[Day-11]

Encapsulation — wrapping data (variables) & methods (functions) together as a single unit. It restricts direct access to some of the objects components, which is a means of preventing accidental interference & misuse of the data.

→ Getter & setter methods
→ Public, Private, Protected variables

```
def __init__(self, name, age, gender):
    self.__name = name      # private var    # Protected: self._name
    self.__age = age        # private var
    self.gender = gender    # Public var
```

```
class Employee(Person):
    def __init__(self, name, age, gender):
        super().__init__(self, name, age, gender)

def get_name(Person):
    return person.__name

person = Person("Shreethe", 21, 'Female')
set_name(Person)
```

Encapsulation with getter & setter

```
def get_name(self):
    return self.__name

def set_name(self, name):
    self.__name = name
```

Abstraction — hiding complex implementation details & showing only the necessary features of an object.

↳ This helps in reducing programming complexity & effort.

```
from abc import ABC, abstractmethod
#Abstract base class
class Vehicle(ABC):
    def fuel_type(self):
        print("Vehicle used for driving")

    @abstractmethod
    def start_engine(self):
        pass

class Car(Vehicle):
    def start_engine(self):
        print("Car engine started")

def operate_vehicle(vehicle):
    vehicle.start_engine()

car = Car()
operate_vehicle(Car())
```

Magic Methods — also known as dunder methods (double underscore methods), are special methods that start & end with double underscores.

↳ These methods enables to define the behaviour of objects for built-in operations, such as arithmetic operations, comparisons and more.

↳ predefined methods that can override to change the behaviour of objects.

↳ Examples:

__init__	: initializes new instance of a class
__str__	: returns String repr. of an object
__repr__	: returns an official string representation of an object
__len__	: returns the length of an object
__getitem__	: gets an item from a container
__setitem__	: sets an item in a container

Operator overloading — Common overloading magic methods are:

__add__, __sub__(self, other), __mul__(self, other), __gt__(self, other)
__truediv__(self, other), __eq__(self, other), __lt__(self, other)

Custom Exceptions (Raise and throw an exception)

```
class Error(Exception):
    pass

class dobException(Error):
    pass

year = int(input("Enter the dob"))
age = 2024 - year
try:
    if age <= 30 and age >= 20:
        print("valid")
    else:
        raise dobException
except dobException:
    print("Not Eligible")
```