Cheat Sheet

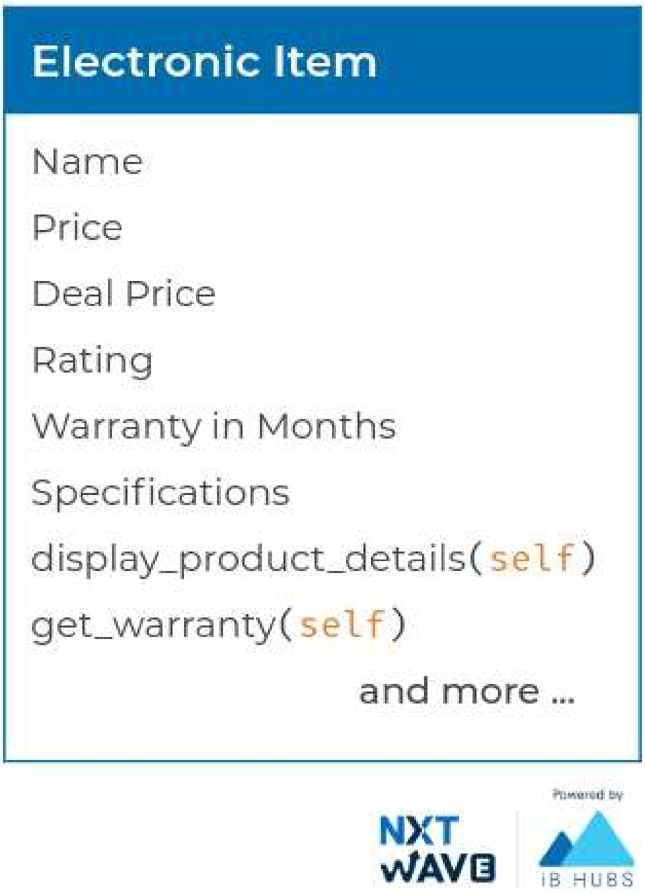
Inheritance

Products

Lets model e-commerce site having different products like Electronics, Kids Wear, Grocery, etc.

Electronic Item

Following are few attributes & methods for an Electronic product.

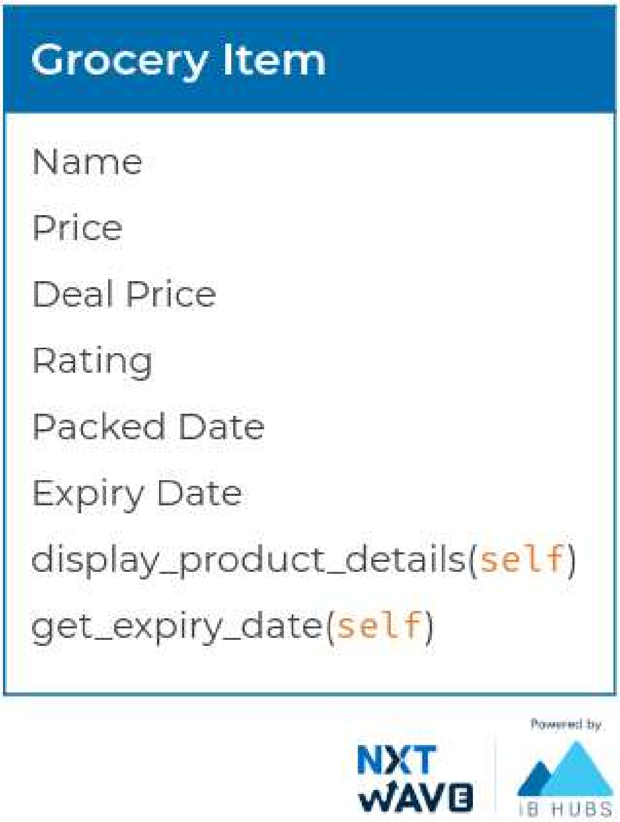


Electronic Item

Name
Price
Deal Price
Rating
Warranty in Months
Specifications
display_product_details(self)
get_warranty(self)
and more ...

