Python Week -7

Assignment Questions

Theory Assignment:

1. **What are the different types of errors that can occur in a Python program, and how are they classified?**

2. **Explain the concept of exceptions in Python programming. How are exceptions raised and handled?**

3. **Describe the significance of syntax errors in Python. How does the interpreter handle syntax errors?**

4. **Discuss the role of built-in exceptions in Python. Provide examples of commonly occurring built-in exceptions and their explanations.**

5. **How can exceptions be forcefully raised in Python programs? Explain the use of the `raise` and `assert` statements with suitable examples.**

6. **What is the purpose of the `try...except` block in Python? How does it help in handling exceptions? Provide a detailed explanation.**

7. **Explain the process of catching exceptions using the `try...except` block in Python. Illustrate with examples.**

8. **Discuss the use of the `finally` clause in Python exception handling. When and why is it used? Provide examples to support your explanation.**

9. **How can multiple exceptions be handled in Python using the `try...except` block? Provide examples demonstrating the handling of multiple exceptions.**

10. **What is the significance of the `else` clause in Python exception handling? How does it complement the `try...except` block? Provide examples to elucidate its usage.**

11. **What is operator overloading in Python, and how does it allow for customization of built-in operators?**

12. **Explain the significance of magic methods (or dunder methods) in Python operator overloading. Provide examples of commonly used magic methods for operator overloading.**

13. **Describe the process of implementing operator overloading in a custom Python class. Provide an example demonstrating the overloading of addition and subtraction operators.**

14. **How do comparison operators and assignment operators contribute to operator overloading in Python? Provide examples illustrating their usage in custom class implementations.**

15. **Discuss the advantages of operator overloading in Python programming. How does it improve code readability, expressiveness, and maintainability?**

Programming Assignment:

1. Implement a temperature converter program in Python that converts Celsius to Fahrenheit and vice versa. However, include exception handling to handle cases where the user enters invalid input (e.g., non-numeric values or out-of-range temperatures).

2. Write a Python program to solve quadratic equations of the form ax^2 + bx + c = 0. Ensure to handle exceptions such as division by zero, and ValueError for invalid input coefficients.

3. Develop a simple calculator program in Python that performs basic arithmetic operations (addition, subtraction, multiplication, division). Incorporate exception handling to deal with division by zero errors and invalid input from the user.

4. Design a ListConcatenator class in Python that overloads the addition operator '+' to concatenate lists. The class should allow users to concatenate lists of any type and provide flexibility in the order of concatenation.

5. Design a student grading system using Python where users can input student names and their corresponding grades. Ensure to handle exceptions for invalid input grades (e.g., grades outside the valid range of 0 to 100).

6. Build a basic banking system simulator in Python where users can deposit and withdraw money from their accounts. Implement exception handling to prevent overdrafts (withdrawals exceeding the account balance) and handle invalid input.

7. Develop a scientific calculator program in Python that includes advanced mathematical functions (e.g., square root, logarithm). Use exception handling to manage errors such as square root of negative numbers or logarithm of zero.

8. Create a number guessing game in Python where the computer generates a random number, and the user has to guess it. Implement error handling to manage situations where the user enters non-numeric input or guesses outside the valid range.

9. Write a Python program for a geometry calculator that calculates the area and perimeter of various shapes (e.g., circle, rectangle). Use exception handling to handle division errors when calculating certain parameters (e.g., area of a circle with zero radius).
10. Create a Matrix class in Python that enables matrix multiplication using operator overloading. The class should allow users to define matrices of arbitrary dimensions and perform matrix multiplication using the '*' operator.

11. Implement a Polynomial class in Python that supports arithmetic operations on polynomials using operator overloading. The class should allow users to define polynomials of any degree and perform addition, subtraction, and multiplication operations on them.