Python Module 2 : Introduction

Purpose :

- Model Real World Entities
- Write Modular Code
- Working with Errors/Exception
- Handling Data from Websites
- Maintaining Code Repositories

Expectation from Learners :

- 75% + Attendance
- 85% PSP
  - → Assignments (mandatory)
  - → Additional/HW (optional)

At the end of Module :

- Module Test (90 mins)
  - → MCQ
  - +
  - → Coding
  - → cutoff to pass (70+/100)
- Mock Interview

Important links

- Git : https://github.com/SachinScaler/July24Python2/
- Whatsapp Group

Instructor Details

1) Whatsapp Number: 8586836627

2) LinkedIn: https://in.linkedin.com/in/sachin-kaushik

Module Overview

1 **Object Oriented Programming**
Classes & Objects - Defining a class, Instantiating an object
4 Pillars of OOP - Abstraction, Encapsulation, Inheritance, Polymorphism
Constructors, Class & Instance Variables

2 **Dunders and Inheritance in OOPs**
Dunder / Magic methods, Private Properties, Inheritance, Multiple Inheritance, Method Resolution Order

3 **Basics of Functional Programming**
Intro to functional programming, Lambda Functions, Higher Order Functions, and Decorators.

4 **Advanced Functional Programming**
Map, Filter, Reduce, Zip functions and their use cases, Args and Kwargs.

5 **Modules and Exception Handling**
Modules in Python: Math and Random modules
Exception Handling: try & except, Raising custom exceptions

6 **Basics of Time & Space Complexity**
Arithmetic and Geometric Progressions
Big-O Notation, Comparison of Order of Time Complexities
Space Complexity

7 **Regular Expressions (Regex)**
Metacharacters, Anchors, Character Sets, Quantifiers and Groups in Regex for Pattern Matching.

8 **Web Scraping**
Understanding a website's structure (HTML/CSS), using **request** module and **beautiful soup** library for scraping data.

9 **Git and GitHub**
Version Control System, Git vs. GitHub
Git with GitHub Desktop & CLI, Creating a repository,
Push & Pull, Branching & Merging, Forking & Cloning

Agenda

- OOP
- Classes and Objects
- 4 pillars of OOP
- Methods
- Constructor
- Class and Instance Variables

Class

OOP

Object Oriented programming

↳ Programming Paradigm

- → Procedural ✓
- → OOP ✓
- → Functional ✓
- → Declarative ✗

Sequential ✗

def func

Classes

↓

Serve as blueprint for data as well a functions

Objects (instances)

OOP

→ Classes : Blueprints

↓

Objects : Instance of Class

* Purpose : Represent Real world Entities or Scenario using Code

⊙ It is used for creating custom Data-Structures with their Own functionalities to represent Entities

Bank Account :

Properties
- → AccId
- → Custname
- → BankBalance
- → * Bank Name

Behaviour
- → ViewBal()
- → Deposit()
- → Withdraw()

4 pillars of OOP

① Encapsulation ✓

② Abstraction

③ Inheritance ✓

④ Polymorphism ✓

① Encapsulation

⇒ Putting things together

⇒ Hiding data and behaviour

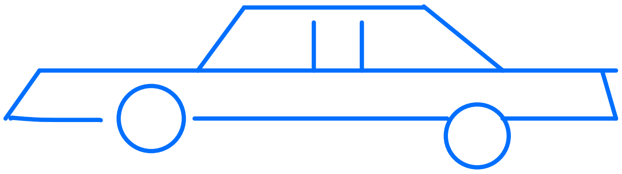
⇒ Class ⟶ instance → Data

instance → Methods

⇒ access Specifiers : Next Session

② Abstraction :

Hiding Details



Behaviour

Properties

- Accelerate
- Break
- Combustion procedure

Colour
Brand
Make
Steering
Engine

* Abstract Classes ×

To drive a car, do I need to know all the inner details?

