**1. @staticmethod**

* A static method **does not take any implicit first argument** (neither cls nor self).
* It behaves like a regular function but is bound to the class's namespace for organizational purposes.
* It **cannot access or modify the class or instance state**.
* You use it when the method logic is independent of the class or instance.

**Usage Scenario**

Use a static method when you need a utility function that logically belongs to the class but doesn't need to interact with class or instance variables.

**Real-World Example: A Utility Function**

class MathUtils:

@staticmethod

def add\_numbers(a, b):

return a + b

# Usage

result = MathUtils.add\_numbers(10, 20)

print(result) # Output: 30

Here, add\_numbers doesn't depend on the class state, but grouping it under MathUtils makes logical sense since it's math-related.

**2. @classmethod**

* A class method takes the class itself (cls) as the first argument.
* It can access and modify the class's state but not an instance's state (i.e., it can modify class variables but not instance variables).
* Use it when you need to create a method that interacts with or modifies the class itself.

**Usage Scenario**

Use a class method when you need to create alternative constructors or work with class-level data.

**Real-World Example: Alternative Constructor**

class Employee:

base\_salary = 50000 # Class-level variable

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

self.name = name

self.salary = salary

@classmethod

def from\_experience(cls, name, years\_of\_experience):

# Modify base salary based on experience

salary = cls.base\_salary + (years\_of\_experience \* 5000)

return cls(name, salary)

# Usage

employee1 = Employee("John", 60000) # Regular constructor

employee2 = Employee.from\_experience("Jane", 5) # Alternative constructor

print(employee1.salary) # Output: 60000

print(employee2.salary) # Output: 75000

Here, from\_experience creates an Employee object by calculating the salary based on years of experience, using the class variable base\_salary.

**When to Use Which**

* Use @staticmethod for **general utility functions** that don't interact with the class or instance.
* Use @classmethod for methods that **interact with or modify class-level data** or when implementing **alternative constructors**.

**Comparison Table**

| **Aspect** | **@staticmethod** | **@classmethod** |
| --- | --- | --- |
| First Parameter | None | cls (reference to the class itself) |
| Access to Class State | No | Yes |
| Access to Instance State | No | No |
| Typical Use Case | Utility functions | Alternative constructors, class-level logic |