Faculty of Engineering & Technology
Electrical & Computer Engineering Department

Linux Laboratory ENCS 3130
Project #2 Report

Python Project

Prepared by:

HibaTullah Jaouni	**ID:** 1201154
Tala Abahra	**ID:** 1200102

Instructor: Dr. Mohammad Jubran
Assistant: Eng. Ibrahim Injas

Section: 3

Date of Submission: 31 / 8 / 2022

Abstract:

The aim of this project is to make us more familiar with python programming and Object Oriented (OOP) concepts. In this project, we are required to build a software system that manages the product items in a warehouse and the distribution of these products to the supermarkets.

Table of Contents:

Abstract: 1

Main Window: 3

Project Components: 6

Product Class: 6

Warehouse Class 7

Supermarket Class 7

Company Class 8

Code Working Principals: 9

Add product to warehouse 11

Add new Supermarket to the System 13

List of Items in the warehouse based on Expiry Date 15

Clear an item from the warehouse: 17

Distribute products from the Warehouse to a Supermarket: 18

Generate a report about the sales status of the warehouse 21

Conclusion: 23

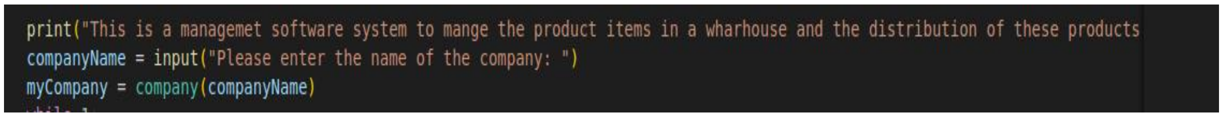
References: 24

Main Window:

The following code display the main window which is the interface between the user and the program code.

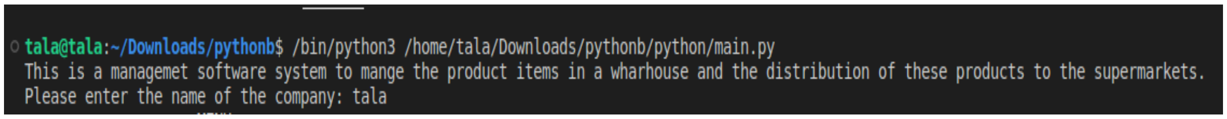
First, the program will ask user to enter the company's name (after importing the company class). Note that this code allows us to establish more than one company and deal with it mainly, that is, the project is not limited to one company only.

Code:



```
print("This is a managemet software system to mange the product items in a wharhouse and the distribution of these products
companyName = input("Please enter the name of the company: ")
myCompany = company(companyName)
```

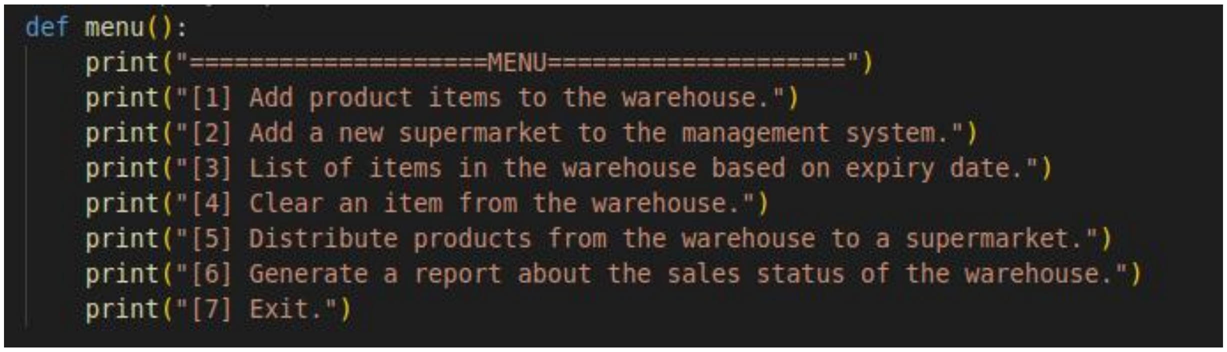
User Interface:



```
tala@tala:~/Downloads/pythonb$ /bin/python3 /home/tala/Downloads/pythonb/python/main.py
This is a managemet software system to mange the product items in a wharhouse and the distribution of these products to the supermarkets.
Please enter the name of the company: tala
```

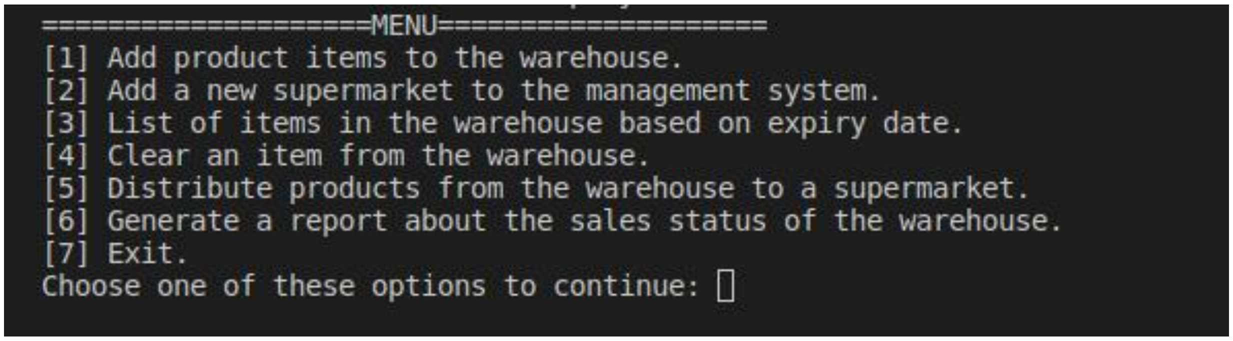
Then, the program will display a menu and let the user enter the number of requested operations.

Code:



```
def menu():
    print("====================MENU====================")
    print("[1] Add product items to the warehouse.")
    print("[2] Add a new supermarket to the management system.")
    print("[3] List of items in the warehouse based on expiry date.")
    print("[4] Clear an item from the warehouse.")
    print("[5] Distribute products from the warehouse to a supermarket.")
    print("[6] Generate a report about the sales status of the warehouse.")
    print("[7] Exit.")
```

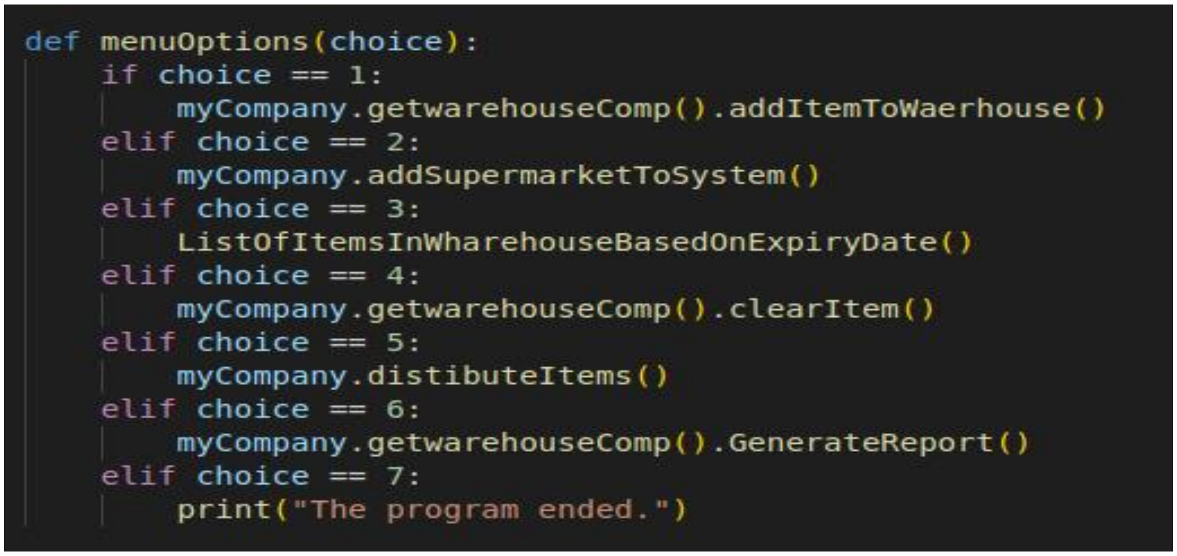
User Interface:



```
====================MENU====================
[1] Add product items to the warehouse.
[2] Add a new supermarket to the management system.
[3] List of items in the warehouse based on expiry date.
[4] Clear an item from the warehouse.
[5] Distribute products from the warehouse to a supermarket.
[6] Generate a report about the sales status of the warehouse.
[7] Exit.
Choose one of these options to continue: 
```

Depending on the user input, the program will call the required method to achieve his purpose.

