Object Oriented Programming - OOPs

Agenda
- → Basic of OOPs
- → Pillars of OOPs → 4 pillars
- → Constructor

⇒ DS → where is OOPs → Open Source Project
→ Day to Day → hot mock

⇒ Paradigm → • Pattern
• model
• Style

Programming Paradigm

→ . If a team / Company follows specific set of rules while writing their program.

Paradigm → OOPs
→ Functional programming

→ Imperative, declarative, sequential, ...

⇒ OOPs → Object Oriented

- Class → blueprint
- Object → instance of class

⇒ Student

Properties
- Name
- phone
- email
- Batch
- Marks

Methods/action
- → attend class
- → take a test
- → apply for interviews

⇒ Class → blueprint of thing → propert, → action → that can be repeated later.

⇒ Class → [name :
class :
email :]

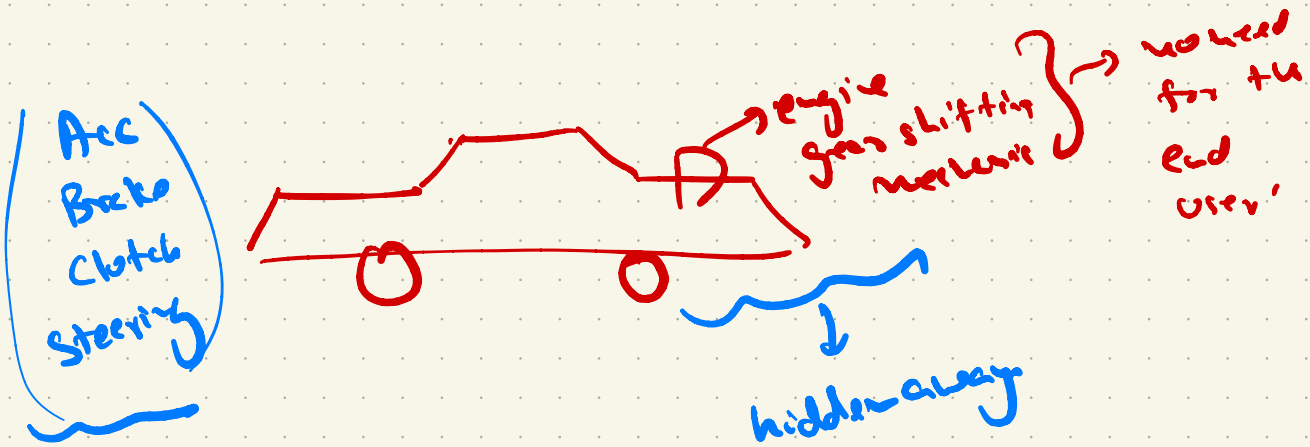
every instance of the class gets the same blueprint.

⇒ 4 Pillars of OOP → [→ → → → → put things together

→ Hide away unnecessary thing → inside class

① Encapsulation → put all related things together.