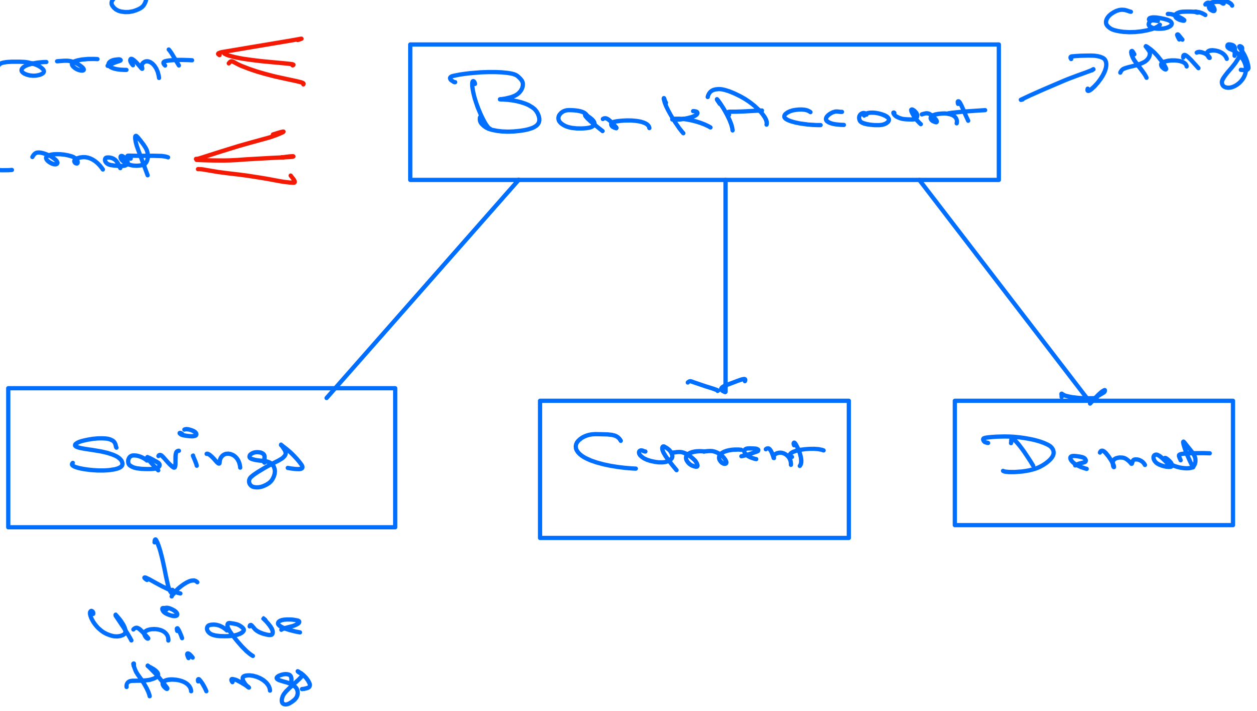
NO

③ Inheritance

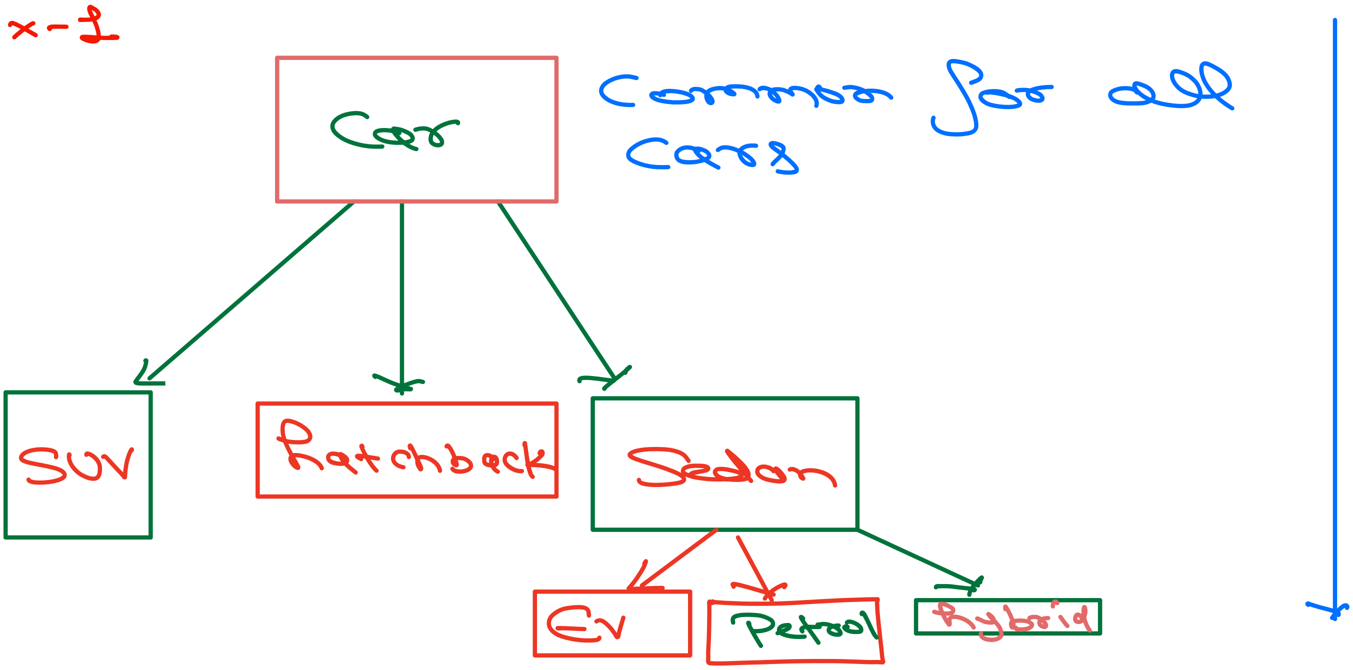
- We use this to write modular code

BankAccounts

- Savings ⇐
- Current ⇐
- Demat ⇐

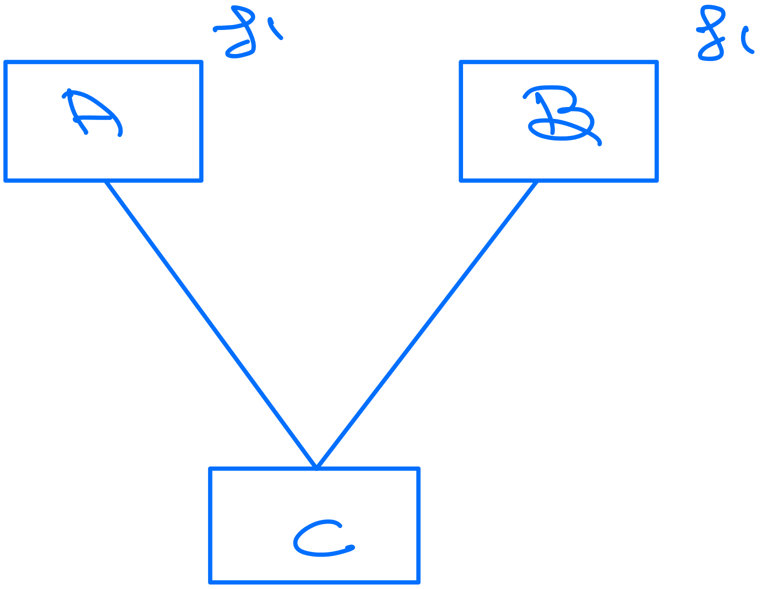


Ex-2



multi-level Inheritance

Ex- 3



Mutiple Inheritance

④ **Polymorphism :** One Entity many Behaviours

Ex:

5 + 6 → 11

Behave differently depending on Type of instance

'5' + '6' → 56

Defining Methods

⑤ Whenever we call a method with instance,

instance is passed as an argument automatically

Special Methods

↳ Dunder Methods

↳ __methodName__

Ex:

__init__

Constructor

① ⇒ A special method that allocates memory

⇒ initializes "instance Variables"

Python doesn't allow direct access to Constructor

⇒ __init__ : initializer

initializes "instance Variables"

⇒ It get called Automatically when an instance is created

Key Takeway

- Updating instance Variable of any instance Doesnot impact Others

Variable Types

* Instance Variable :

    * A variable/property that is unique to each instance

* Class Variable :

    * A shared Variable, Common to all the instances.