If the user entered number 7, the program will break out of the loop and end the program.



```python
def menuOptions(choice):
    if choice == 1:
        myCompany.getwarehouseComp().addItemToWaerhouse()
    elif choice == 2:
        myCompany.addSupermarketToSystem()
    elif choice == 3:
        ListOfItemsInWharehouseBasedOnExpiryDate()
    elif choice == 4:
        myCompany.getwarehouseComp().clearItem()
    elif choice == 5:
        myCompany.distibuteItems()
    elif choice == 6:
        myCompany.getwarehouseComp().GenerateReport()
    elif choice == 7:
        print("The program ended.")
```

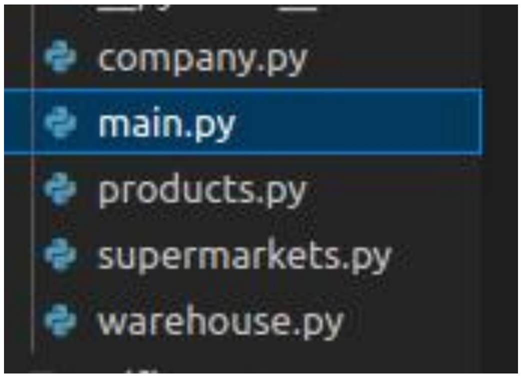
The code of the main window (main.py class).

```
impor From company
* from company import#
:()menu def
("====================MENU====================")print
(".Add product items to the warehouse [1]")print
(".Add a new supermarket to the management system [2]")print
(".List of items in the warehouse based on expiry date [3]")print
(".Clear an item from the warehouse [4]")print
(".Distribute products from the warehouse to a supermarket [5]")print
(".Generate a report about the sales status of the warehouse [6]")print
(".Exit [7]")print

:()ListOfItemsInWharehouseBasedOnExpiryDate def
(" :Please enter the date (dd/mm/yyyy)")input = inputDate
(inputDate)listItems_ExpiryDate.myCompany

:(choice)menuOptions def
:1 == choice if
()addItemToWaerhouse.()getwarehouseComp.myCompany
:2 == choice elif
()addSupermarketToSystem.myCompany
:3 == choice elif
()ListOfItemsInWharehouseBasedOnExpiryDate
:4 == choice elif
()clearItem.()getwarehouseComp.myCompany
:5 == choice elif
()distibuteItems.myCompany
:6 == choice elif
```

```
()GenerateReport.()getwarehouseComp.myCompany
:7 == choice elif
(".The program ended")print


This is a managemet software system to mange the product items in a ")print
(".wharhouse and the distribution of these products to the supermarkets
(" :Please enter the name of the company")input = companyName
(companyName)company = myCompany
:1 while
()menu
:try
((" :Choose one of these options to continue")input)int = choice
:((7 < choice) | (1 > choice))if
(".Option not defined")print
("...Please try again")print
continue
:ValueError except
("...Error in entering the type of data")print
("...Please try again")print
continue

(choice)menuOptions
:7 == choice if
Break
```

Project Components:

The project consists of mainly five classes; which are the main class that we discussed in the previous section, the company class which keep tracking with its supermarkets and its warehouse. The supermarket class and warehouse class are part of the company and the product class is a class that describes the features of any product.

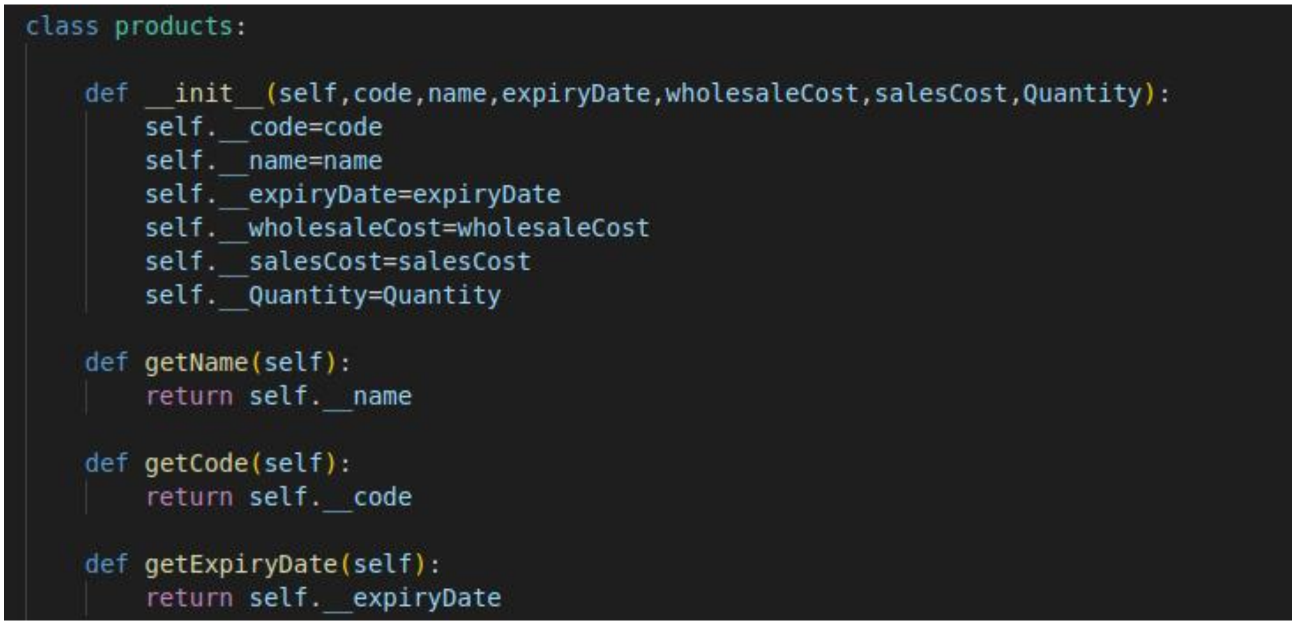


Product Class:

This class is designed to describe the features of any product we have using the constructor which are the product code, name, expiry date, whole sale cost, sales cost and it's available quantity.

The class also contains getters for all attributes, but not all setters; because there are some attributes that I won't change their values.

Note that all the data variables in that class are private to get encapsulation features for it.



