Cheat Sheet

Attributes & Methods

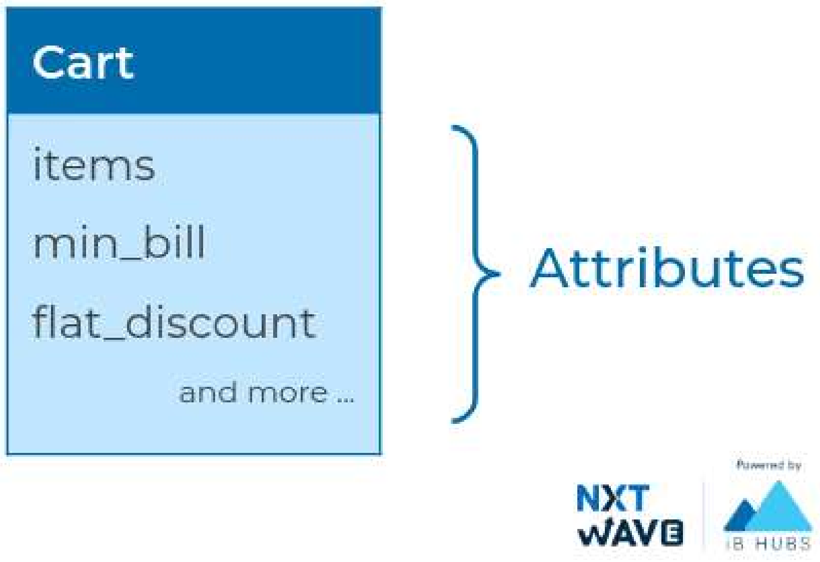
Shopping Cart

- Users can add different items to their shopping cart and checkout.
- The total value of the cart should be more than a minimum amount (Rs. 100/-) for the checkout.
- During Offer Sales, all users get a flat discount on their cart and the minimum cart value will be Rs. 200/-.

Attributes

Broadly, attributes can be categorized as

- Instance Attributes
- Class Attributes



Instance Attributes

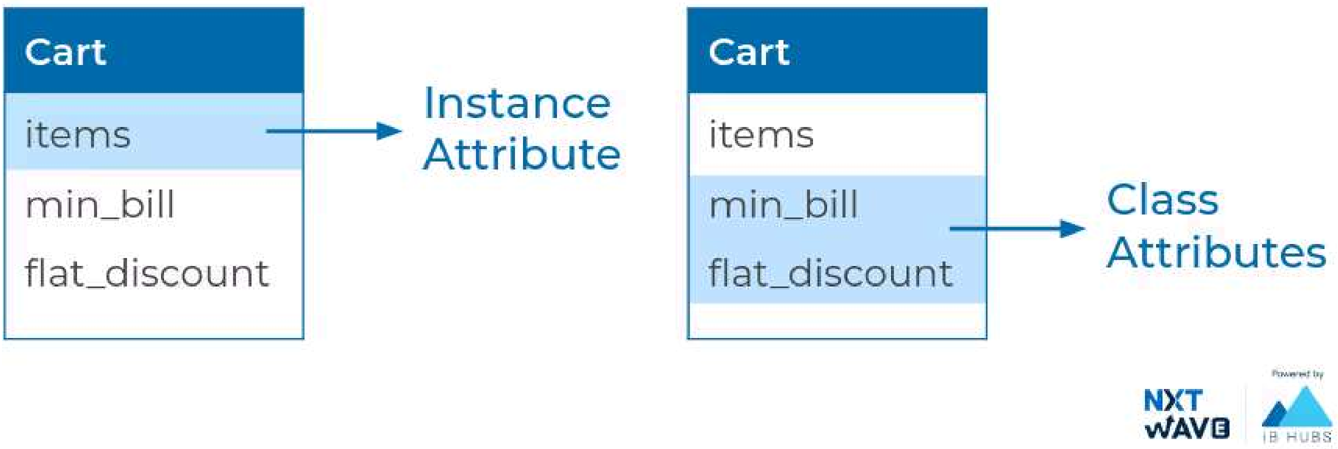
Attributes whose value can differ for each instance of class are modeled as instance attributes.

Ex: Items in Cart

Class Attributes

Attributes whose values stay common for all the objects are modelled as Class Attributes.

Ex: Minimum Cart Bill,
Flat Discount



Accessing Instance Attributes

Code

PYTHON

```python
class Cart:
  flat_discount = 0
  min_bill = 100
  def __init__(self):
      self.items = {}
  def add_item(self,..):
      self.items[item_name] = quantity
  def display_items(self):
      print(items)
a = Cart()
```

Expand

Output

```
NameError: name 'items' is not defined
```

Instance attributes can only be accessed using instance of class.