Grocery Item

Similarly, attribute & methods for a Grocery item.

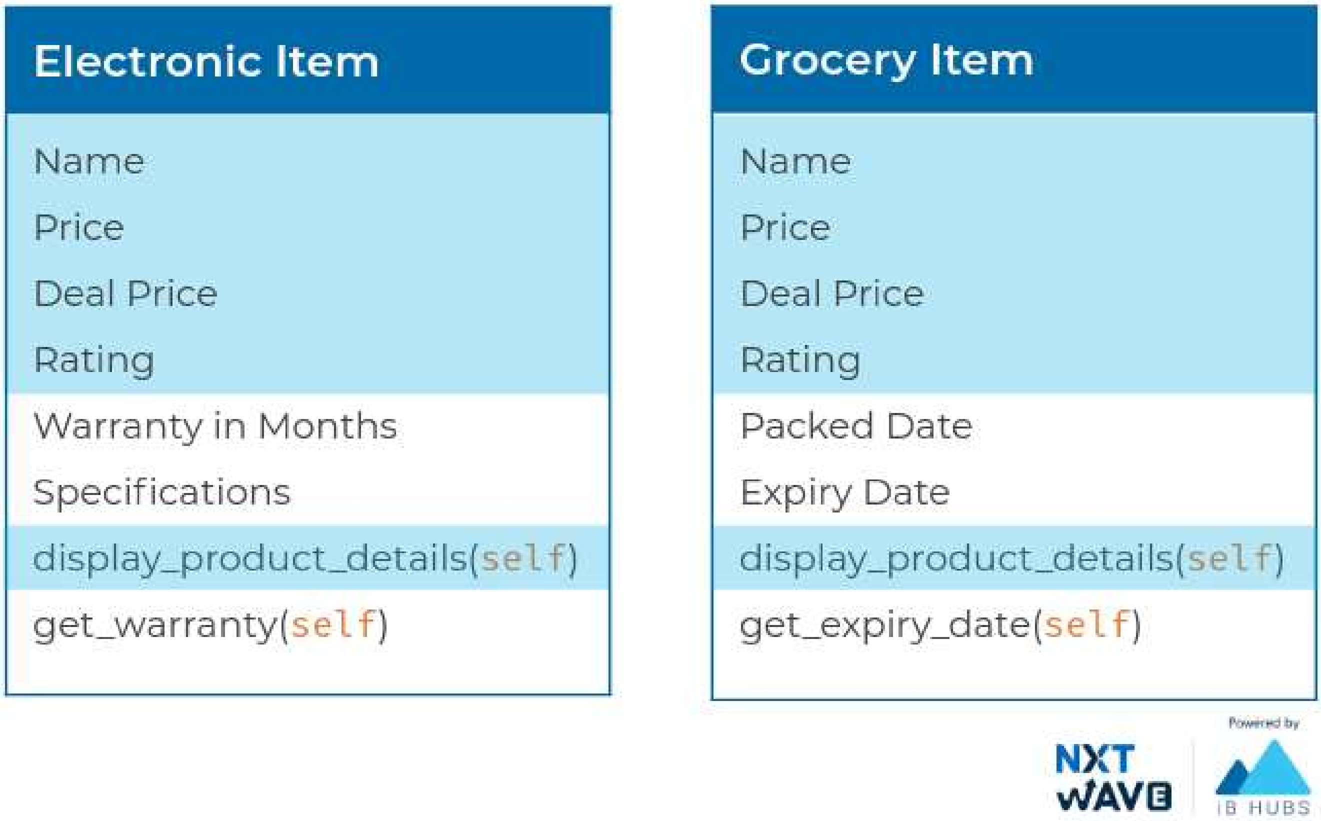


Grocery Item

Name
Price
Deal Price
Rating
Packed Date
Expiry Date
display_product_details(self)
get_expiry_date(self)