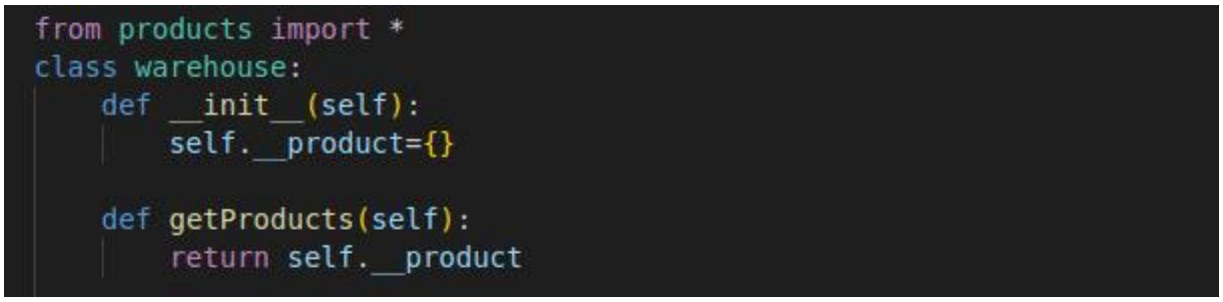
```
class products:

    def __init__(self,code,name,expiryDate,wholesaleCost,salesCost,Quantity):
        self.__code=code
        self.__name=name
        self.__expiryDate=expiryDate
        self.__wholesaleCost=wholesaleCost
        self.__salesCost=salesCost
        self.__Quantity=Quantity

    def getName(self):
        return self.__name

    def getCode(self):
        return self.__code

    def getExpiryDate(self):
        return self.__expiryDate
```

This class contains some other method that we want to user later in our program.

Warehouse Class

This class is designed to keep tracking with the products stored in the warehouse. This class is created by default when each company is created. This class has no-arg constructor that only create an empty dictionary of products indicating that there are no products can be involved when a warehouse is created.

The class has a ***getProducts()*** method that returns only the products in the warehouse, and it contains another method that we are going discuss them later.

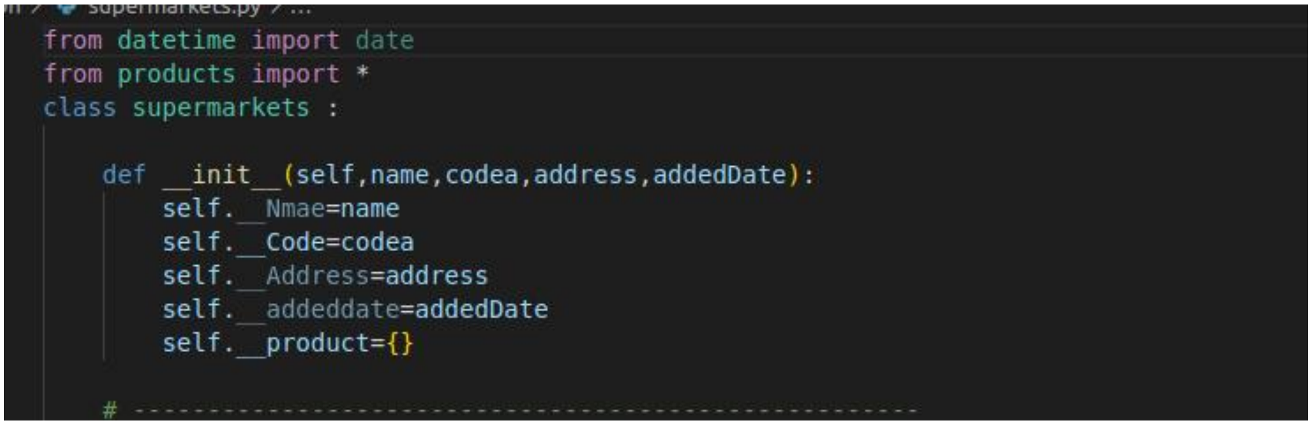


```
from products import *
class warehouse:
    def __init__(self):
        self.__product={}

    def getProducts(self):
        return self.__product
```

Supermarket Class

This class is specialized when creating any supermarket that is related to the company. The supermarket has some attributes such as name, code, address and added date and products it stores. When first creating a supermarket for the company, the dictionary of products will be empty as shown in the figure below.

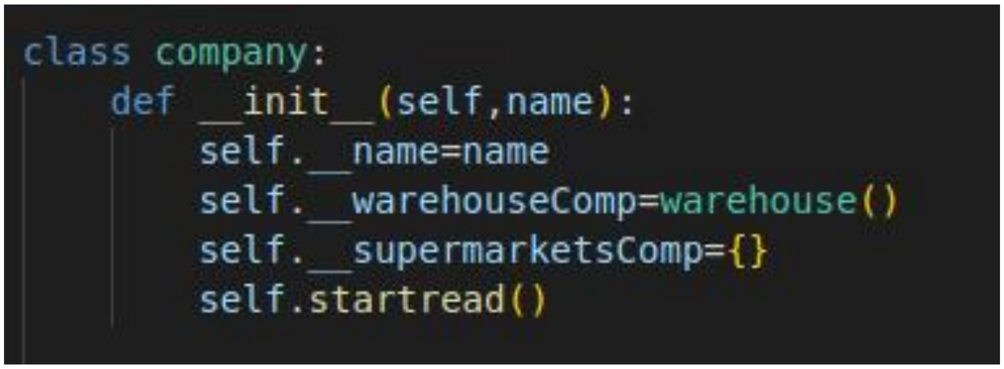
Note that the added date is current date; we imported datetime library to establish the current date.



```
from datetime import date
from products import *
class supermarkets :

    def __init__(self,name,codea,address,addedDate):
        self.__Nmae=name
        self.__Code=codea
        self.__Address=address
        self.__addeddate=addedDate
        self.__product={}

    # ----------------------------------------------------
```

Company Class

This class has one attribute which is the name, and it's a part of the constructor as shown in the figure. Also, an empty warehouse and a dictionary of supermarket will be created.

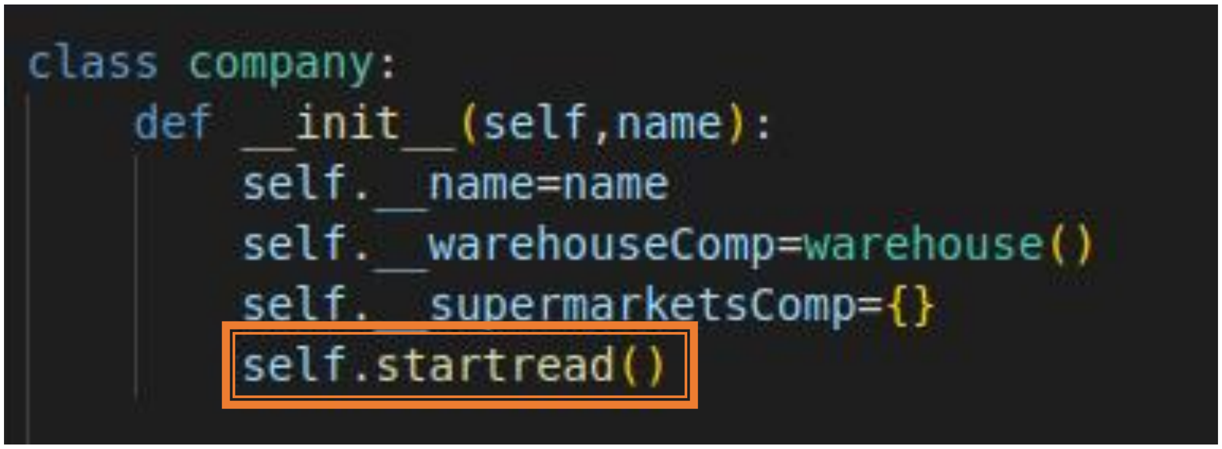


```
class company:
    def __init__(self,name):
        self.__name=name
        self.__warehouseComp=warehouse()
        self.__supermarketsComp={}
        self.startread()
```

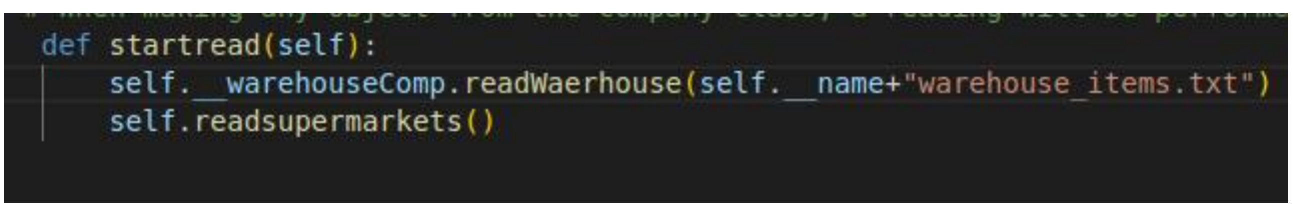
Code Working Principals:

When creatin a company object, any company has its own data stored in special files for it, and when starting work, this data must be taken to deal with. If there is no data, we will start creating it.

