Python for Computer Science and Data Science 2 (CSE 3652)

MINOR ASSIGNMENT-1: OBJECT-ORIENTED PROGRAMMING (OOP)

1. What is the significance of classes in Python programming, and how do they contribute to object-oriented programming?

2. Create a custom Python class for managing a bank account with basic functionalities like deposit and withdrawal?

3. Create a Book class that contains multiple Chapters, where each Chapter has a title and page count. Write code to initialize a Book object with three chapters and display the total page count of the book.

4. How does Python enforce access control to class attributes, and what is the difference between public, protected, and private attributes?

5. Write a Python program using a Time class to input a given time in 24-hour format and convert it to a 12-hour format with AM/PM. The program should also validate time strings to ensure they are in the correct HH:MM:SS format. Implement a method to check if the time is valid and return an appropriate message.

6. Write a Python program that uses private attributes for creating a BankAccount class. Implement methods to deposit, withdraw, and display the balance, ensuring direct access to the balance attribute is restricted. Explain why using private attributes can help improve data security and prevent accidental modifications.

7. Write a Python program to simulate a card game using object-oriented principles. The program should include a Card class to represent individual playing cards, a Deck class to represent a deck of cards, and a Player class to represent players receiving cards. Implement a shuffle method in the Deck class to shuffle the cards and a deal method to distribute cards to players. Display each player's hand after dealing.

8. Write a Python program that defines a base class Vehicle with attributes make and model, and a method display_info(). Create a subclass Car that inherits from Vehicle and adds an additional attribute num_doors. Instantiate both Vehicle and Car objects, call their display_info() methods, and explain how the subclass inherits and extends the functionality of the base class.

9. Write a Python program demonstrating polymorphism by creating a base class Shape with a method area(), and two subclasses Circle and Rectangle that override the area() method. Instantiate objects of both subclasses and call the area() method. Explain how polymorphism simplifies working with different shapes in an inheritance hierarchy.

10. Implement the CommissionEmployee class with __init__, earnings, and __repr__ methods. Include properties for personal details and sales data. Create a test script to instantiate the object, display earnings, modify sales data, and handle data validation errors for negative values.

11. What is duck typing in Python? Write a Python program demonstrating duck typing by creating a function describe() that accepts any object with a speak() method. Implement two classes, Dog and Robot, each with a speak() method. Pass instances of both classes to the describe() function and explain how duck typing allows the function to work without checking the object's type.

12. WAP to overload the + operator to perform addition of two complex numbers using a custom Complex class?

13. WAP to create a custom exception class in Python that displays the balance and withdrawal amount when an error occurs due to insufficient funds?

14. Write a Python program using the Card data class to simulate dealing 5 cards to a player from a shuffled deck of standard playing cards. The program should print the player's hand and the number of remaining cards in the deck after the deal.

15. How do Python data classes provide advantages over named tuples in terms of flexibility and functionality? Give an example using python code.

16. Write a Python program that demonstrates unit testing directly within a function's docstring using the doctest module. Create a function add(a, b) that returns the sum of two numbers and includes multiple test cases in its docstring. Implement a way to automatically run the tests when the script is executed.

17. Scope Resolution: object's namespace → class namespace → global namespace → built-in namespace.

```
species = Global Species

class Animal:
    species = Class Species

    def __init__(self, species):
        self.species = species

    def display_species(self):
        print("Instance species:", self.species)
        print("Class species:", Animal.species)
        print("Global species:", globals()['species'])

a = Animal("Instance Species")
a.display_species()
```

What will be the output when the above program is executed? Explain the scope resolution process step by step.

18. Write a Python program using a lambda function to convert temperatures from Celsius to Kelvin, store the data in a tabular format using `pandas`, and visualize the data using a plot.