UNIT-2

LOOPING, CONDITIONAL & CONTROL STATEMENTS

- The statements in the program are executed one by one by Python interpreter. This type of execution is called **'sequential execution'**.
- **Decision Making** is required when we want to execute a code only if certain conditions are satisfied.

CONTROL STATEMENTS

- Control or change the flow of execution. **(Consider the PPT as well)**
    1. If Statements
    2. If...else Statements
    3. If...elif...else statement
    4. While loop
    5. For loop
    6. Else suite
    7. Break statement
    8. Continue statement
    9. Pass statement
    10. Assert statement
    11. Return statement

    NOTE: Switch statement is not available in Python.

1. IF STATEMENT

- The group of statements in Python is called a **suite**.

Example:

#Python program to display a digit in words

```
num=1
if num==1:
   print("ONE")
```

 # To display a group of messages when condition is true

```
str='Yes'
if str=='Yes':
   print("Yes")
```

```
print("This is Python Session")
print("Control Structures")
```

A Word on Indentation

- Refers to spaces that are used in the beginning of a statement.

Example:

```
if x==1:
   print('a')
   print('b')
   if y==2:
      print('c')
      print('d')
print('end')
```

2. IF ELSE STATEMENT

Example:

```
# To check number is even or odd
x=16
if x%2==0:
   print(x,"is even number")
else:
   print(x,"is odd number")
```

3. IF...ELIF...ELSE STATEMENT

Example:

```
# To check number is zero, positive or negative
num=-17
if num==0:
   print(num,"is zero")
elif num>0:
   print(num,"is positive")
else:
   print(num,"is negative")
```

4. THE WHILE LOOP

- Useful to execute a group of statements several times repeatedly depending on the whether a condition is True or False.

Example:
 # To display number from 1 to 10

```
x=1
while x<=10 :
   print(x)
   x+=1
print("Finish")
```

5. THE FOR LOOP

- Called **iterators.**
- Iterating over a sequence is called **traversal.**

Example:

 # To display characters of string

```
str="Hello"
n=len(str)
print(n)
for ch in str:
   print(ch)
```

#Difference between while and for loop

```
#with while loop
x=0
while(x<5):
   print(x)
   x=x+1;

#with For loop
for x in range(0,5):
   print(x)
```

The range function ( )

- Range(10) –generate 0 to 9
- **range (start, stop, step_size) –** step_size defaults to 1 if not provided.

Example:

```
print(range(10))
print(list(range(10)))
print(list(range(2,8)))
print(list(range(20,40,2)))
```

 # Program to iterate through a list using indexing

```
genre=['pop','rock','jazz']
for i in range(len(genre)):      #iterate over the list using index
   print("I like",genre[i])

list=[10,20, 30, 'A', 'Amit']
for element in list:
   print(element)
```

#For Loop for string

```
Months=["Jan", "Feb", "March", "April", "May"]
for m in Months:
   print(m)
```

INFINITE LOOPS

- It is a loop that executes forever.
- Press **Control+C** at system prompt to stop the program.
- Avoid infinite loops in any program.

Example 1:

```
i=1
while i<=10:
        print(i)
```

Example 2:

```
while(True):
        print("Good Morning")
```

NESTED LOOPS

- One loop inside another loop.

Example:

```
for i in range(3):
    for j in range(4):
        print('i=', i, '\t','j=',j)
```

 # To display stars in right angled triangular format

```
for i in range (1,11):
    for j in range(1,i+1): # no of stars=row number
        print('* ', end='')
print()
```

6. The ELSE SUITE

Syntax: For loop with else

```
For (var in sequence):
    Statements
Else:
    Statements
```

Syntax: While loop with else

```
While(condition):
    Statements
Else:
    Statements
```

Example:

```
for i in range(5):
   print("Yes")
else:
   print("No")
```

7. THE BREAK STATEMENT

- Used to come out of the loop.

Example:

```
x=10
while x>=1:
   print('x= ',x)
   x-=1
   if x==5:
      break
print("End of the Loop")
```

 # Use of break statement inside the loop

```
for val in "string":
    if val=="i":
        break
    print(val)
print("End of the Program")
```

#Another example

```
for x in range(10,20):
   if(x==15):
      break
   print(x)
```

8. CONTINUE STATEMENT

- Used in a loop to go back to the beginning of the loop.
- Used to skip the rest of the code inside a loop for the current iteration only.
- Loop does not terminate but continues on with the next iteration.

Example 1:

```
x=0
while(x<10):
   x=x+1
   if x>5:
      continue
   print('x= ',x)
print("End of the Loop")
```

Example 2:

```
for val in "string":
   if val=="i":
      continue
   print(val)
print("End of the Program")
```

How to use "Continue statement" in For Loop

```
for x in range(10,20):
   if(x%5==0):continue
   print(x)
```

What is enumerate() in python?

- Built-in function
- Used for assigning an index to each item of the iterable object.
- This object can be used in a for loop to convert it into a list by using list() method.
- Used for the numbering or indexing the members in the list.

Example:

```
#Use of a for loop over a collection
Months=["Jan", "Feb","March", "April", "May", "June"]
for i,m in enumerate(Months):
   print(i,m)
```

How to use for loop to repeat the same statement over and again?

```
for i in '123':
   print("Hello",i)
```

9. THE PASS STATEMENT

- Does not to nothing.
- To represent no operation(NOP).
- Null Statement.
- **Syntax:** pass
- As a placeholder inside loops, functions, class if-statement that is meant to be implemented later.

Example:

```
x=0
while(x<10):
   x+=1
   if x>5:
      pass
   print('x= ',x)
print("End of the Program")

#pass is just a placeholder for functionality to be added later
sequence = {'p','a','s','s'}
for val in seqience:
   pass
```

Difference between Break and Continue Statement

- When break is encountered, it will exit the loop.
- Continue, the current iteration that is running will be stopped, and it will proceed with next iteration.

10.THE ASSERT STATEMENT

- Useful to check if a particular condition is fulfilled or not

Syntax: assert expression, message

Example:

```
x=int(input("Enter a number greater than 0:"))
assert x>0, "Wrong Input Entered By the USER"
print("You Entered :",x)
```

- The **'Assertion Error'** shown in the output is called an **exception.**
- An exception is an error that occurs during runtime.
- To avoid such type of exceptions, we can handle them using **'try..except' statement.**
- After 'try', we use 'assert' and after 'except', we write the statements that are executed in case of exception.

Example:

```
x=int(input("Enter a number greater than 0:"))
try:
   assert x>0, "Wrong Input Entered By the USER"
   print("You Entered :",x)
except AssertionError:
   print("Wrong Input Entered")
```

11.THE RETURN STATEMENT

- A function starts with the keyword **def** that represents the definition of the function.

  Example:

```
def sum(a,b):
        print(a,b)
```

  We can call this function as: sum(5,2)

- Return statement is used inside a function to return some value to the calling place.

  Syntax: return expression

  Example:

```
#A function to return sum of two numbers
def sum(a,b):
        return a+b

result=sum(5,2)
print("The result is: ", result)
```