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** are functions that are already defined in the Python language. They can be used without having to define them first. Some examples of built-in functions are print (), len(), and sum().

Example of a built-in function:

Print ("Hello, world!")

**User-defined functions** are functions that are defined by the programmer. They can be used to perform specific tasks that are not built into the Python language. To define a user-defined function, you use the def keyword.

def factorial(n):

"""

Calculates the factorial of a number.

Args:

n: The number to calculate the factorial of.

Returns:

The factorial of n.

"""

result = 1

for i in range(1, n + 1):

result \*= i

return result

print(factorial(5))

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

**Positional arguments** are arguments that are passed to a function in the order that they are defined. For example, the following function definition defines a function called add () that takes two positional arguments:

def add(a, b):

return a + b

**Keyword arguments** are arguments that are passed to a function by name. For example, the following function definition defines a function called greet() that takes two keyword arguments:

def greet(name, greeting):

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

1. 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 Python function is used to exit the function and return a value. The value that is returned can be any Python object, such as a number, a string, a list, or a dictionary.

A function can have multiple return statements, but only one of them will be executed. The first return statement that is encountered will cause the function to exit and return the value that is associated with that statement.

1. 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 different from regular functions in several ways:

They are anonymous. Lambda functions do not have a name, so they cannot be called using the function name. They can only be called using the lambda expression.

They can only have one expression. Lambda functions can only have one expression, which is evaluated and the result is returned. Regular functions can have multiple statements, which are executed in order.

They are typically used for short, simple tasks. Lambda functions are typically used for short, simple tasks that do not need to be named. They are often used in conjunction with other functions, such as map () and filter ().

Here is an example of where a lambda function can be useful:

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

doubled\_numbers = map(lambda x: x \* 2, numbers)

print(doubled\_numbers)

O/P: [2, 4, 6, 8, 10]

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

In Python, scope is a region of a program where a name can be used to refer to an object. There are two types of scope in Python: local scope and global scope.

Local scope is the scope of a function. Variables defined inside a function are only visible inside that function.

Global scope is the scope of the entire program. Variables defined outside of any function are visible in the global scope.

1. 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 from a function by separating them with commas.

For example, the following function definition defines a function called get\_coordinates () that returns the x-coordinate and y-coordinate of a point:

def get\_coordinates(x, y):

"""

Returns the coordinates of a point.

Args:

x: The x-coordinate of the point.

y: The y-coordinate of the point.

Returns:

A tuple containing the x-coordinate and y-coordinate of the point.

"""

return x, y

1. 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** means that a copy of the argument is passed to the function. Any changes made to the argument inside the function will not affect the original argument.

**Pass by reference** means that a reference to the argument is passed to the function. Any changes made to the argument inside the function will affect the original argument.

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

a. Logarithmic function (log x)

import math

def math\_operations(x):

"""

This function takes an integer or decimal value and performs the following operation:

\* Logarithmic function (log x)

Args:

x: The integer or decimal value to perform the operation on.

Returns:

The logarithm of x.

"""

log\_x = math.log(x)

return log\_x

b. Exponential function (exp(x))

import math

def math\_operations(x):

"""

This function takes an integer or decimal value and performs the following operation:

\* Exponential function (exp(x))

Args:

x: The integer or decimal value to perform the operation on.

Returns:

The exponential of x.

"""

exp\_x = math.exp(x)

return exp\_x

c. Power function with base 2 (2x)

def math\_operations(x):

"""

This function takes an integer or decimal value and performs the following operation:

\* Power function with base 2 (2x)

Args:

x: The integer or decimal value to perform the operation on.

Returns:

The power of x with base 2.

"""

two\_x = 2 \*\* x

return two\_x

d. Square root

import math

def math\_operations(x):

"""

This function takes an integer or decimal value and performs the following operation:

\* Square root

Args:

x: The integer or decimal value to perform the operation on.

Returns:

The square root of x.

"""

sqrt\_x = math.sqrt(x)

return sqrt\_x

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

def get\_name(full\_name):

"""

This function takes a full name as an argument and returns the first name and last name.

Args:

full\_name: The full name of the person.

Returns:

A tuple containing the first name and last name.

"""

split\_name = full\_name.split(" ")

first\_name = split\_name[0]

last\_name = split\_name[-1]

return first\_name, last\_name