CIS 103: Fundamentals of Programming

Lab 4: Python Practice – Group Assignment

Due Date: 10/05/2024 @ 11:59pm
Presentation Date: 09/28/2024 @ 10:00am

Objective:
The purpose of this lab is to practice and reinforce Python programming concepts to set a foundation of programming

Lab Instructions:
- Complete each of the exercises below.
- Write and test your code in Python.
- Make sure to follow best practices for coding (e.g., commenting, proper variable names, etc.).
- Work in groups, you will be presenting next lab day
- Submit your Python scripts via Brightspace and/or your GitHub repo.

Part 1: Short Practice – Individual
- Write a program that checks if a number is positive, negative, or zero
- Write a for loop that prints every second number between 0 and 10
- Create a while loop that prints numbers from 1 to 100
- Loop through rows and columns of a matrix that you created
- Write a function that takes two numbers as arguments and returns their sums
- Write a function that calculate that area of a rectangle, with a default length and width
- Use a try-except mentioned in last lecture to handle user input errors in the calculator program from lab 1

Part 2: Mini Project – Group Assignment
In this lab assignment, students will work in groups to design, implement, and present a programming solution for a chosen topic. Each group will select one topic from the list below. The goal is to apply what you've learned in class to solve a real-world problem or create a functional application.

Each group will:

1. Choose a topic.
2. Plan the solution.
3. Write the code as a group.
4. Test the solution thoroughly.
5. Present your code and demonstrate its functionality in the next class.

1) Text-Based Calculator

- Build a text-based calculator that can handle basic operations: addition, subtraction, multiplication, and division using user input with try-catch blocks for exception handling.
- Bonus: Add support for advanced operations like exponents, square roots, or percentages.
- Focus Areas: Variables, arithmetic operators, user input, loops, and conditional statements.

2) Student Grades Management System

- Create a program that allows users to input student names and grades. The program should be able to calculate the class average, highest and lowest grades, and display all students' grades.
- Bonus: Add features such as grade categorization (e.g., A, B, C) and handling missing data.
- Focus Areas: Lists, dictionaries, loops, conditional logic, and file handling (optional).

3) Simple To-Do List Application

- Design a console-based to-do list application that allows users to add tasks, remove tasks, mark them as complete, and display the current list of tasks.
- Bonus: Store the tasks in a file so that the list is saved even when the program is closed.
- Focus Areas: Lists, dictionaries, file handling (optional), and loops.

4) Library Book Management System

- Create a program that manages a collection of books in a library. The system should allow users to add books, remove books, and search for a book by title or author.
- Bonus: Add functionality to track whether a book is checked out or available.
- Focus Areas: Lists, dictionaries, loops, and user input.

5) Rock, Paper, Scissors Game

- Build a two-player or player-vs-computer Rock, Paper, Scissors game. The game should allow for multiple rounds and declare a winner at the end.
- Bonus: Keep track of player scores and display a final scoreboard.
- Focus Areas: Functions, loops, conditional statements, and randomness (`random` module).

6) Simple Quiz Application

- Develop a quiz application that allows users to take a multiple-choice quiz. The program should keep track of correct answers and display the final score at the end.
- Bonus: Allow the user to input their own questions and answers before taking the quiz.
- Focus Areas: Lists, dictionaries, loops, conditional logic, and string formatting.

7) Number Guessing Game

- Create a number guessing game where the program randomly selects a number, and the user has to guess it. The program should give feedback on whether the guess is too high, too low, or correct.
- Bonus: Add a limit to the number of guesses and include difficulty levels.
- Focus Areas: Random numbers (`random` module), loops, and conditional statements.

Part 3: Peer Evaluation

After the project, each group member will submit a Peer Evaluation Form. On the form, evaluate your group members' contributions on a scale from 1 to 5 (1 = minimal contribution, 5 = outstanding contribution) in the following areas:

- Research

- Writing

- Teamwork

Provide a brief written explanation for each rating. Your peer evaluation score will affect 30% of your individual project grade.

Part 4: Submission

- **Code**: The group must submit a working Python program. Be sure to use comments in your code to explain key parts of the logic.
- **Documentation**: Provide a brief write-up (one page) explaining the approach you took, how you divided the tasks among group members, and any challenges you encountered.
- **Presentation**: Each group will present their solution in the next class. Be prepared to:
    1. Run the code and demonstrate how it works.
    2. Explain the logic behind your solution.
    3. Answer any questions from the instructor or classmates.
- **Individual Work**: Please submit in one zip file on Brightspace

Grading Criteria:

- Each exercise will be graded for correctness, code clarity, and proper use of Python constructs.
- Make sure your code is well-documented with comments explaining each part of the process.