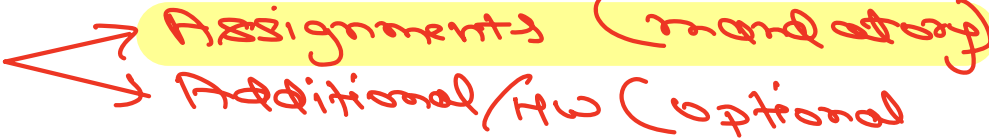


Python Module 2: Introduction

① Purpose :

- ① Model Real World Entities
- ① Write Modular Code
- ① Working With Errors/Exception
- ① Handling Data from Websites
- ① Maintaining Code Repositories

② Expectation from Learners :

- ① 75% + Attendance
- ① 85% PSP 

③ At the end of Module :

- ① Module Test (30 mins)
 - MCQ + Coding
 - cutoff to pass (70+/100)
- ① Mock Interview

④ Important Links :

- ① Git : <https://github.com/SachinScaler/May24Python2>
- ① Whatsapp Group :

③ Instructor Details:

③ WhatsApp Number: 8586836627

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

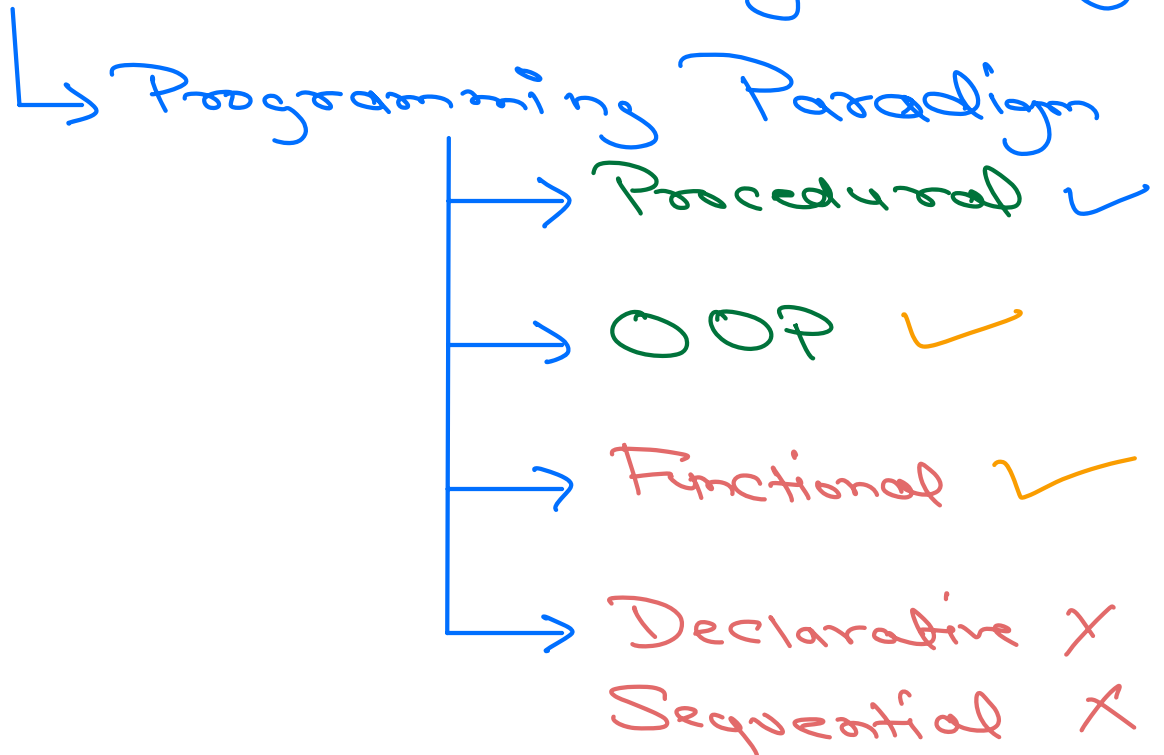
⇒ Methods

⇒ Constructor

⇒ Class and Instance Variable

OOP

Object Oriented programming



OOP

→ **Classes** : Blueprints

↓
Objects : Instance of Class

* Purpose : Represent Real world Entities or Scenario using Code

① It is used for creating custom Data-Structures with their own functionalities to represent Entities

Bank Account :

Properties

- AccId
- CusName
- Bank Balance

Behaviour

- ViewBal()
- Deposit()
- Withdraw()

create Diction

Properties

Functions

($k : v$)