The method highlighted below is required to do so when creating any new company.

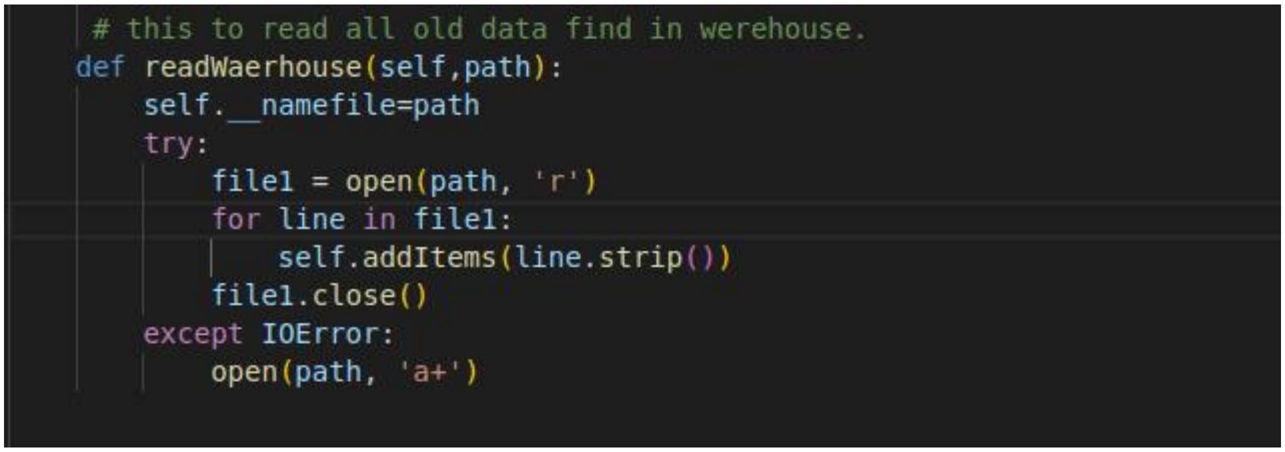


```
class company:
    def __init__(self,name):
        self.__name=name
        self.__warehouseComp=warehouse()
        self.__supermarketsComp={}
        self.startread()
```



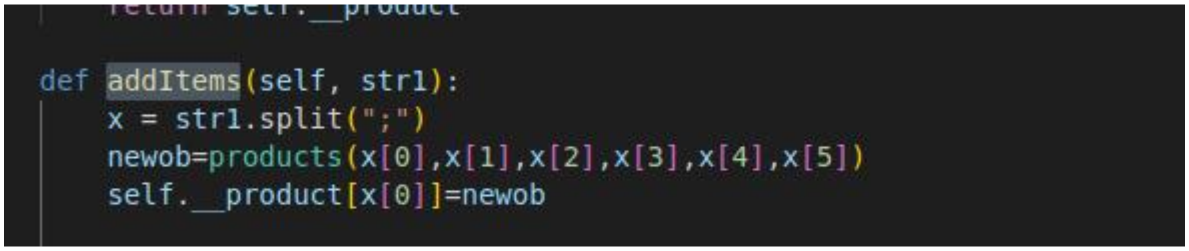
```
# when making any object from the company class, a reading will be performe
def startread(self):
    self.__warehouseComp.readWaerhouse(self.__name+"warehouse_items.txt")
    self.readsupermarkets()
```

The first line in the method calls another method in the warehouse class, this class reads the available products in the warehouse from a file called ***self._name+"warehouse_items.txt"*** and store the data in a dictionary; this is done by calling another method in warehouse class called ***addItems(line***). the key of the dictionary is the product code and its value is the product object. If the file does not exist (a new company is created for the first time), the program will create a file called ***self._name+"warehouse_items.txt"*** to store the company's warehouse data.



```
    # this to read all old data find in werehouse.
    def readWaerhouse(self,path):
        self.__namefile=path
        try:
            file1 = open(path, 'r')
            for line in file1:
                self.addItems(line.strip())
            file1.close()
        except IOError:
            open(path, 'a+')
```

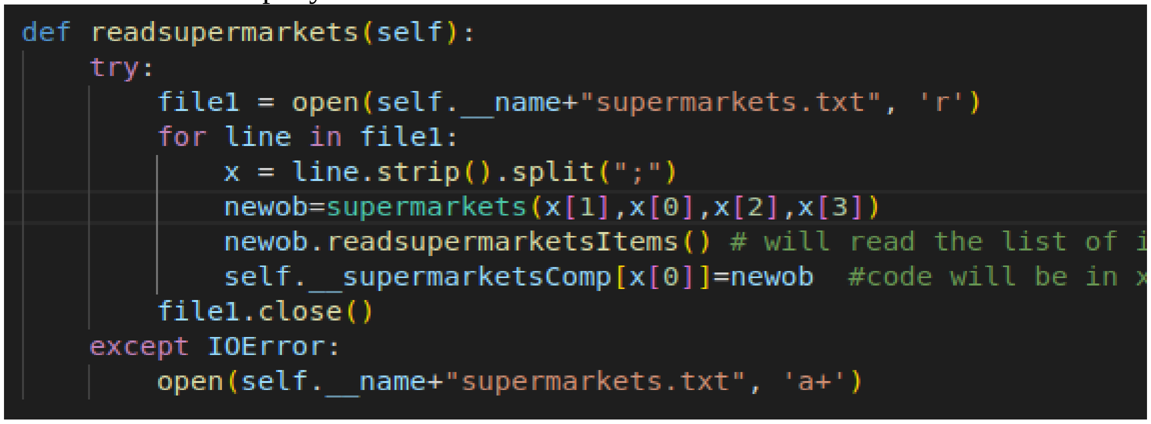
addItems() method is one on the methods in the warehouse class that takes a string and split that string according to Semillon (;). Then, it will create a new instance of product class and add it to the dictionary of products in the warehouse.



```
        return self.__product

    def addItems(self, str1):
        x = str1.split(";")
        newob=products(x[0],x[1],x[2],x[3],x[4],x[5])
        self.__product[x[0]]=newob
```

For the next function call, when creating any company, you must have a file for the names of its supermarkets (file name: company name + supermarket) and each supermarket has a file for the items inside it (file name: supermarket name + items).
The process of obtaining this data:

- We try to access the supermarket data at the beginning. If the company already exists, we will refer to it ،else we will create it.
  the next code in company class.



```
def readsupermarkets(self):
    try:
        file1 = open(self.__name+"supermarkets.txt", 'r')
        for line in file1:
            x = line.strip().split(";")
            newob=supermarkets(x[1],x[0],x[2],x[3])
            newob.readsupermarketsItems() # will read the list of i
            self.__supermarketsComp[x[0]]=newob  #code will be in x
        file1.close()
    except IOError:
        open(self.__name+"supermarkets.txt", 'a+')
```