Common Attributes & Methods

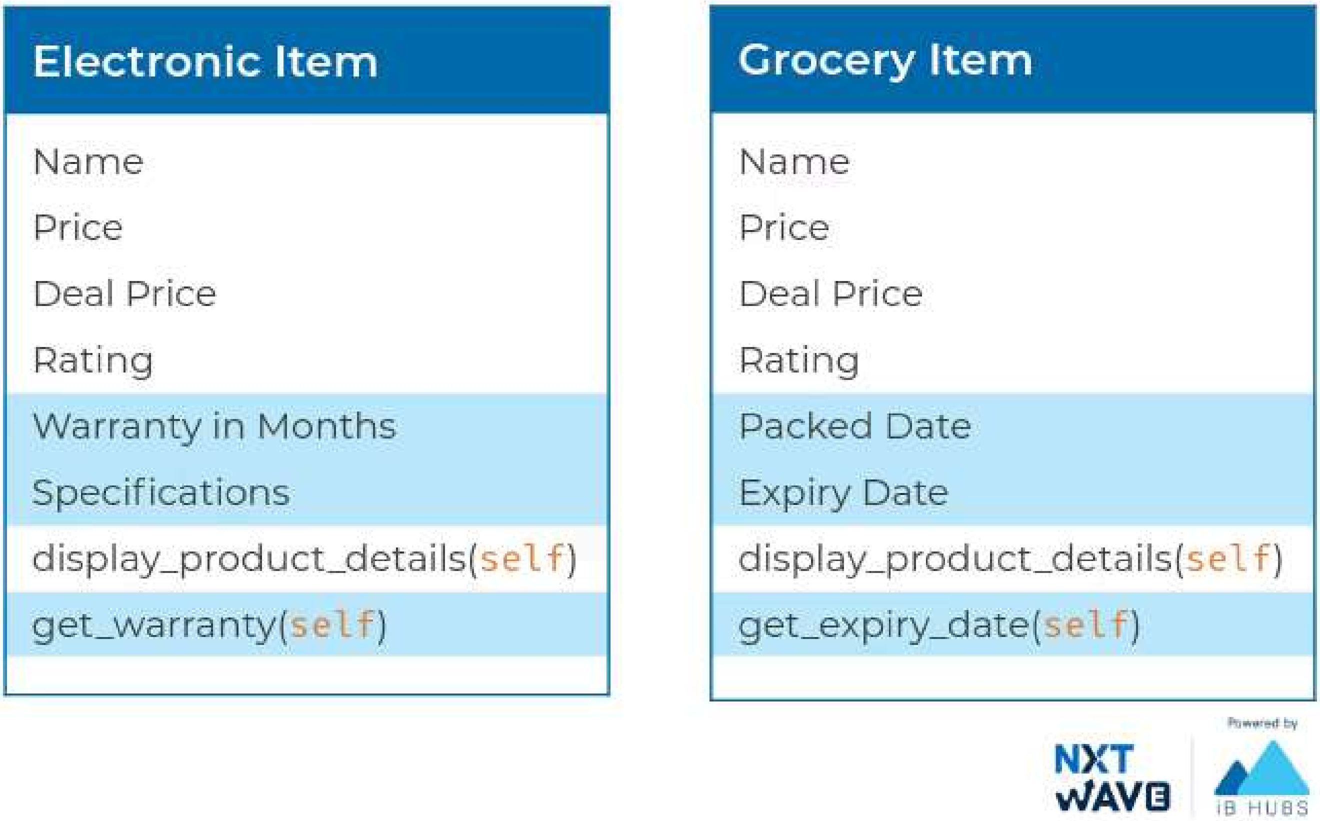
All these products Electronics, Kids Wear, Grocery etc.. have few common attributes & methods.



Electronic Item	Grocery Item
Name	Name
Price	Price
Deal Price	Deal Price
Rating	Rating
Warranty in Months	Packed Date
Specifications	Expiry Date
display_product_details(self)	display_product_details(self)
get_warranty(self)	get_expiry_date(self)

Specific Attributes & Methods

Also, each product has specific attributes & methods of its own.



