LESSON 18

LOCAL AND GLOBAL VARIABLES

TOPICS COVERED

- WHAT IS SCOPE?
- VARIABLE SCOPE
- TYPES OF VARIABLE SCOPE
- LOCAL SCOPE
- GLOBAL SCOPE
- GLOBAL KEYWORD
- CALLING FUNCTIONS

LEARNING CONTENT

WHAT IS SCOPE?

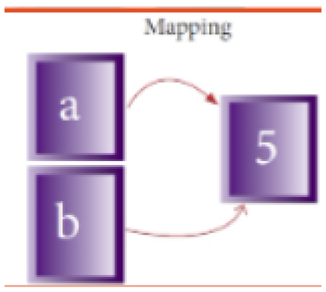
- Scope refers to the visibility of variables, parameters and functions in one part of a program to another part of the same program.
- Every variable defined in a program has global scope.
- Once defined, every part of the program can access that variable. But it is a good practice to limit a variable's scope to a single definition.

- This ensures that changes inside the function can't affect the variable on the outside of the function in unexpected ways.

Example:

a:=5

b:=a



WHAT IS VARIABLE SCOPE?

- Variables are addresses to an object in memory.
- When assigning a variable with := to an instance (object), we are binding (or mapping) the variable to that instance.
- Multiple variables can be mapped to the same instance.
- The process of binding a variable name with an object is called mapping.
- **: =** (colon equal to sign) is used in programming languages to map the variable and object.
- Programming languages keeps track of all the mappings with namespaces.
- Namespaces are containers for mapping names of variables to objects.

TYPES OF VARIABLE SCOPE:

- There are 4 types of variable scopes namely:

Local(**L**)	Defined inside function/class
Enclosed(**E**)	Defined inside enclosing functions (Nested function concept)
Global(**G**)	Defined at the uppermost level
Built-in (**B**)	Reserved names in built-in functions (modules)

- Out of these, local and global are widely used and we shall learn about them in detail.

LOCAL SCOPE:

- A variable created inside a function belongs to the local scope of that function, and can only be used inside that function.
- Local scope refers to variables defined in current function.
- A function will first look up for a variable name in its local scope. Only if it does not find it there, the outer scopes are checked.
- When we try to access variables with local scope outside the defined function, the python interpreter throws an error.

PROGRAM:

```
def myfunc():
   x = 300
   print(x)
myfunc()
print(x)
```

OUTPUT:

```
300
Traceback (most recent call last):
  File "C:/Users/Hp/OneDrive/Desktop/ed.py", line 6, in <module>
    print(x)
```

NameError: **name 'x'** is not defined

GLOBAL SCOPE:

- A variable which is declared outside of all the functions in a program is known as global variable. **Global variable** can be accessed inside or outside of all the functions in a program.
- A variable created in the main body of the Python code is a global variable and belongs to the global scope.
- **EXAMPLE**:

```
x = 300
def myfunc():
   print(x)
myfunc()
print(x)
```

OUTPUT:

```
300
300
```

GLOBAL KEYWORD:

- Global keyword helps to modify the variable outside of the current scope.
- It is used to create a global variable and make changes to the variable in a local context.

RULES OF GLOBAL KEYWORD:

- When we define a variable outside of a function, it is global by default. No need to use global keyword.

- We use global keyword to read and write a global variable inside a function.
- Use of global keyword outside a function has no effect.

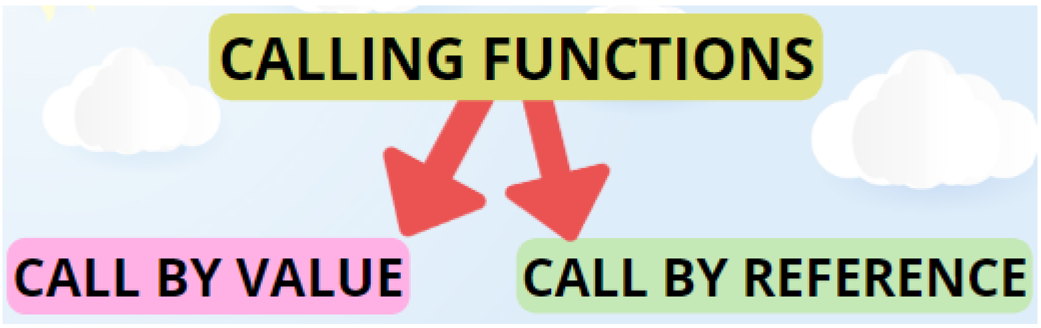
CALLING FUNCTIONS:

- If we declare the same variable name inside and outside of a function, Python will treat them as two separate variables, that is, one is available in the global scope (outside the function) and one is available in the local scope (inside the function).
- **EXAMPLE**:

```
x = 300
def myfunc():
   x = 200
   print(x)
myfunc()
print(x)
```

OUTPUT:

```
200
300
```



CALL BY VALUE-

It refers to the way of passing arguments to a function in which the arguments get copied to the formal parameters of a function and are stored in different memory locations.

