Python OOPS Theory Assignment

What is Object-Oriented Programming (OOP)?

Object-Oriented Programming (OOP) is a programming paradigm that uses "objects" to represent data and methods to manipulate that data. OOP focuses on using objects to model real-world entities, promoting code reusability, scalability, and maintainability. Key concepts include classes, objects, inheritance, encapsulation, abstraction, and polymorphism.

What is a class in OOP?

A class is a blueprint for creating objects. It defines a set of attributes (data) and methods (functions) that the created objects will have. Classes encapsulate data for the object and provide methods to manipulate that data.

What is an object in OOP?

An object is an instance of a class. It is created from a class and can hold data and perform actions defined by the class. Each object can have different values for its attributes.

What is the difference between abstraction and encapsulation?

Abstraction is the concept of hiding the complex reality while exposing only the necessary parts. It helps in reducing programming complexity and increasing efficiency. Encapsulation, on the other hand, is the bundling of data (attributes) and methods (functions) that operate on the data into a single unit or class, restricting access to some of the object's components.

What are dunder methods in Python?

Dunder methods, or "double underscore" methods, are special methods in Python that start and end with double underscores. They allow you to define the behavior of objects for built-in operations (e.g., __init__ for object initialization, __str__ for string representation, __add__ for addition).

Explain the concept of inheritance in OOP.

Inheritance is a mechanism where a new class (child class) derives properties and behavior (methods) from an existing class (parent class). This promotes code reusability and establishes a relationship between classes.

What is polymorphism in OOP?

Polymorphism allows methods to do different things based on the object it is acting upon, even if they share the same name. It can be achieved through method overriding (inherited classes) and method overloading (same method name with different parameters).

How is encapsulation achieved in Python?

Encapsulation in Python is achieved by using private and protected access modifiers. Attributes can be made private by prefixing them with double underscores (__), restricting access from outside the class.

What is a constructor in Python?

A constructor is a special method called when an object is instantiated. In Python, the constructor is defined using the __init__ method. It initializes the object's attributes.

What are class and static methods in Python?

Class methods are methods that are bound to the class and not the instance of the class. They are defined using the @classmethod decorator and take cls as the first parameter.

Static methods are similar but do not take any reference to the class or instance. They are defined using the @staticmethod decorator.

What is method overloading in Python?

Method overloading allows multiple methods with the same name but different parameters. However, Python does not support method overloading directly; instead, you can achieve similar functionality using default arguments or variable-length arguments.

What is method overriding in OOP?

Method overriding occurs when a child class provides a specific implementation of a method that is already defined in its parent class. This allows the child class to modify or extend the behavior of the parent class.

What is a property decorator in Python?

The property decorator in Python is used to define methods in a class that can be accessed like attributes. It allows for getter, setter, and deleter methods to manage access to private attributes.

Why is polymorphism important in OOP?

Polymorphism is important because it allows for flexibility and the ability to use a unified interface for different data types. It enables code to be more generic and reusable.

What is an abstract class in Python?

An abstract class is a class that cannot be instantiated and is meant to be subclassed. It can contain abstract methods that must be implemented by derived classes. Abstract classes are defined using the `abc` module in Python.

What are the advantages of OOP?

- Advantages of OOP include:
  - Code reusability through inheritance.
  - Improved maintainability and scalability.
  - Encapsulation for data protection.
  - Abstraction for simplifying complex systems.
  - Polymorphism for flexibility in code.

What is the difference between a class variable and an instance variable?

A class variable is shared among all instances of a class and is defined within the class but outside any instance methods. An instance variable, on the other hand, is unique to each instance of a class and is defined within instance methods, typically in the constructor.

What is multiple inheritance in Python?

Multiple inheritance is a feature in Python where a class can inherit attributes and methods from more than one parent class. This allows for a more complex hierarchy but can lead to ambiguity if not managed carefully.

Explain the purpose of __str__ and __repr__ methods in Python.

The __str__ method is used to define a human-readable string representation of an object, while the __repr__ method is intended to provide an unambiguous string representation that can be used to recreate the object. The __str__ method is called by the print() function, whereas __repr__ is called by the repr() function.

What is the significance of the super() function in Python?

The super() function is used to call methods from a parent class in a child class. It allows for the proper initialization of inherited attributes and methods, facilitating the use of multiple inheritance.

What is the significance of the __del__ method in Python?

The __del__ method is a destructor that is called when an object is about to be destroyed. It can be used to release resources or perform cleanup actions before the object is removed from memory.

What is the difference between @staticmethod and @classmethod in Python?

@staticmethod defines a method that does not require access to the instance or class and can be called on the class itself.

@classmethod, on the other hand, takes a reference to the class (cls) as its first parameter and can modify class state that applies across all instances.

How does polymorphism work in Python with inheritance?

Polymorphism in Python with inheritance allows derived classes to implement methods defined in a base class. When a method is called on an object, Python determines which method to execute based on the object's class, enabling different behaviors for the same method name.

What is method chaining in Python OOP?

Method chaining is a technique where multiple method calls are made on the same object in a single statement. This is achieved by returning self from each method, allowing for a fluent interface.

What is the purpose of the __call__ method in Python?

The __call__ method allows an instance of a class to be called as if it were a function. This can be useful for creating callable objects that maintain state.