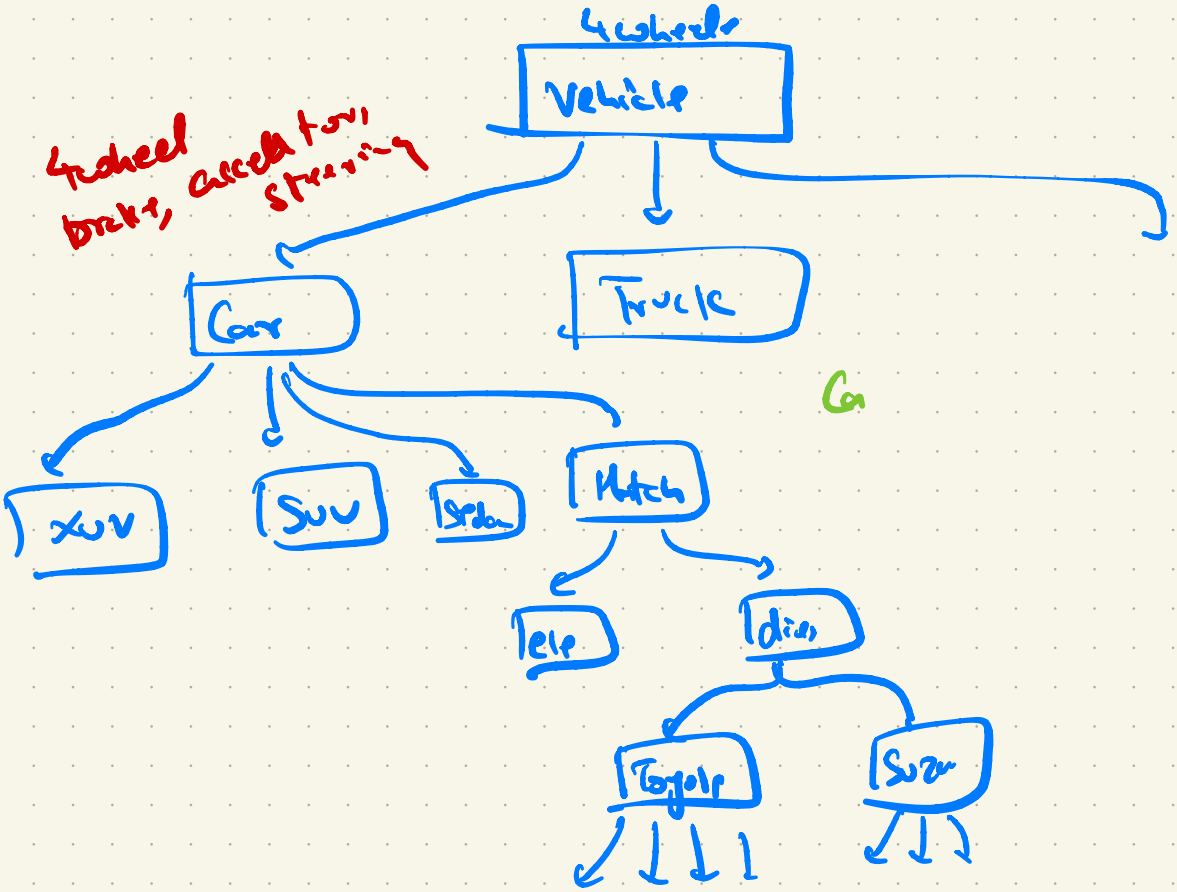
② Abstraction → Hiding the irrelevant features



→ properties

→ functionalities → !!

③ Inheritance : Parent Children.



⇒ 4 wheeler → 4 wheel, brake, tyres, body, ...

Car → everything in 4 wheeler

{

}

Sedan → everything in Car

{

}

Class → Storing all common attributes

*① Class Vehicle →

② Class Car → • Copy Class Vehicle
- [

③ Class Sedan → • Copy Class Car
[

Func → Copy full [ ]

Inheritance → Children inherit properties from parent class.

Polymorphism → Different behaviours for same entity.

Same Symbol / Object / function / -- which behave in different manners for different input.

[We are only talking about datatypes]

⇒ Constructor → first function that will be called whenever an object is created to allocate memory to that object

⇒ Should user be given access to constructor func?

Python does not give access of Constructor to user

↓

Instead

Python gives access to initialising function to user → . (A function that is called whenever a object is created in memory)

a=5 → Constructor → Store variable a as 5.

Init → (Coder tells what to do after a is created)

↙

Steps to do once a variable is created

a = Random() | 5 | "Ankit"

→ Stored in memory

→ __init__ (if defined (Used) is not a Constructor

{ . } lifetime of the object

→ __del__ → destructor() → removed from memory

init: Any repeated task which needs to be done as soon as any object created → Can be put into init.

⇒ Class variable → Shared across instances

① Immutable → overwrite in instance A → Only A is impacted

② Mutable (List, dict, Set) → overwrite in A → All other instances also get impacted

⇒ Instance Variable

→ Immutable → only impact A
→ Mutable

⇒ OOPs → 4pillars

↳ • Create a Class
• Store class variable
instance variable
• Store / Call functions
↳ Special function → init

