Programming Fundamentals - Week 3 Lectures

Authors: Refat Othman and Diaeddin Rimawi

Lecture 1: Functions and File Handling in Python

Lecture Goals:

- Understand the importance of modularity through functions.
- Learn function types, definitions, parameters, return values, and scope.
- Understand recursion and how functions can call other functions.
- Learn how to handle file input and output operations in Python.

Topic 1: Introduction to Functions

What is a function and why we need it?

A function is a reusable block of code that performs a specific task. It helps reduce redundancy and improves code clarity.

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

greet()
```

Topic 2: Function Definitions and Types

Function without parameters and no return:

```
def say_hello():
    print("Hello!")

say_hello()
```

Function with parameters but no return:

```python
def greet_user(name):
    print("Hello,", name)

greet_user("Ali")
```

Function with parameters and a return value:

```python
def add(a, b):
    return a + b

result = add(3, 5)
print(result)
```

Function without parameters but with return:

```python
def get_number():
    return 42

print(get_number())
```

Topic 3: Function Calls and Recursion

Function calling another function:

```python
def square(x):
    return x * x

def print_square(y):
    print("Square is:", square(y))

print_square(4)
```

Recursive function (e.g., factorial):

```python
def factorial(n):
    if n == 0:
        return 1
    return n * factorial(n - 1)

print(factorial(5))
```

Topic 4: Bubble Sort as a Function

Sorting an array using bubble sort: This function takes a list of numbers and sorts them in ascending order using the bubble sort algorithm.

```
def bubble_sort(arr):
    n = len(arr)
    for i in range(n):
        for j in range(0, n - i - 1):
            if arr[j] > arr[j + 1]:
                arr[j], arr[j + 1] = arr[j + 1], arr[j]

numbers = [64, 34, 25, 12, 22, 11, 90]
bubble_sort(numbers)
print("Sorted array:", numbers)
```

Topic 5: Local and Global Variables

Local vs Global Example:

```
x = 10  # global variable

def func():
    x = 5  # local variable
    print("Local x:", x)

func()
print("Global x:", x)
```

Modifying global variable inside function:

```
count = 0

def increment():
    global count
    count += 1

increment()
print(count)
```

Topic 6: Files in Python

Writing to a file:

```
with open("data.txt", "w") as f:
    f.write("Hello File\\n")
    f.write("Python I/O")
```

Reading from a file:

```
import os

if os.path.exists("data.txt"):
    with open("data.txt", "r") as f:
        content = f.read()
        print(content)
else:
    print("File does not exist.")
```

Appending to a file:

```
with open("data.txt", "a") as f:
    f.write("\\nAppended line")
```

Lecture 2: Practical Python Exercises

Lecture Goals:

- Reinforce programming fundamentals through practice.
- Encourage problem solving and structured thinking.

Exercise 1: Temperature Converter

- Write a function that takes a Celsius temperature and returns the Fahrenheit equivalent.
- Use input/output functions and function call.
- Save the result to a file.

Exercise 2: Grade Evaluator

- Read a list of student marks from a file (one mark per line).
- Calculate the average of the class.
- Display the average on the screen.

Exercise 3: Number Statistics

- Let the user enter a list of integers.
- Use loops to find max, min, count positives/negatives.
- Display summary in the console and write it to a file.

Exercise 4: Login System

- Predefine usernames and passwords.
- Use a loop to validate user input.
- Use a function for validation.
- Log login attempts to a file.

Exercise 5: Recursive Calculator

- Implement a recursive function to calculate the power of a number (e.g., a^b).
- Create a menu that allows the user to choose to calculate power, square, or exit.

Assignments (To be solved individually by students)

Assignment 1: Word Counter

- Ask the user for a file name.
- Count and display the number of words in the file.
- Also, identify the most frequent word and how many times it appears.

Assignment 2: Prime Number Analyzer

- Let the user input a range of numbers.
- Use a function to check if a number is prime.
- Write all prime numbers in that range to a file, one per line.