Functions in Python

Lab Objective:

By the end of this lab, students will:

1. Understand how to define and call **functions** in Python.
2. Learn about **parameters** and **return values**.
3. Explore **scope** (local and global variables) and the importance of reusability through functions.
4. Work with **default arguments**, **keyword arguments**, and **arbitrary arguments**.
5. Solve practical problems using functions.

Part 1: Introduction to Functions

Objective:

- Understand what functions are and how to define and use them to encapsulate reusable code.

Concept:

- A **function** is a block of organized, reusable code that performs a specific task. Functions help in breaking the program into smaller, manageable sections.

Syntax:

```
def function_name(parameters):
    # Function body
    return result  # Optional return statement
```

Tasks:

1. **Defining a Simple Function:**
    - Write a function that prints a message.

    Code Example:

    ```
    def greet():
        print("Hello, welcome to Python functions!")

    greet()  # Call the function
    ```

2. **Function with Parameters:**
    - Write a function that takes two numbers and prints their sum.

    Code Example:

```
def add_numbers(a, b):
    print(f"The sum of {a} and {b} is {a + b}")

add_numbers(10, 5)  # Call the function with arguments
```

3. **Function with Return Value:**
    - Write a function that returns the square of a number.

    Code Example:

```
def square(num):
    return num * num

result = square(4)
print(f"The square of 4 is {result}")
```

Exercise 1:

- Write a function named `multiply` that takes two numbers as input, multiplies them, and returns the result.

Part 2: Parameters and Arguments

Objective:

- Understand the difference between **parameters** and **arguments** in functions. Learn how to work with **default arguments** and **keyword arguments**.

Tasks:

1. **Positional Arguments:**
    - Write a function that calculates the area of a rectangle and takes length and width as positional arguments.

    Code Example:

```
def rectangle_area(length, width):
    return length * width

print("Area of the rectangle:", rectangle_area(5, 3))
```

2. **Default Arguments:**
    - Write a function with a default argument.

    Code Example:

```
def greet(name="Guest"):
    print(f"Hello, {name}!")
```

```
greet()  # Default value is used
greet("Alice")  # Argument is passed, overriding the default
```

3. **Keyword Arguments:**
    - Write a function that takes keyword arguments.

    Code Example:

```
def introduce(name, age):
    print(f"My name is {name} and I am {age} years old.")

introduce(name="John", age=30)
```

Exercise 2:

- Write a function `student_info` that takes two arguments: `name` and `age`. If `age` is not provided, use a default value of `18`. Test the function by calling it with and without the `age` argument.

Part 3: Variable-Length Arguments

Objective:

- Learn how to work with functions that accept a variable number of arguments using **`*args`** and **`**kwargs`**.

Tasks:

1. **Using `*args`:**
    - Write a function that accepts a variable number of arguments and returns their sum.

    Code Example:

```
python
Copy code
def sum_numbers(*args):
    total = 0
    for num in args:
        total += num
    return total

print("Sum of numbers:", sum_numbers(1, 2, 3, 4, 5))
```

2. **Using `**kwargs`:**
    - Write a function that accepts keyword arguments and prints them.

Code Example:

```
def display_info(**kwargs):
    for key, value in kwargs.items():
        print(f"{key}: {value}")

display_info(name="Alice", age=25, city="New York")
```

Exercise 3:

- Write a function named `product_of_numbers` that accepts any number of arguments using `*args` and returns their product.

Part 4: Scope of Variables (Local and Global)

Objective:

- Understand the scope of variables and the difference between local and global variables.

Tasks:

1. **Local Variables:**
   - Write a function where variables declared inside the function are local and cannot be accessed outside.

   Code Example:

   ```
   def local_scope():
       message = "This is a local variable."
       print(message)

   local_scope()
   # print(message)  # This will raise an error because 'message' is not
   defined outside the function
   ```

2. **Global Variables:**
   - Write a function that modifies a global variable.

   Code Example:

   ```
   count = 0  # Global variable

   def increment():
       global count
       count += 1

   increment()
   print("Count after incrementing:", count)
   ```

Exercise 4:

- Write a Python function that modifies a global variable to track the number of times a function is called.

Part 5: Lambda Functions

Objective:

- Learn how to use **lambda functions** (anonymous functions) in Python.

Tasks:

1. **Basic Lambda Function:**
   - Write a lambda function to calculate the square of a number.

   Code Example:

   ```
   square = lambda x: x * x
   print("Square of 5:", square(5))
   ```

2. **Lambda with Multiple Arguments:**
   - Write a lambda function to calculate the product of two numbers.

   Code Example:

   ```
   product = lambda a, b: a * b
   print("Product of 3 and 4:", product(3, 4))
   ```

Exercise 5:

- Write a lambda function that takes three arguments and returns their average.

Part 6: Recursive Functions

Objective:

- Understand and implement **recursive functions**, where a function calls itself to solve a smaller version of the problem.

Tasks:

1. **Factorial using Recursion:**

- Write a recursive function to calculate the factorial of a number.

Code Example:

```
def factorial(n):
    if n == 1:
        return 1
    else:
        return n * factorial(n - 1)

print("Factorial of 5:", factorial(5))
```

Exercise 6:

- Write a recursive function that calculates the **nth Fibonacci number**.

Assignment:

1. **Problem 1:**
   - Write a Python function `max_of_three` that takes three numbers as arguments and returns the **largest** of the three.
2. **Problem 2:**
   - Create a function `is_prime` that checks if a given number is prime. The function should return `True` if the number is prime and `False` otherwise.
3. **Problem 3:**
   - Write a Python program that defines a function `reverse_string` which takes a string as input and returns the string reversed. Use the function to reverse the user's input string.
4. **Problem 4:**
   - Write a recursive function to calculate the **sum of all numbers** in a list.
5. **Problem 5:**
   - Create a Python program that defines a **lambda function** to filter out the even numbers from a list of integers.