1000 customers



Create a Class

① Act as a template

② It will contain

↳ Property

↳ Functionality

↓
Instance : Will contain data of
Customers

Object

↓
Everything in Python is
derived,

Everything in Python is an Object

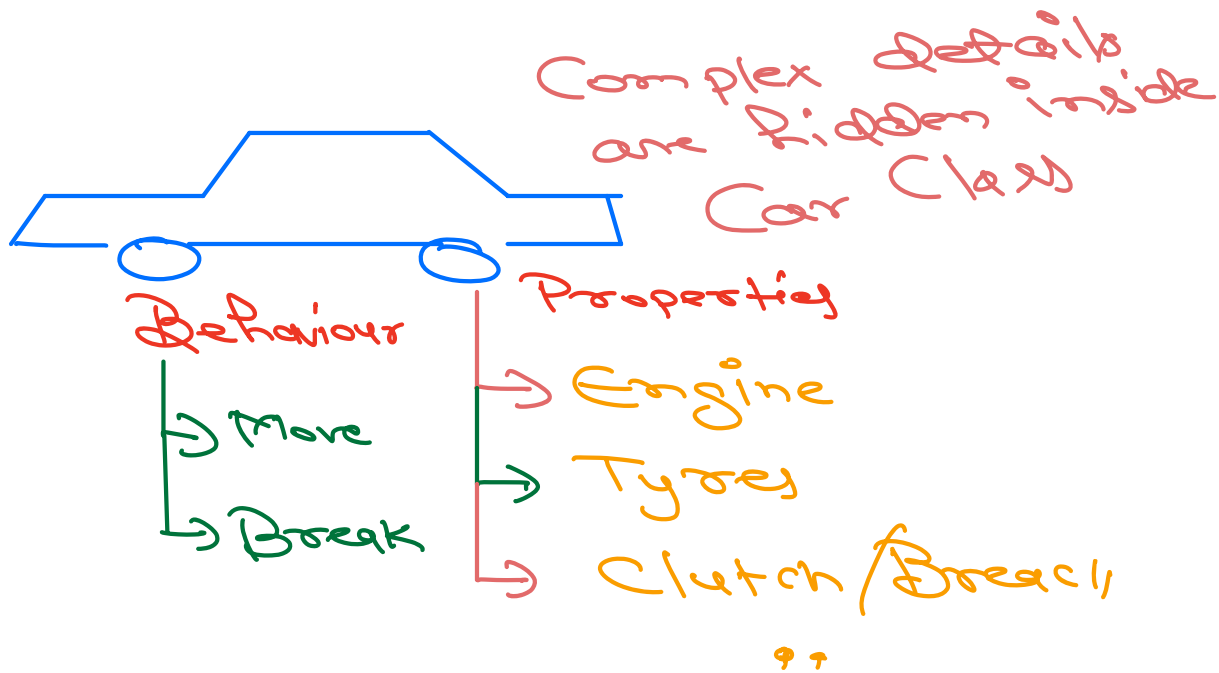
4 pillars of OOP

1 Encapsulation

Packing things together

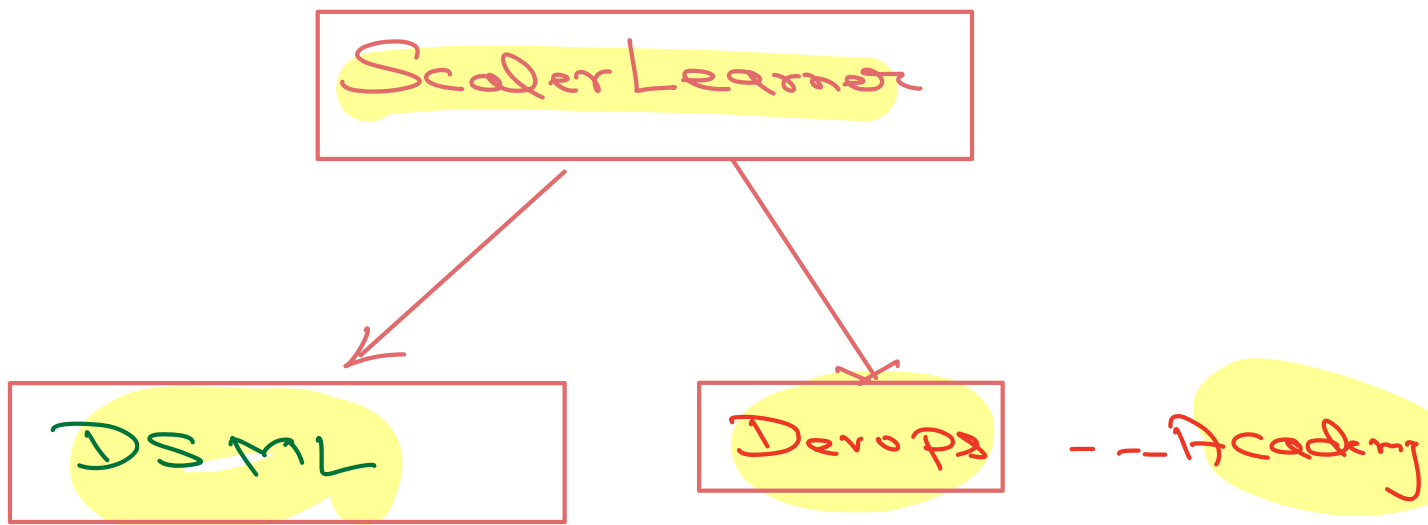
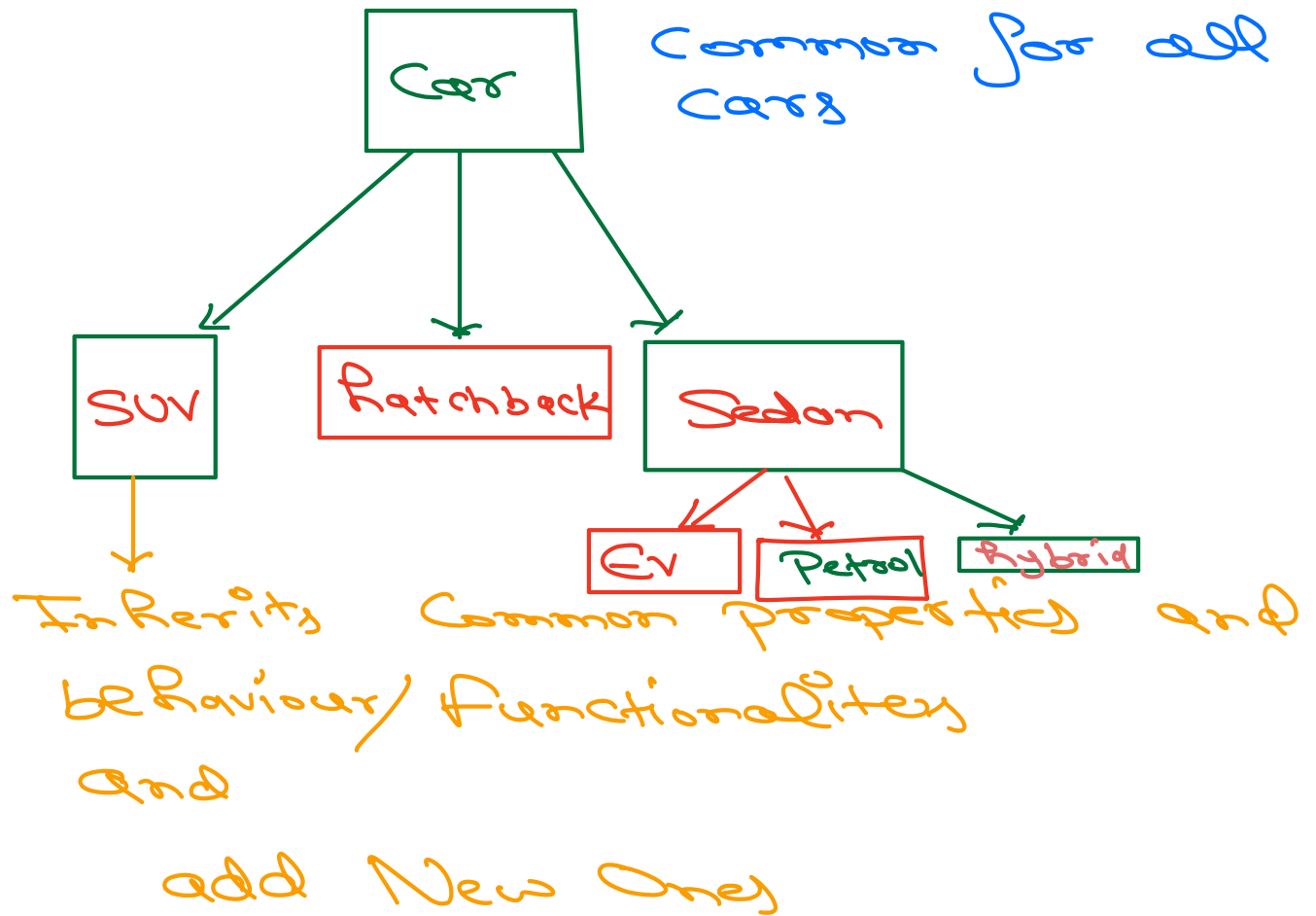


2 Abstraction :



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

⑨ Inheritance



③ Polymorphism: Same Entity
Multiple Behaviour

$$5 + 6 \Rightarrow 11$$

addition for int

$$'5' + '6' \Rightarrow '56'$$

concatenation for str

Defining Behaviour

(Functionalities)

class A:

def Func1(): → Method of Class A

|||

Q1 = A()

Q1.Func1() →

Func1(Q1)

Object is
automatically
Passed as an
argument

Special Methods

→ Under Methods
→ `--method Name--`

`--init--`

① Initializer NOT a Constructor

② Purpose:

Creating properties/attributes
that each instance must
have

↓
instance Variables

③ `--init--` gets automatically
executed when instance is
getting created

Key Takeaway

③ Updating instance Variable of any instance does not impact Others

* Instance Variable :

* A variable/property that is unique to each instance

* Class Variable :

* A shared Variable, Common to all the instances.

Empty - dict = {}

loop over string

↳ if char is not present in Empty-dict

empty-dict[char] = 1

else

increment old freq by 1