Python OOP: Class Attributes

Class Attributes

Key Takeaways

- **Class Attributes**
  - They **belong to the class** and all instances share the same class attribute. There is only one copy of the attribute.
    - ✓ For example: if we want our class to keep track of how many accounts have been created, the BankAccount class could have a accounts_created class attribute and all the instances of this class would access that same value.
  - The value of a class attribute is **shared across instances**. They all access the value from the same source, the class.
  - **Changing the value** of a class attribute **affects all instances**, since they take the value from the same source.
  - You can access and modify the values of class attributes.
  - The value of a class attribute can be accessed using the name of the class. No instance is required to access class attributes.

Class Attributes

Key Takeaways

- **General Syntax to Assign a Value to a Class Attribute Within the Class**

```
<class_attribute> = <value>
```

- **Example**

```
class BankAccount:

    accounts_created = 0

    def __init__(self, number, client):
        self.number = number
        self.client = client
        self.balance = balance
        BankAccount.accounts_created += 1

    def display_balance(self):
        print(self.balance)
```

Shared across instance
s

Class Attributes

Key Takeaways

- **General Syntax to Access the Value of a Class Attribute**

```
<ClassName>.<class_attribute>
```

- **Example**

```
class BankAccount:

    accounts_created = 0

    def __init__(self, number, client):
        self.number = number
        self.client = client
        self.balance = balance
        BankAccount.accounts_created += 1

    def display_balance(self):
        print(self.balance)
```

You can access and work with this value

Key Takeaways

- **General Syntax to Modify the Value of a Class Attribute**

```
<ClassName>.<class_attribute> = <value>
```

- **Example**

```
class BankAccount:

    accounts_created = 0

    def __init__(self, number, client):
        self.number = number
        self.client = client
        self.balance = balance
        BankAccount.accounts_created += 1

    def display_balance(self):
        print(self.balance)

BankAccount.accounts_created = 5
```

The value is changed for all instances