PPS UNIT 3:

CONTROL FLOW, FUNCTIONS

Conditional Statements:

Conditional statements are used in programming to control the flow of a program.

There are three main types of conditional statements in Python:

I. Conditional Statements

1) if

if condition : statement

OR

if condition :
statement-1
statement-2
statement-3

If condition is true then statements will be executed.

```
x = 10        #Initialize the value of x
if(x>0):      #test the value of x
   x = x+1      #Increment the value of x if it is > 0
print(x)       #Print the value of x
```

OUTPUT

```
x = 11
```

2) if-else:

```
if condition:
    Action-1
else:
    Action-2
```

if condition is true then Action-1 will be executed otherwise Action-2 will be executed.

```
age = int(input("Enter the age : "))
if(age>=18):
   print("You are eligible to vote")
else:
        yrs = 18 - age
        print("You have to wait for another " + str(yrs) +" years to cast your vote")
```

OUTPUT

```
Enter the age : 10
You have to wait for another 8 years to cast your vote
```

3) if-elif-else:

```
if condition1:
        Action-1
elif condition2:
        Action-2
elif condition3:
        Action-3
elif condition4:
        Action-4
       ...
else:
     Default Action
```

Based condition the corresponding action will be executed.

The if-elif-else construct works in the same way as a usual if-else statement. **if-elif-else** construct is also known as **nested-if** construct.

```
num = int(input("Enter any number : "))
if(num==0):
    print("The value is equal to zero")
elif(num>0):
    print("The number is positive")
else:
    print("The number is negative")
```

OUTPUT

```
Enter any number : -10
The number is negative
```

Q) Write a Program to find Biggest of given 2 Numbers from the Commad Prompt?

```
1) n1=int(input("Enter First Number:"))
2) n2=int(input("Enter Second Number:"))
3) if n1>n2:
4)     print("Biggest Number is:",n1)
5) else :
6)     print("Biggest Number is:",n2)
```

Q) Write a Program to find Biggest of given 3 Numbers from the Commad Prompt?

```
1) n1=int(input("Enter First Number:"))
2) n2=int(input("Enter Second Number:"))
3) n3=int(input("Enter Third Number:"))
4) if n1>n2 and n1>n3:
5)     print("Biggest Number is:",n1)
6) elif n2>n3:
7)     print("Biggest Number is:",n2)
8) else :
9)     print("Biggest Number is:",n3)
```

Home work:

Q) Write a program to find smallest of given 2 numbers?
Q) Write a program to find smallest of given 3 numbers?
Q) Write a program to check whether the given number is even or odd?

Q) **Write a Program to take a Single Digit Number from the Key Board and Print is Value in English Word?**

```
1) 0 → ZERO
2) 1 → ONE
3)
4) n=int(input("Enter a digit from o to 9:"))
5) if n==0 :
6)     print("ZERO")
7) elif n==1:
8)     print("ONE")
9) elif n==2:
10)    print("TWO")
11) elif n==3:
12)    print("THREE")
13) elif n==4:
14)    print("FOUR")
15) elif n==5:
16)    print("FIVE")
17) elif n==6:
18)    print("SIX")
19) elif n==7:
20)    print("SEVEN")
21) elif n==8:
22)    print("EIGHT")
23) elif n==9:
24)    print("NINE")
25) else:
26)    print("PLEASE ENTER A DIGIT FROM 0 TO 9")
```

Iteration Statements (Loops)

Iteration statements are used to execute a piece of code a certain number of times or until a certain condition is met.
There are two types of iteration statements in Python:

- The “for” loop
- The “while” loop

1) for loop:

If we want to execute some action for every element present in some sequence (it may be string or collection) then we should go for for loop.

Syntax:
```
for x in sequence:
    Body
```

Where sequence can be string or any collection.
Body will be executed for every element present in the sequence.

Example 1

To print Hello 10 times

```
for x in range(10) :
    print("Hello")
```

Example 2

To display numbers from 0 to 10

```
for x in range(11) :
    print(x)
```

Example 3

To display odd numbers from 0 to 20

```
for x in range(21) :
    if (x%2!=0):
        print(x)
```

2) while loop:

If we want to execute a group of statements iteratively until some condition false,then we should go for while loop.

Syntax:
```
while condition :
    body
```

Example 1

To print numbers from 1 to 10 by using while loop

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

Example 2

display the sum of first n numbers

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

Using break statement:

1) break:

We can use break statement inside loops to break loop execution based on some condition.

```
1)  for i in range(10):
2)      if i==7:
3)          print("processing is enough..plz break")
4)          break
5)      print(i)
```

```
D:\Python_classes>py test.py
0
1
2
3
4
5
6
processing is enough..plz break
```

2) continue:

We can use continue statement to skip current iteration and continue next iteration.

