Python Functions – Continuation

Call by value:

- In the event that you pass arguments like whole numbers, strings or tuples to a function, the passing is like call-by-value because you cannot change the value of the immutable objects being passed to the function

Example:

```
def swap_numbers(a,b):
  c=a
  a=b
  b=c
  print("\nValues inside function:")
  print("a: ",a)
  print("b: ",b)

a=5
b=10
print("\nValues before function call:")
print("a: ",a)
print("b: ",b)

swap_numbers(a,b)

print("\nValues after function call:")
print("a: ",a)
print("b: ",b)
```

```
Values before function call:
a:  5
b:  10

Values inside function:
a:  10
b:  5

Values after function call:
a:  5
b:  10
```

Example:

```
string = "India"

print("\nOutside Function (before calling the function):", string)

def test(string):

    string = "I Love India"

    print("\nInside Function:", string)

test(string)

print("\nOutside Function (after calling the function):", string)
```

```
Outside Function (before calling the function): India

Inside Function: I Love India

Outside Function (after calling the function): India
```

Call by Reference:

- Whereas passing mutable objects can be considered as call by reference because when their values are changed inside the function, then it will also be reflected outside the function.
- The advantage of call-by-reference consists in the advantage of greater time- and space-efficiency, because arguments do not need to be copied.
- On the other hand, this harbours the disadvantage that variables can be "accidentally" changed in a function call.
- So special care has to be taken to "protect" the values, which shouldn't be changed.

```
def list_function(sub_list):
  sub_list.append("Electronics")
  print("\nList Items inside function:")
  print(subject_list)
subject_list=["English", "Mathematics"]
print("\nList items before function call:")
print(subject_list)
list_function(subject_list)
print("\nList items after function call:")
print(subject_list)
```

```
List items before function call:
['English', 'Mathematics']

List Items inside function:
['English', 'Mathematics', 'Electronics']

List items after function call:
['English', 'Mathematics', 'Electronics']
```

Scope of Variables

- All variables in a program may not be accessible at all locations in that program.
- This depends on where you have declared a variable.
- The scope of a variable determines the portion of the program where you can access a particular identifier.
- There are two basic scopes of variables in Python −
  - Global variables
  - Local variables

Global vs. Local variables

- Variables that are defined inside a function body have a local scope, and those defined outside have a global scope.
- This means that local variables can be accessed only inside the function in which they are declared.
- But global variables can be accessed throughout the program body by all functions.
- When you call a function, the variables declared inside it are brought into scope.

Example

```
total = 0; # This is global variable.

# Function definition is here

def sum( arg1, arg2 ):

   # Add both the parameters and return them."

   total = arg1 + arg2; # Here total is local variable.

   print ("Inside the function local - total : ", total)

sum( 10, 20 ) # calling sum function

print ("Outside the function - global total : ", total)
```

```
Inside the function local - total :  30
Outside the function - global total :  0
```

Activity:

1. **Write a python program with the following requirements.**
    - Read numbers from users and insert them into a list (say **list_numbers**).
    - Define a function (say **function_sorting**), which receives the **list_numbers** as an argument and capable of sorting the numbers in the **list_numbers**.
    - Read option from users. If the option is '1', sort the **list_numbers** in ascending order. If the option is '2', sort the **list_numbers** in descending order.
    - Legibly print the **list_numbers** before and after sorting.

2. **Write a python program with the following requirements.**
    - Read a string (say **input_string**) from user.
    - Define a function (say **function_vowel_count**) that receives the **input_string** and count the number of vowel characters in the string and returns the count to the calling function. Print the result properly.
    - Define a function (say **function_numeric_digit_count**) that receives the **input_string** and count the number of **numeric digits (0-9)** in the string and returns the count to the calling function. Print the result properly.
3. **Write a python program with the following requirements.**
    - Read two numbers (say n1, n2).
    - Define four separate functions to add, subtract, multiply and divide n1, n2. Give suitable names to these four functions.
    - Read choice from user.
        - If choice is 1, call the function that performs n1+n2.
        - If choice is 2, call the function that performs n1-n2.
        - If choice is 3, call the function that performs n1*n2.
        - If choice is 4, call the function that performs n1/n2.
    - Read n1, n2, choice from the users and call the corresponding function, as long as the they like.
    - Print the necessary information wherever required.