Python OOP

Procedural Programming: This approach uses functions or procedures to manipulate data, focusing on sequential execution with loops and conditionals.
- Data separated from functions
- Lack of modularity
- High coupling
- Re-use difficult
- Does not model the real world accurately

Object Oriented Programming: OOP centres around objects that combine data and functions (methods), encouraging reusability and modelling real-world entities using classes, inheritance, and polymorphism.
- Data and functions combined
- Better modularity
- Low coupling
- Re-use easier
- Closely models the real world

Abstraction: Abstraction in Object-Oriented Programming (OOP) refers to the process of simplifying complex reality by modelling classes based on their essential characteristics and ignoring unnecessary details. It involves representing the relevant features of an object in a way that hides the underlying complexity while providing a clear and manageable interface for interacting with that object.

Class:
- A class is a user-defined data type. It consists of data members and member functions, which can be accessed and used by creating an instance of that class
- Hide data and implementation details
- Each class has a well-defined responsibility

Object:
- An Object is an instance of a Class

Encapsulation:
- Provide a simple, consistent interface to use the object
- Each class has a well defined responsibility
- All that is provided outside is a set of functions that are called and gives users access to capabilities of the class
- Encapsulation in OOP refers to the practice of enclosing data (attributes) and the methods (functions) that operate on that data within a single unit (class).
- It restricts direct access to data from outside the class, promoting controlled interactions and abstraction of implementation details.
- Use of getter and setter

Inheritance:
- Inheritance in OOP is the mechanism by which a new class (subclass or derived class) can inherit attributes and methods from an existing class (superclass or base class).
- It allows for the creation of a hierarchy of classes, where the subclass inherits the behaviour of the superclass and can also extend or override it as needed
- Refactor pull members up, can pull members into superclass automatically

Polymorphism:
- Polymorphism in OOP allows objects of different classes to be treated as instances of a common superclass
- It enables the use of a single interface to represent various types of objects, promoting flexibility and code reusability.
- By combining data and functionality into a single unit, we can tell an object to do a task
- Call add order function on customer object, and pass data in; and the customer object knows what to do to perform the task
- For instance, tell car engine or appliance to start, and each do something different
- Different number of classes, all capable of dealing with the same operation being called, but classes know how to perform the action for their particular case

Summary

Abstraction
An object is an instance of a class
A class is a type of object
- Involves hiding complex implementation details of how an object works exposing only the relevant and essential features
- Focus on what an object does rather than how it does it
- This reduces complexity and enhances reusability.

Encapsulation
- Hides data and implementation details
- Provides a simple, consistent interface to use the object
- Encapsulation is about controlling access to the internal components of a class
- Each class has a well-defined responsibility
- _ (underscore) convention is that if a feature of a class has underscore at start, it's a private feature aimed at restricting direct access to certain attributes or methods of a class
- It helps in controlling the visibility and accessibility of the internal workings of an object

Inheritance
-   A fundamental concept in OOP that allows a new class (subclass or derived class) to inherit properties and behaviours (attributes and methods) from an existing class (superclass or base class).
-   This promotes code reuse and hierarchical organisation of classes.
-   The subclass can extend, modify, or override the attributes and methods inherited from the superclass.

Polymorphism
-   Refers to the ability of different classes to be treated as instances of a common superclass
-   A number of different classes all capable of dealing with the same message or operation being called on them, but each class know how to perform actions for their own particular case where the underlying implementation is unknown due to encapsulation
-   Different classes that can all respond to the same message or operation
-   Even though each class knows how to perform the action specific to its purpose (due to encapsulation), you can simply tell a class to do a job, and it will know how to execute it
-   This concept enhances code reusability and adaptability