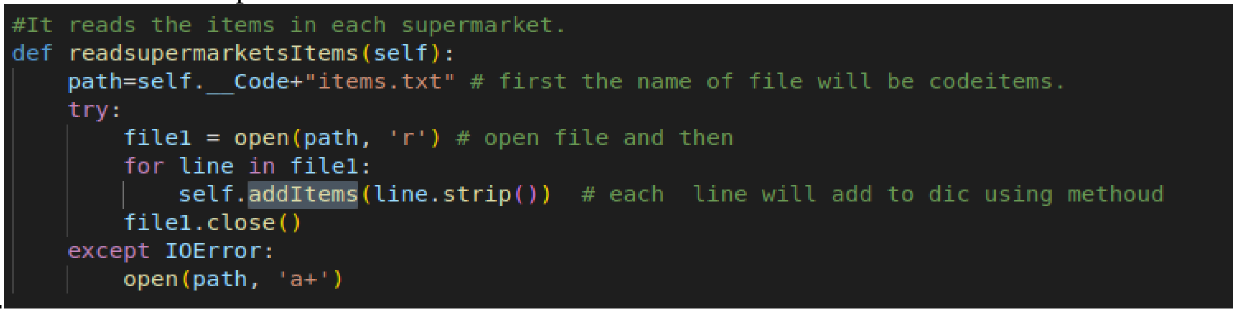
- We start reading the data from the file, and each line is a supermarket so we will create object supermarket then add to supermarket dictionary.
  using this code

```
newob=supermarkets(x[1],x[0],x[2],x[3])
self.__supermarketsComp[x[0]]=newob
```

- We start by taking the special items for each established supermarket through its own file. (The following code calls the method for that.)

```
newob.readsupermarketsItems()
```

the next code in supermarket class



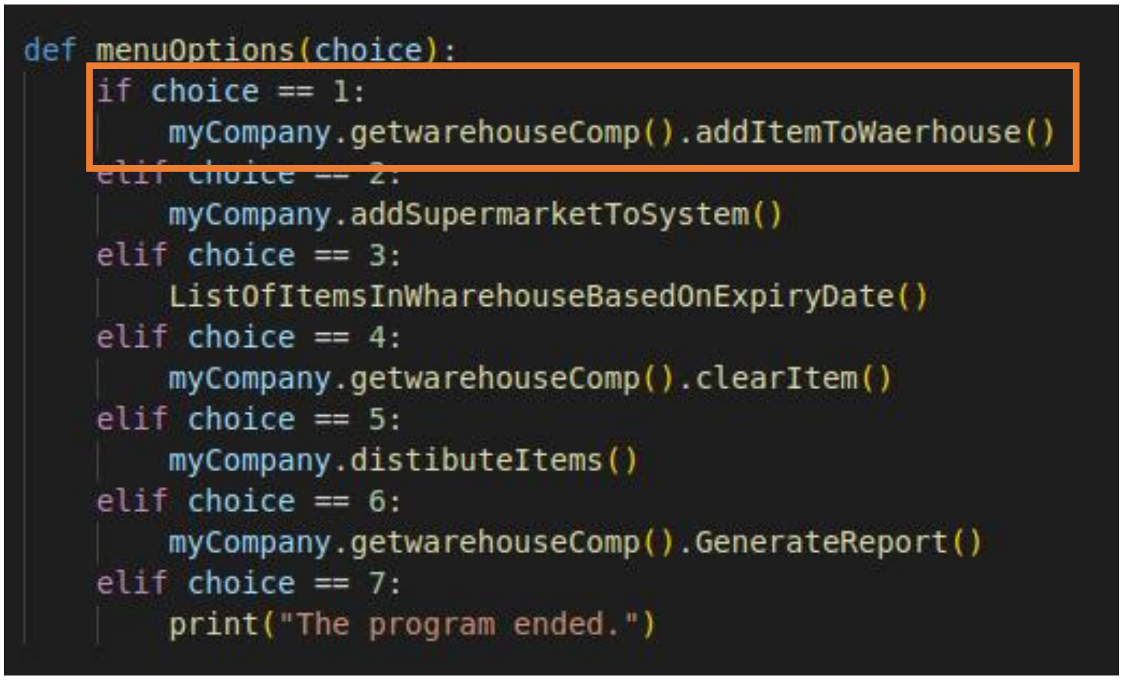
```python
#It reads the items in each supermarket.
def readsupermarketsItems(self):
    path=self.__Code+"items.txt" # first the name of file will be codeitems.
    try:
        file1 = open(path, 'r') # open file and then
        for line in file1:
            self.addItems(line.strip())  # each  line will add to dic using methoud
        file1.close()
    except IOError:
        open(path, 'a+')
```

.

- We start with one of each line and create object product using the ***addItem***() method which else add this product to this dictionary in supermarket.

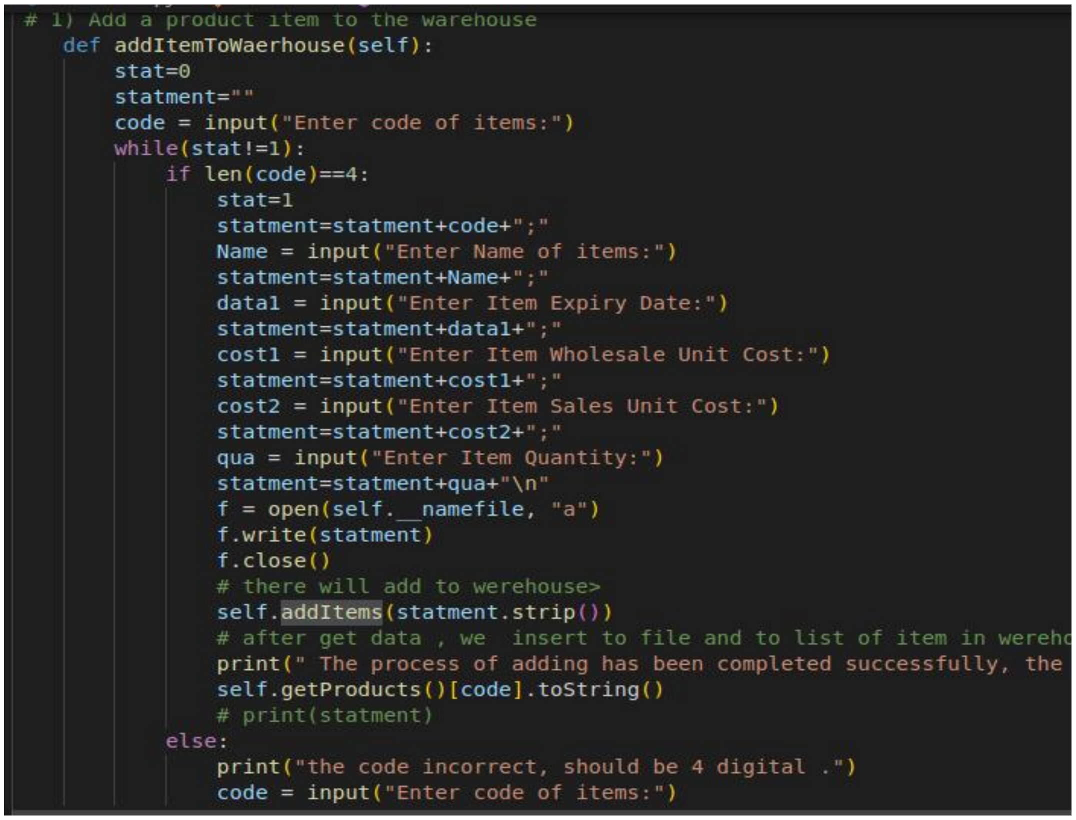
Add product to warehouse

When choosing the first option on the main menu; which is add product to warehouse, first we call a method in the company class called ***getwarehouseComp()*** that will return an object of the warehouse, in other words, return the products stored in the warehouse.