Electronic Item

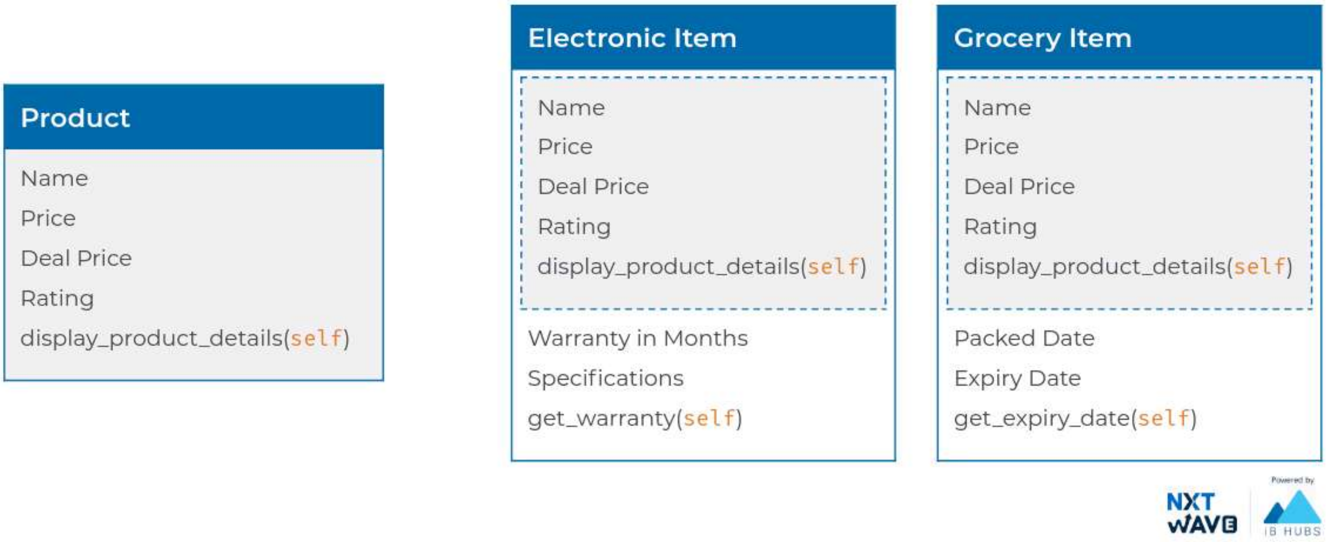
Name
Price
Deal Price
Rating
Warranty in Months
Specifications
display_product_details(self)
get_warranty(self)

Grocery Item

Name
Price
Deal Price
Rating
Packed Date
Expiry Date
display_product_details(self)
get_expiry_date(self)

Electronic & Grocery Items

Electronic Item & Grocery Item will have all attributes & methods which are common to all products.
Lets Separate the common attributes & methods as Product



Product

Name
Price
Deal Price
Rating
display_product_details(self)

