Day 39

Class Methods in Python:

What is a Class Method?

A class method is a method that:

- Belongs to the class, not to an individual object
- Works with class-level data
- Uses @classmethod decorator
- Takes cls (class itself) as the first parameter

Think of it as a method that knows about the class, not a specific object.

Why Do We Need Class Methods?

We use class methods when:

- We want to access or modify class variables
- We want alternative ways to create objects (factory methods)
- The logic is related to the class as a whole

Syntax of a Class Method

```
class MyClass:
    @classmethod
    def my_class_method(cls):
        print(cls)
```

- @classmethod → tells Python this is a class method
- cls → refers to the class itself

Example:

```
1  class Student:
2      school_name = "ABC School"
3      @classmethod
4      def get_school_name(cls):
5          return cls.school_name
6  print(Student.get_school_name())
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
PS E:\Python\Day31-40\Day39> python main.py
ABC School
```

Example: Modifying Class Variables Using Class Method

```python
class Student:
    school_name = "ABC School"
    @classmethod
    def change_school(cls, new_name):
        cls.school_name = new_name
Student.change_school("XYZ School")
print(Student.school_name)
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
PS E:\Python\Day31-40\Day39> python main.py
XYZ School
```

Example: Factory Method (Very Important Use Case) -> Class methods are often used to create objects.

```python
class Student:
    def __init__(self, name, age):
        self.name = name
        self.age = age
    @classmethod
    def from_string(cls, data):
        name, age = data.split(",")
        return cls(name, int(age))
s1 = Student.from_string("Aditya,22")
print(s1.name, s1.age)
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  POR
PS E:\Python\Day31-40\Day39> python main.py
Aditya 22
```

Difference Between @staticmethod and @classmethod

Feature	Class Method	Static Method
Decorator	@classmethod	@staticmethod
First parameter	cls	None
Access class variables	✅ Yes	❌ No
Knows class info	✅ Yes	❌ No

Real-Life Analogy

- Class → School
- Class Method → School rules
- Instance Method → Student behavior

Summary:

- Class methods belong to the class, not objects
- Defined using @classmethod
- Use cls instead of self
- Used to access/modify class variables
- Commonly used as factory methods

Class Methods as Alternative Constructors in Python:

What is a Constructor?

A constructor is a special method that:

- Creates an object
- Initializes data
- In Python, it is __init__()

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

This is the main (default) constructor.

What is an Alternative Constructor?

An alternative constructor is:

- Another way to create an object
- Uses class methods
- Useful when input data is in a different format

Python does not support multiple constructors directly. So we use class methods instead.

Why Use Class Methods as Alternative Constructors?

We use them when:

- Data comes as string / file / list / dict
- We want clean and readable code
- Object creation logic differs

Example:

```python
class Student:
    def __init__(self, name):
        self.name = name
    @classmethod
    def from_string(cls, data):
        return cls(data)

s1 = Student("Aditya")          # normal constructor
s2 = Student.from_string("Rahul")  # alternative constructor
print(s1.name)
print(s2.name)
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS E:\Python\Day31-40\Day39> python main.py
Aditya
Rahul
```

--The End--