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 provided by the Python programming language itself. These functions are pre-defined and are readily available for use without the need for additional code. Examples of built-in functions include `print()`, `len()` etc.

Example of a built-in function (`len()`):

my\_list = [1, 2, 3, 4, 5]

length = len(my\_list)

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

**User-defined functions** are functions created by the programmer to perform specific tasks. These functions are defined using the `def` keyword and can be customized to suit the programmer's needs.

Example:

def greet(name):

return "Hello, " + name + "!"

user\_name = "Alice"

greeting = greet(user\_name)

print(greeting)

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

**Positional arguments** are passed to a function based on their position or order. The function receives the arguments in the same order as they are passed. **Example:**

def add\_numbers(a, b):

return a + b

result = add\_numbers(5, 3)

**Keyword arguments** are passed to a function using the parameter names as keywords. This allows you to specify arguments out of order, making the function call more readable and self-explanatory.

**Example:**

def print\_info(name, age, country):

print(f"Name: {name}, Age: {age}, Country: {country}")

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 function specifies what value the function should produce as its result. When a `return` statement is encountered within a function, the function immediately stops executing.

A function can indeed have multiple `return` statements.

def calculate\_grade(score):

if score >= 90:

return "A"

elif score >= 80:

return "B"

elif score >= 70:

return "C"

else:

return "F"

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, also known as anonymous functions, are small, single-expression functions in It can have any number of arguments but can only have one expression. Defined using the `lambda` keyword, followed by the argument list and the expression.

**Syntax:** lambda arguments: expression

**Different:** 1. Lambda functions use a more compact syntax compared to regular functions

2. Lambda functions can only have one expression, whereas regular functions can have multiple expressions.

3. Lambda functions are anonymous; they don't require a name.

Usually used for sorting, filtering, or mapping operations **Example:**

points = [(1, 5), (3, 2), (2, 8), (4, 3)]

sorted\_points = sorted(points, key=lambda x: x[1])

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

The concept of "scope" in Python refers to the visibility and accessibility of variables within different parts of code.

**Local scope** refers to the area within a function where variables are defined. They are only accessible and meaningful within that function. Local variables are created when the function is called and are destroyed when the function exits.

**Global scope** refers to the variables defined outside of any function, at the main script level, have global scope. Global variables can be accessed and modified from anywhere in the code, including within functions.

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

In Python, you can use the `return` statement in a function to return multiple values by returning them as part of a data structure, such as a tuple, list, or dictionary.

**For Example:** You can pack multiple values into a list and return the list using the `return` statement.

def get\_numbers():

num1 = 5

num2 = 10

num3 = 15

return [num1, num2, num3]

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

In a **pass by value** scenario, a copy of the actual value of the variable is passed to the function. This means that any changes made to the parameter within the function do not affect the original variable outside the function.

In a **pass by reference** scenario, a reference or memory address of the variable is passed to the function. Any modifications made to the parameter inside the function are reflected in the original variable outside the function.

1. **Create a function that can intake integer or decimal value and do following operations:**
   1. **Logarithmic function (log x)**
   2. **Exponential function (exp(x))**
   3. **Power function with base 2 (2x)**
   4. **Square root**

import math

def perform\_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

value = float(input("Enter a number: "))

log\_res, exp\_res, power\_res, sqrt\_res = perform\_operations(value)

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

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

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

print(f"Square root of {value}: {sqrt\_res}")

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

def extract\_first\_last\_name(full\_name):

names = full\_name.split()

first\_name = names[0]

last\_name = names[-1]

return first\_name, last\_name

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

first\_name, last\_name = extract\_first\_last\_name(full\_name)

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

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