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

Built-in functions: These functions are pre-defined in Python and are available for use without any additional code. Examples include print(), len(), and sum().

# Built-in function example

my\_list = [1, 2, 3, 4]

length = len(my\_list)

print("Length of the list:", length)

User-defined functions: These functions are created by the user to perform specific tasks as needed. They are defined using the def keyword followed by a function name, parameters, and a function body.

# User-defined function example

def greet(name):

print("Hello, " + name + "!")

greet("Alice")

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

arguments and keyword arguments.

In Python, you can pass arguments to a function when you call it.

There are two types of arguments:

Positional arguments: These are passed in the same order as the function parameters are defined.

Keyword arguments: These are passed with the parameter name explicitly, which allows you to pass them in any order.

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 in a function is used to specify the value that the function should produce as its output. It allows the function to send a result back to the caller. A function can have multiple return statements, but only one of them will be executed during the function call.

Example:

def get\_grade(score):

if score >= 90:

return "A"

elif score >= 80:

return "B"

else:

return "C"

grade = get\_grade(85)

print("Grade:", grade)

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, also known as anonymous functions, are small, one-line functions that don't have a name. They are created using the lambda keyword and can take any number of arguments but can only have one expression.

Example # Regular function

def square(x):

return x \*\* 2

# Equivalent lambda function

square\_lambda = lambda x: x \*\* 2

print(square(5)) # Output: 25

print(square\_lambda(5)) # Output: 25

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

Local scope: Variables defined inside a function have local scope. They are accessible only within the function and are destroyed when the function exits.

Global scope: Variables defined outside any function have global scope. They are accessible throughout the program.

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

In Python, you can use the return statement to return multiple values as a tuple.

def calculate\_stats(numbers):

total = sum(numbers)

average = total / len(numbers)

return total, average

nums = [1, 2, 3, 4, 5]

total\_sum, avg = calculate\_stats(nums)

print("Total sum:", total\_sum)

print("Average:", avg)

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

Python uses "pass by object reference." When you pass an object (like a list or dictionary) to a function, you are passing a reference to that object, not a copy of the object. Modifications made to the object inside the function will affect the original object outside the function.

For immutable objects (like integers or strings), Python behaves like "pass by value." When you pass an immutable object to a function, a copy of the value is created, and modifications inside the function do not affect the original value outside the function.

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

import math

def math\_operations(x):

log\_result = math.log(x)

exp\_result = math.exp(x)

power\_result = 2 \*\* x

sqrt\_result = math.sqrt(x)

return log\_result, exp\_result, power\_result, sqrt\_result

num = 10

log\_val, exp\_val, pow\_val, sqrt\_val = math\_operations(num)

print("log(", num, ") =", log\_val)

print("exp(", num, ") =", exp\_val)

print("2 ^", num, "=", pow\_val)

print("Square root of", num, "=", sqrt\_val)

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

def extract\_names(full\_name):

names = full\_name.split()

first\_name = names[0]

last\_name = names[-1]

return first\_name, last\_name

full\_name = "John Doe"

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

print("First Name:", first)

print("Last Name:", last)