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

**Ans:** Built-in functions are predefined functions that are already been designed by someone else and

we can use it anytime for example: len(), type() etc

User defined functions are the functions defined by the programmer for their use for any repetitive task. So that they write the code one time and use it as many times simply by calling it.

For ex: function to add two numbers.

Def add\_two\_num(a,b):

C = a+b

Return C

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

**Ans:** Arguments are specified after the function name, inside the parentheses for example :

Def add\_two\_num(a,b):

C = a+b

Return C

Here ‘a’ and ‘b’ are arguments

Positional arguments must be included in the correct order. Keyword arguments are included with a keyword and equals sign.

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

**Ans:** return statement is used to return the output from a function.

Python functions can have multiple return statements

def infiniteReturn(x):

if x > 0:

return 1

elif x < 0:

return -1

else:

return 0

1. What are lambda functions in Python? How are they different from regular functions? Provide an example where a lambda function can be useful.

**Ans:** A lambda function is an anonymous function (i.e., defined without a name) that can take any number of arguments but, unlike normal functions, evaluates and returns only one expression.

Expressions that do not include complex structures such as if-else, for-loops, etc can be made using lamda functions for ex: lambda x: x % 2 == 0

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

Ans: The concept of scope rules how variables and names are looked up in your code.

1. Global scope**:** The names that you define in this scope are available to all your code.
2. Local scope: The names that you define in this scope are only available or visible to the code within the scope.
3. How can you use the "return" statement in a Python function to return multiple values?

**Ans:** Using tuple, dictionary, class, list

For example: def func(x):

   y0 = x+ 1

   y1 = x \* 3

    y2 = y0 \*\* 3

    return (y0, y1, y2)

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

Ans: When you give function parameters via reference, you're just passing references to values that already exist. When you pass arguments by value, on the other hand, the arguments become independent copies of the original values.

**Pass by value**

Any other operation performed will not have any effect on the original value as the argument passed to the function is copied. It only changes the value of the copy created within the function.

**Pass by reference**

This method passes a reference to the original arguments, and any operation performed on the parameter affects the actual value. It alters the value in both function and globally.

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

**Ans: (a):** def logarithm():

import math

try:

x = int(input("Please input any number to find log: "))

return math.log(x)

except:

print("Sorry, you have not entered an integer value")

(b) **):** def logarithm():

import math

try:

x = int(input("Please input any number to find log: "))

return math.log(x)

except:

print("Sorry, you have not entered an integer value")

(c) pow(x)

(d)import math

def calculate\_square\_root(number):

return math.sqrt(number)

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

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

name\_parts = full\_name.split()

first\_name = name\_parts[0]

last\_name = ' '.join(name\_parts[1:])

return first\_name, last\_name