Python OOPs

Assignment Solutions

SKILLS

Python OOPs Solutions

Q1. What is Object-Oriented Programming (OOP)?

- OOP is a programming paradigm based on the concept of "objects," which can contain data in the form of fields (attributes) and code in the form of methods (functions). It provides principles like inheritance, encapsulation, polymorphism, and abstraction to design programs.

Q2. What is a class in OOP?

- A class is a blueprint or template for creating objects. It defines the properties (attributes) and behaviors (methods) that the objects created from the class will have.

Q3. What is an object in OOP?

- An object is an instance of a class that holds actual data and can perform actions using methods defined in the class.

Q4. What is the difference between abstraction and encapsulation?

- Abstraction focuses on hiding the complexity of a system and showing only the essential features.
- Encapsulation is about bundling the data and methods that operate on the data into a single unit and restricting direct access to some components.

Q5. What are dunder methods in Python?

- Dunder (double underscore) methods, also called magic methods, are special methods in Python with a double underscore before and after their name, such as ‘__init__’, ‘__str__’, and ‘__len__’. These methods allow customization of basic operations in a class.

Q6. Explain the concept of inheritance in OOP.

- Inheritance is the mechanism by which a class (child class) can derive properties and behaviors from another class (parent class). It allows code reuse and creates a hierarchy between classes.

Q7. What is polymorphism in OOP?

- Polymorphism means "many forms" and allows the same function or method to behave differently based on the object or data type. For example, method overloading and overriding are examples of polymorphism.

Q8. How is encapsulation achieved in Python?

- Encapsulation in Python is achieved using private and protected variables. Private variables are prefixed with ‘__’, and protected variables are prefixed with ‘_’.

Q9. What is a constructor in Python?

- A constructor is a special method in Python, defined as ‘__init__’, that is automatically called when an object of a class is created. It initializes the object’s attributes.

Q10. What are class and static methods in Python?

- Class methods: These are methods bound to the class, not the instance. They use the @classmethod decorator and take cls as the first parameter.
- Static methods: These methods are not bound to the class or instance. They use the @staticmethod decorator and do not take self or cls as a parameter.

Q11. What is method overloading in Python?

- Method overloading is the ability to define multiple methods in a class with the same name but different parameters. However, Python does not support traditional method overloading directly.

Q12. What is method overriding in OOP?

- Method overriding occurs when a subclass defines a method with the same name as a method in its parent class, providing a new implementation.

Q13. What is a property decorator in Python?

- A property decorator, @property, is used to make methods act like attributes. It allows getter, setter, and deleter functionality for managing attributes in an elegant way.

Q14. Why is polymorphism important in OOP?

- Polymorphism allows for flexibility and reusability by letting objects of different types be treated as objects of a common supertype, thus enabling generic programming.

Q15. What is an abstract class in Python?

- An abstract class is a class that cannot be instantiated and may contain one or more abstract methods. Abstract methods are declared using the @abstractmethod decorator and must be implemented in derived classes.

Q16. What are the advantages of OOP?

- OOP provides advantages like code reusability, modularity, scalability, easy maintenance, and better data security through encapsulation.

Q17. What is the difference between a class variable and an instance variable?

- Class variable: Shared by all instances of a class. Defined within the class but outside any methods.
- Instance variable: Unique to each instance of a class. Defined inside a constructor or methods using self.

Q18. 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.

Q19. Explain the purpose of ‘’__str__’ and ‘__repr__’ ‘ methods in Python.

- ‘__str__’: Used to provide a human-readable representation of an object, often used with the print() function.
- ‘__repr_’_: Used to provide a developer-friendly representation of an object, often used in debugging.

Q20. What is the significance of the 'super()' function in Python?

- The super() function is used to call the parent class methods or constructor from the child class. It ensures proper initialization in the inheritance chain.

Q21. What is the significance of the __del__ method in Python?

- '__del__' method is a special method in Python that is called when an object is about to be destroyed. It is used for cleanup, such as releasing resources or closing files. However, its use is discouraged because it can lead to issues with reference counting.

Q22. What is the difference between @staticmethod and @classmethod in Python?

- @staticmethod: Defines a method that does not require access to the instance or the class. It is independent of both.
- @classmethod: Defines a method that takes the class as its first parameter (cls). It can modify class-level attributes but cannot access instance-level attributes.

Q23. How does polymorphism work in Python with inheritance?

- Polymorphism allows methods in different classes to have the same name but different implementations. With inheritance, subclasses can override a method from the parent class, enabling different behaviors for objects of different types.

Q24. What is method chaining in Python OOP?

- Method chaining is a programming technique where multiple method calls are made in a single line, where each method returns the object itself (self). It allows you to call methods on an object one after another, making the code more concise.

Q25. What is the purpose of the __call__ method in Python?

- The __call__ method allows an instance of a class to be called as a function. By implementing this method, you can make objects callable and pass arguments to them, essentially turning an object into a function.

For answers to practical questions please refer to the link given below.
https://colab.research.google.com/drive/1r2Yt4ZJVsmZZiIzIgF_3xmc6CgdLoKrg