[Day 9]

OOP — programming paradigm that uses 'objects & classes' to design applications & computer programs.

↳ OOP allows for modeling real-world scenarios using classes and objects.

↳ Class is a blueprint for creating objects — Attributes methods

instance variable and methods (Here variables)

```
class Dog:
    ## constructor
    def __init__(self, name, age):        # parameters
        self.name = name
        self.age = age

    ##Create objects
    dog1 = Dog("Coco", 3)
    print(dog1)          # prints obj location
    print(dog1.name)     # prints Coco
    print(dog1.age)      # prints 3
```

Define a class with instance methods

```
class Dog:
    def __init__(self, name, age):
        self.name = name
        self.age = age

    def bark(self):          # instance method
        print(f"{self.name} says woof")
dog1 = Dog("Coco", 3)
dog1.bark()                  # prints Coco says woof
```

Inheritance — concept in OOP that allows a class to inherit attributes & methods from another class.

↳ super() method is used to inherit the properties of base class.

↳ Single inheritance & multiple inheritance
↳ when class inherits from more than one base class

Example: Multiple inheritance

```
class Animal:
    def __init__(self, name):
        self.name = name

    def speak(self):
        print("Subclass must implement this method")

#Base class 2
class Pet:
    def __init__(self, owner):
        self.owner = owner

#Derived class
class Dog(Animal, Pet):
    def __init__(self, name, owner):
        Animal.__init__(self, name)
        Pet.__init__(self, owner)

    def speak(self):
        return f"{self.name} say woof"

dog = Dog("Coco", "Kristy")
print(dog.speak())   ——— # prints Coco says woof.
print(dog.owner())   ——— # prints Kristy
```