print("I am done!")
  break
 print(i)
```

Output:

```
1
2
3
4
5
6
7
8
9
I am done!
```

Example 2:

```
l=[10,20,30,400,600,700,30,40,50]
for i in l:
 if i >=500:
```

```
    print("You cannot buy the item priced more than 500")
    break
  print(i)
print("Shopping completed.")
```

Output:

```
10
20
30
400
You cannot buy the item priced more than 500
Shopping completed.
```

- **continue statement:**
  If the current iteration needs to be skipped, then we use the continue statement.

  Example 1:

```
for i in range(10):
 if i==5:
  continue
 print(i)
```

Output:

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

Example 2:

```
l=[10,20,30,400,600,700,30,40,800,50]
for i in l:
 if i >=500:
    print("You cannot buy the item priced more than 500")
    continue
 print(i)
print("Shopping completed.")
```

Output:

```
10
```

```
20
30
400
You cannot buy the item priced more than 500
You cannot buy the item priced more than 500
30
40
You cannot buy the item priced more than 500
50
Shopping completed.
```

- **Loops with else block:**
  This is valid only in python. ‘else’ is part of the for/while loop and not ‘if’.

  Example:

```
l=[10,20,30,400,600,700,30,40,50]
for i in l:
 if i >=500:
   break
 print(i)
else:
 print("This is else part of for loop.")
print("Shopping completed.")
```

  Output:

```
10
20
30
400
Shopping completed.
```

- **pass:**
  If we need to make an empty block, pass acts as a placeholder.
  Example 1:

```
def hello():
 pass
a=2
```

  Here, the hello() function can be left empty by using pass as a placeholder.

  Example 2:

```
name = input("Enter name: ")
if name == "John":
```

```
    print("Name is correct")
else:
    pass
```

Output:

```
Enter name: Ram
```

- **del statement:**
  If we want to delete the reference of any object, we use del statement. The reference of the object is only deleted and not the object.
  Example 1:

```
a = 5
print(a)
del a
print(a)
```

Output:

```
5
Traceback (most recent call last):
  File
"/Users/ishhh/Downloads/Intern/Practice_python/session9.py", line
95, in <module>
    print(a)
          ^
NameError: name 'a' is not defined
```

Example 2:

```
s='Hello'
a=s
b=a
print(s)
print(a)
print(b)
del s
print(a)
print(b)
print(s)
```

Output:

```
Hello
Hello
Hello
Hello
Hello
```

```
Traceback (most recent call last):
  File
"/Users/ishhh/Downloads/Intern/Practice_python/session9.py", line
106, in <module>
    print(s)
          ^
NameError: name 's' is not defined
```

Difference between del and none:

Using del:

```
x = 10
del x
print(x)
```

Output:

```
Traceback (most recent call last):
  File
"/Users/ishhh/Downloads/Intern/Practice_python/session9.py", line
110, in <module>
    print(x)
          ^
NameError: name 'x' is not defined
```

But,

Using none:

```
x=10
x= None
print(x)
```

Output:

```
None
```

Assignment Link to following questions: (Redirect to Github)

1. Prime Number
2. Strong Number
3. Palindrome Number
4. Perfect Number
5. Armstrong Number
6. Fibonacci Number
7. Harshad(niven) number (ex: 18 = 9)
8. Twin number (ex: 11,13)
9. Triangular Number