🎨Python Methods Cheat Sheet (Colorful Notes)

🏅 Instance Method

- **Default method** (no decorator).
- First parameter → `self` (the **object**).
- Can access **instance variables** and **class variables**.

```
class Demo:
    def instance_method(self):
        print("I work with objects")
```

🏅 Class Method

- Defined with `@classmethod`.
- First parameter → `cls` (the **class**).
- Can **modify class variables**.
- Often used as **alternative constructors**.

```
class Student:
    school = "ABC University"

    @classmethod
    def change_school(cls, new_school):
        cls.school = new_school
```

📌Usage:

```
Student.change_school("XYZ College")
```

🏅 Static Method

- Defined with `@staticmethod`.
- Takes **no ** `** or **`.
- Works like a normal function inside a class.

```
class Math:
    @staticmethod
    def add(a, b):
        return a + b
```

📷Side-by-Side Comparison

Method Type	Decorator	First Param	Refers To
Instance Method	None	`self`	Object
Class Method	`@classmethod`	`cls`	Class
Static Method	`@staticmethod`	None	Nothing passed

Without Decorators

If you forget the decorator:

- Python **always passes the object** as first argument.
- Even if you call it `cls`, it's really just `self` (the instance).

```
class Demo:
    def wrong_method(cls):
        print(cls)

obj = Demo()
obj.wrong_method()  # prints object, not class
```

🧱Tip:

- Use `self` → when you work with **object data**.
- Use `cls` → when you work with **class-level data**.
- Use `@staticmethod` → when your function is just a helper, doesn't need object or class.

Perfect! Let's make a **clear, colorful, and concise note** on **Instance vs Class Variables** that fits nicely with your previous class method notes.

🎨Python Variables Cheat Sheet (Colorful Notes)

🏅 Instance Variables

- Belong to **object/instance**.
- Each object has its **own copy**.
- Defined **inside **`` using `self`.
- Can access **both instance & class variables**, but changes affect **only that object**.

```
class Student:
    school = "ABC University"   # class variable

    def __init__(self, name, age):
        self.name = name        # instance variable
        self.age = age

s1 = Student("Ali", 20)
s2 = Student("Aisha", 22)

print(s1.name, s1.age)  # Ali 20
print(s2.name, s2.age)  # Aisha 22
```

🏅 Class Variables

- Belong to the **class**, shared by **all instances**.
- Defined **directly in the class**, outside any methods.
- Changes affect **all instances** (unless overridden by an instance).

```
class Student:
    school = "ABC University"   # class variable

s1 = Student()
s2 = Student()

print(s1.school)  # ABC University
print(s2.school)  # ABC University

Student.school = "XYZ College"   # modify class variable
print(s1.school)  # XYZ College
print(s2.school)  # XYZ College
```

📷Comparison Table

Feature	Instance Variable	Class Variable
Belongs to	Object/instance	Class
Shared by all	🔩No	🔗Yes
Defined with	`self.var` in `__init__`	Directly in class
Accessed via	`obj.var`	`Class.var` or `obj.var`
Changes affect	Only that object	All objects

Tips

- Use **instance variables** for **object-specific data** (e.g., name, age).
- Use **class variables** for **shared data** (e.g., school, company, config).
- Can **override class variable** in an instance, but that creates a new **instance variable**.

```
s1.school = "Local College"  # creates new instance variable, doesn't change
class
print(s1.school)  # Local College
print(s2.school)  # XYZ College
```