

Python OOPs

Assignment Questions



Python OOPs Questions

1. What is Object-Oriented Programming (OOP)?
2. What is a class in OOP?
3. What is an object in OOP?
4. What is the difference between abstraction and encapsulation?
5. What are dunder methods in Python?
6. Explain the concept of inheritance in OOP.
7. What is polymorphism in OOP?
8. How is encapsulation achieved in Python?
9. What is a constructor in Python?
10. What are class and static methods in Python?
11. What is method overloading in Python?
12. What is method overriding in OOP?
13. What is a property decorator in Python?
14. Why is polymorphism important in OOP?
15. What is an abstract class in Python?
16. What are the advantages of OOP?
17. What is the difference between a class variable and an instance variable?
18. What is multiple inheritance in Python?
19. Explain the purpose of “__str__” and ‘__repr__’ methods in Python.
20. What is the significance of the ‘super()’ function in Python?
21. What is the significance of the __del__ method in Python?
22. What is the difference between @staticmethod and @classmethod in Python?
23. How does polymorphism work in Python with inheritance?
24. What is method chaining in Python OOP?
25. What is the purpose of the __call__ method in Python?

Practical Questions

1. Create a parent class `Animal` with a method `speak()` that prints a generic message. Create a child class `Dog` that overrides the `speak()` method to print "Bark!".
2. Write a program to create an abstract class `Shape` with a method `area()`. Derive classes `Circle` and `Rectangle` from it and implement the `area()` method in both.
3. Implement a multi-level inheritance scenario where a class `Vehicle` has an attribute `type`. Derive a class `Car` and further derive a class `ElectricCar` that adds a `battery` attribute.
4. Demonstrate polymorphism by creating a base class `Bird` with a method `fly()`. Create two derived classes `Sparrow` and `Penguin` that override the `fly()` method.
5. Write a program to demonstrate encapsulation by creating a class `BankAccount` with private attributes `balance` and methods to `deposit`, `withdraw`, and `check balance`.
6. Demonstrate runtime polymorphism using a method `play()` in a base class `Instrument`. Derive classes `Guitar` and `Piano` that implement their own version of `play()`.
7. Create a class `MathOperations` with a class method `add_numbers()` to add two numbers and a static method `subtract_numbers()` to subtract two numbers.
8. Implement a class `Person` with a class method to count the total number of persons created.
9. Write a class `Fraction` with attributes `numerator` and `denominator`. Override the `str` method to display the fraction as "numerator/denominator".
10. Demonstrate operator overloading by creating a class `Vector` and overriding the `add` method to add two vectors.
11. Create a class `Person` with attributes `name` and `age`. Add a method `greet()` that prints "Hello, my name is {name} and I am {age} years old."
12. Implement a class `Student` with attributes `name` and `grades`. Create a method `average_grade()` to compute the average of the grades.
13. Create a class `Rectangle` with methods `set_dimensions()` to set the dimensions and `area()` to calculate the area.
14. Create a class `Employee` with a method `calculate_salary()` that computes the salary based on hours worked and hourly rate. Create a derived class `Manager` that adds a bonus to the salary.

15. Create a class `Product` with attributes `name`, `price`, and `quantity`. Implement a method `total_price()` that calculates the total price of the product.
16. Create a class `Animal` with an abstract method `sound()`. Create two derived classes `Cow` and `Sheep` that implement the `sound()` method.
17. Create a class `Book` with attributes `title`, `author`, and `year_published`. Add a method `get_book_info()` that returns a formatted string with the book's details.
18. Create a class `House` with attributes `address` and `price`. Create a derived class `Mansion` that adds an attribute `number_of_rooms`.