Eg 1: To print odd numbers in the range 0 to 9

```
1)  for i in range(10):
2)      if i%2==0:
3)          continue
4)      print(i)
```

```
D:\Python_classes>py test.py
1
3
5
7
9
```

3) pass statement:

- pass is a keyword in Python.
- In our programming syntactically if block is required which won't do anything then we can define that empty block with pass keyword.

pass
- |- It is an empty statement
- |- It is null statement
- |- It won't do anything

Eg: if True:

SyntaxError: unexpected EOF while parsing

if True: pass → valid

The Pass Statement Example:

```
for letter in "HELLO":
        pass      #The statement is doing nothing
        print("Pass : ", letter)
print("Done")
```

OUTPUT

```
Pass :  H
Pass :  E
Pass :  L
Pass :  L
Pass :  O
Done
```

Functions in Python

If a program contains a large piece of code that must be run repeatedly, it is preferable to implement that code as a function and then call it using a loop. Functions promote code reuse, modularity, and integrity.

Consider the following scenario: you must add two numbers 100 times. You will have to assign values to two variables 100 times, perform the addition, and print the result on the console if you don't use a function. If you're asked to conduct subtraction, you'll have to change the plus sign back and forth 100 times. It is more convenient in this case to write a function that accepts two numbers and performs addition between them. The function can then be invoked inside a loop. Similarly, if you want to move from addition to subtraction, you must do so in one position.

1) Built in Functions:

The functions which are coming along with Python software automatically, are called built in functions or pre defined functions.

Eg: id()
type()
input()
eval()
etc..

2) User Defined Functions:

The functions which are developed by programmer explicitly according to business requirements, are called user defined functions.

Syntax to Create User defined Functions:

```
def   function_name(parameters) :
      """ doc string"""
            ----
            -----
      return value
```

Eg 1: Write a function to print Hello

test.py

```
1) def wish():
2)     print("Hello Good Morning")
3) wish()
```

Parameters

Parameters are inputs to the function. If a function contains parameters, then at the time of calling,compulsory we should provide values otherwise,otherwise we will get error.

Eg: Write a function to take number as input and print its square value

```
1) def squareIt(number):
2)     print("The Square of",number,"is", number*number)
3) squareIt(4)
4) squareIt(5)
```

Return Statement:

Function can take input values as parameters and executes business logic, and returns output to the caller with return statement.

Q) Write a Function to accept 2 Numbers as Input and return Sum

```
1) def add(x,y):
2)     return x+y
3) result=add(10,20)
4) print("The sum is",result)
5) print("The sum is",add(100,200))
```

Q) Write a Function to check whether the given Number is Even OR Odd?

```
1) def even_odd(num):
2)     if num%2==0:
3)         print(num,"is Even Number")
4)     else:
5)         print(num,"is Odd Number")
6) even_odd(10)
7) even_odd(15)
```

Q) Write a Function to find Factorial of given Number?

```
1) def fact(num):
2)     result=1
3)     while num>=1:
4)         result=result*num
5)         num=num-1
6)     return result
7) for i in range(1,5):
8)     print("The Factorial of",i,"is :",fact(i))
```

Returning Multiple Values from a Function:

In other languages like C,C++ and Java, function can return atmost one value. But in Python, a function can return any number of values.

Eg 1:

```
1) def sum_sub(a,b):
2)     sum=a+b
3)     sub=a-b
4)     return sum,sub
5) x,y=sum_sub(100,50)
6) print("The Sum is :",x)
7) print("The Subtraction is :",y)
```

Eg 2:

```
1)  def calc(a,b):
2)      sum=a+b
3)      sub=a-b
4)      mul=a*b
5)      div=a/b
6)      return sum,sub,mul,div
7)  t=calc(100,50)
8)  print("The Results are")
9)  for i in t:
10)     print(i)
```

Types of Variables

Python supports 2 types of variables.

1) Global Variables
2) Local Variables

1) Global Variables

- The variables which are declared outside of function are called global variables.
- These variables can be accessed in all functions of that module.

```
1) a=10  # global variable
2) def f1():
3)     print(a)
4)
5) def f2():
6)     print(a)
7)
8) f1()
9) f2()
```

Output

```
10
10
```

2) Local Variables:

- The variables which are declared inside a function are called local variables.
- Local variables are available only for the function in which we declared it.i.e from outside of function we cannot access.

```
1)  def f1():
2)      a=10
3)      print(a)    # valid
4)
5)  def f2():
6)      print(a)     #invalid
7)
8)  f1()
9)  f2()
```

NameError: name 'a' is not defined

Note: If global variable and local variable having the same name then we can access global variable inside a function as follows

```
1)  a = 10 → Global Variable
2)  def f1():
3)      a=777 → Local Variable
4)      print(a)
5)      print(globals()['a'])
6)  f1()
```

Output

777
10

Recursive Functions

A function that calls itself is known as Recursive Function.

