🧠 Lecture 17: Methods in OOP — Python

🔷 What Are Methods?

- Methods = **Functions inside a class**
- Allow objects to interact with their **attributes**
- 3 Types:
  - **Instance Methods**
  - **Class Methods**
  - **Static Methods**

🔷 1. Instance Methods

- Operate on **individual objects**
- First argument is always `self`
- Can **access & modify instance attributes**

✅ Example:

```
python
CopyEdit
class House:
    def __init__(self, color):
        self.color = color

    def show_color(self):
        print(f"This house is painted {self.color}")
python
CopyEdit
my_house = House("red")
my_house.show_color()
```

📝 Output:

```
csharp
CopyEdit
This house is painted red
```

🔷 2. Class Methods

- Shared across **class level**

- First argument is always `cls`
- Defined using `@classmethod`
- Can create new instances (like factory methods)

✅ Example:

```
python
CopyEdit
class House:
    def __init__(self, color, age):
        self.color = color
        self.age = age

    @classmethod
    def from_construction_year(cls, color, year):
        return cls(color, 2024 - year)
python
CopyEdit
my_house = House.from_construction_year("white", 2000)
print(my_house.age)  # Output: 24
```

🔷 3. Static Methods

- Independent **utility functions**
- No `self` or `cls`
- Use `@staticmethod` decorator
- Do not access/modify class or instance data

✅ Example:

```
python
CopyEdit
class House:
    @staticmethod
    def is_valid_area(area):
        return area > 0
python
CopyEdit
print(House.is_valid_area(120))   # True
print(House.is_valid_area(-50))   # False
```

🔁 Difference: Function vs Method

Feature	Function	Method
Defined in	Global scope	Inside a class

Feature	Function	Method
First argument	None	`self`, `cls`, or nothing
Called with	function()	obj.method() / Class.method()

□ Story-Driven Example: MathTool Class

```
python
CopyEdit
class MathTool:
    pi = 3.14159

    def __init__(self, radius):
        self.radius = radius

    def area(self):
        return MathTool.pi * self.radius * self.radius

    @classmethod
    def from_diameter(cls, diameter):
        return cls(diameter / 2)

    @staticmethod
    def is_valid_radius(radius):
        return radius > 0
```

◈ Usage:

```
python
CopyEdit
tool = MathTool(5)
print(tool.area())                  # 78.53975

tool2 = MathTool.from_diameter(10)
print(tool2.radius)                 # 5.0

print(MathTool.is_valid_radius(-1)) # False
```

□ Summary Table

Type	Decorator	First Param	Can Access	Use Case
Instance Method	None	`self`	Instance	Read/write object data
Class Method	@classmethod	`cls`	Class	Alternate constructors, shared logic

Type	Decorator	First Param	Can Access	Use Case
Static Method	@staticmethod	None	None	Utility, validation, math

---

✓ Final Thoughts

- **Instance methods** → Modify individual object
- **Class methods** → Work on class-level data or alternate constructor
- **Static methods** → Independent helpers inside class
- Practice to **master when & where to use each**