⇒ : →] • Can you crack an interview
. →

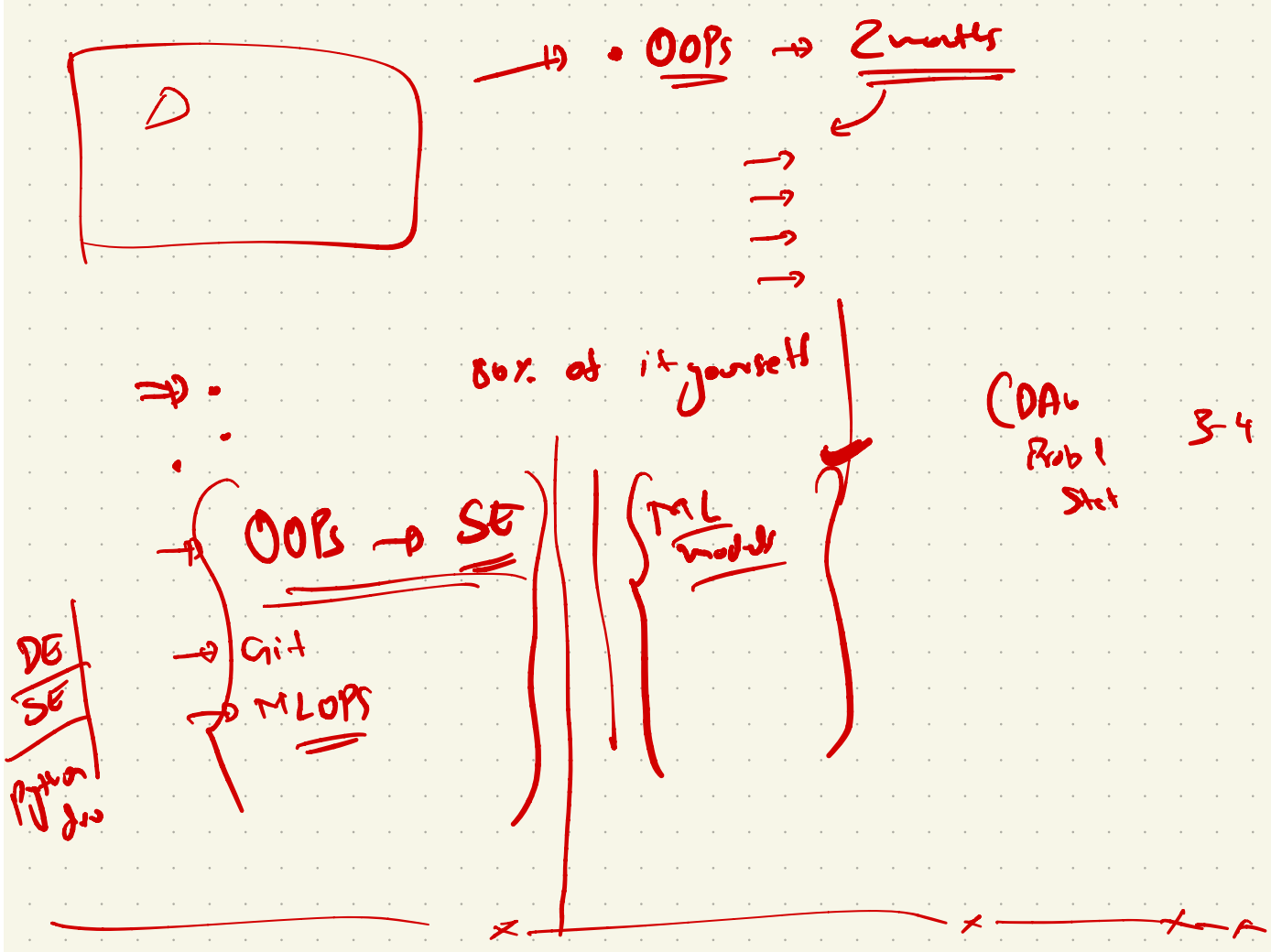
Theoretical perspective →

→ (Basic Concepts)
→
→

① 4pillars or not

→ very basic of
② (→ Implemented)

[ ]



① Solve assignment

② Post Read + Lecture notes
+ Python notebooks

③ Reattempt assignment

④ Visiting Lectures again

⇒ Object

Machine store data → RAM Harddisk

a

a=5

print(a)

a= 5*5

print(a)

First time

a=5 → Python → Create a place for a in memory and store value 5 there

(piece of code) done by a Constructor

→ C → gives access of Construction to programmer

→ Python → you don't have access to Constructor

C → • Combine routine initial steps with Constr.

