Lab 03 – Functions, Strings, and Modular Programming

Faculty of Engineering
Egyptian Russian University
Department of Mechatronics and Robotics

1. Objectives

After completing this lab, students will be able to:

- Define and call functions in Python.
- Pass arguments and return values from functions.
- Explain variable scope (local vs. global).
- Manipulate strings using built-in methods.
- Combine loops, conditions, and functions in modular programs.

2. Introduction

Functions and strings are key structural elements in Python programs. A **function** is a reusable block of code that performs a specific task, while a **string** is an immutable sequence of characters. Using functions promotes modularity and reduces code repetition. Strings are used in almost every program—for input/output, file names, messages, and text processing.

3. Functions in Python

3.1 Function Syntax

```
def function_name(parameters):
"""optional docstring"""
# statements
return expression
```

Listing 1: Simple addition function

```
def add(a, b):
"""Return the sum of two numbers."""
return a + b

x = float(input("Enter first number: "))
y = float(input("Enter second number: "))
print("Sum =", add(x, y))
```

Key Points:

- The `def` keyword defines a function.
- Parameters are optional.
- The `return` statement sends a result back to the caller.

3.2 Parameters and Return Values

```
        def average(a, b, c):
        return (a + b + c) / 3

        print("Average:", average(10, 20, 30))
```

Output:

```
Average: 20.0
```

3.3 Default and Keyword Arguments

```
        def greet(name, message="Welcome to Python!"):
        print("Hello", name + "!", message)

        greet("Elsayed")
        greet("Mina", "Good luck in the lab.")
```

3.4 Scope of Variables

Variables declared inside a function are **local**. Global variables exist outside any function.

```
        x = 10          # global
        def test():
        x = 5       # local
        print("Inside:", x)
        test()
        print("Outside:", x)
```

4. Strings in Depth

4.1 String Indexing and Slicing

```
        name = "Python"
        print(name[0])        # P
        print(name[-1])       # n
        print(name[1:4])      # yth
```

4.2 String Methods

Common methods: `upper()`, `lower()`, `strip()`, `replace()`, `split()`, `join()`, `count()`, `find()`.

```
        text = "  python programming  "
        print(text.strip().title())
        print(text.replace("python", "Python 3"))
```

4.3 Iterating Through Strings

```
        word = input("Enter a word: ")
        for ch in word:
        print(ch)
```

4.4 Counting Vowels Example

```
        def count_vowels(s):
        vowels = "aeiouAEIOU"
        count = 0
        for ch in s:
        if ch in vowels:
        count += 1
        return count

        sentence = input("Enter sentence: ")
        print("Number of vowels:", count_vowels(sentence))
```

5. Functions with Lists and Loops

Listing 2: Average grade using functions

```
        def average_grade(grades):
        total = sum(grades)
        return total / len(grades)

        grades = []
        for i in range(3):
        g = float(input(f"Enter grade {i+1}: "))
        grades.append(g)

        avg = average_grade(grades)
        print("Average =", avg)

        if avg >= 60:
        print("Status: PASS")
        else:
        print("Status: FAIL")
```

6. Exercises

Exercise 1: Factorial Function

Write a function that computes factorial of a number using a loop. **Hint:** $n! = n \times (n-1) \times (n-2) \ldots 1$

```
        def factorial(n):
        result = 1
        for i in range(1, n+1):
        result *= i
        return result
```

Exercise 2: String Reversal

Write a function `reverse_string(text)` that reverses a string manually.

Exercise 3: Word Counter

Count how many words appear in a sentence using `split()` and `len()`.

Exercise 4: Menu Program

Implement a menu using functions:

```
1. Count vowels
2. Reverse string
3. Exit
```

Each choice should call a separate function.

7. Mini Project: Student Gradebook System

Objective: Integrate loops, lists, strings, and functions.

Description:

- Input names and grades for several students.
- Compute each student's average.
- Print a summary report.

Listing 3: Gradebook Program

```
        def compute_average(marks):
        return sum(marks)/len(marks)

        def print_report(names, all_marks):
        print("\n---- Student Report ----")
        for i in range(len(names)):
```

```
        avg = compute_average(all_marks[i])
        print(f"{names[i]:10} -> Average: {avg:.2f}")

        students = int(input("Number of students: "))
        names = []
        marks = []

        for i in range(students):
        name = input(f"Enter name of student {i+1}: ")
        scores = []
        for j in range(3):
        s = float(input(f"  Enter score {j+1}: "))
        scores.append(s)
        names.append(name)
        marks.append(scores)

        print_report(names, marks)
```

8. Assessment Rubric

Task	Marks	Remarks
Function Basics (Def/Call/Return)	15	Correct syntax and logic
String Processing Exercises	20	Proper use of methods and loops
List + Function Integration	15	Working average computation
Mini Project Implementation	25	Modular design and testing
Documentation & Clarity	10	Comments and formatting
Viva / Discussion	15	Conceptual understanding
Total	**100**	

9. Viva Questions

1. What are the advantages of using functions?
2. What happens if a function has no return statement?
3. Are strings mutable or immutable in Python? Why?
4. How do you pass multiple values back from a function?
5. Explain the difference between local and global variables.

10. Homework

Task: Write a program that checks if a string is a palindrome using a function. **Hint:** A palindrome reads the same forward and backward.

Example:

```
Input: radar
Output: Yes, it is a palindrome.
```

11. Summary

In this lab, we learned to:

- Write and use functions with parameters and return values.
- Manipulate strings using slicing and methods.
- Integrate loops and functions for structured programming.
- Build modular programs for reusability and clarity.

End of Lab 03 – Functions, Strings, and Modular Programming