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

example of each.

Ans : a) Built-in Function: These are functions that are provided by Python's standard library and are readily available for use without the need for any additional import. Eg - print(), len()

b) User-defined functions: These are the functions created by the user. These functions are defined to perform specific tasks and can be reused throughout the code.

Eg - def rectangle\_area(length, width):

return length \* width

area = rectangle\_area(5, 10)

print("Area of rectangle:", area)

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

arguments and keyword arguments.

Ans : In Python, we can pass arguments to a function in several ways. When defining a function, we can specify parameters to receive these arguments. When calling the function, we pass the actual values to these parameters. There are two main ways to pass arguments to a function: positional arguments and keyword arguments.

1. Positional Arguments: Positional arguments are the most common way to pass arguments to a function. They are matched based on their position in the function call. The first argument is assigned to the first parameter, the second argument to the second parameter, and so on. The order of the arguments matters when using positional arguments.
2. Keyword Arguments: Keyword arguments are passed to a function using the name of the parameter as a keyword, followed by the value. When using keyword arguments, the order of the arguments 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.

Ans : The return statement in a function is used to specify the value that the function will send back as the result of its execution. When a return statement is encountered in a function, it immediately terminates the function's execution and returns the specified value to the caller.

Yes, a function can have multiple return statements. However, only one return statement is executed during the function's execution. As soon as the first return statement is encountered, the function terminates, and the specified value is returned.

Eg - def get\_grade(score):

if score >= 70:

return "A"

elif score >= 40:

return "B"

elif:

return "C"

4. What are lambda functions in Python? How are they different from regular functions? Provide an

example where a lambda function can be useful.

Ans : Lambda functions are single-line functions in Python that can be defined without a name. They are defined using the lambda keyword and are useful for situations where a simple function is needed for a short period of time. They are different from normal functions as their syntax is completed in one line only. Also, Lambda functions are limited to a single expression, while regular functions can contain multiple statements and have a block of code with a return statement.

Syntax of lambda function - lambda arguments: expression

Eg - add = lambda a, b: a + b

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

scope and global scope.

Ans : In Python, the concept of "scope" refers to the accessibility of variables within different parts of a program. It determines where a variable can be accessed and modified. Python has two main types of scope: local scope and global scope.

1. Local Scope: When a variable is defined inside a function, it belongs to the local scope of that function. This means the variable is only accessible within the function in which it is defined.
2. Global Scope: Variables defined outside any function or at the top level of a module belong to the global scope. Global variables are accessible from any part of the code, including inside functions. To use a global variable inside a function, you need to explicitly declare it using the global keyword.

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

Ans : def info():

    name = "Aditya"

    age = 30

    email = "aditya@google.com"

    return name, age, email

print(info())

7. What is the difference between the "pass by value" and "pass by reference" concepts when it

comes to function arguments in Python?

Ans : In Python, the terms "pass by value" and "pass by reference" are often used to describe how function arguments are handled.

1. Pass by Value: In "pass by value," the function receives a copy of the actual value of the argument passed to it. Any changes made to the parameter within the function do not affect the original value outside the function. The function works with its own local copy of the variable.
2. Pass by Reference: In "pass by reference," the function receives a reference (memory address) to the original variable. Any changes made to the parameter within the function directly affect the original value outside the function.

Python's argument-passing mechanism involves both "pass by value" for immutable objects and "pass by reference" for mutable objects.

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 (2x)

d. Square root

Ans : import math

def cal():

    var = float(input("Give a value : "))

    func = input("Give a function (power/log/exp/sq\_rt) : ")

    result = None

    if func == "power":

        result = pow(var,2)

    elif func == "log":

        result = math.log(var)

    elif func == "exp":

        result = math.exp(var)

    elif func == "sq\_rt":

        result = math.sqrt(var)

    else:

        return "TRY AGAIN"

    return result

print(cal())

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

Ans :

def get\_full\_name():

    full\_name = input("Enter your full name: ")

    names = full\_name.split()

    first\_name = names[0]

    last\_name = names[1]

    return first\_name, last\_name

first\_name, last\_name = get\_full\_name()

print("First Name:", first\_name)

print("Last Name:", last\_name)