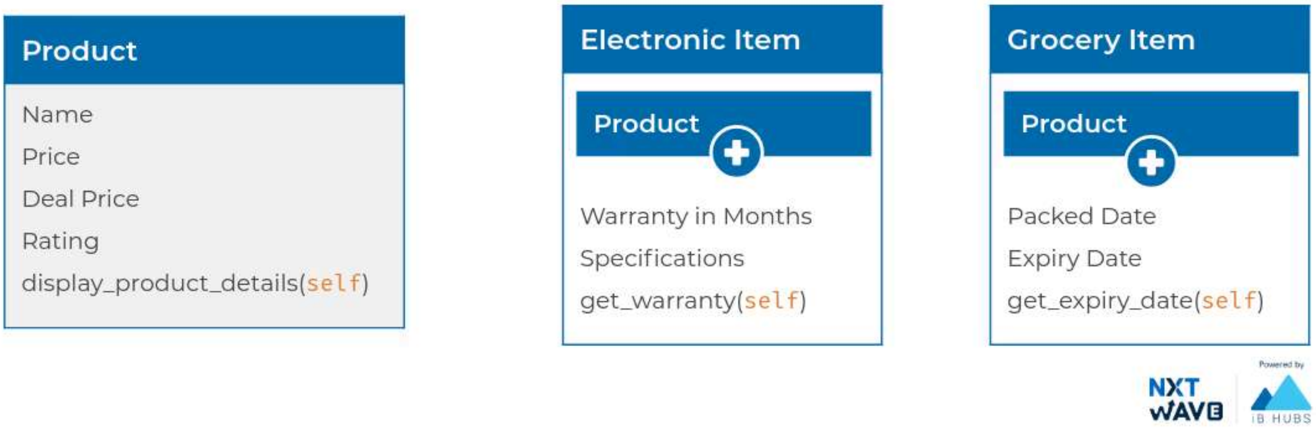
Electronic Item

Name
Price
Deal Price
Rating
display_product_details(self)
Warranty in Months
Specifications
get_warranty(self)

Grocery Item

Name
Price
Deal Price
Rating
display_product_details(self)
Packed Date
Expiry Date
get_expiry_date(self)

Modelling Classes



Advantages of Modelling Classes as above

- Reusability
- Clear Separation
- More Organized

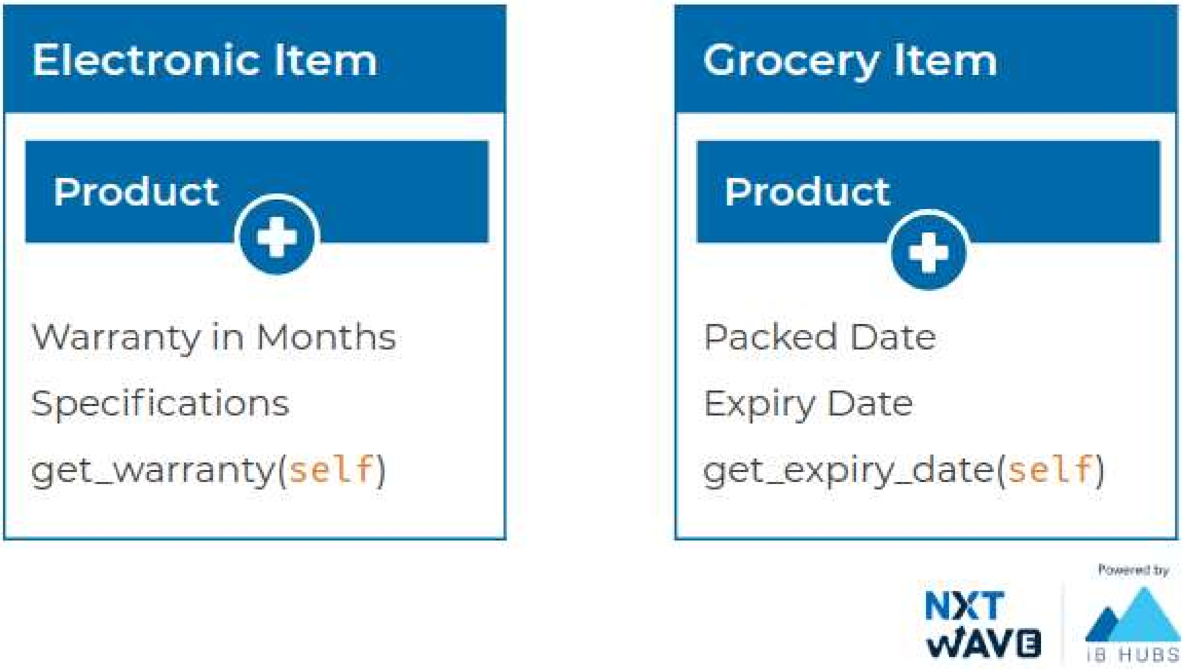
Inheritance

Inheritance is a mechanism by which a class inherits attributes and methods from another class.

