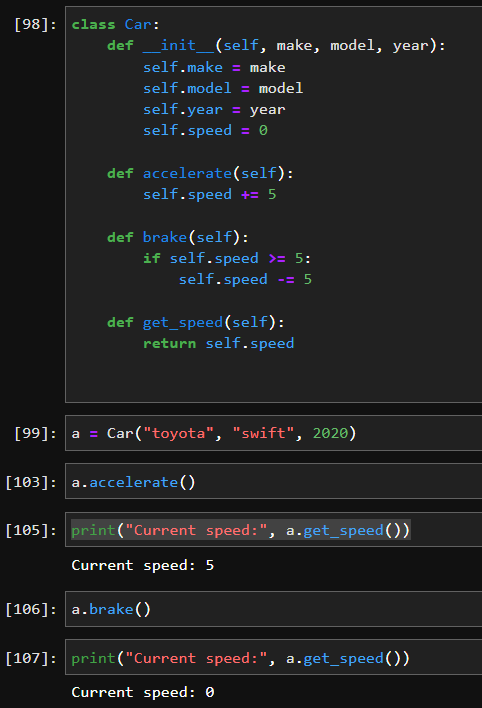
Week 4 assignment 1

Q1. Explain Class and Object with respect to Object-Oriented Programming. Give a suitable example.

Class:

- A class is a blueprint or template for creating objects. It defines a set of attributes (data members) and methods (functions) that characterize the objects of that class.

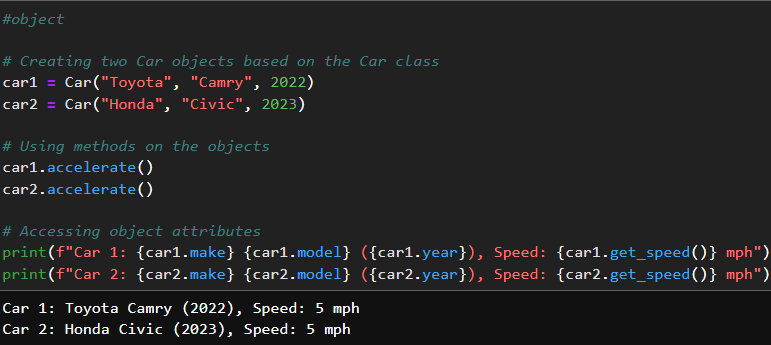
- Think of a class as a blueprint for a specific type of object. It specifies what data can be stored in an object and what actions (methods) can be performed on the object. Classes are used to encapsulate and group related data and functions together, promoting code reusability and maintainability.



Object:

- An object is an instance of a class. It represents a specific instance or occurrence of the class, with its own set of attribute values.

- Objects are created based on the blueprint defined by a class. They can have unique data while sharing the same methods defined in the class.Objects allow you to interact with and manipulate data in a structured way, as defined by the class.

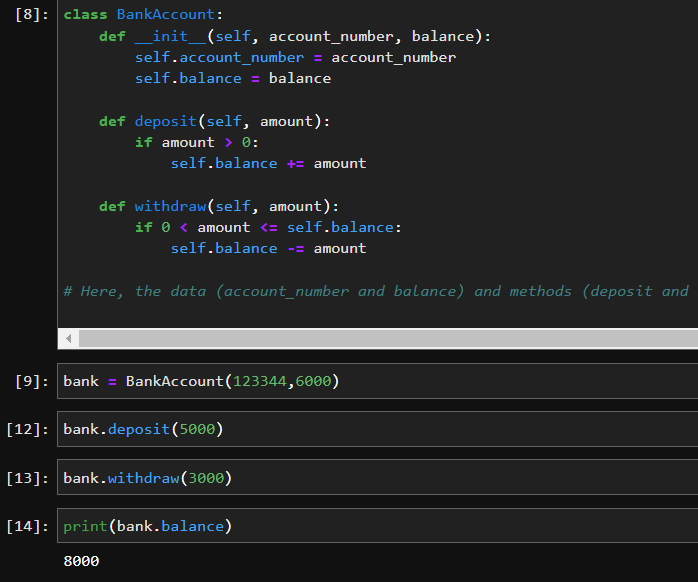


Q2. Name the four pillars of OOPs.

