Function Assignment 1

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

- **Built-in function**: A built-in function is a function that is already provided by the Python programming language. These functions are readily available for use without requiring any additional code or external dependencies.

  ex
  ```
  my_list = [1, 2, 3, 4, 5]
  length = len(my_list)
  print(length)
  ```
  User-defined function: A user-defined function is created and defined by the user to perform a specific task or a series of tasks. These functions are not part of the Python language itself and need to be defined explicitly by the programmer.

  ex
  ```
  def calculate_sum(a, b):
    return a + b

   result = calculate_sum(3, 4)

  print(result)
  ```

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

- Information can be passed into functions as arguments. Arguments are specified after the function name, inside the parentheses. You can add as many arguments as you want, just separate them with a comma.
  Positional arguments: Positional arguments are passed to a function based on their position or order. The function's parameters are matched with the arguments in the same order they are provided.
  Keyword arguments: Keyword arguments are passed to a function using the names of the parameters. Each argument is associated with a specific parameter, regardless of their 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 serves the purpose of specifying the value that should be returned by the function when it is called. It allows a function to compute a result and pass it back to the caller. When a **return** statement is encountered in a function, it immediately terminates the execution of the function and returns the specified value.

A function can have multiple **return** statements, but only one of them is executed.

```
Example: def get_grade(score):
    if score >= 90:
        return "A"
    elif score >= 80:
        return "B"
    elif score >= 70:
        return "C"
    else:
        return "F"


student_score = 85
grade = get_grade(student_score)
print(f"The grade for a score of {student_score} is {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 are defined without a function name. They are created using the **lambda** keyword and are typically used for simple and concise operations

```
example:add = lambda x, y: x + y
         result = add(3, 4)
           print(result)
```

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

- The concept of "scope" refers to the visibility and accessibility of variables within different parts of a program.

Local Scope: Local scope refers to variables defined within a specific function. These variables are only accessible within that function and are considered local to it. They cannot be accessed from outside the function or from other functions

Global Scope: Global scope refers to variables that are defined outside of any function or at the top level of a module

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

- To return multiple values from a function in Python, you can use the **return** statement followed by a comma-separated list of values.

7. 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: In Python, the concept of "pass by value" and "pass by reference" can be a bit different from some other programming languages.
pass by reference: Since everything is passed by object reference in Python, if the parameter is a mutable object (e.g., a list or dictionary), modifications made to the parameter within the function will affect the original object 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 perform_operations(value):

   logarithmic = math.log(value)

   exponential = math.exp(value)

   power_two = math.pow(2, value)

   square_root = math.sqrt(value)

     return logarithmic, exponential, power_two, square_root

 result = perform_operations(4)

print(result)
```

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

```
➢ def get_first_last_name(full_name):
   names = full_name.split()
   first_name = names[0]
   last_name = names[-1]
   return first_name, last_name
name = "John Doe"
first, last = get_first_last_name(name)
print("First Name:", first)
print("Last Name:", last)
```