```
#------------------shopping_cart.py------------------

# create class shopcart
#     class member nmae quantiy price
    # class method init addtocart updatecart rempvefromcart printbil

import  csv

list_item = [{"name": "Banana", "price": 30},
             {"name": "Boost", "price": 130},
             {"name": "Fizz", "price": 30},
             {"name": "Milk", "price": 20},
             {"name": "Bread", "price": 80},
             {"name": "Cheese", "price": 70}]
class ShoppingCart:
    def __init__(self,list_item):
        self.list_item = list_item
        for items in self.list_item:
            self.name = items['name']
            self.price = items['price']
        self.cart = {}
    def _showcart(self):
        return self.cart
    def _addcart(self, name, quantity):
        if quantity <= 0:
            return self.cart
        elif name in self.cart:
            q = self.cart[name]
            qty = q + quantity
            self.cart[name] = qty
        else:
            self.cart[name] = quantity
        # return self.cart

    # def _updatecart(self, name,quantity): # for pytest purpose
    def _updatecart(self, name):
        if name in self.cart:
            quantity=int(input("Enter the quantity"))
            if quantity == 0:
                del self.cart[name]
                return self.cart
            self.cart[name] = quantity
        return self.cart

    def _removecart(self, name):
        if name in self.cart:
            del self.cart[name]
        return self.cart

    def _printbill(self):
        gtotal = 0
        with open('ShopBill.csv', mode='w') as pfile:
            pfile_writer = csv.writer(pfile, delimiter=" ", quotechar=' ',
quoting=csv.QUOTE_MINIMAL)
```

```
            pfile_writer.writerow("{:<10} {:<10} {:<10} {:<10}".format('Items',
'Quantity', 'Price', 'Total'))
            pfile_writer.writerow('*' * 80)
            for i in self.cart:
                qty = self.cart[i]
                for x in self.list_item:
                    if i==x['name']:
                        self.price=x['price']

                priceofsingleitem = qty * self.price
                gtotal += priceofsingleitem
                pfile_writer.writerow("{:<10} {:<10} {:<10} {:<10}".format(i,
qty, self.price, priceofsingleitem))
            pfile_writer.writerow('*' * 80)
            pfile_writer.writerow("{:<10} {:<10} {:<10} {:<10}".format('', '',
'Grand_Total', gtotal))
            suc="Bill Printed"
            return  suc

#
def main():
    sh=ShoppingCart(list_item)
    print("Welcome to VBazar")
    while True:
        x = int(input("1.View Item \n 2.Add item to cart \n 3.Update cart \n
4.Remove item from cart \n 5.Exit and print bill"))
        if x == 1:
            for i in list_item:
                print(i['name'], " ", i["price"])
        elif x == 2:
            while True:
                item = input("Enter the item to add")
                qty = int(input("enter the quantity"))
                sh._addcart(item, qty)
                print(sh._showcart())
                con = input("do you want to add more y/n")
                if con == 'n':
                    break
        elif x == 3:
            while True:
                print(sh._showcart())
                item = input("Enter the item to update")
                sh._updatecart(item)
                print(sh._showcart())
                con = input("do you want to update more y/n")
                if con == 'n':
                    break
        elif x == 4:
            print(sh._showcart())
            item = input("Enter the item to delete")
            sh._removecart(item)
            print(sh._showcart())
        elif x == 5:
            sh._printbill()
            print("bill printed")
            break
        else:
            raise Exception("Invalid selection")

main()
```

```
#-----------------Shoppytest.py------------------
```

```
import pytest
from shopping_cart import ShoppingCart

# initialising the list item

list_item = [{"name": "Banana", "price": 30},
             {"name": "Boost", "price": 130},
             {"name": "Fizz", "price": 30},
             {"name": "Milk", "price": 20},
             {"name": "Bread", "price": 80},
             {"name": "Cheese", "price": 70}]
#Test case 1 check wheather cart is initialsed to 0
@pytest.fixture
def Shop():
    return ShoppingCart(list_item)

# check if cart is initially 0
def test_cartis_initaly_zero(Shop):
    assert Shop.cart=={}

# check adding an item
def test_add_single_item(Shop):
    Shop._addcart("Banana",5)
    assert Shop.cart=={'Banana':5}

# check adding same item update the quantity
def test_add_multiple_item(Shop):
    Shop._addcart("Banana",6)
    Shop._addcart("Banana", 5)
    assert Shop.cart == {'Banana': 11}

# test case to check updation
def test_update_existing_item(Shop):
    Shop._addcart("Banana",6)
    Shop._updatecart("Banana", 2)
    assert Shop.cart == {'Banana': 2}
```

```
# test case to check updation of an non exisiting item
def test_update_non_existing_item(Shop):
    Shop._addcart("Banana",6)
    Shop._updatecart("Milk", 2)
    assert Shop.cart == {'Banana': 6}

# test case to check the removal if an existing item
def test_remove_existing_item(Shop):
    Shop._addcart("Banana",6)
    Shop._removecart("Banana")
    assert Shop.cart == {}

# test case to check the removal if an non existing item
def test_remove_non_existing_item(Shop):
    Shop._addcart("Banana",6)
    Shop._removecart("Milk")
    assert Shop.cart == {'Banana': 6}

#Test case for multiplr operation
def test_multiple_operation(Shop):
    Shop._addcart("Banana", 5)
    Shop._addcart("Banana", 5)
    Shop._addcart("Milk", 3)
    Shop._updatecart("Milk", 2)
    Shop._addcart("Fizz", 3)
    Shop._removecart("Milk")
    assert Shop.cart == {'Banana': 10,'Fizz':3}

# Test case for bill print
def test_print_bill(Shop):
    Shop._addcart("Milk", 3)
    Shop._updatecart("Milk", 2)
    Shop._addcart("Fizz", 3)
    assert Shop._printbill() == "Bill Printed"

# Test case for add item with zero quantity
def test_add_item_with_zero_quantity(Shop):
    Shop._addcart("Milk", 0)
    assert Shop._showcart()=={}

# Test case for add item with negative quantity
def test_add_item_with_negative_quantity(Shop):
    Shop._addcart("Milk", -5)
    assert Shop._showcart()=={}

# Test case to update item with zero which delete the item
def test_update_item_with_zero_quantity(Shop):
    Shop._addcart("Milk",5)
    Shop._updatecart("Milk", 0)
    assert Shop._showcart()=={}
```