Self

`self` passed to method contains the object, which is an instance of class.

Code

PYTHON

```python
class Cart:
  flat_discount = 0
  min_bill = 100
  def __init__(self):
      self.items = {}
  def add_item(self,item_name, quantity):
      self.items[item_name] = quantity
  def display_items(self):
      print(self)

```

Expand

Output

```
<__main__.Cart object at 0x7f6f83c9dfd0> <__main__.Cart object at 0x7f6f83c9dfd0>
```

Accessing Using Self

Code

PYTHON

```python
class Cart:
    flat_discount = 0
    min_bill = 100
    def __init__(self):
        self.items = {}
    def add_item(self, item_name,quantity):
        self.items[item_name] = quantity
    def display_items(self):
        print(self.items)
a = Cart()
```

Expand

Output

```
{"book": 3}
```

Accessing Using Object

Code

PYTHON

```python
class Cart:
    flat_discount = 0
    min_bill = 100
    def __init__(self):
        self.items = {}
    def add_item(self, item_name,quantity):
        self.items[item_name] = quantity
    def display_items(self):
        print(self.items)
a = Cart()
```

Expand

Output

```
{'book': 3}
```

Accessing Using Class

Code

PYTHON

```python
class Cart:
    flat_discount = 0
    min_bill = 100
    def __init__(self):
        self.items = {}
    def add_item(self, item_name,quantity):
        self.items[item_name] = quantity
    def display_items(self):
        print(self.items)
print(Cart.items)
```

Output

```
AttributeError: type object 'Cart' has no attribute 'items'
```

Accessing Class Attributes

Example 1

Code

PYTHON

```python
class Cart:
    flat_discount = 0
    min_bill = 100
    def __init__(self):
        self.items = {}

print(Cart.min_bill)
```

Output

```
100
```

Example 2

Code

PYTHON

```python
class Cart:
    flat_discount = 0
    min_bill = 100
    def __init__(self):
        self.items = {}
    def print_min_bill(self):
        print(Cart.min_bill)

a = Cart()
a.print_min_bill()
```

Output

```
100
```

Updating Class Attribute

Code

PYTHON

```python
class Cart:
    flat_discount = 0
    min_bill = 100
    def print_min_bill(self):
        print(Cart.min_bill)
a = Cart()
b = Cart()
Cart.min_bill = 200
print(a.print_min_bill())
print(b.print_min_bill())
```