Any changes made within the function are not reflected in the actual parameters of the function when called.
When immutable objects such as whole numbers, strings, etc are passed as arguments to the function call, it can be considered as Call by Value.

When the values are modified within the function, then the changes do not get reflected outside the function.

EXAMPLE:

```
def myFunc(a):
 print("\t Value received in 'a' =", a)
 a+=2
 print("\tValue of 'a' changes to :",a)
num=13
print("Initial number: ", num)
myFunc(num)
print("Value of number= ", num)
```

OUTPUT:

```
Initial number: 13
Value received in 'a'=13
Value of 'a' changes to: 15
Value of number=13
```

CALL BY REFERENCE-

It is a way of passing arguments to a function call in which both the actual argument and formal parameters refer to the same memory locations.

Changes made within the function are reflected in the actual parameters of the function when called.

When mutable objects such as list, dict, set, etc are passed as arguments to the function call, it can be considered as Call by reference in Python.
When the values are modified within the function then the change also gets reflected outside the function.

EXAMPLE:

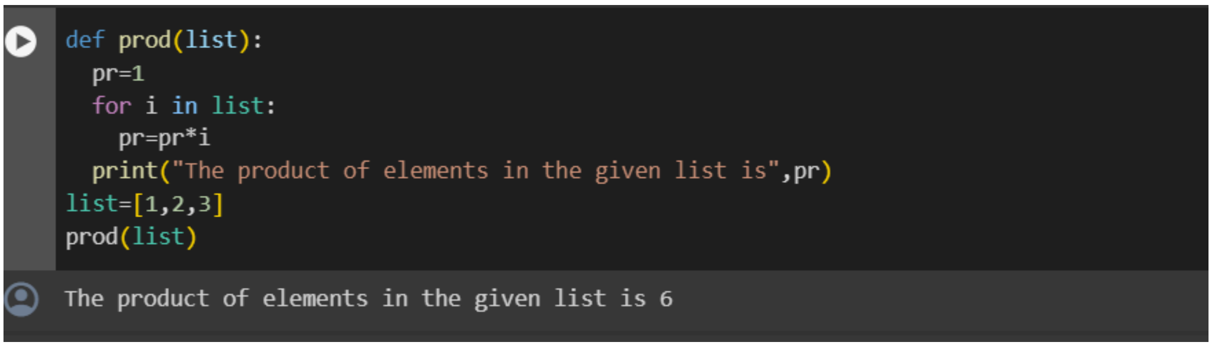
```
def myFunc(myList):
print("\t List received: ",myList)
myList.append(3)
    myList.extend([7,1])
print("\t List after adding some elements:", myList)
myList.remove(7)
print("\t List within called function:", myList)
return
List1=[1]
print("List before function call :",List1)
myFunc(List1)
print("\t List after function call: ",List1)
```

IN CLASS CHALLENGES

CHALLENGE1-CHIRAGS BIG PROBLEM:

Chirag was given a list containing n numbers. His teacher asked him to find the product of all the elements present in the list. He did it perfectly when the list was small. As the size of the next lists were becoming larger, he found it difficult to calculate the product. He thought of creating a python program that gets a list as argument and calculates the product and displays the result. But he is not good at function concepts. Help him to create a function to do the above.

SOLUTION:



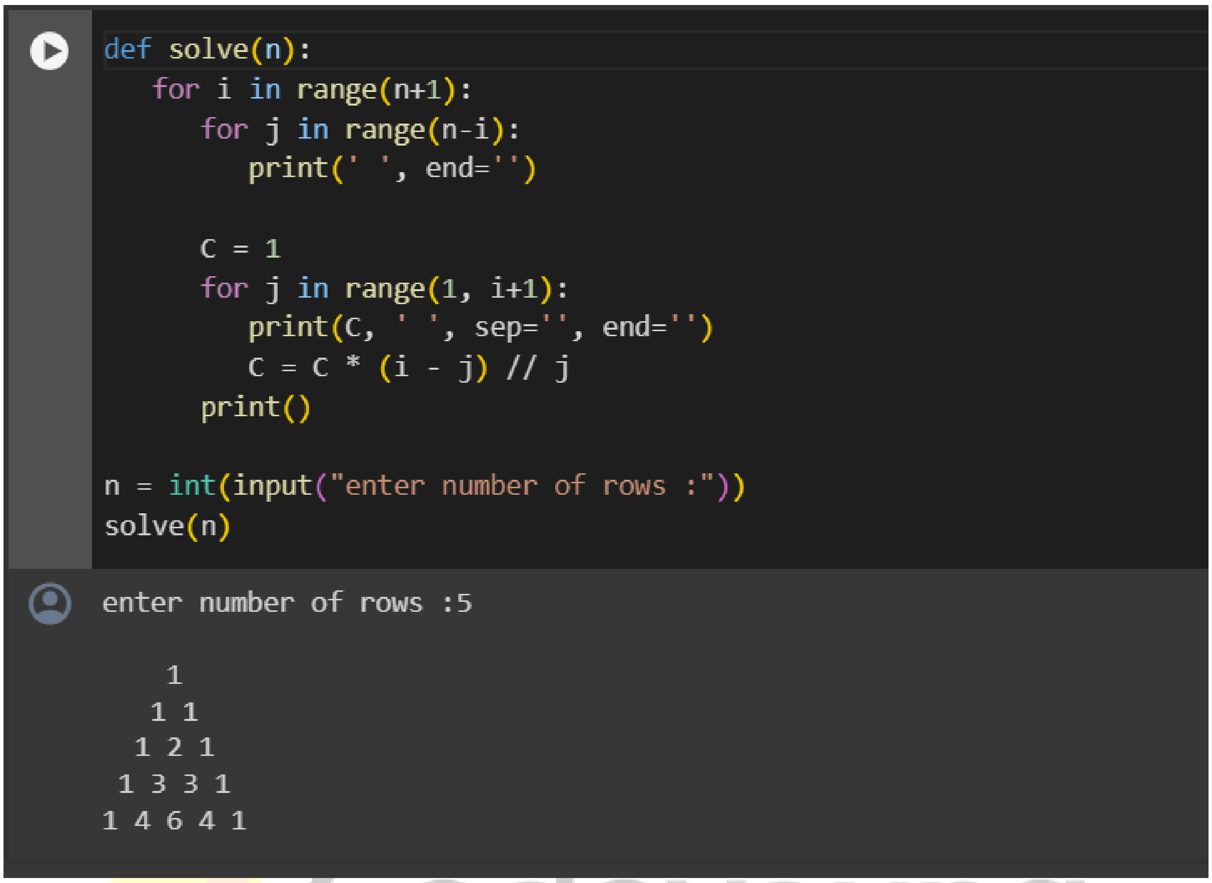
```
def prod(list):
  pr=1
  for i in list:
    pr=pr*i
  print("The product of elements in the given list is",pr)
list=[1,2,3]
prod(list)
```

```
The product of elements in the given list is 6
```

CHALLENGE 2- PASCALS TRIANGLE:

Sahil came to know about Pascal's triangle. He tries the small patterns of pascal's triangle. But he was not sure how the triangle looks when the number of rows were increased. Help him to create a function which gets the number of rows as input from the user and prints the pascal's triangle upto that row.

SOLUTION-



```
def solve(n):
   for i in range(n+1):
      for j in range(n-i):
         print(' ', end='')

      C = 1
      for j in range(1, i+1):
         print(C, ' ', sep='', end='')
         C = C * (i - j) // j
      print()

n = int(input("enter number of rows :"))
solve(n)
```

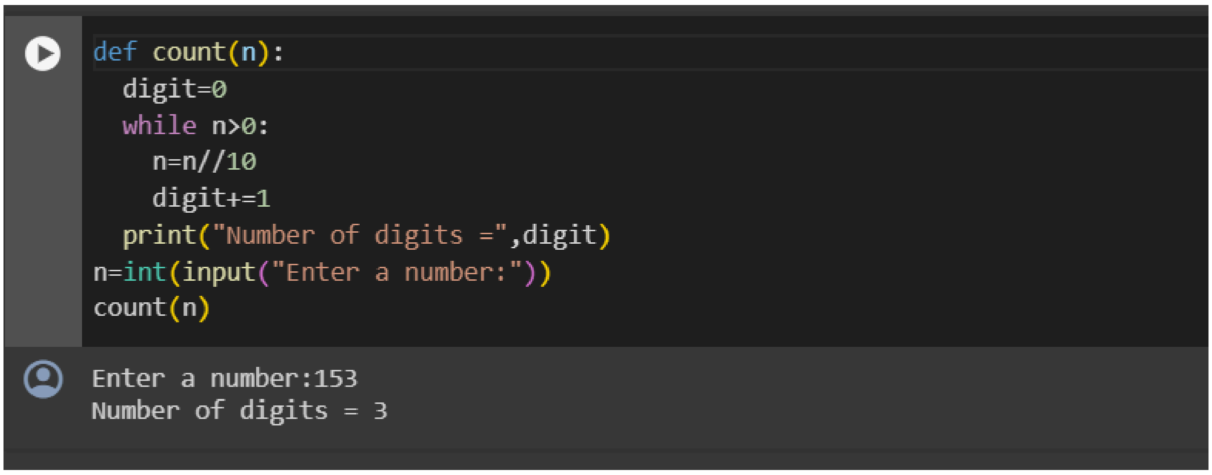
```
enter number of rows :5

    1
   1 1
  1 2 1
 1 3 3 1
1 4 6 4 1
```

<u>**HOME TASK**</u>

<u>TASK1- NUMBER OF DIGITS:</u>

Nidhi is writing a program to count the number of digits in a number given by user. But she got stuck in the middle. Help her to create a function that counts the number of digits in a given number.