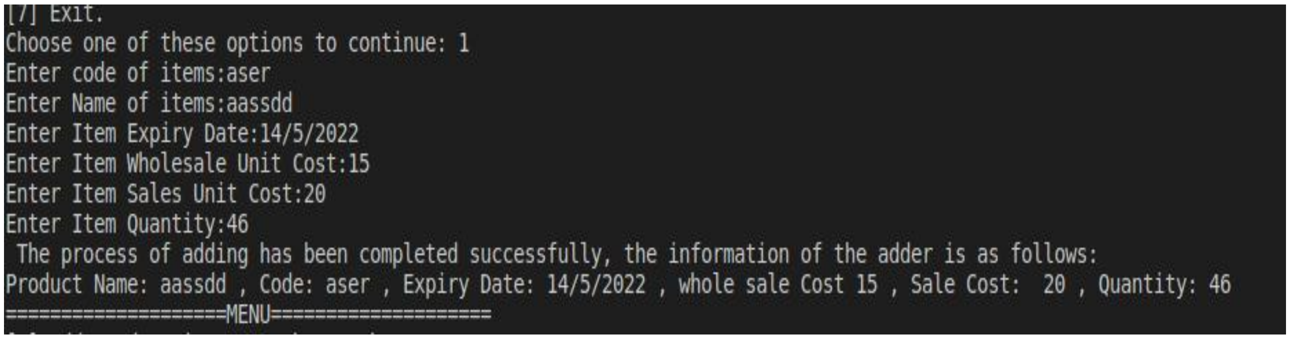
```python
def menuOptions(choice):
    if choice == 1:
        myCompany.getwarehouseComp().addItemToWaerhouse()
    elif choice == 2:
        myCompany.addSupermarketToSystem()
    elif choice == 3:
        ListOfItemsInWharehouseBasedOnExpiryDate()
    elif choice == 4:
        myCompany.getwarehouseComp().clearItem()
    elif choice == 5:
        myCompany.distibuteItems()
    elif choice == 6:
        myCompany.getwarehouseComp().GenerateReport()
    elif choice == 7:
        print("The program ended.")
```

Then we will call a method called ***addItemToWaerhouse()*** which is a method created in the warehouse class. This method asks the user to enter the product's information such as name, code, etc. After that, we append this product to the warehouse file and call a method called ***addItems()*** to add it to the warehouse.



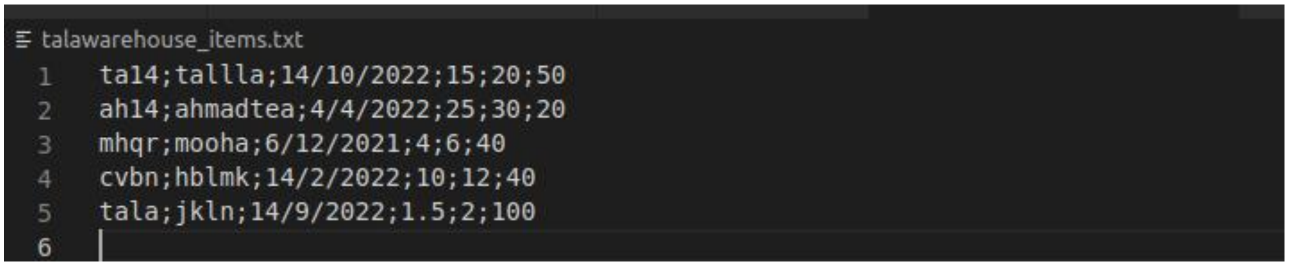
```python
# 1) Add a product item to the warehouse
    def addItemToWaerhouse(self):
        stat=0
        statment=""
        code = input("Enter code of items:")
        while(stat!=1):
            if len(code)==4:
                stat=1
                statment=statment+code+";"
                Name = input("Enter Name of items:")
                statment=statment+Name+";"
                data1 = input("Enter Item Expiry Date:")
                statment=statment+data1+";"
                cost1 = input("Enter Item Wholesale Unit Cost:")
                statment=statment+cost1+";"
                cost2 = input("Enter Item Sales Unit Cost:")
                statment=statment+cost2+";"
                qua = input("Enter Item Quantity:")
                statment=statment+qua+"\n"
                f = open(self.__namefile, "a")
                f.write(statment)
                f.close()
                # there will add to werehouse>
                self.addItems(statment.strip())
                # after get data , we  insert to file and to list of item in wereh
                print(" The process of adding has been completed successfully, the
                self.getProducts()[code].toString()
                # print(statment)
            else:
                print("the code incorrect, should be 4 digital .")
                code = input("Enter code of items:")
```

User Interface:



```
[7] Exit.
Choose one of these options to continue: 1
Enter code of items:aser
Enter Name of items:aassdd
Enter Item Expiry Date:14/5/2022
Enter Item Wholesale Unit Cost:15
Enter Item Sales Unit Cost:20
Enter Item Quantity:46
 The process of adding has been completed successfully, the information of the adder is as follows:
Product Name: aassdd , Code: aser , Expiry Date: 14/5/2022 , whole sale Cost 15 , Sale Cost:  20 , Quantity: 46
===================MENU===================
```

File:

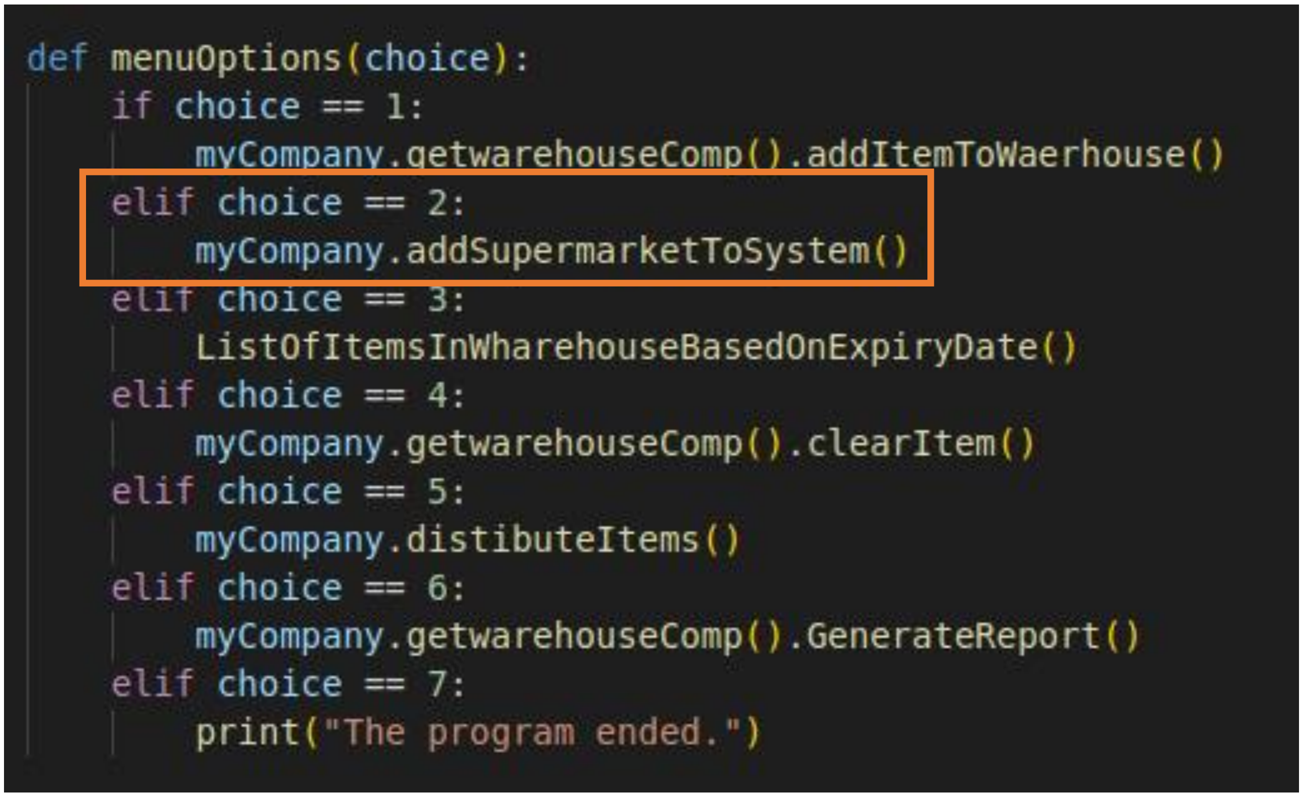


talawarehouse_items.txt

```
tal4;tallla;14/10/2022;15;20;50
ah14;ahmadtea;4/4/2022;25;30;20
mhqr;mooha;6/12/2021;4;6;40
cvbn;hblmk;14/2/2022;10;12;40
tala;jkln;14/9/2022;1.5;2;100

