Object Oriented Programming

In Python

OOP vs Procedural Programming

By Definition

- OOP paradigm uses objects and classes for creating models based on the real-world environment.
- Procedural paradigm makes use of a step by step approach for breaking down a task into a collection of routines (or subroutines) and variables by following a sequence of instructions. It carries out each step systematically in order so that a computer easily gets to understand what to do.

In terms of Security

- Hiding data is possible with Object Oriented Programming due to the abstraction. Thus, it is more secure than the Procedural Programming.

- Procedural Programming does not offer any method of hiding data. Thus, it is less secure when compared to Object Oriented Programming.

In terms of Division of Program

- Object Oriented Programming divides the program into small parts and refers to them as objects.

- Procedural Programming divides the program into small parts and refers to them as functions.

In terms of Reusability

- Object Oriented Programming offers the feature to reuse any existing codes in it by utilizing a feature known as inheritance.

- No feature of reusing codes is present in Procedural Programming.

Relationship b/w Class & its object

What is a Class?

- A class is a blueprint of how our real world objects will be like and how will they behave.
- A class consist of two things:
  - Attributes
  - Methods

Let’s look at an example!

Types of Attributes

- Instance Variables
- Class Variables

Constructors

Definition

- is a member function that constructs an instantiation (object) of the class
- Is defined with ‘__init__’
- Can be parameterized or not

Behavior of Constructor

- A constructor is called automatically whenever a new instance of a class is created.
- You can supply the arguments to the constructor when a new instance is created.
- If you do not specify a constructor, the interpreter generates a default constructor for you

The 4 Principles of OOP

Encapsulation

- Encapsulation can be defined as the binding of data and attributes or methods and data members in a single unit.
- This is to prevent the access to the data directly, the access to them is provided through the functions of the class.
- It is one of the popular feature of Object Oriented Programming(OOPs) that helps in data hiding.

Real Life Example of Encapsulation

- In a company there are different sections like the accounts section, finance section, sales section etc. The finance section handles all the financial transactions and keep records of all the data related to finance. Similarly the sales section handles all the sales related activities and keep records of all the sales. Now there may arise a situation when for some reason an official from finance section needs all the data about sales in a particular month. In this case, he is not allowed to directly access the data of sales section. He will first have to contact some other officer in the sales section and then request him to give the particular data.

Abstraction

- Data abstraction is one of the most essential and important feature of object oriented programming.
- Data abstraction refers to providing only essential information about the data to the outside world, hiding the background details or implementation.
- Abstraction is selecting data from a larger pool to show only the relevant details to the object.
- Abstraction is a way to cope with complexity.
- Principle of abstraction:
- “Capture only those details about an object that are relevant to current perspective”

Features of Abstraction

- **Security:** With Abstraction, any outsider doesn't get to know the internal implementation so it makes the classes more secure, and unauthorized user will not have access to the data.
- **Easy Enhancement:** If we need to change the implementation in a class, we don't have to change the entire external logic which is there for the user. We only need to make changes for the methods and it will not affect the functionality.
- **Maintainability will improve:** Without affecting the user, it can able to perform any types of changes internally, so maintainability will improve.

Inheritance

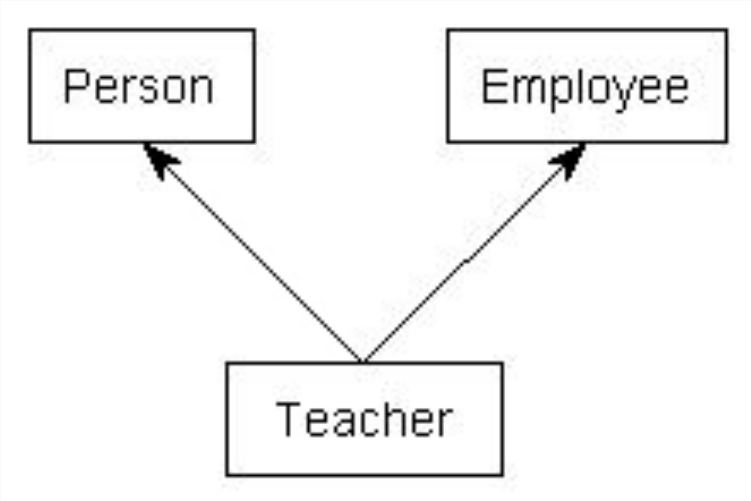
- Inheritance is the concept of acquiring features of the existing class into the new class.
- Suppose there is a class, considered as the parent class, that has some methods associated with it. And we declare a new class, considered as the child class, that has its own methods as well as the one that it inherits from the parent class.

