**1.Procedural programming is a programming style where code is written as a set of step-by-step instructions**

a = 10

b = 10

add = a + b

print(add)

**2. Functional programming is a style of programming where you write code using pure functions**

it is used to avoid **redundancy** and make the code **cleaner** by using pure functions

def add(num1 , num2):

  print(num1 + num2)

add(10,20)

add(30,40)

add(10,40)

add(40,40)

add(60,40)

add(70,40)

**Object-Oriented Programming is a way of writing programs by creating objects — real-world-like things that have data (variables) and behavior (functions). It focuses on organizing code using classes and objects**.

class student:

    name = "Atharva"

s1 = student()

print(s1.name)

s2 = student()

print(s2.name)

s3 = student()

print(s3.name)

s4 = student()

print(s4.name)

class car:

    color = "blue"

    brand = "mercendes"

    tyre = "4"

car1 = car()

print(car1.color)

print(car1.brand)

print(car1.tyre)

**init function**

**When you create something using a class, Python needs a way to set it up. That’s what \_\_init\_\_() does — it initializes (starts) the object with some values.**

self refers to the **current object** (also called **instance**) of a class.

Because when many objects are created from a class, Python needs a way to **know which object you're referring to** when calling a method.

class car:

  def \_\_init\_\_(self, color, brand, tyres):

    self.car\_color = color

    self.car\_brand = brand

    self.car\_tyres = tyres

car1 = car("blue","mercedes","4")

print(car1.car\_color)

print(car1.car\_brand)

print(car1.car\_tyres)

car2 = car("red","tata","3")

print(car2.car\_color)

print(car2.car\_brand)

print(car2.car\_tyres)

car3 = car("green","BMW","4")

print(car3.car\_color)

print(car3.car\_brand)

print(car3.car\_tyres)

**\_\_str\_\_() defines what you see when you print an object of a class.**

**# without using \_\_str\_\_ function**

class student:

    def \_\_init\_\_(self, firstname):

        self.firstname = firstname

s1 = student("Atharva")

print(s1)

**# after using \_\_Str\_\_ function**

class student:

    def \_\_init\_\_(self, firstname, lastname ):

        self.firstname = firstname

        self.lastname = lastname

    def \_\_str\_\_(self):

        return f"student name: {self.firstname} {self.lastname}"

s1 = student("Atharva","kesarkar")

print(s1)

**What are Object Methods?**

* **When we create a class, we can also create functions inside that class.**
* **These functions are called methods.**
* **They belong to the object (**they work *with* the object’s data**).**

class Person:

    def \_\_init\_\_(self, name):

        self.name = name

    def say\_hello(self):

        print("Hello! My name is", self.name)

p1 = Person("Atharva")

p1.say\_hello()

**Modify Object Properties**

**You can modify properties on objects like this:**

class Person:

  def \_\_init\_\_(self,  age):

    self.age = age

p1 = Person(36)

p1.age = 40

print(p1.age)

**Delete Object Properties**

**You can delete properties on objects by using the del keyword**

class Person:

def \_\_init\_\_(self, name, age):

self.name = name

self.age = age

def myfunc(self):

print("Hello my name is " + self.name)

p1 = Person("John", 36)

del p1.age

print(p1.age)

**The pass Statement**

**class definitions cannot be empty, but if you for some reason have a class definition with no content, put in the pass statement to avoid getting an error.**

class Person:

pass

**Inheritance**

**Inheritance** is a feature in Python where **one class (child class)** can **use the code** (variables and methods) from **another class (parent class)** without writing it again.

class human:

    def eat(self):

        print("yes i have eat")

    def work(self):

        print("yes i have done the work")

class male(human):

    pass

male\_1 = male()

male\_1.eat()

class human:

    def eat(self):

        print("yes i have eat")

    def work(self):

        print("yes i have done the work")

class male(human):

    def play(self):

        print("yes i have played the game")

male\_1 = male()

male\_1.eat()

male\_1.play()

male\_1.work()

class human:

    def eat(self):

        print("yes i have eat")

    def work(self):

        print("yes i have done the work")

class male(human):

    def play(self):

        print("yes i have played the game")

class female(male):

    pass

female\_1 = female()

female\_1.eat()

male\_1 = male()

male\_1.eat()

male\_1.play()

male\_1.work()

**# overidding**

class human:

    def eat(self):

        print("yes i have eat")

    def work(self):

        print("yes i have done the work")

class male(human):

    pass

male\_1 = male()

male\_1.eat()

**using attributes**

class Human:

    def \_\_init\_\_(self, eyes, hand):

        self.eyes = eyes

        self.hand = hand

class Male(Human):

    pass

male\_1 = Male(2,2)

print(male\_1.eyes)

print(male\_1.hand)