OOPS_Part-1

Assignment Questions

PW SKILLS

Theory Assignment:

1. What is Object-Oriented Programming (OOP), and how does it differ from procedural programming?

2. Define the terms Class, Attribute, and Object in the context of OOP in Python.

3. Explain the role of instances in Python classes. How are instances created, and what purpose do they serve?

4. Describe the process of defining a class in Python, including the syntax and components involved.

5. How are functions defined within a class in Python? What is the significance of the 'self' parameter?

6. Explain how class functions can be called both from outside and inside the class in Python, providing examples for each scenario.

7. Differentiate between class attributes and instance attributes in Python, and provide examples of each.

8. Differentiate between class method ,instance method and static method in Python, and provide examples of each.

9. What is a constructor in Python classes? How is it defined, and what is its purpose?

10. What are access specifiers in Python ,explain with examples.

11. What is encapsulation in python , explain it with an example.How does it promote data hiding and abstraction, and why is it important in software design.

12. Explain the Getter and Setter method in Python with examples, and explain its importance.

13. Discuss the concept of inheritance and explain all types of inheritance with suitable examples in Python. How are subclasses created, and what advantages does inheritance offer?

14. Explain the concept of method overriding in Python classes. How does it allow subclasses to modify the behavior of superclass methods?

15. Discuss the significance of the 'super()' function in Python inheritance. How is it used, and what role does it play in accessing superclass methods and attributes?

16. Explain MRO (Method Resolution Order) in detail with example.Explain its importance.

Programming Assignments:

1. Design and implement a Python program for managing student information using object-oriented principles. Create a class called `Student` with encapsulated attributes for name, age, and roll number. Implement getter and setter methods for these attributes. Additionally, provide methods to display student information and update student details.

Tasks:

- Define the `Student` class with encapsulated attributes.
- Implement getter and setter methods for the attributes.
- Write methods to display student information and update details.
- Create instances of the `Student` class and test the implemented functionality.

2.Develop a Python program for managing library resources efficiently. Design a class named `LibraryBook` with attributes like book name, author, and availability status. Implement methods for borrowing and returning books while ensuring proper encapsulation of attributes.

Tasks:

- Create the `LibraryBook` class with encapsulated attributes.
- Implement methods for borrowing and returning books.
- Ensure proper encapsulation to protect book details.
- Test the borrowing and returning functionality with sample data.

3.Create a simple banking system using object-oriented concepts in Python. Design classes representing different types of bank accounts such as savings and checking. Implement methods for deposit, withdraw, and balance inquiry. Utilize inheritance to manage different account types efficiently.

Tasks:

- Define base class(es) for bank accounts with common attributes and methods.
- Implement subclasses for specific account types (e.g., SavingsAccount, CheckingAccount).
- Provide methods for deposit, withdraw, and balance inquiry in each subclass.
- Test the banking system by creating instances of different account types and performing transactions.

4.Write a Python program that models different animals and their sounds. Design a base class called `Animal` with a method `make_sound()`. Create subclasses like `Dog` and `Cat` that override the `make_sound()` method to produce appropriate sounds.

Tasks:

- Define the `Animal` class with a method `make_sound()`.
- Create subclasses `Dog` and `Cat` that override the `make_sound()` method

- Implement the sound generation logic for each subclass.
- Test the program by creating instances of `Dog` and `Cat` and calling the `make_sound()` method.

5.Write a code for Restaurant Management System Using OOPS:

- Create a MenuItem class that has attributes such as name, description, price, and category.
- Implement methods to add a new menu item, update menu item information, and remove a menu item from the menu.
- Use encapsulation to hide the menu item's unique identification number.
- Inherit from the MenuItem class to create a FoodItem class and a BeverageItem class, each with their own specific attributes and methods.

6.Write a code for Hotel Management System using OOPS :

- Create a Room class that has attributes such as room number, room type, rate, and availability (private).
- Implement methods to book a room, check in a guest, and check out a guest.
- Use encapsulation to hide the room's unique identification number.
- Inherit from the Room class to create a SuiteRoom class and a StandardRoom class, each with their own specific attributes and methods.

7.Write a code for Fitness Club Management System using OOPS:

- Create a Member class that has attributes such as name, age, membership type, and membership status (private).
- Implement methods to register a new member, renew a membership, and cancel a membership.
- Use encapsulation to hide the member's unique identification number.
- Inherit from the Member class to create a FamilyMember class and an IndividualMember class, each with their own specific attributes and methods.

8.Write a code for Event Management System using OOPS:

- Create an Event class that has attributes such as name, date, time, location, and list of attendees (private).
- Implement methods to create a new event, add or remove attendees, and get the total number of attendees.
- Use encapsulation to hide the event's unique identification number.
- Inherit from the Event class to create a PrivateEvent class and a PublicEvent class, each with their own specific attributes and methods.

9.Write a code for Airline Reservation System using OOPS:

- Create a Flight class that has attributes such as flight number, departure and arrival airports, departure and arrival times, and available seats (private).
- Implement methods to book a seat, cancel a reservation, and get the remaining available seats.
- Use encapsulation to hide the flight's unique identification number.
- Inherit from the Flight class to create a DomesticFlight class and an InternationalFlight class, each with their own specific attributes and methods.

10.Write a code for HR Management System using OOPS:

- Create an Employee class that has attributes such as name, employee ID (private), job title, and department.
- Implement methods to hire a new employee, update an employee's information, and terminate an employee.
- Use encapsulation to hide the employee's unique identification number.
- Inherit from the Employee class to create a Manager class and an Intern class, each with their own specific attributes and methods.

11.Write a Python program to create a class representing a shopping cart. Include methods for adding and removing items, and calculating the total price.

12.Write a Python program to create a calculator class. Include methods for basic arithmetic operations.