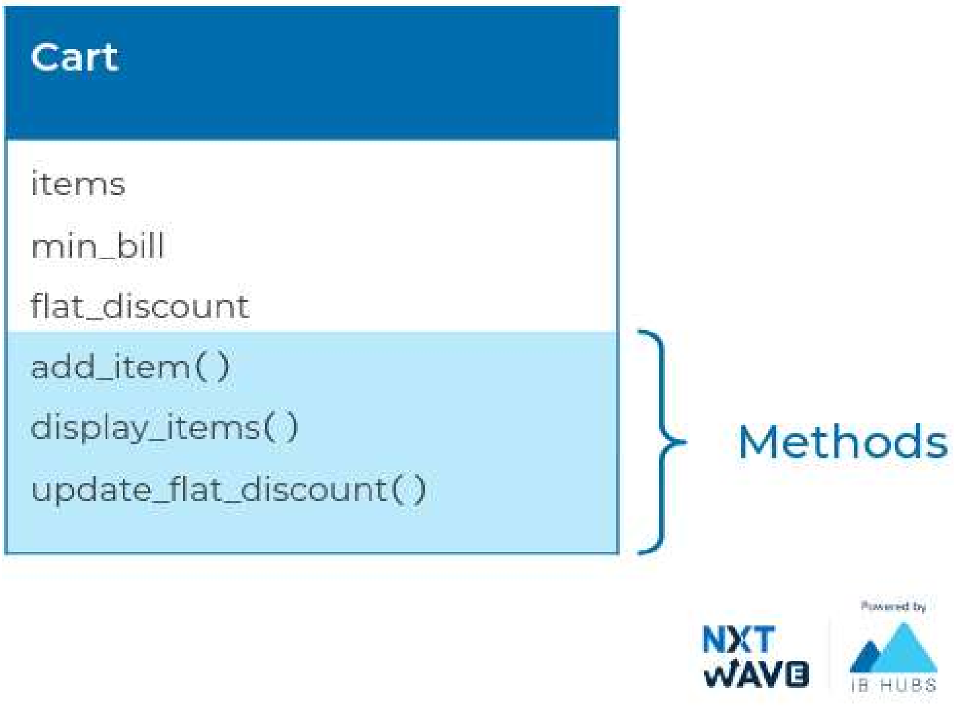
Output

```
200
200
```

Method

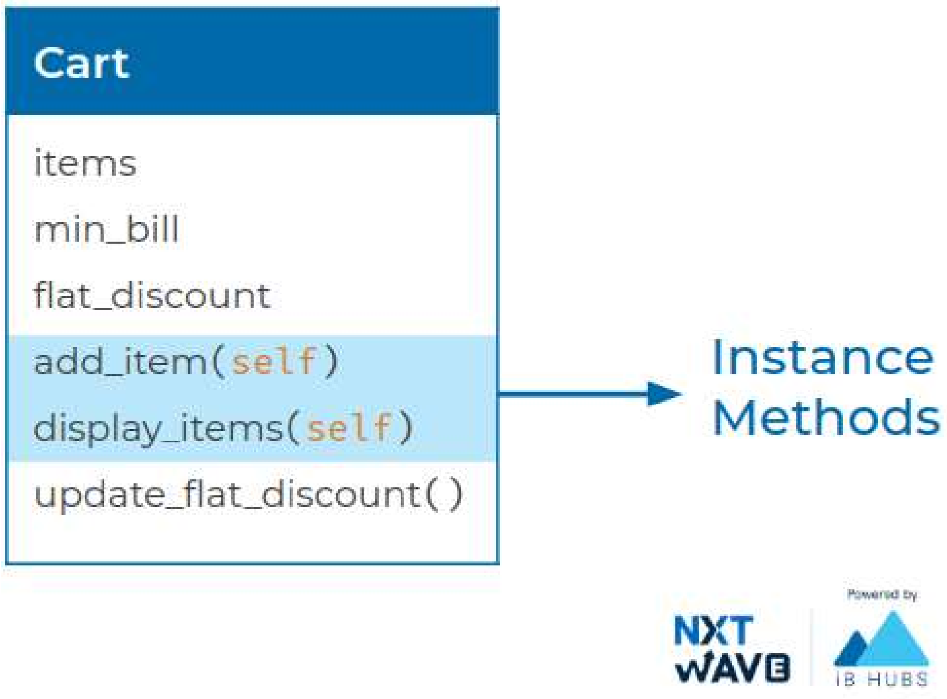
Broadly, methods can be categorized as

- Instance Methods
- Class Methods
- Static Methods



Instance Methods

Instance methods can access all attributes of the instance and have self as a parameter.



Example 1

Code

PYTHON

```python
class Cart:
   def __init__(self):
       self.items = {}
   def add_item(self, item_name,quantity):
      self.items[item_name] = quantity
   def display_items(self):
      print(self.items)

a = Cart()
a.add_item("book", 3)
```

Expand

Output

```
{'book': 3}
```

Example 2

Code

PYTHON

```python
class Cart:
  def __init__(self):
      self.items = {}
  def add_item(self, item_name,quantity):
     self.items[item_name] = quantity
     self.display_items()
  def display_items(self):
     print(self.items)

a = Cart()
```

