Why It’s a Static Method

- It doesn’t use any **instance data (`self`)** or **class data (`cls`)**.
- It just provides **general information** relevant to the class.
- You can call it **without creating an object**.

💡 Analogy:

Think of it like a **note pinned to the class** —

> "Hey, all cars are vehicles meant for transport."

No matter which car you create (or even if you don’t create any), that statement remains true.

difference between a **static method** and a **class method** in practice.

Let’s look at both side by side 👇

🚗 Example: `@staticmethod` vs `@classmethod`

```
class Car:
    total_cars = 0  # class variable

    def __init__(self, brand, model):
        self.brand = brand
        self.model = model
        Car.total_cars += 1   # increases every time a Car object is
created

    def full_name(self):
        return f"{self.brand} {self.model}"
    # ✅ Static Method: General info, independent of any object or
class
    @staticmethod
    def general_description():
        return "Cars are means of transport — they help people
travel."

    # ✅ Class Method: Works with class-level data (like total_cars)
    @classmethod
    def show_total_cars(cls):
        return f"Total cars created: {cls.total_cars}"
```

🧪 Example Usage

```
# Create some cars
car1 = Car("Tata", "Nexon")
car2 = Car("Tesla", "Model S")

# Static method → general info
print(Car.general_description())

# Class method → class-level info
print(Car.show_total_cars())
```

🧾 Output

```
Cars are means of transport — they help people travel.
Total cars created: 2
```

⚙️ Comparison Table

Feature	`@staticmethod`	`@classmethod`
First parameter	None	`cls` (class reference)

Can access instance variables (`self`)?	❌ No	❌ No
Can access class variables (`cls.total_cars`)?	❌ No	✅ Yes
Typical use case	Utility or general info	Factory methods, counters, class-level logic
Called using	Class or object	Class or object

🧠 Think of it this way:

You're saying...	Use this
"I just want a generic helper that's related to the class, not any object."	`@staticmethod`
"I want to do something related to the whole class, not one instance."	`@classmethod`

🧩 Quick Analogy:

Imagine the `Car` class is a **car factory**:

- `@classmethod` → talks to the **factory itself** (how many cars built so far)
- `@staticmethod` → just gives a **general fact** about cars (they're used for travel)