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 readily available functions with python programming language, we do not need to import or define it in line of codes. Eg: print(), len(), input(), etc.

   User-defined functions are defined by the user as per requirement. User define custom code for specific requirement. These functions helps in code reusability, improved readability and maintability.

   Eg: calculate_square()

```
def calculate_square(number):
    square = number**2
    return square
result = calculate_square(10)
    print(result)
```

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

   We can pass arguments to a function in python in two ways i) By positional arguments ii) by keywords arguments

   Positional arguments are passed in a function in a certain order based on their position or order within the function definition.

   Eg: def Greet(name, age):
   print(f ' Hello {name} ! ,You have {age} years' of wisdom')

   Greet('juhi', 30)

   Output ➔ Hello juhi ! ,You have 30 years' of wisdom

   Keywords arguments: These are passed with a keyword and associated values, irrespective of its order within the function definition. Only rule is keyword should correctly match with its corresponding parameters. These provide flexibility to the programmer and is useful when a function takes many arguments.

   Eg: def Greet(name, age):
   print(f ' Hello {name} ! ,You have {age} years' of wisdom')

   Greet(age= 30, name = "juhi")

   Output ➔ Hello juhi ! You have 30 years' of wisdom

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

   Return statement is essential for the functions that performs some calculations or have to produce results that need to be used or stored for further processing in the calling code. Function can have multiple returns but only one can be executed at a time based on problem conditions.

   Eg: def divide(a/b):
   if b==0:
   return "Error: division by zero not allowed"
   else:
   result= a/b
   return result

```
print(divide(10,2)) #output➔ 5
print(divide(10,0)) #output➔ Error: division by zero not allowed
```

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 are single line expression functions. We do not need to define a named function. They are anonymous simple single line functions.

```
Eg: num = [1,3,4,5]
square_number = map(lambda x : x**2, num)
my_lst= list(square_number)
print(my_lst)
#output➔[1,9,16,25]
```

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

   In python scope refers to visibility and accessibility of a variable within different parts of a program. This concept applies to functions also.

   Local scope variables are defined within a function. They cannot be called or modified outside the function.

```
Eg: def func():
        X=10
        print(X)
func() #output ➔ 10
print(X) #output➔ Error: variable not defined
```

   Global scope variables are defined outside of any function, at top of script or any module. Global means they can be access from anywhere within the program including inside functions.

```
Eg:  X=12
def func():
    print(X)
func() #output ➔ 12
print(X) #output➔ 12
```

   However, if we assign new value to X inside a function, it will create new local variable instead of modifying the value. We need to explicitly indicate that it is a global variable by using keyword **global** before X.

```
Eg:  X=12
def func():
    global X   #modifying global variable
    X= 8
    print(X)
func() #output ➔ 8

print(X) #output➔ 8 #modified value
```

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

   “Return” statement can return multiple values in form of a tuple.

```
Eg:   def personal_info():
          name = “juhi”
          age = 30
          city = “indore”
          return name, age, city
```

```
print(personal_info()) #output➔ (‘juhi’, 30, indore)
```

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

   When we pass argument to a function, a reference to the object is passed. However, the behavior can be similar to both “pass by value” and “Pass by reference” depending on the behavior of the object.

   If Object is mutable, it behaves like pass by reference (Modifications are global, changes original value) Eg: lists, dictionaries

   if object is immutable, it behaves like pass by value (modifications are local, only within function. It does not reflect outside the function.) eg: strings, integer, tuple

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 perform_operations(x):
    logarithm = math.log(x)
    exponential = math.exp(x)
    power = math.pow(2, x)
    square_root = math.sqrt(x)

    return logarithm, exponential, power, square_root

# Test the function
value = 2.5
result = perform_operations(value)
print("Logarithm:", result[0])
print("Exponential:", result[1])
print("Power (base 2):", result[2])
print("Square Root:", result[3])
```

Output➔ Logarithm: 0.9162907318741551
Exponential: 12.182493960703473
Power (base 2): 5.656854249492381
Square Root: 1.5811388300841898

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

```
def Extract_name():
    a = input('Enter your full name')
    b=a.split()
    print(f'First name is {b[0]} and last name is {b[-1]}')

Extract_name()
```

Output➔ Enter your full name juhi tejwani
First name is juhi and last name is tejwani