Expand

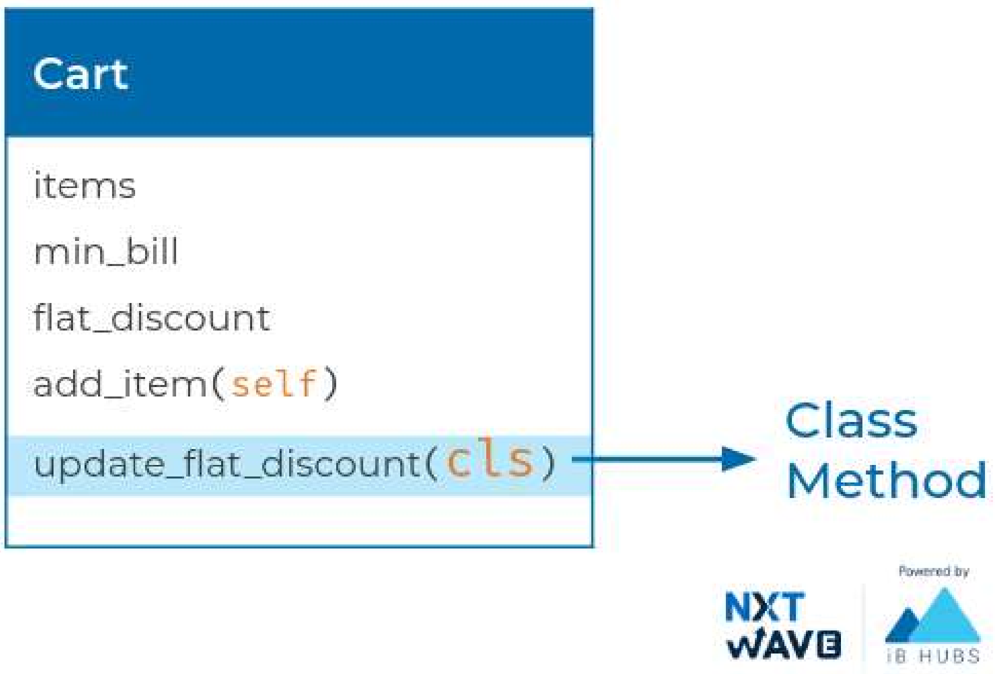
Output

```
{'book': 3}
```

Class Methods

Methods which need access to class attributes but not instance attributes are marked as Class Methods.
For class methods, we send

`cls` as a parameter indicating we are passing the class.



Code

PYTHON

```python
class Cart:
    flat_discount = 0
    min_bill = 100
    @classmethod
    def update_flat_discount(cls,
                                new_flat_discount):
        cls.flat_discount = new_flat_discount

Cart.update_flat_discount(25)
print(Cart.flat_discount)
```

Output

```
25
```

@classmethod decorator marks the method below it as a class method.

We will learn more about decorators in upcoming sessions.

Accessing Class Method

Code

PYTHON

```python
class Cart:
  flat_discount = 0
  min_bill = 100
  @classmethod
  def update_flat_discount(cls, new_flat_discount):
      cls.flat_discount = new_flat_discount

  @classmethod
  def increase_flat_discount(cls, amount):
      new_flat_discount = cls.flat_discount + amount
```

Expand

Output

```
50
```

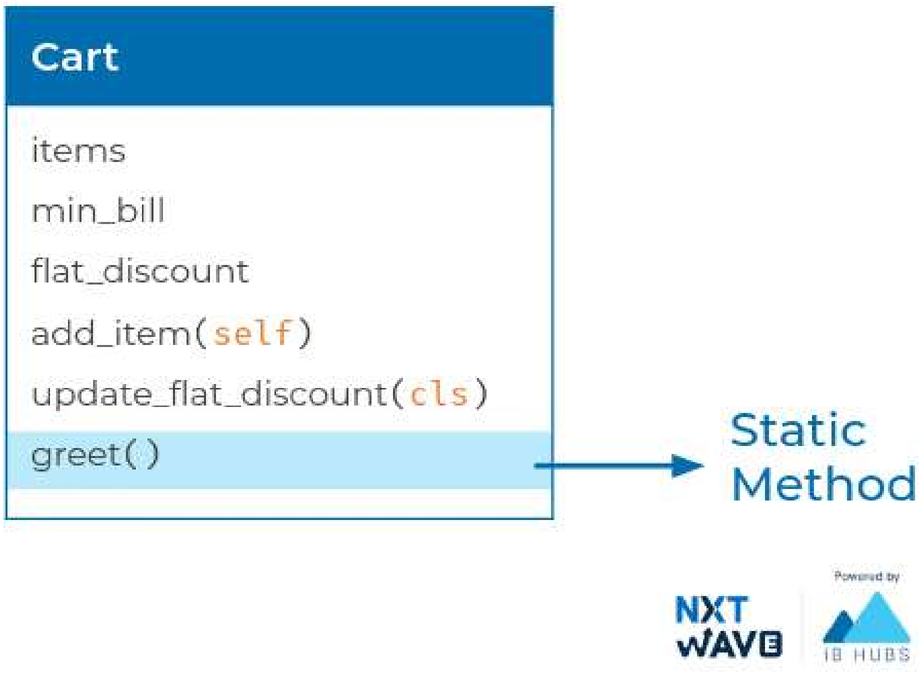
Static Method

We might need some generic methods that don't need access to either instance or class attributes. These type of methods are called Static Methods.

Usually, static methods are used to create utility functions which make more sense to be part of the class.

`@staticmethod` decorator marks the method below it as a static method.

We will learn more about decorators in upcoming sessions.



Code

PYTHON

```python
class Cart:

    @staticmethod
    def greet():
        print("Have a Great Shopping")

Cart.greet()
```

Output

```
Have a Great Shopping
```

Overview of Instance, Class & Static Methods

Instance Methods	Class Methods	Static Methods
self as parameter	cls as parameter	No cls or self as parameters
No decorator required	Need decorator @classmethod	Need decorator @staticmethod

Instance Methods	Class Methods	Static Methods
Can be accessed through object(instance of class)	Can be accessed through class	Can be accessed through class

Submit Feedback