1. In Python, what is the difference between a built-in function and a user-defined function? Provide an example of each.

- Difference between them is as per their usage and availability
- Built in function:
    - These are predefined and can be used directly
    - E.g print(“My name is Pritam”)
- User defined function:
    - These are defined by user in form of block of code, it is reusable
    - E.g

```
def print_name(name):
        print(“My name is, {name} ”)
```

2. How can you pass arguments to a function in Python? Explain the difference between positional arguments and keyword arguments.

- Arguments can be passed to function in following ways
    - Positional arguments
    - Keyword arguments
    - Default arguments
- In case of positional arguments, arguments are passed to function based on their position or order while in case of keyword arguments, parameters are passed as key-value pair, so ordering does not matter

3. What is the purpose of the return statement in a function? Can a function have multiple return statements? Explain with an example.

- The return statement has two purpose
    - Value output: outputting value back to caller, this value can be assigned to other variable or used in other part of program
    - Termination: to terminate the function
- Yes function can have multiple return statement, E.g below

```
def divide(a/b):
        If b == 0:
                Return (“can not divide by 0”)
        Else:
                Return(a/b)
```

4. What are lambda functions in Python? How are they different from regular functions? Provide an example where a lambda function can be useful.

- Lambda functions are anonymous functions, without name, with single expressions and to be used immediately, syntax is lambda argument: expression
- E.g below

```
Multiplication = lambda x,y : x* y
Sorting_list = sorted(list_name, lambda x: x[1])
```

5. How does the concept of "scope" apply to functions in Python? Explain the difference between local scope and global scope.

- Scope refers to visibility and accessibility of variables, objects and functions
- Local scope variables are defined within function and can not be used outside of function, while global scope variables are defined outside function and can be used anywhere.
- Global scope variable can be edited inside a function using global keyword

6. How can you use the "return" statement in a Python function to return multiple values?

- Returning as a list:

```
def return_values(x,y,z):
    x = 10, y = 20, z = 30
    Return [x,y,z]
```

- Returning as a tuple:

```
def return_values(x,y,z):
    x = 10, y = 20, z = 30
    Return x,y,z
```

- Returning as a dictionary:

```
def return_values(x,y,z):
    x = 10, y = 20, z = 30
    Return {'x':x, 'y':y, 'z':z}
```

7. What is the difference between the "pass by value" and "pass by reference" concepts when it comes to function arguments in Python?

Pass by value (for immutable objects):

- For immutable objects like numbers, strings, tuples; the value of object itself can not be changed, when argument gets passed to function, function creates copy of it and works on it, but original value does not change.

```
[2] def modified_val(x):
      x = x+1
      print('insider the function', x)

    val = 5
    modified_val(val)
```

```
insider the function 6
```

```
[3] print('outside the function', val)
```

```
outside the function 5
```

Pass by Reference (for mutable objects)

- For lists, dictionaries, user defined objects - the reference to the object is passed to the function
- So if value of object is modified inside function, it will impact value outside as well.

```
[4] def modified_ls(lst):
      lst.append(4)
      print('inside the function',lst)

    my_list = [1,2,3]
    modified_ls(my_list)
```

```
    inside the function [1, 2, 3, 4]
```

```
[5] print('outside the function', my_list)
```

```
    outside the function [1, 2, 3, 4]
```

8. Create a function that can intake integer or decimal value and do following operations:

a. Logarithmic function (log x)
b. Exponential function (exp(x))
c. Power function with base 2 (2^x)
d. Square root

```
#operations on int
import math
def math_ops(x):
  log_op = math.log(x)
  exp_op = math.exp(x)
  power_op = math.pow(2,x)
  sqrt_op = math.sqrt(x)
  return log_op, exp_op, power_op, sqrt_op
```

```
[9] math_ops(2)
```

```
    (0.6931471805599453, 7.38905609893065, 4.0, 1.4142135623730951)
```

9. Create a function that takes a full name as an argument and returns first name and last name.

```
[11] #seperate first n last name
     def extract_name(full_name):
       names = full_name.split()
       first_name = names[0]
       last_name = names[-1]
       return first_name,last_name
```

```
[12] extract_name('Pritam Ganpatrao Badwar')

     ('Pritam', 'Badwar')
```

```
[14] first_name,last_name = extract_name('Pritam Ganpatrao Badwar')
     print(first_name,last_name)

     Pritam Badwar
```