Python Fundamentals day 11

Today's Agenda

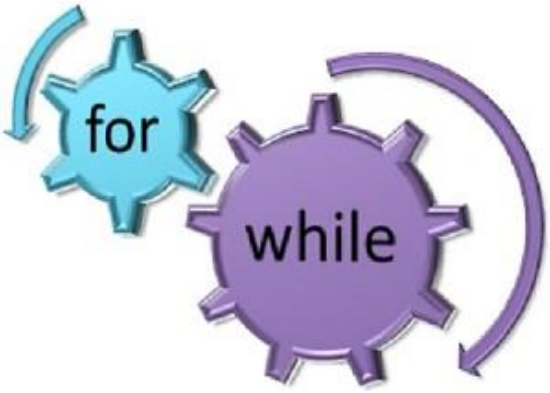
- Looping statements
- While loop
- For loop
- range()
- When to use for and while loop

Looping Statements

We can also change the flow of control by using looping statements. Previously we saw how some a part of code gets skipped by not satisfying the condition. Here we shall see how to make a certain section of code repeat itself for given number of times. This is only called as iteration or looping of statements.

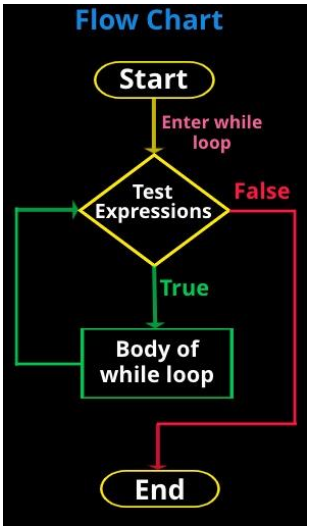
In python we have two types of looping statements

- while loop
- for loop



While loop

In while loop as long as the condition is satisfied the set of statements repeats its execution. And once the condition is false the control exits the loop and executes the rest of the code if any.



Syntax:

while <condition>:

body of the loop

Let us see an example

```
lst=[10,20,30,40,50]

i=0 #initialisation of variable
while i<5:  # condition check
    print(lst[i])
    i=i+1  #incrementation of i
```

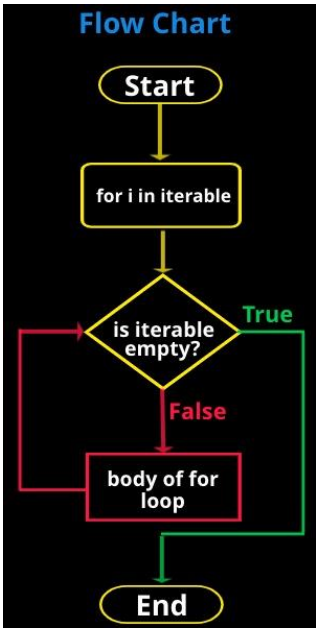
Output:

```
In [8]: runfile('C:/Users/rooman/OneDrive/
Desktop/python/test.py', wdir='C:/Users/rooman/
OneDrive/Desktop/python')
10
20
30
40
50
```

Note: initialisation, condition check and incrementation is necessary in while loop.

For loop

This is less like the for keyword in other programming languages, a for loop is used for iterating over a sequence (that is either a list, a tuple, a dictionary, a set, or a string).



Iterable is something which can work on/with for loop.

Syntax:

for var in iterable:

body of the loop

Let us see an example

```
lst=[10,20,30,40,50]

for i in lst:
    print(i)
```

Output:

```
In [9]: runfile('C:/Users/rooman/OneDrive/
Desktop/python/test.py', wdir='C:/Users/rooman/
OneDrive/Desktop/python')
10
20
30
40
50
```

Above example we see that no condition was checked and no initialisation is required.

range( )

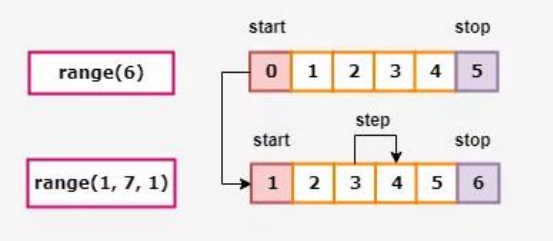
If you want to write for loop in traditional way, that is also possible in python. Where you can specify the number of times your loop should execute by using the function called as range().

Syntax: range(start, stop, step)

Start: Which position to start from. By default it is 0. //optional

Stop: Which position to stop at (not included). //required

Step: Number specifying incrementation. Default is 1. //optional



Let us see some examples

```
print(range(5))
```

Output:

```
In [1]: runfile('C:/Users/rooman/OneDrive/
Desktop/python/test.py', wdir='C:/Users/rooman/
OneDrive/Desktop/python')
range(0, 5)
```

Internally a range object is created which is ranging from 0 to 5.

Let us see it in list format

```
print(list(range(5)))
```

Output:

```
In [2]: runfile('C:/Users/rooman/OneDrive/
Desktop/python/test.py', wdir='C:/Users/rooman/
OneDrive/Desktop/python')
[0, 1, 2, 3, 4]
```

Now let us see range starting from non-zero element and with a different step

```
print(list(range(2,10,2)))
```

Output:

```
In [3]: runfile('C:/Users/rooman/OneDrive/
Desktop/python/test.py', wdir='C:/Users/rooman/
OneDrive/Desktop/python')
[2, 4, 6, 8]
```

When to use for and while loop

- While loop must be used when you are not sure how many times a loop should repeat itself. As long as the condition is true the loop must execute.
- For loop should be used when you are sure about the number of times the iterations should be performed. Here there is no condition checked.

Let us consider an example of banking where you don't know how many times you have to withdraw the money. As long as the min balance is less than actual balance the withdrawal should continue

```
balance=15500
min_balance=500

print("Balance before transaction:", balance)

while  min_balance< balance:
    balance=balance-1000

print("Balance after transaction:", balance)
```

Output:

```
In [6]: runfile('C:/Users/rooman/OneDrive/
Desktop/python/test.py', wdir='C:/Users/rooman/
OneDrive/Desktop/python')
Balance before transaction: 15500
Balance after transaction: 500
```

Above example we don't know how many times the loop should execute, but we know the condition. In such cases while loop should be used.

Let's say we know how much to withdraw, considering 5000rs is amount to be withdrawn

```
balance=15500
min_balance=500

print("Balance before transaction:", balance)

for i in range(5):
    balance=balance-1000

print("Balance after transaction:", balance)
```

Output:

```
In [7]: runfile('C:/Users/rooman/OneDrive/
Desktop/python/test.py', wdir='C:/Users/rooman/
OneDrive/Desktop/python')
Balance before transaction: 15500
Balance after transaction: 10500
```

Above case we know how many times a loop has to execute. And there is no condition to be satisfied. In such cases we use for loop.

Let us see the difference between while loop and for loop

while loop	for loop
1. Initialization of looping variable is required.	Initialization of looping variable is not required.
2. Looping variable need to be incremented/decremened.	Looping variable need not be incremented/decremened.
3. Condition should be checked	Condition need not be checked.
Use: When we are not sure how many iterations must be performed	Use: When we are sure how many iterations must be performed