Day -2

A program's **control flow** is the order in which the program's code executes.

The control flow of a Python program is regulated by conditional statements, loops, and function calls.

Python has *three* types of control structures:

- **Sequential** - default mode
- **Selection** - used for decisions and branching
- **Repetition** - used for looping, i.e., repeating a piece of code multiple times.

- **Sequential statements** are a set of statements whose execution process happens in a sequence. The problem with sequential statements is that if the logic has broken in any one of the lines, then the complete source code execution will break.

```python
## This is a Sequential statement

a=20
b=10
c=a-b
print("Subtraction is : ",c)
```

Some decision control statements are:

- `if`
- `if-else`
- `nested if`
- `if-elif-else`

`if` – It help us to run a particular code, but only when a certain condition is met or satisfied. A `if` only has one condition to check.

`if-else` – The `if-else` statement evaluates the condition and will execute the body of `if` if the test condition is `True`, but ifthe condition is `False`, then the body of `else` is executed.

Nested `if`: Nested `if` statements are an `if` statement inside another `if` statement.

```
a = 20
b = 10
c = 15
if a > b:
   if a > c:
      print("a value is big")
   else:
       print("c value is big")
elif b > c:
    print("b value is big")
else:
     print("c is big")
```

Output

```
a value is big
```

```
x = 15
y = 12
if x == y:
   print("Both are Equal")
elif x > y:
    print("x is greater than y")
else:
    print("x is smaller than y")
```

Output

```
x is greater than y
```

3. Repetition

A **repetition statement** is used to repeat a group(block) of programming instructions.

In Python, we generally have two loops/repetitive statements:

- `for` loop
- `while` loop

`for` loop – A `for` loop is used to iterate over a sequence that is either a list, tuple, dictionary, or a set. We can execute a set of statements once for each item in a list, tuple, or dictionary.

```
print("1st example")

lst = [1, 2, 3]
for i in range(len(lst)):
     print(lst[i], end = " \n")

print("2nd example")

for j in range(0,5):
    print(j, end = " \n")
```

Output

```
1st example 1 2 3

2nd

example 0 1 2 3
```

```
m = 5
i = 0
while i < m:
     print(i, end = " ")
     i = i + 1
print("End")
```

Output

```
0 1 2 3 4 End
```

Logical operators

In Python, Logical operators are used on conditional statements (either True or False). They perform **Logical AND**, **Logical OR** and **Logical NOT** operations.

OPERATOR	DESCRIPTION	SYNTAX
and	Logical AND: True if both the operands are	x and y

OPERATOR	DESCRIPTION	SYNTAX
	true	
or	Logical OR: True if either of the operands is true	x or y
not	Logical NOT: True if operand is false	not x

Logical AND operator in Python

```
a = 10
b = 10
c = -10

if a > 0 and b > 0:
    print("The numbers are greater than 0")

if a > 0 and b > 0 and c > 0:
    print("The numbers are greater than 0")
else:
    print("Atleast one number is not greater than 0")
```

Output

```
The numbers are greater than 0

Atleast one number is not greater than 0
```

```
a = 10
b = 12
c = 0

if a and b and c:
    print("All the numbers have boolean value as True")
else:
```

```python
    print("Atleast one number has boolean value as False")
```

Output

```
Atleast one number has boolean value as False
```

Logical OR operator in Python

```python
a = 10
b = -10
c = 0

if a > 0 or b > 0:
    print("Either of the number is greater than 0")
else:
    print("No number is greater than 0")

if b > 0 or c > 0:
    print("Either of the number is greater than 0")
else:
    print("No number is greater than 0")
```

Output

```
Either of the number is greater than 0

No number is greater than 0
```

Logical not operator in Python

```python
a = 10

if not a:
    print("Boolean value of a is True")

if not (a%3 == 0 or a%5 == 0):
    print("10 is not divisible by either 3 or 5")
else:
    print("10 is divisible by either 3 or 5")
```

Output

```
10 is divisible by either 3 or 5
```

What is randomisation in Python?

The Python Random module is a built-in module for generating random integers in Python. These numbers occur randomly and does not follow any rules or instructuctions. We can therefore use this module to generate random numbers, display a random item for a list or string, and so on.

```
import random
r1 = random.randint(5, 15)
print("Random number between 5 and 15 is % s" % (r1))
r2 = random.randint(-10, -2)
print("Random number between -10 and -2 is % d" % (r2))
```

Output:

```
Random number between 5 and 15 is 7
Random number between -10 and -2 is -9
```

List

```
List = []
print("Blank List: ")
print(List)

# Creating a List of numbers
List = [10, 20, 14]
print("\nList of numbers: ")
print(List)

# Creating a List of strings and accessing
# using index
List = ["Geeks", "For", "Geeks"]
print("\nList Items: ")
print(List[0])
print(List[2])
```

Output

```
Blank List:

[]


List of numbers:

[10, 20, 14]
```

```
List Items:
Geeks
Geeks
```

2nd code

```python
List = [1, 2, 'Geeks', 4, 'For', 6, 'Geeks']
print("\nList with the use of Mixed Values: ")
print(List)
```

Output

```
List with the use of Numbers:
[1, 2, 4, 4, 3, 3, 3, 6, 5]


List with the use of Mixed Values:
[1, 2, 'Geeks', 4, 'For', 6, 'Geeks']
```