Store something in memory

⇒ logging → Send email

Python → Python Constructor → create object in memory → (__init__)

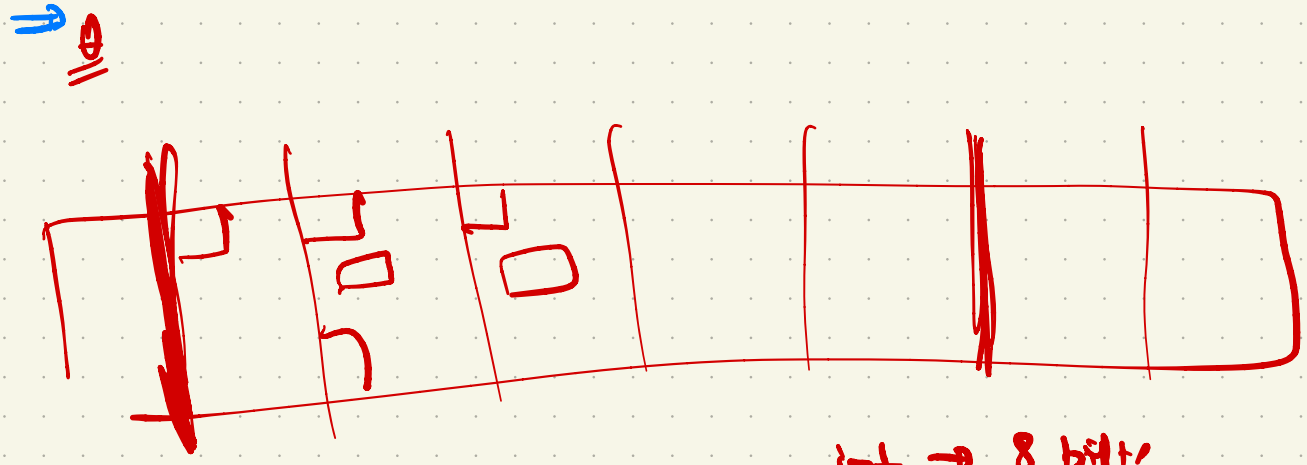
anything inside

will be called as soon as
object is created

a=5 → object get created [Constructor]

↳ Python Internal

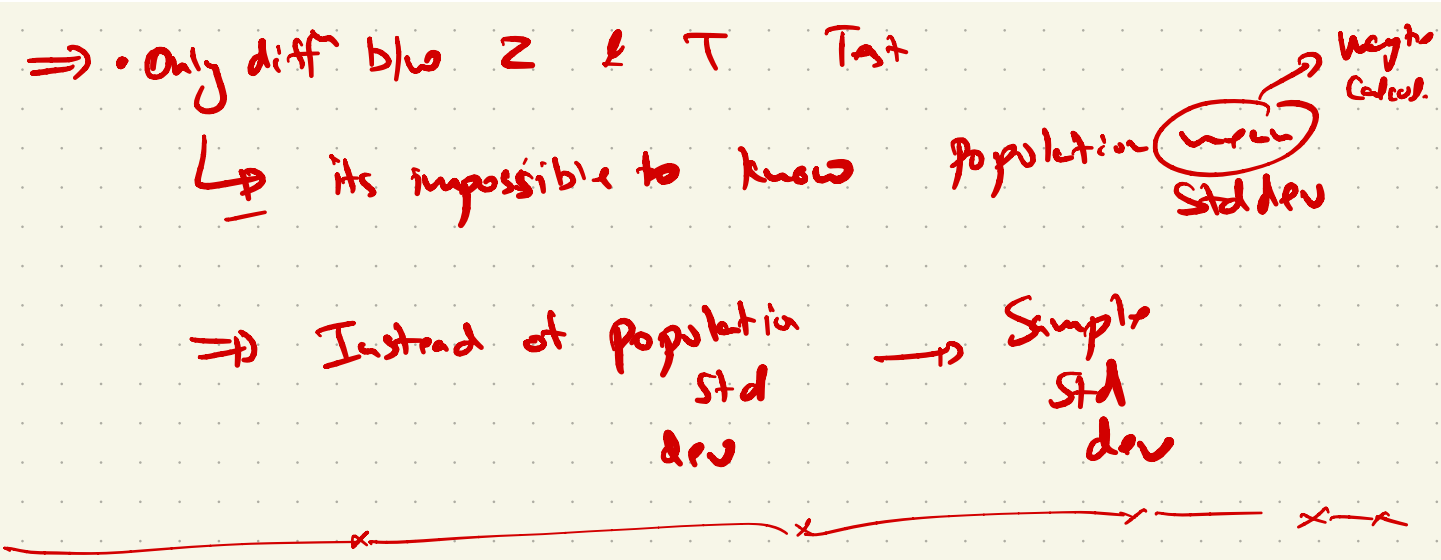
[init function]



Stored Objects

int → 8 bytes
32 bits

Others → 256



Collab : https://colab.research.google.com/drive/1QcXQGePSTKQ_veWg01Dbssr3SYG2PKHH?usp=sharing