The four pillars of object-oriented programming (OOP) are:

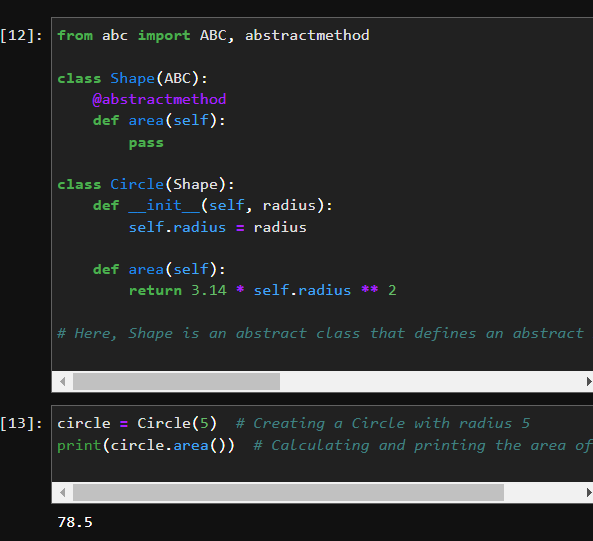
1. Encapsulation:

- Encapsulation is the concept of bundling data (attributes) and the methods (functions) that operate on that data into a single unit called a class. It restricts direct access to some of the object's components and prevents unintended interference and misuse of data.



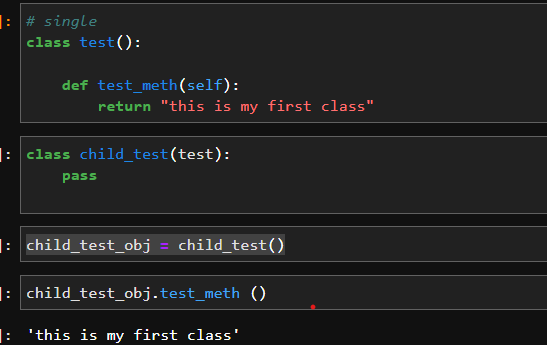
2. Abstraction:

- Abstraction is the process of simplifying complex reality by modelling classes based on the essential properties and behaviours an object should have. It hides the unnecessary details of an object and only exposes the necessary features.



3. Inheritance:

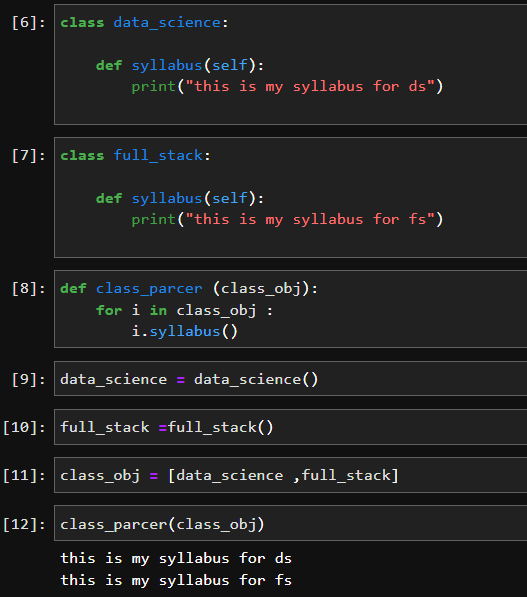
- Inheritance is a mechanism that allows a new class (derived or child class) to inherit properties and behaviours from an existing class (base or parent class).It promotes code reuse and the creation of a hierarchy of classes.



4. Polymorphism:

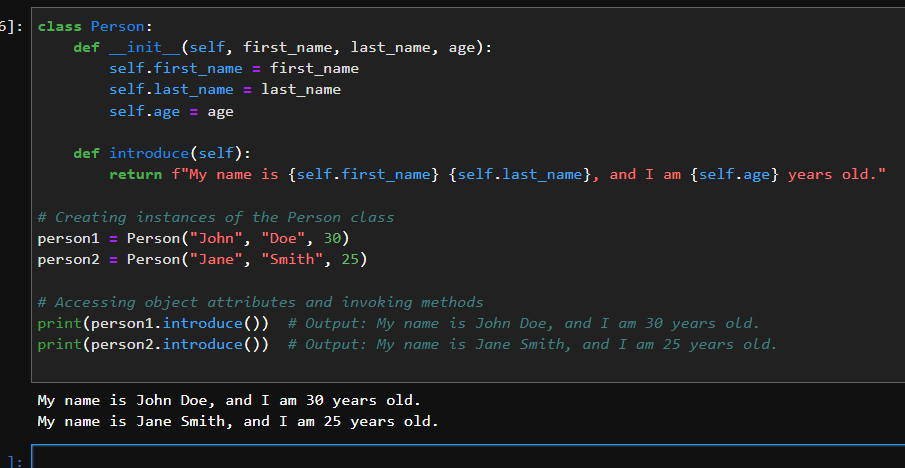
- Polymorphism allows objects of different classes to be treated as objects of a common superclass.