Eg:

```
factorial(3) = 3 * factorial(2)
             = 3 * 2 * factorial(1)
             = 3 * 2 * 1 * factorial(0)
             = 3 * 2 * 1 * 1
             = 6
factorial(n) =  n * factorial(n-1)
```

The main advantages of recursive functions are:

1) We can reduce length of the code and improves readability.
2) We can solve complex problems very easily.

Q) Write a Python Function to find Factorial of given Number with Recursion

```
1)  def  factorial(n):
2)      if n==0:
3)          result=1
4)      else:
5)          result=n*factorial(n-1)
6)      return result
7)  print("Factorial of 4 is :",factorial(4))
8)  print("Factorial of 5 is :",factorial(5))
```

Output

```
Factorial of 4 is : 24
Factorial of 5 is : 120
```

Function Composition in Python

Function composition is the way of combining two or more functions in such a way that the output of one function becomes the input of the second function and so on. For example, let there be two functions “F” and “G” and their composition can be represented as F(G(x)) where “x” is the argument and output of G(x) function will become the input of F() function.

Example:

```
# Function to add 2
# to a number
def add(x):
    return x + 2

# Function to multiply
# 2 to a number
def multiply(x):
    return x * 2

# Printing the result of
# composition of add and
# multiply to add 2 to a number
# and then multiply by 2
print("Adding 2 to 5 and multiplying the result with 2: ",
      multiply(add(5)))
```

Output:

```
Adding 2 to 5 and multiplying the result with 2: 14
```

Explanation

First the `add()` function is called on input 5. The `add()` adds 2 to the input and the output which is 7, is given as the input to `multiply()` which multiplies it by 2 and the output is 14

Strings: string slices:

Python slice string syntax is:

```
str_object[start_pos:end_pos:step]
```

The slicing starts with the start_pos index (included) and ends at end_pos index (excluded). The step parameter is used to specify the steps to take from start to end index. Python String slicing always follows this rule: **s[:i] + s[i:] == s** for any index 'i'. All these parameters are optional - start_pos default value is 0, the end_pos default value is the length of string and step default value is 1. Let's look at some simple examples of string slice function to create substring.

```
s = 'HelloWorld'


print(s[:])


print(s[::])
```

Output:

```
HelloWorld
HelloWorld
```

Note that since none of the slicing parameters were provided, the substring is equal to the original string. Let's look at some more examples of slicing a string.

```
s = 'HelloWorld'
first_five_chars = s[:5]
print(first_five_chars)

third_to_fifth_chars = s[2:5]
print(third_to_fifth_chars)
```

Output:

```
Hello
llo
```

Note that index value starts from 0, so start_pos 2 refers to the third character in the string.

Reverse a String using Slicing

We can reverse a string using slicing by providing the step value as -1.

```
s = 'HelloWorld'
reverse_str = s[::-1]
print(reverse_str)
```

Output: `dlroWolleH`

Let's look at some other examples of using steps and negative index values.

```
s1 = s[2:8:2]
print(s1)
```

Output: `loo` Here the substring contains characters from indexes 2,4 and 6.

```
s1 = s[8:1:-1]
print(s1)
```

Output: `lroWoll` Here the index values are taken from end to start. The substring is made from indexes 1 to 7 from end to start.

Some additional important Programs:

```
#Name      Roll No-K
#Python Program to convert temperature in celsius to fahrenheit

celsius=float(input("Enter value in celsius:"))
fahrenheit = (celsius * 1.8) + 32
print("Temp in fahrenheit=",fahrenheit)
```

```
Enter value in celsius:35.5
Temp in fahrenheit= 95.9
```

```
# prime number
#num=20
num=int(input("Enter the number:"))
flag=False
if(num>1):
  for i in range(2,num):
    if(num%i==0):
      flag=True
      break
if flag:
  print("Number is not prime number")
else:
  print("Number prime number")
```

```
Enter the number:20
True
Number is not prime number
```

```python
#Fibonacci series
#nterms=8
nterms=int(input("Enter nterms:"))
n1=0
n2=1
count=0
while(count<nterms):
  print(n1)
  n3=n1+n2
  #update the values of n1 & n2
  n1=n2
  n2=n3
  count=count+1
```

```
Enter nterms:8
0
1
1
2
3
5
8
13
```