Assignment-01_July_OOPs

1.What is the primary goal of Object-Oriented Programming (OOP)?

Solution: The primary goal of Object-Oriented Programming is to enable the development of more modular, flexible, and maintainable software by organizing code into objects that encapsulate both data and the methods that operate on that data. OOP promotes inheritance, polymorphism, and encapsulation to improve code reusability, readability, and scalability.

2. What is an object in Python?

Solution: In Python, an object is an instance of a class. It can be a data structure, a variable, or a combination of both. Objects have attributes, which are variables that store data, and methods, which are functions that define the behavior of the object. Objects can also interact with other objects by calling their methods or accessing their attributes. Everything is an object in Python, including built-in data types like integers and strings.

3. What is a class in Python?

Solution: A class in Python is a blueprint for creating objects, which are instances of the class. It defines the structure and behavior of the objects that will be created. A class can have attributes (variables) and methods (functions) that can be accessed by the objects created from the class. It allows for code reusability and provides a way to organize and manage code.

4. What are attributes and methods in a class?
Solution: In object-oriented programming, attributes and methods are the two main components of a class.

Attributes, also known as member variables or properties, are variables that store data within an object. They define the state or characteristics of an object. Attributes can be of any data type such as integers, strings, or custom objects. Each object of a class has its own set of attributes.

Methods, also known as member functions or behaviors, are functions that define the actions or operations that an object can perform. They provide the functionality or behavior of the class. Methods can be used to manipulate the attributes of an object, perform calculations, or interact with other objects.

In summary, attributes represent the data or state of an object, while methods define the behavior or actions that an object can perform. Together, they encapsulate the data and behavior of an object within a class.

5. What is the difference between class variables and instance variables in Python?
Solution: In Python, class variables and instance variables are both used to store data within classes, but they differ in terms of their scope, access, and usage.

1) Scope and Access:

Class variables: These variables are defined within the class and are shared by all instances of the class. They are typically declared at the top level of the class definition, outside of any methods.
Instance variables: These variables are specific to each instance of a class and are created inside the constructor method (__init__) using the "self" keyword. Each instance of the class has its own copy of the instance variables.

2) Usage:
Class variables: They are commonly used to store data that is shared by all instances of a class, such as constants or properties that do not change for different instances. Class variables can be accessed through the class itself or any instance of the class.
Instance variables: They are used to store unique data for each instance of a class. These variables are accessed and modified using the instance of the class they belong to.

3) Modification:
Class variables: When a class variable is modified, its value is changed for all class instances.
Instance variables: Each instance has its own copy of instance variables, so modifying an instance variable only affects that particular instance.

Here's an example to illustrate the difference:

https://colab.research.google.com/drive/1SY7PvGrsaiksekEpKEE9xRQsHWQ7oX44?usp=sharing

6. What is the purpose of the self-parameter in Python class methods?
Solution: The self-parameter is used to refer to the instance of the class itself. It allows class methods to access and manipulate the attributes and methods of that instance. By convention, class methods in Python have the first parameter named 'self', although it can technically be named anything. Using the self-parameter, you can access instance variables, call other methods of the same instance, or even create new instance variables.

7. For a library management system, you have to design the "Book" class with OOP principles in mind. The "Book" class will have following attributes:
a. title: Represents the title of the book.
b. author: Represents the author(s) of the book.
c. isbn: Represents the ISBN (International Standard Book Number) of the book.
d. publication_year: Represents the year of publication of the book.
e. available_copies: Represents the number of copies available for checkout.
The class will also include the following methods:
a. check_out(self): Decrements the available copies by one if there are copies available for checkout.
b. return_book(self): Increments the available copies by one when a book is returned.

c. display_book_info(self): Displays the information about the book, including its attributes and the number of available copies.
Solution: https://colab.research.google.com/drive/1SY7PvGrsaiksekEpKEE9xRQsHWQ7oX44?usp=sharing

8. For a ticket booking system, you have to design the "Ticket" class with OOP principles in mind. The "Ticket" class should have the following attributes:
a. ticket_id: Represents the unique identifier for the ticket.
b. event_name: Represents the name of the event.
c. event_date: Represents the date of the event.
d. venue: Represents the venue of the event.
e. seat_number: Represents the seat number associated with the ticket.
f. price: Represents the price of the ticket.
g. is_reserved: Represents the reservation status of the ticket.
The class also includes the following methods:
a. reserve_ticket(self): Marks the ticket as reserved if it is not already reserved.
b. cancel_reservation(self): Cancels the reservation of the ticket if it is already reserved.
c. display_ticket_info(self): Displays the information about the ticket, including its attributes and reservation status.
Solution:
https://colab.research.google.com/drive/1SY7PvGrsaiksekEpKEE9xRQsHWQ7oX44?usp=sharing

9. You are creating a shopping cart for an e-commerce website. Using OOP to model the "ShoppingCart" functionality the class should contain following attributes and methods:
a. items: Represents the list of items in the shopping cart.
The class also includes the following methods:
a. add_item(self, item): Adds an item to the shopping cart by appending it to the list of items.
b. remove_item(self, item): Removes an item from the shopping cart if it exists in the list.
c. view_cart(self): Displays the items currently present in the shopping cart.
d. clear_cart(self): Clears all items from the shopping cart by reassigning an empty list to the items attribute.
Solution:
https://colab.research.google.com/drive/1SY7PvGrsaiksekEpKEE9xRQsHWQ7oX44?usp=sharing

10. Imagine a school management system. You have to design the "Student" class using OOP concepts. The "Student" class has the following attributes:
a. name: Represents the name of the student.
b. age: Represents the age of the student.
c. grade: Represents the grade or class of the student.
d. student_id: Represents the unique identifier for the student.

e. attendance: Represents the attendance record of the student.
The class should also include the following methods:
a. update_attendance(self, date, status): Updates the attendance record of the student for a given date with the provided status (e.g., present or absent).
b. get_attendance(self): Returns the attendance record of the student.
c. get_average_attendance(self): Calculates and returns the average attendance percentage of the student based on their attendance record.
Solution:
https://colab.research.google.com/drive/1SY7PvGrsaiksekEpKEE9xRQsHWQ7oX44?usp=sharing