With Inheritance, we can have

`ElectronicItem` inherit the attributes & methods from `Product` instead of defining them again.

Product is Super/Base/Parent Class and ElectronicItem is Sub/Derived/Child Class.



Super Class

Code

PYTHON

```python
class Product:
    def __init__(self, name, price, deal_price, ratings):
        self.name = name
        self.price = price
        self.deal_price = deal_price
        self.ratings = ratings
        self.you_save = price - deal_price
    def display_product_details(self):
        print("Product: {}".format(self.name))
        print("Price: {}".format(self.price))
```

Expand

Output

```
Product: Shoes Price: 500 Deal Price: 250 You Saved: 250 Ratings: 3.5
```

Sub Class

The subclass automatically inherits all the attributes & methods from its superclass.

Example 1

Code

PYTHON

```python
class Product:
    def __init__(self, name, price, deal_price, ratings):
        self.name = name
        self.price = price
        self.deal_price = deal_price
        self.ratings = ratings
        self.you_save = price - deal_price
    def display_product_details(self):
        print("Product: {}".format(self.name))
        print("Price: {}".format(self.price))
```

Expand

Output

```
Product: TV Price: 45000 Deal Price: 40000 You Saved: 5000 Ratings: 3.5
```

Example 2

Code

PYTHON

```python
class Product:
    def __init__(self, name, price, deal_price, ratings):
        self.name = name
        self.price = price
        self.deal_price = deal_price
        self.ratings = ratings
        self.you_save = price - deal_price
    def display_product_details(self):
        print("Product: {}".format(self.name))
        print("Price: {}".format(self.price))
```

Expand

Output

```
Product: milk Price: 25 Deal Price: 20 You Saved: 5 Ratings: 3
```

Example 3

Code

PYTHON

```python
class Product:
   def __init__(self, name, price, deal_price, ratings):
        self.name = name
        self.price = price
        self.deal_price = deal_price
        self.ratings = ratings
        self.you_save = price - deal_price
    def display_product_details(self):
        print("Product: {}".format(self.name))
        print("Price: {}".format(self.price))
```

Expand

Output

```
24
```

In the above example, calling `set_warranty` will create an attribute `warranty_in_months` .

Super Class & Sub Class

Superclass cannot access the methods and attributes of the subclass.

Code

PYTHON

```
class Product:
    def __init__(self, name, price, deal_price, ratings):
        self.name = name
        self.price = price
        self.deal_price = deal_price
        self.ratings = ratings
        self.you_save = price - deal_price
    def display_product_details(self):
        print("Product: {}".format(self.name))
        print("Price: {}".format(self.price))
```

Expand

Output

```
AttributeError: 'Product' object has no attribute 'set_warranty'
```

Sub Class Method

Code

PYTHON

```python
class Product:
   def __init__(self, name, price, deal_price, ratings):
       self.name = name
       self.price = price
       self.deal_price = deal_price
       self.ratings = ratings
       self.you_save = price - deal_price
   def display_product_details(self):
       print("Product: {}".format(self.name))
       print("Price: {}".format(self.price))
```

Expand

Output

```
Product: TV Price: 45000 Deal Price: 40000 You Saved: 5000 Ratings: 3.5
```

Calling Super Class Method

We can call methods defined in superclass from the methods in the subclass.

Code

PYTHON

```python
class Product:
   def __init__(self, name, price, deal_price, ratings):
       self.name = name
       self.price = price
       self.deal_price = deal_price
       self.ratings = ratings
       self.you_save = price - deal_price
   def display_product_details(self):
       print("Product: {}".format(self.name))
       print("Price: {}".format(self.price))
```

Expand

Output

```
Product: TV Price: 45000 Deal Price: 40000 You Saved: 5000 Ratings: 3.5 Warranty 24 months
```

Submit Feedback