**1. Instance Variables and Instance Methods**

**Instance Variables:**

* **Definition**: Instance variables are variables that are associated with a specific instance (object) of a class. These variables are defined using self inside methods and are unique to each instance of the class.
* **Scope**: They can be accessed by any method within the class using the self keyword.
* **Purpose**: To store the state or data of an object.

**Instance Methods:**

* **Definition**: Instance methods are functions defined within a class that operate on instance variables. They take self as their first parameter, which refers to the current object.
* **Scope**: They can access and modify instance variables and other instance methods.

**Example:**

* class Employee:
* def \_\_init\_\_(self, name, age):
* # Instance variables
* self.name = name
* self.age = age
* def display\_info(self):
* # Instance method accessing instance variables
* print(f"Name: {self.name}, Age: {self.age}")
* # Creating instances
* emp1 = Employee("Alice", 30)
* emp2 = Employee("Bob", 25)
* # Calling instance method
* emp1.display\_info()  # Output: Name: Alice, Age: 30
* emp2.display\_info()  # Output: Name: Bob, Age: 25
* self.name and self.age are **instance variables**.
* display\_info() is an **instance method** because it operates on instance variables.

**2. Class Variables and Class Methods**

**Class Variables:**

* **Definition**: Class variables are shared by all instances of the class. These variables are defined inside the class but outside any methods.
* **Scope**: They are accessible by both instance methods and class methods, but they are shared by all instances of the class.
* **Purpose**: To store data that is shared across all instances of the class.

**Class Methods:**

* **Definition**: Class methods are methods that take the class itself as the first argument (cls), not the instance (self). They are defined using the @classmethod decorator.
* **Scope**: Class methods can access and modify class variables, but they cannot access instance variables unless passed an instance explicitly.

**Example:**

* class Employee:
* company\_name = "ABC Corp"  # Class variable
* def \_\_init\_\_(self, name, age):
* self.name = name  # Instance variable
* self.age = age  # Instance variable
* @classmethod
* def change\_company\_name(cls, new\_name):
* cls.company\_name = new\_name  # Class method modifying class variable
* def display\_info(self):
* print(f"Name: {self.name}, Age: {self.age}, Company: {Employee.company\_name}")
* # Creating instances
* emp1 = Employee("Alice", 30)
* emp2 = Employee("Bob", 25)
* # Calling instance method
* emp1.display\_info()  # Output: Name: Alice, Age: 30, Company: ABC Corp
* emp2.display\_info()  # Output: Name: Bob, Age: 25, Company: ABC Corp
* # Calling class method to change class variable
* Employee.change\_company\_name("XYZ Ltd.")
* # Displaying info again after changing the class variable
* emp1.display\_info()  # Output: Name: Alice, Age: 30, Company: XYZ Ltd.
* emp2.display\_info()  # Output: Name: Bob, Age: 25, Company: XYZ Ltd.
* company\_name is a **class variable**.
* change\_company\_name() is a **class method** because it modifies the class variable.
* display\_info() is an **instance method**, which still accesses the class variable.

**3. Static Variables and Static Methods**

**Static Variables:**

* **Definition**: Python does not have a direct concept of static variables like some other languages (e.g., C++ or Java). However, you can define **static variables** using a class-level variable that is **not tied to any specific instance** or **class** method.
* **Scope**: They exist at the class level and are shared by all instances, similar to class variables.
* **Purpose**: To store values that are not specific to any instance or class method but are still associated with the class.

**Static Methods:**

* **Definition**: Static methods are methods that do not require access to any instance or class-specific data (i.e., they don't need self or cls parameters). They are defined using the @staticmethod decorator.
* **Scope**: Static methods cannot access instance variables or class variables unless explicitly passed as arguments.

**Example:**

* class Employee:
* # Static variable
* company\_name = "ABC Corp"
* def \_\_init\_\_(self, name, age):
* self.name = name
* self.age = age
* @staticmethod
* def company\_info():
* print(f"Company: {Employee.company\_name}")
* def display\_info(self):
* print(f"Name: {self.name}, Age: {self.age}")
* # Creating instances
* emp1 = Employee("Alice", 30)
* emp2 = Employee("Bob", 25)
* # Calling static method
* Employee.company\_info()  # Output: Company: ABC Corp
* # Calling instance method
* emp1.display\_info()  # Output: Name: Alice, Age: 30
* emp2.display\_info()  # Output: Name: Bob, Age: 25
* company\_info() is a **static method** because it does not access self or cls.
* company\_name is a **class variable**, which is not tied to any particular instance, but can be accessed in the static method.

**Summary of Differences:**

| **Feature** | **Instance Variable** | **Class Variable** | **Static Variable** |
| --- | --- | --- | --- |
| **Definition** | Variable bound to an instance of the class. | Variable shared by all instances of the class. | Class-level variable that is independent of instances. |
| **Access** | Accessed via self. | Accessed via cls or self. | Accessed via class\_name or self. |
| **Scope** | Specific to an instance of the class. | Shared by all instances of the class. | Shared across the class, not tied to instances or methods. |
| **Purpose** | Store data specific to the object. | Store data shared by all instances. | Typically used for data that is not tied to an instance or class. |
| **Modification** | Modified via self in instance methods. | Modified via cls in class methods. | Modified via the class itself. |

| **Feature** | **Instance Method** | **Class Method** | **Static Method** |
| --- | --- | --- | --- |
| **Definition** | Method that operates on an instance's data. | Method that operates on class-level data. | Method that does not access instance or class-level data. |
| **First Parameter** | self (refers to the instance). | cls (refers to the class). | None (does not access instance or class-level data). |
| **Access** | Access instance variables and methods. | Access class variables and other class methods. | Can only access class variables or external parameters. |
| **Purpose** | To operate on individual instance data. | To operate on class data or modify class state. | To perform operations not dependent on class or instance state. |

**Conclusion:**

* **Instance variables** are for storing object-specific data, and **instance methods** act on that data.
* **Class variables** are shared across all instances of the class, and **class methods** modify or interact with these class-level variables.
* **Static variables** (though not a formal concept in Python) behave similarly to class variables but are independent of instances, and **static methods** perform operations that don’t need access to instance-specific or class-specific data.

Each of these types of variables and methods has its place depending on whether you're working with object-specific data, class-level data, or performing utility-like tasks that don’t depend on any class data.