Types of Inheritance: Simple Inheritance

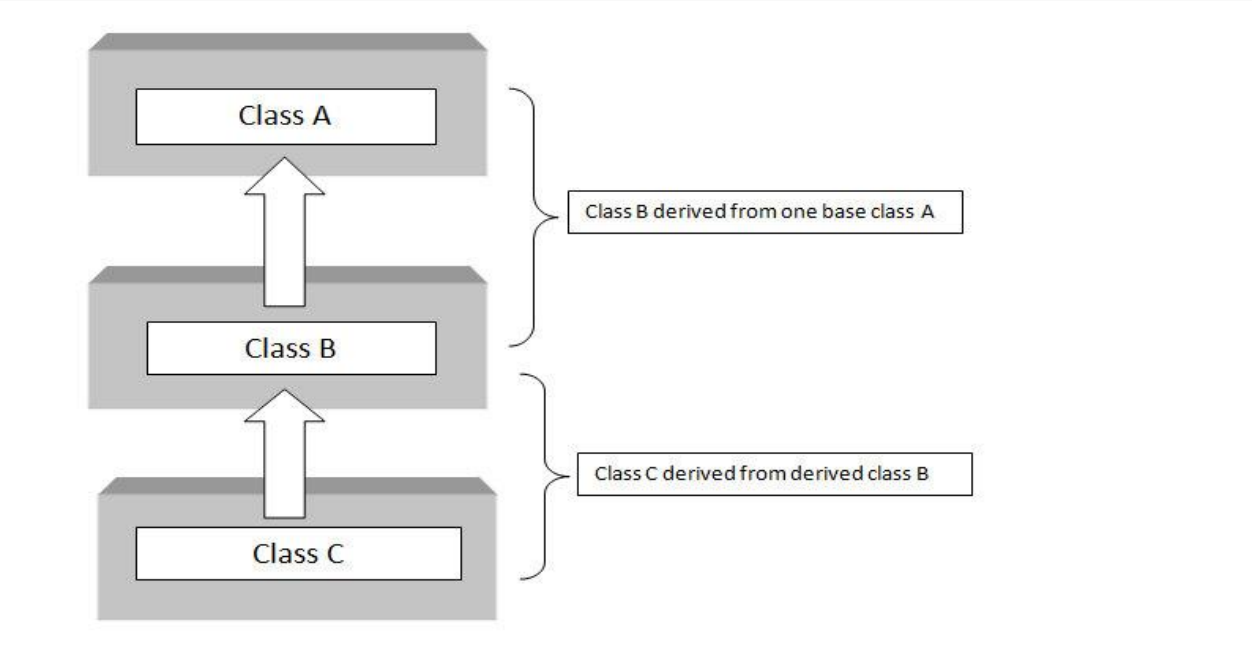
- When one class inherits from another class

Types of Inheritance: Multiple Inheritance

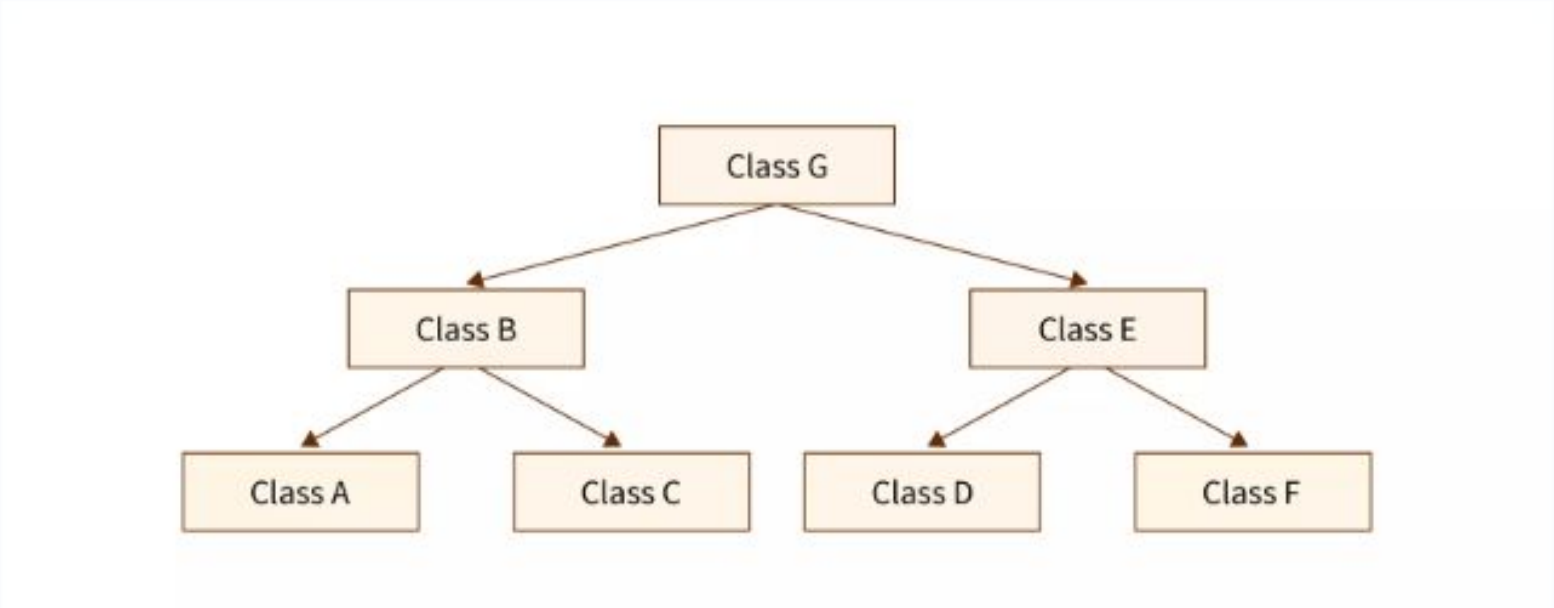
- When one class inherits from multiple classes