```

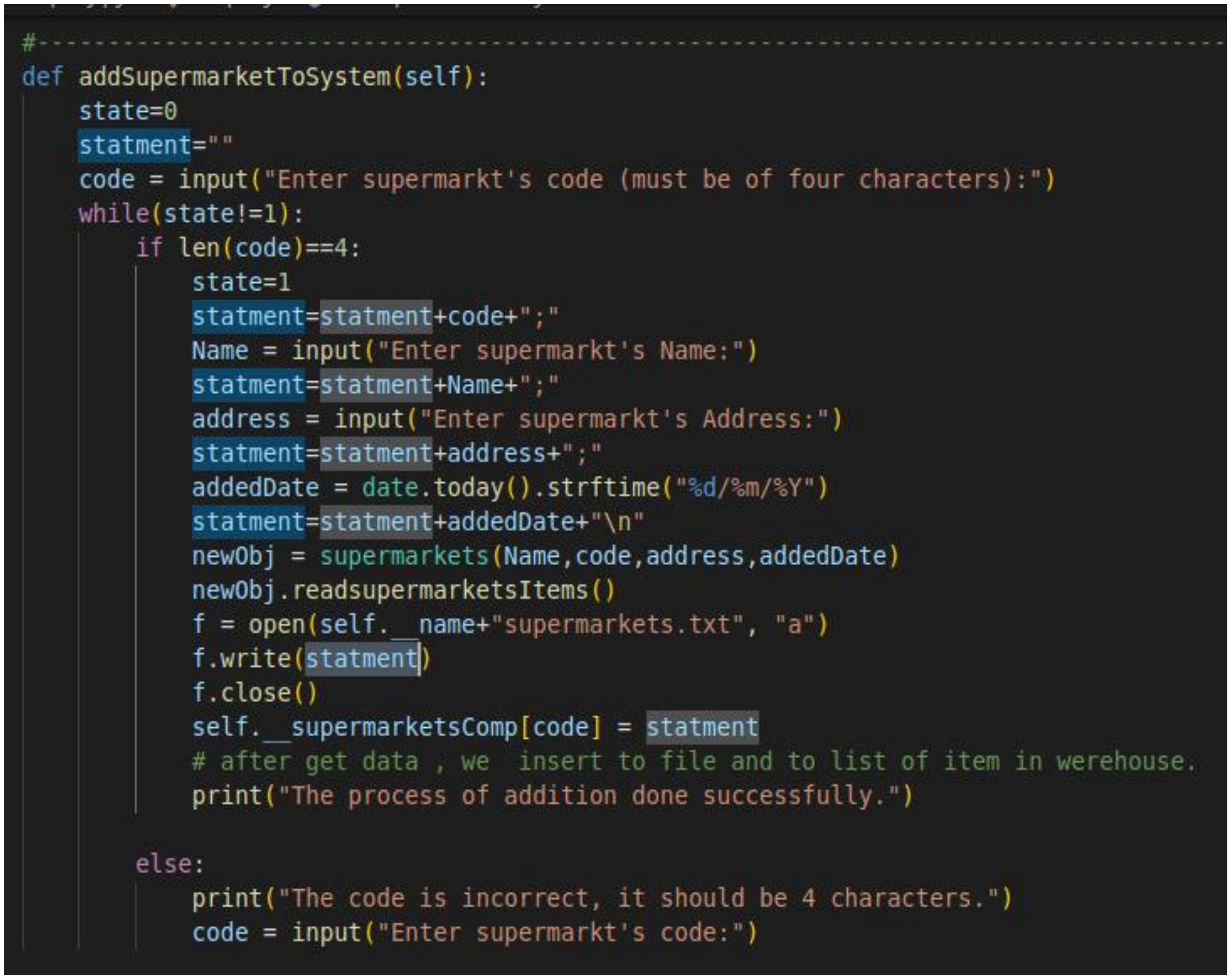
Add new Supermarket to the System

It is the second option in the menu. First, a method called ***addSupermarketToSystem()*** from the company class is called.



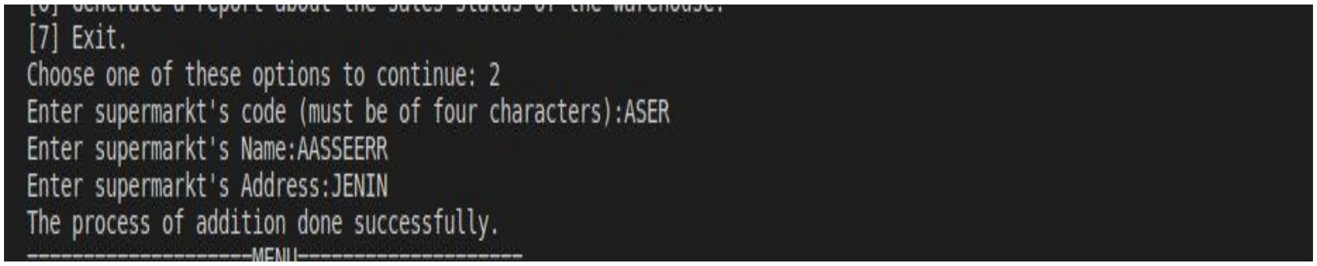
```
def menuOptions(choice):
    if choice == 1:
        myCompany.getwarehouseComp().addItemToWaerhouse()
    elif choice == 2:
        myCompany.addSupermarketToSystem()
    elif choice == 3:
        ListOfItemsInWharehouseBasedOnExpiryDate()
    elif choice == 4:
        myCompany.getwarehouseComp().clearItem()
    elif choice == 5:
        myCompany.distibuteItems()
    elif choice == 6:
        myCompany.getwarehouseComp().GenerateReport()
    elif choice == 7:
        print("The program ended.")
```

This method askes the user to enter the supermarket data which are supermarket code, name and address. But to insert the added date, we use a method called ***date().today().strftime("%d/%m/%Y")*** from the datetime class the we imported before. We create an instance from the supermarket class to add it to the dictionary of supermarkets in the warehouse. After creating the supermarket, we create an empty file for it's products contained using the ***readsupermarketsItems()*** that we illustrated in the previous sections. Finally, we add the supermarket to the supermarket file for the company.



