Practical No 1

Aim : Write A Program In Order To Explain The Concepts Of Class And Objects.

1. Making A Class For Circle

Class Definition: This class represents a Circle with attributes radius, circumference, and area.
Class Methods:

- `circumferenceOfCircle(this)`: Calculates the circumference of the circle using the formula 2 * π * radius and updates the `circumference` attribute.
- `areaOfCircle(this)`: Calculates the area of the circle using the formula π * radius^2 and updates the `area` attribute.
- `displayCircumference(this)`: Prints the calculated circumference of the circle.
- `displayArea(this)`: Prints the calculated area of the circle.

This code snippet is a simple interactive program that uses a `Circle` object to calculate and display the circumference and area of a circle. Here's a step-by-step breakdown:

1. An instance of the `Circle` class is created and assigned to the variable `abc`.
2. The program enters a `while` loop that continues as long as the user chooses to proceed (`choice == "y"`).
3. Inside the loop:
    - The user is prompted to enter a radius for the circle, which is stored as an integer in the `abc.radius` attribute.
    - Two methods are called on the `abc` object: `circumferenceOfCircle()` and `areaOfCircle()`. These methods likely calculate the circumference and area of the circle based on the provided radius, but the actual implementation is not shown in this snippet.
    - The program then prints the calculated circumference and area using the `displayCircumference()` and `displayArea()` methods, respectively

2. Making Bike Class

This class definition defines a `Bike` class with two class-level variables: `name` and `gear`.

Here is a list explaining what each part of the class does:

- `name = ""`: Initializes a class-level variable `name` with an empty string value.
- `gear = 0`: Initializes a class-level variable `gear` with an integer value of 0.

This code creates four instances of the `Bike` class (`B1`, `B2`, `B3`, `B4`) and sets their `name` and `gear` attributes. It then prints out the name and gear of each bike.

3. Constructor And Destructor

a. Default Constructor

This code snippet demonstrates the concept of a default constructor in Python.

In Python, the **`init`** method is a special method that is automatically called when an object of the class is created. This method is used to initialize the attributes of the class.

In this case, the `Car` class has a default constructor (**`init`**) that sets the default values of `name`, `model`, and `year` to "BMW", "X1", and 2020 respectively.

The code then creates an instance of the `Car` class and prints the default values. It also updates the `name` attribute to "Audi" and prints the updated value.

b. Parameterized Constructor And Display Function

This Python code is an example of a class definition. The `Car` class is defined with a constructor (`__init__`) that takes three arguments: `name`, `model`, and `year`. These arguments are used to set the corresponding attributes of the `Car` object.

The `display` method of the `Car` class is used to print out the details of the car. It uses the `self` keyword to refer to the current instance of the `Car` object. The `print` statement uses string formatting to display the car's name, model, and year.

The code then creates an instance of the `Car` class called `car`, with the arguments "BMW", "X1", and 2020. This instance is then used to demonstrate accessing the car's attributes directly and through the `display` method.

d. Destructor

This code snippet demonstrates the concept of a destructor in Python.

A destructor is a special method in a class that is called when an object is about to be destroyed. In this code, the `Car` class has a destructor defined as **`del`**. When the `car` object is deleted using the `del` keyword, the destructor is automatically called and it prints "Object Destroyed".

The code creates an instance of the `Car` class with the parameters "BMW", "X1", and 2020. It then prints the name, model, and year of the `car` object. Finally, the `car` object is deleted using the `del` keyword, triggering the destructor and printing "Object Destroyed".

This code snippet is useful for performing cleanup tasks or releasing resources when an object is no longer needed.

4. Project Of Class

This code snippet is a simple employee management system. It allows users to input the number of employees, their names, and then displays the employee ID and name. The system continues to prompt the user for input until they choose to stop.

Here's a breakdown:

1. An `Employee` class is defined with attributes `id` and `name`, and methods to display employee information and handle object destruction.
2. An empty list `employees` is created to store `Employee` objects.
3. The `employeeInput` function takes user input for the number of employees and their names, creates `Employee` objects, and adds them to the `employees` list.
4. The `employeeInput` function is called repeatedly until the user chooses to stop by entering 'n'.

5. Types Of Variable

This Python code snippet declares a global variable `num` and then defines a function `change()` that modifies the global variable by adding 10 to it. The `global` keyword is used to indicate that `num` is a global variable, but in this case, it's unnecessary because the value of `num` isn't being reassigned to a new value, it's just being incremented. The function then prints the updated value of `num`.