OOP – 101

Programming Paradigm ?

$$\begin{tikzcd} & \text{microwave} & \\ \text{Make maggy.} \arrow[ur] \arrow[dr] & & \Rightarrow \text{same end result.} \\ & \text{kadhai.} & \end{tikzcd}$$

✓ 1) OOP ⇒ Classes and Objects

✓ 2) Functional.

$$\begin{tikzcd} \text{Car} \arrow[d] & \text{Mercedes.} \arrow[d] \\ \text{Class.} & \text{Object} \end{tikzcd}$$

(Blueprint.) (Instance)

Human ⇒ diff attributes

age, name,
gender,
height

Common

→ walk

→ eat

→ breathe.

→ sleep

---

class Student → properties / attribute.
name, age, roll-num

↳ methods / actions.
study, class, solve

4 pillars of OOP

$$\begin{tikzcd}[column sep=small] & & A\,P\,I\,E \arrow[dll] \arrow[dl] \arrow[d] \arrow[dr] & \\ \text{Abstraction} & \text{Polymorphism} & \text{Inheritance} & \text{Encapsulation} \end{tikzcd}$$

Abstraction ✓ Polymorphism $\underbrace{\quad}$ Inheritance Encapsulation ✓

Encapsulation

bringing together all common items at one place.

data + functions

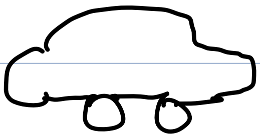
a fn inside class

=> called as method.

Abstraction

↓

Only the relevant info is known.



Car

relevant => how to drive

clutch position, gear,

how Engines work etc => abstracted.

Construction and Initialisation

→ purchased a plot of land.

Build a house

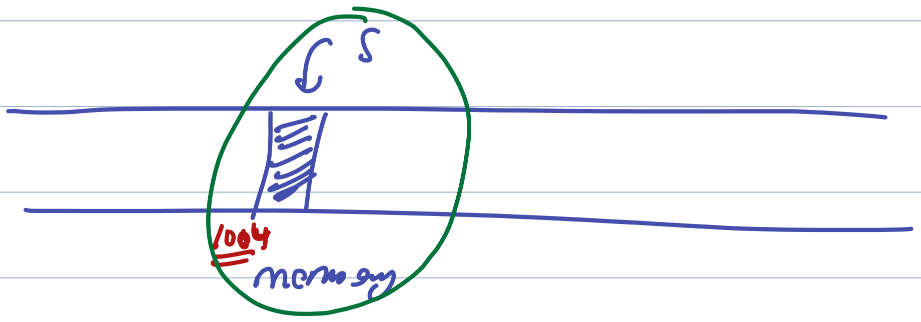
Bought & cleared.

Constructor ① Free up the space → clean up plot of land

Initialiser ② Allocating the values → creating the house

```
class Student:
    pass
```

s = Student() # ① Object is Constructed/created



s.name = '___'

s.age = __

---

not the constructor

this only assigns values

```
class Student:
    def __init__(self, n, a):
        self.name = n
        self.age = a
```

```
s = Student('Ram ai', 25)
```

```
class FamilyPerson:

        def __init__(self, name, sur_name) :
instance  { -> self.name = name
variables.{ -> self.sur_name = sur_name
```

```
p1 = FamilyPerson('Rahul', 'Jangha')
p2 = FamilyPerson('Parveen', 'Jangha')
```

```
class FamilyPerson:
        sur_name= 'Janghu'   → class variable.
        def __init__(self, name):
instance → {   self.name= name
variables.
```

```
p1 = FamilyPerson('Rahul')
p2 = FamilyPerson('Parveen)

 print(p1.sur_name)
 print (p2.surname)
```