Types of Inheritance: Multi-level Inheritance



Types of Inheritance: Hierarchical Inheritance



Advantage of Inheritance

- The biggest advantage of inheritance is code reusability. The methods which are already declared in the base class need not be rewritten in the derived class, They can be directly used. It is automatically available for use on the derived class.

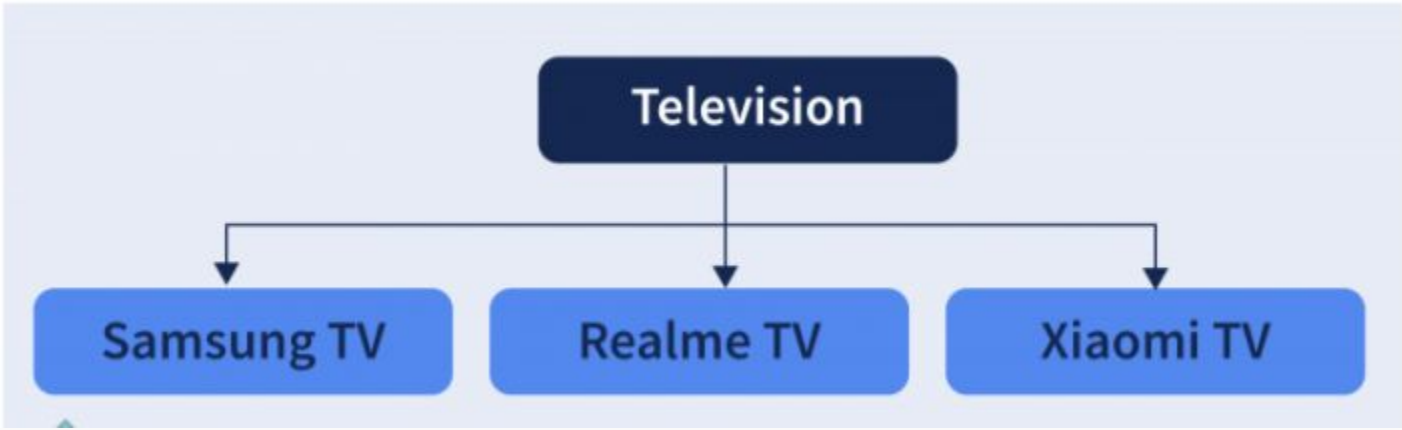
Polymorphism

- Polymorphism is the most essential concept of the Object-Oriented Programming principle. It has meaning **‘poly’** – many, **‘morph’** – forms. So polymorphism means many forms.

- In Object-Oriented Programming, any object or method has more than one name associated with it. That is nothing but polymorphism.

Polymorphism for Objects

- When any object can hold the reference of its parent object then it is a polymorphism.



Polymorphism for Methods

- Methods can also have multiple features associated with them with the same name. That we also consider as a type of polymorphism.
- There is two type of polymorphism for methods:
  1. Compile Time Polymorphism. Also called Static binding.
  2. Runtime Polymorphism. Also called Dynamic binding.

Compile Time Polymorphism

- Compile Time Polymorphism is achieved using Method Overloading.

RunTime Polymorphism

- Runtime polymorphism is achieved using Method Overriding.

- **Method Overriding** – When more than one method is declared as the same name and same signature but in a different class. Then the derived class method will override the base class method. This means the Base class method will be shadowed by the derived class method. And when the method is called then it can be called based on the overridden method.

Methods

Constructors & Destructors

- **Constructors-** are the methods that are used to construct an instantiation/object of a class.

- **Destructors -** are the methods that are used to destroy the created object of a class.

Accessors and Mutators

- **Accessors -** are the methods that are used to **retrive** the current value of a specific attribute.

- **Mutators -** are the methods that are used to **change** the value of a specific attribute of the object.

Access Modifiers

“No programming paradigm is necessarily bad, it only depends on what you want to achieve.”