<u>SOLUTION:</u>



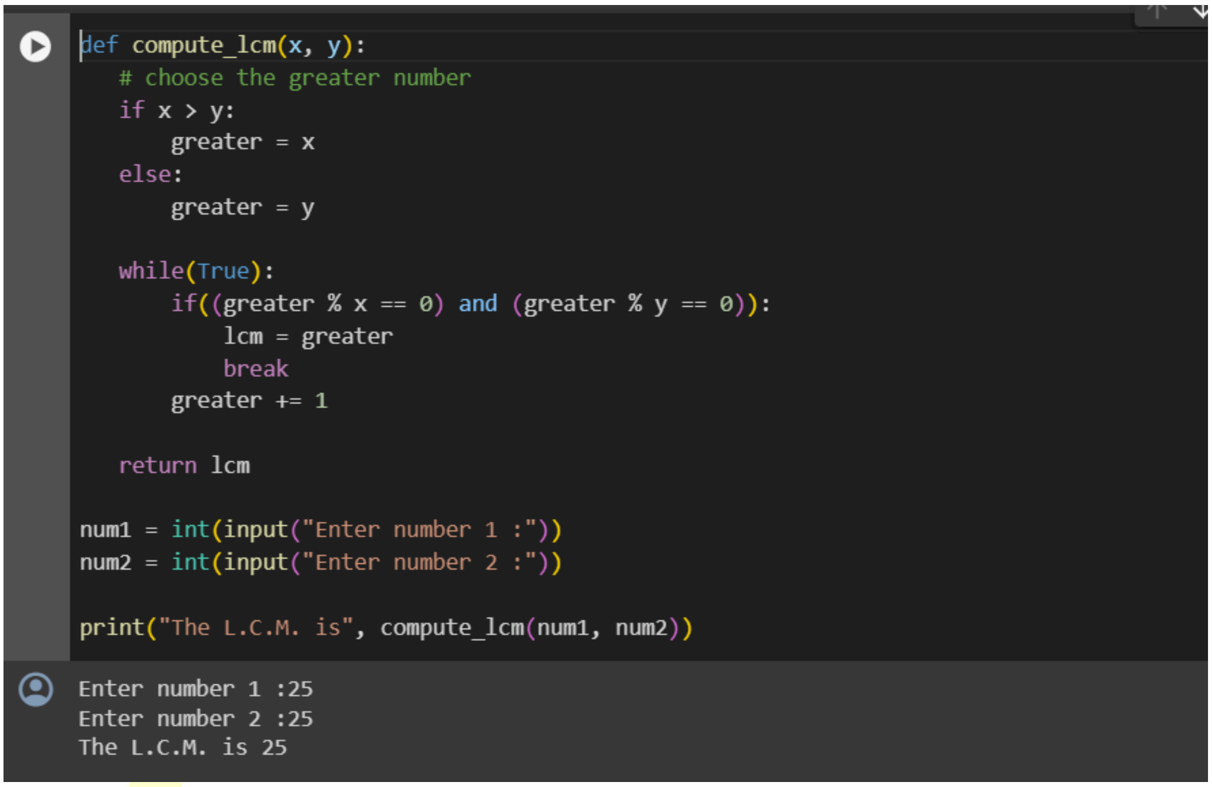
```
def count(n):
  digit=0
  while n>0:
    n=n//10
    digit+=1
  print("Number of digits =",digit)
n=int(input("Enter a number:"))
count(n)
```

```
Enter a number:153
Number of digits = 3
```

TASK 2-LCM OF NUMBERS:

Mahesh learnt about LCM of numbers and is eager to find LCM for more numbers. But he finds it tough to calculate the LCM for larger numbers. Help him to create a function that gets two numbers as input, finds the LCM of those two numbers and displays the result as output.

SOLUTION:



```
def compute_lcm(x, y):
    # choose the greater number
    if x > y:
        greater = x
    else:
        greater = y

    while(True):
        if((greater % x == 0) and (greater % y == 0)):
            lcm = greater
            break
        greater += 1

    return lcm

num1 = int(input("Enter number 1 :"))
num2 = int(input("Enter number 2 :"))

print("The L.C.M. is", compute_lcm(num1, num2))
```

```
Enter number 1 :25
Enter number 2 :25
The L.C.M. is 25
```

<u>WHAT’S NEXT?</u>

PROBLEM SOLVING 3