**Class Attributes in Python Programming**

In Python, **class attributes** are variables that are shared across all instances of a class. These attributes belong to the class itself rather than to any individual object created from the class.

**1. Defining Class Attributes**

A **class attribute** is defined inside the class but outside any instance methods. It is shared among all instances of the class.

**Example:**

class Car:

wheels = 4 # Class attribute

def \_\_init\_\_(self, brand, model):

self.brand = brand # Instance attribute

self.model = model # Instance attribute

# Creating instances

car1 = Car("Toyota", "Camry")

car2 = Car("Honda", "Civic")

print(car1.wheels) # Output: 4

print(car2.wheels) # Output: 4

print(Car.wheels) # Output: 4

Here, wheels is a **class attribute** shared among all instances of the Car class.

**2. Difference Between Class and Instance Attributes**

* **Class attributes**: Shared across all instances.
* **Instance attributes**: Unique to each instance.

**Example:**

class Employee:

company = "TechCorp" # Class attribute

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

self.name = name # Instance attribute

self.age = age # Instance attribute

emp1 = Employee("Alice", 30)

emp2 = Employee("Bob", 25)

print(emp1.company) # Output: TechCorp

print(emp2.company) # Output: TechCorp

# Changing class attribute for all instances

Employee.company = "NewTech"

print(emp1.company) # Output: NewTech

print(emp2.company) # Output: NewTech

Since company is a class attribute, changing it affects all instances.

**3. Modifying Class Attributes**

**A. Modifying via Class Name (Recommended)**

To modify a class attribute for all instances, change it using the class name:

Car.wheels = 6

Now, all instances will reflect this change.

**B. Modifying via an Instance (Creates an Instance Attribute)**

If an instance modifies a class attribute directly, it creates a new **instance attribute** instead of modifying the class attribute.

car1.wheels = 5 # Creates an instance attribute 'wheels' for car1

print(car1.wheels) # Output: 5

print(car2.wheels) # Output: 6 (remains unchanged)

Here, car1.wheels = 5 does not change the class attribute but creates a new instance attribute.

**4. Using Class Attributes in Methods**

**Class Method (@classmethod) to Modify Class Attributes**

If you want to modify a class attribute inside a method, use the @classmethod decorator:

class Car:

wheels = 4 # Class attribute

@classmethod

def set\_wheels(cls, new\_wheels):

cls.wheels = new\_wheels # Modifies class attribute

Car.set\_wheels(6)

print(Car.wheels) # Output: 6

**5. When to Use Class Attributes?**

* When you need a value that should be the same for all instances (e.g., number of wheels, company name, tax rate).
* When a shared state or configuration is required.
* When you want to optimize memory usage (instead of storing redundant data in each instance).

**Conclusion**

* **Class attributes** are shared among all instances.
* **Instance attributes** are unique to each object.
* Modifying a class attribute using an instance creates a new instance attribute.
* Use **class methods** to update class attributes safely.