- It enables dynamic binding and late binding, where the method to be executed is determined at runtime based on the actual type of object.



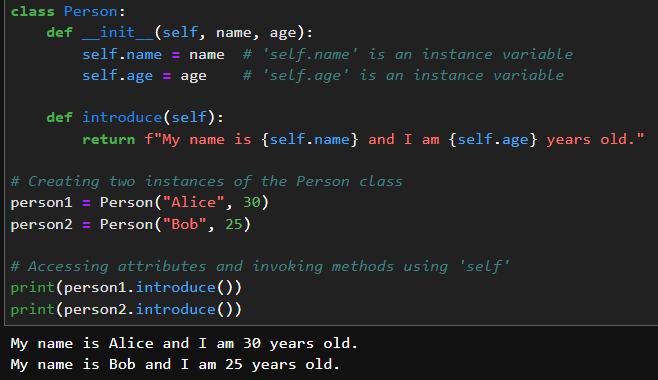
Q3. Explain why the \_\_init\_\_() function is used. Give a suitable example.

The `\_\_init\_\_()` function in Python is a special method also known as the constructor. It is used to initialize and set up the initial state of an object when an instance of a class is created. This method is automatically called when you create a new object of a class, and it allows you to specify how the object should be initialized with default values or values provided during object creation.



Q4. Why self is used in OOPs?

In object-oriented programming (OOP), `self` is a convention used in many programming languages, including Python, to refer to the current instance of a class. It is a reference to the object itself and is used within methods of the class to access and manipulate the object's attributes and methods.



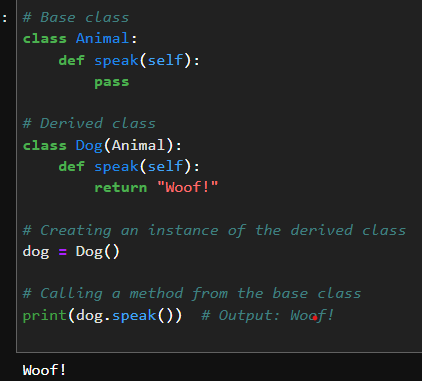
Q5. What is inheritance? Give an example for each type of inheritance.

Inheritance is a fundamental concept in object-oriented programming (OOP) that allows a new class (derived or child class) to inherit attributes and methods from an existing class (base or parent class). It promotes code reuse and the creation of a hierarchy of classes where common attributes and behaviours can be shared among related classes. Inheritance establishes an "is-a" relationship between the base class and the derived class, indicating that the derived class is a specialized version of the base class.

1. Single Inheritance:

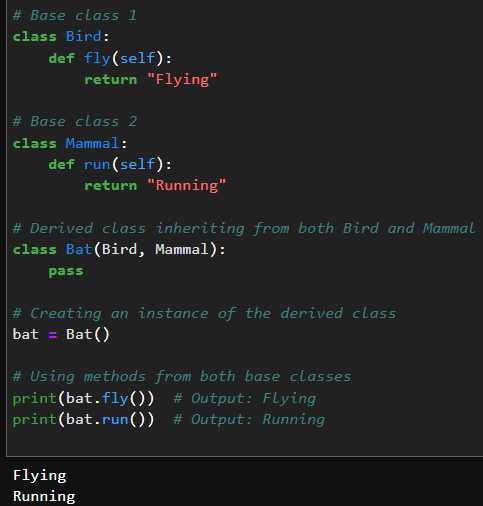
- In single inheritance, a derived class inherits from a single base class.

- It represents a one-to-one relationship between a base class and a derived class.



2. Multiple Inheritance:

- In multiple inheritance, a derived class can inherit from more than one base class. It allows a class to inherit attributes and methods from multiple parent classes.



3. Multilevel Inheritance:

- In multilevel inheritance, a derived class inherits from a base class, and then another class inherits from the derived class, forming a chain of inheritance. It represents a hierarchy of classes.

