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

**Ans.1**

* Built-in functions are functions that come with Python and are readily available for use without the need for additional definitions. Examples include print(), len(), and input().

**Example of a Built-in Function:**

print("Hello, World!")

* User-defined functions are functions created by users to perform specific tasks. They are defined using the def keyword and can be customized as needed.

**Example of a User-defined Function:**

def greet(name):

print(f"Hello, {name}!")

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

**Ans.2** In Python, you can pass arguments to a function in two ways:

* **Positional Arguments:** These are passed in the order defined in the function's parameter list. The order matters.

Example of Positional Arguments:

def add(a, b):

return a + b

result = add(3, 5) **# Here, 3 is assigned to 'a', and 5 is assigned to 'b'.**

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

**Ans.3**

The **return** statement is used to specify the value that a function should return when it's called. A function can have multiple **return** statements, but only one will be executed, and it will exit the function when encountered.

**Example with Multiple Return Statements**:

def compare\_numbers(a, b):

if a > b:

return "a is greater"

elif a < b:

return "b is greater"

else:

return "a and b are equal"

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.4**

Lambda functions are small, anonymous functions defined using the **lambda** keyword. They are often used for short, simple operations. Unlike regular functions, lambda functions can have only one expression and are typically used when a function is needed for a short duration.

**Example of a Lambda Function**

double = lambda x: x \* 2

print(double(5)) # Output: 10

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

**Ans.5 Scope in Functions:**

* Local Scope: Variables defined within a function are in the local scope and are only accessible within that function.
* Global Scope: Variables defined outside of any function are in the global scope and can be accessed from anywhere in the program.

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

Ans.6 **Returning Multiple Values:**

You can return multiple values from a function by returning a tuple, a list, or multiple values separated by commas. **For example:**

def get\_values():

return 1, 2, 3

a, b, c = get\_values() **# Unpack the returned values into individual variables**

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

comes to function arguments in Python?

**Ans.7**

**Pass by Value vs. Pass by Reference:**

In Python, function arguments are passed by object reference. This means that when you pass a mutable object (like a list) to a function, changes made to the object within the function will affect the original object. For immutable objects (like integers), a new object is created within the function, and changes do not affect the original object.

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.8**

Python function that performs the specified mathematical operations on an input value (integer or decimal) as follows:

**import math**

def math\_operations(x):

**# Logarithmic function (log x)**

log\_x = math.log(x)

**# Exponential function (exp(x))**

exp\_x = math.exp(x)

**# Power function with base 2 (2^x)**

power\_2x = 2 \*\* x

**# Square root**

sqrt\_x = math.sqrt(x)

return log\_x, exp\_x, power\_2x, sqrt\_x

**# Example usage:**

value = 4.0 **# You can change this value to any integer or decimal**

logarithm, exponential, power\_of\_2, square\_root = math\_operations(value)

print(f"Logarithmic function (log {value}): {logarithm}")

print(f"Exponential function (exp({value})): {exponential}")

print(f"Power function with base 2 (2^{value}): {power\_of\_2}")

print(f"Square root of {value}: {square\_root}")

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

**Ans.9**

def extract\_names(full\_name):

**# Split the full name into parts using whitespace as the separator**

name\_parts = full\_name.split()

if len(name\_parts) == 1:

**# If there's only one part, consider it as the first name**

first\_name = name\_parts[0]

last\_name = "" # Set the last name to an empty string

elif len(name\_parts) >= 2:

**# If there are multiple parts, consider the first part as the first name**

**# and the last part as the last name**

first\_name = name\_parts[0]

last\_name = name\_parts[-1]

else:

**# If there are no parts, set both first name and last name to empty strings**

first\_name = ""

last\_name = ""

return first\_name, last\_name

**# Example usage:**

full\_name = "John Doe"

first, last = extract\_names(full\_name)

print(f"First Name: {first}, Last Name: {last}")