LAB # 08

FUNCTIONS

OBJECTIVE

Create python function using different argument types.

THEORY

- Functions can be used to define reusable code and organize and simplify code. Basically, we can divide functions into the following two types:

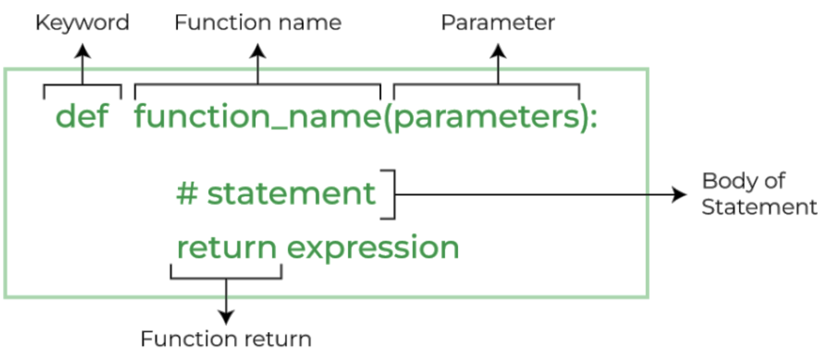
    1. **Built-in functions -** Functions that are built into Python.
    2. **User-defined functions** - Functions defined by the users themselves.

- A function is a block of organized, reusable code that is used to perform a single, related action.
- Functions provide better modularity for your application and a high degree of code reusing.
- Python gives you many built-in functions like print(), etc. but you can also create your own functions. These functions are called **user-defined functions**.

Defining a Function:

- A function definition consists of the function's name, parameters, and body.
- You can define functions to provide the required functionality.
- Here are simple rules to define a function in Python.
    - ✓ Function blocks begin with the **keyword *def*** followed by the **function name and parentheses (( )).**
    - ✓ Any input parameters or arguments should be placed within these parentheses.
    - ✓ You can also define parameters inside these parentheses.
    - ✓ The code block within every function starts with a colon (:) and is indented.

Syntax:



LAB # 08

Example

```
# Define a function

def greet():
  print("Hello from a function")
```

Output:

```
>>> %Run Lab_08.py
>>>
```

Example:

```
# Define a function

def greet_user(username):
    """Display a simple greeting"""
    print("Hello," , username , "!")
```

Output:

```
>>> %Run Lab_08.py
>>>
```

Calling a Function:

Calling a function executes the code in the function. If the function returns a value, a call to that function is usually treated as a value.

Example:

```
# Define a function
def greet():
  print("Hello from a function")

#Calling a function
greet()
```

Output:

```
>>> %Run Lab_08.py
Hello from a function
>>>
```

Example:

```
# Define a function

def greet_user(username):
    """Display a simple greeting"""
    print("Hello," , username , "!")

# Calling a function
greet_user("Abid Class")
```

LAB # 08

Output:

```
>>> %Run Lab_08.py
Hello, Abid Class !
>>>
```

Return Value:

The python return statement is used in a function to return something to the caller program. Use the return statement inside a function only.

Example:

```
# Define Function
def xFunction(x, y):
    print("Addition: ", x + y)
    return

# Calling a function
xFunction(2, 3)
```

Output:

```
>>> %Run Lab_08.py
Addition:  5
>>>
```

When a function doesn't have a return statement and return statement has no value, the function returns ***None.***

Argument Types:

- An argument is a piece of information that is passed from a function call to a function.
- Python supports various types of arguments that can be passed at the time of the function call.
- When we call the function, we place the value we want the function to work with in parentheses.
- You can call a function by using the following types of formal arguments:
  1. Required arguments/ Positional arguments
  2. Keyword arguments
  3. Default arguments

LAB # 08

Required Arguments:

Required arguments are the arguments passed to a function in correct positional order. Here, the number of arguments in the function call should match exactly with the function definition.

Example:

```
# Define a function
def square(x):
    y=x*x
    return y
x=10

# calling a function and storing to result variable
result= square(x)
print("The result of" , x , "squared is", result)
```

Output:

```
>>> %Run Lab_08.py
The result of 10 squared is 100
>>>
```

Keyword Arguments:

- Keyword arguments are related to the function calls.
- When you use keyword arguments in a function call, the caller identifies the arguments by the parameter name.
- This allows you to skip arguments or place them out of order because the Python interpreter is able to use the keywords provided to match the values with parameters.

Example:

```
# define a function having 2 parameters
def print_info(name,age):
    "This prints a passed info into this function"
    print("Name:", name)
    print("Age:" , age)
    return

# calling a function, passsing arguments
print("======== Arguments Passed In Same Order ========")
print_info(name ='Gh', age=27)
print("======== Arguments Passed In Different Order ========")
print_info(age=27, name ='Gh')
```

LAB # 08

Output:

```
>>> %Run Lab_08.py
======== Arguments Passed In Same Order ========
Name: Gh
Age: 27
======== Arguments Passed In Different Order ========
Name: Gh
Age: 27
>>> Age: 50
>>>
```

Default Arguments:

A default argument is an argument that assumes a default value if a value is not provided in the function call for that argument.

Example:

```
# define a function having 2 parameters, with a default value for y
def info(x, y=50):
    """This prints the passed info into this function"""
    print("X:", x)
    print("Y:", y)
    return

# calling the function, passing both arguments
print("======== Arguments Passed ========")
info(x=12, y=27)

# calling the function, passing only one argument
print("======== Only One Argument Passed ========")
info(x=10)  # Must pass x, because it has no default
```

Output:

```
>>> %Run Lab_08.py
======== Arguments Passed ========
X: 12
Y: 27
======== Only One Argument Passed ========
X: 10
Y: 50
>>>
```

LAB # 08

EXERCISE

A. Point out the errors, if any, in the following Python programs.

1. Code

```
define sub(x, y)
return x + y
```

Output

2. Code

```
define describe_pet(pet_name, animal_type='dog'):
    print("\nI have a " , animal_type ,".")
    print("My " , animal_type + "'s name is " , pet_name + ".")
```

Output

3. Code

```
def type_of_int(i):
    if i // 2 == 0:
        return 'even'
    else:
        return 'odd'
```

Output

LAB # 08

B. What will be the output of the following programs:

1. Code

```
def test(a):
   def add(b):
      a =+ 1
      return a+b
   return add
func = test(4)
print(func(4))
```

Output

2. Code

```
def return_none():
  return
print(return_none())
```

Output

3. Code

```
 def test_range(n):
   if n in range(3,9):
      print( n,"is in the range")
   else :
      print("The number is outside the given range.")
test_range(7)
test_range(10)
test_range(4)
```

Output

LAB # 08

C. Write Python programs for the following:

1. Write a user-defined function called favorite_book() that accepts one parameter, title. The function should print a message, such as One of my favorite books is Alice in Wonderland. Call the function, making sure to include a book title as an argument in the function call.

2. Write a user-defined function called max( ), that returns the maxium of three integer numbers.

3. Write a Python program to find GCD of two numbers.

4. Write a user-defined function called describe_city() that accepts the name of a city and its country. The function should print a simple sentence, such as Reykjavik is in Iceland. Give the parameter for the country a default value. Call your function for three different cities, at least one of which is not in the default country.