DAY-1 OOP In Python

- OOP concept
  - ↳ A different way to write codes

- OOPs concept helps us in two things
  1) Define our own datatype
  2) Generality to specifity

★ • In python everything is an object

→ Reason :

for eg : (1) L = [1, 2, 3, 4]

L.upper()

error :
List object has no attribute upper()

(2) Str = "Zain"

Str.append('a')

error :
Str object has no attribute append()

- **OOPs :** 1) Object.
  2) Class.
  3) Polymorphism.
  4) Encapsulation.
  5) Inheritance.
  6) Abstraction.

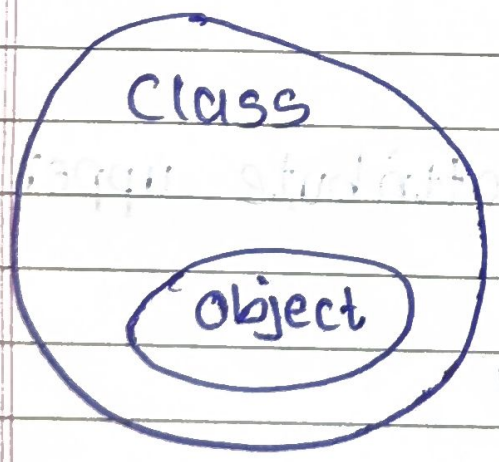
Class

→ Blueprint.

→ Self defined datatype.

- We know that in python everything is an object. So whatever object it is must be in a class.

- Class is a Blueprint



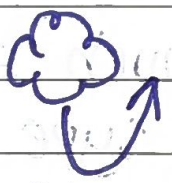
- All datatypes are built in classes in python.

For eg : a = 2 (int object)
a → variable
a → object

- From now onwards variables = objects

"So, List, tuples... are datatypes and while making these datatypes classes are used"

↳ For eg: L = [1, 2]

L.  If we hit tab here we can see many functions, and these functions can only be accessible by list datatypes.

↳ For eg:

```
Str = "Zain"
Str.upper()
str.lower()
str.isalpha()
str.isnum()
```

These are functions which can only be accessible by string datatype.

And these functions are made and stored in a class.

Class

Data or Property

Functions or Behaviour

Page No.
Date

↳ For eg:- If we are making class named "Car",

So, what are the data related to Cars?

↳ Answer: 1) Number of wheels
2) Colour
3) Brand
4) How much seater
5) Engine type
6) Auto or Manual

And what's there behaviour?

↳ Answer: 1) Calculate Mileage
2) Calculate average speed

Class basic structure:

Class Car: [Class's name should be pascal case]

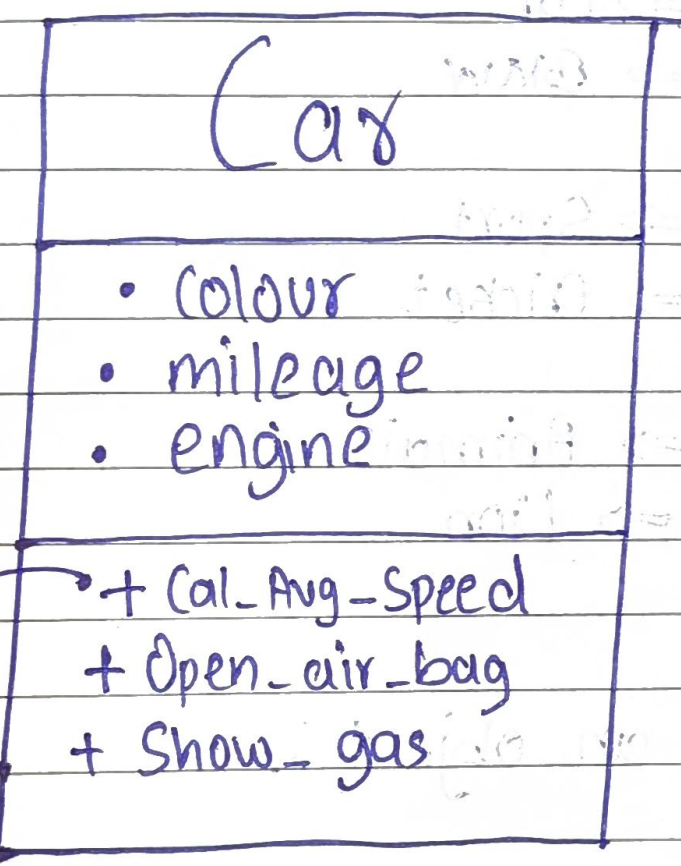
colour = "Blue"
Model = "Sport"
} Data

def CalculateAvgSpeed(km, time)
Your logic here
} Behaviour

→ Pascal Case :

→ eg: 1) ThisIsPascalCase.
2) IAmZain.

Diagrammaticaly Representation of a class :



Public : It can be used outside the class.

Object

↳ An instance of a class

- Datatype is class.

↳ Variable in it is an object.

e.g.: 1) class => Car
object => BMW

2) class => Sport
object => Cricket

3) class => Animals
object => Lion

- How to create an object ?

↳ BMW = Car ()
Cricket = Sport ()
Lion = Animals ()

* for built in classes:

↳ We use "object literal"

↳ L = [1,2,3]
L => List

* for not built in classes:

L = List ()
Str = Str ()
BMW = Car ()

Let's make a class :

Atm class

↳ Function vs Methods

Method : A special function written inside a class.

function : Normal function we create or use outside a class.

↳ For eg : L = [1, 2]

(1) print(len(L)) → Function "len"

↳ 2

(2) L.append(3) : Method "append"

↳ [1, 2, 3]

Constructors

- Whenever we create/construct an object of a class, there is an inbuilt method (__init__) which is called constructor.

- It is a special method or a function within a class that gets automatically call, when an object is created.

- It allocated memory to the object.

- It is used to inilize the attributes of the object.

- It is written as __init__

=> Class Student :

    def __init__() → here we have not given any arguments.
        print ('Hello')

s1 = Student() → here also 0 arguments

→ This will throw an error.

Let's discuss ...

- We will get an error: Student.init__() takes 0 arguments but 1 was given

But we have not given any arguments, ...

So, what happen here is, the id(s1) is the argument given to the constructor every time we try to create an object.

So, we will take 1 argument in __init__(self)

we can name it anything but because it is self taking argument so we name it 'self'.

```
class student:

    def __init__(self):
        print(id(s1))

s1 = student()
```

Now the code will work!