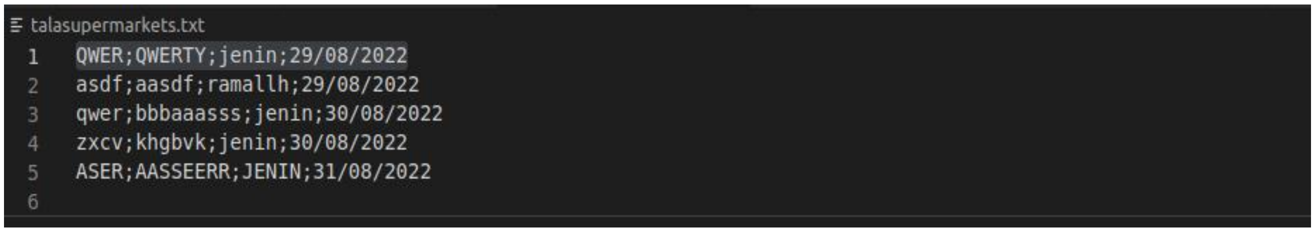
```python
#--------------------------------------------------------------------------------------
def addSupermarketToSystem(self):
    state=0
    statment=""
    code = input("Enter supermarkt's code (must be of four characters):")
    while(state!=1):
        if len(code)==4:
            state=1
            statment=statment+code+";"
            Name = input("Enter supermarkt's Name:")
            statment=statment+Name+";"
            address = input("Enter supermarkt's Address:")
            statment=statment+address+";"
            addedDate = date.today().strftime("%d/%m/%Y")
            statment=statment+addedDate+"\n"
            newObj = supermarkets(Name,code,address,addedDate)
            newObj.readsupermarketsItems()
            f = open(self.__name+"supermarkets.txt", "a")
            f.write(statment)
            f.close()
            self.__supermarketsComp[code] = statment
            # after get data , we  insert to file and to list of item in werehouse.
            print("The process of addition done successfully.")

        else:
            print("The code is incorrect, it should be 4 characters.")
            code = input("Enter supermarkt's code:")
```

User Interface:



```
[7] Exit.
Choose one of these options to continue: 2
Enter supermarkt's code (must be of four characters):ASER
Enter supermarkt's Name:AASSEERR
Enter supermarkt's Address:JENIN
The process of addition done successfully.
-------------------MENU-------------------
```

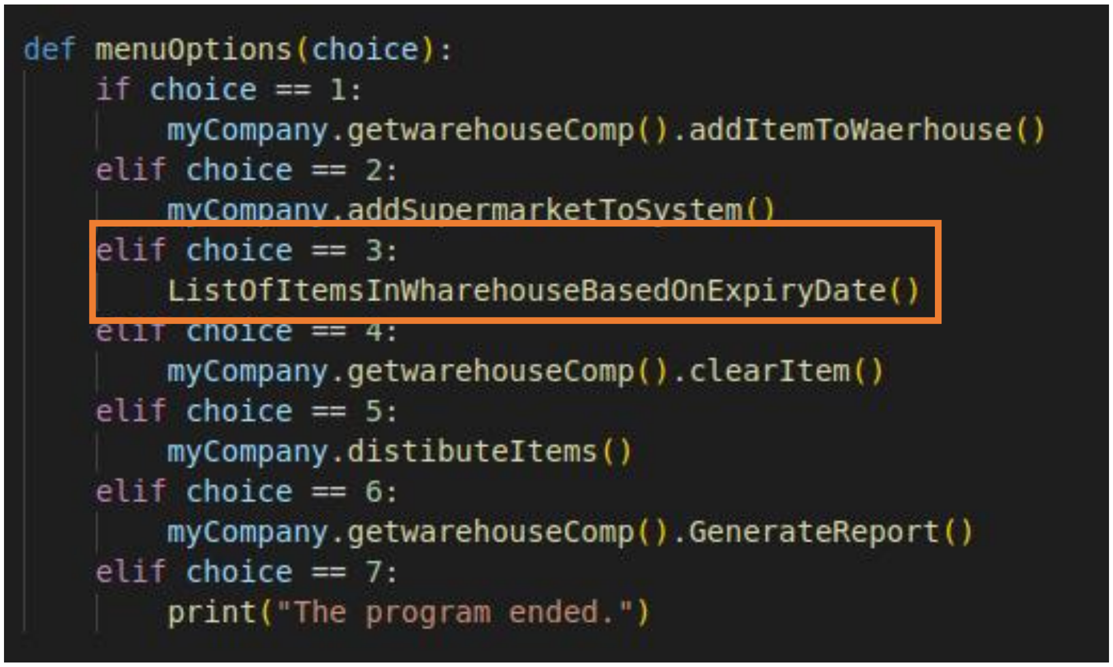
File:



```
talasupermarkets.txt
QWER;QWERTY;jenin;29/08/2022
asdf;aasdf;ramallh;29/08/2022
qwer;bbbaaasss;jenin;30/08/2022
zxcv;khgbvk;jenin;30/08/2022
ASER;AASSEERR;JENIN;31/08/2022

```

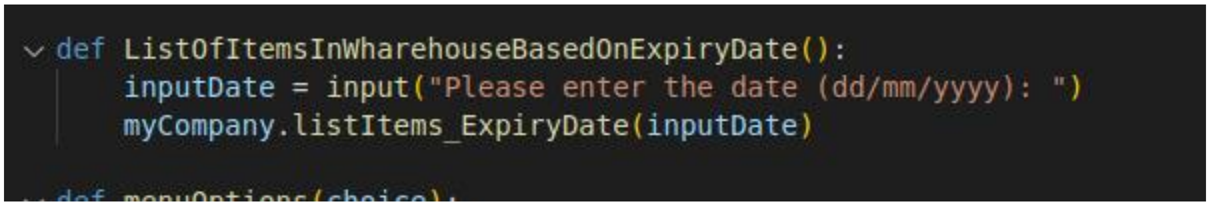
List of Items in the warehouse based on Expiry Date

A function called ***listOfItmesInTheWarehouseBasedOnExpiryDate()*** will be called from the main class when clicking on the third option.



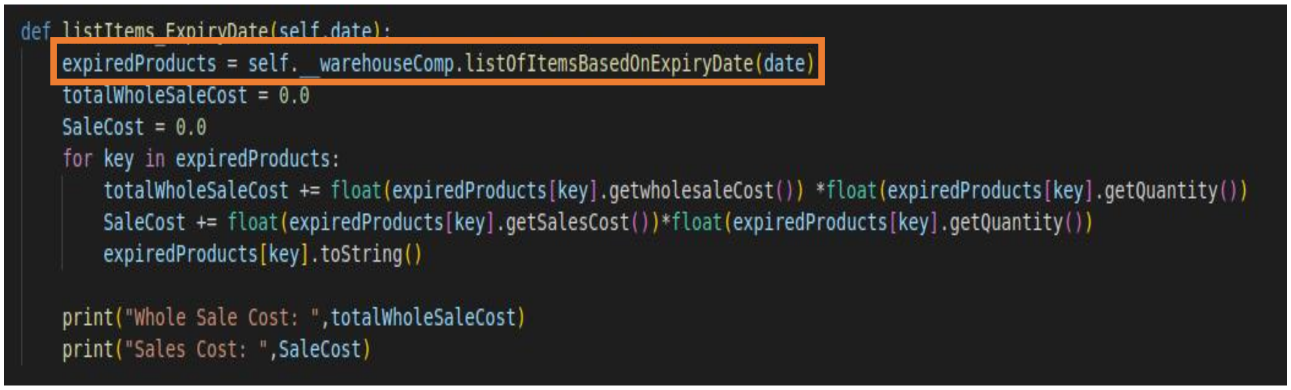
```python
def menuOptions(choice):
    if choice == 1:
        myCompany.getwarehouseComp().addItemToWaerhouse()
    elif choice == 2:
        myCompany.addSupermarketToSystem()
    elif choice == 3:
        ListOfItemsInWharehouseBasedOnExpiryDate()
    elif choice == 4:
        myCompany.getwarehouseComp().clearItem()
    elif choice == 5:
        myCompany.distibuteItems()
    elif choice == 6:
        myCompany.getwarehouseComp().GenerateReport()
    elif choice == 7:
        print("The program ended.")
```

This function askes the user enters a date. Then we call a method called ***listItems_ExpiryDate(inputDate)*** in the company class.



```python
def ListOfItemsInWharehouseBasedOnExpiryDate():
    inputDate = input("Please enter the date (dd/mm/yyyy): ")
    myCompany.listItems_ExpiryDate(inputDate)

def menuOptions(choice):
```

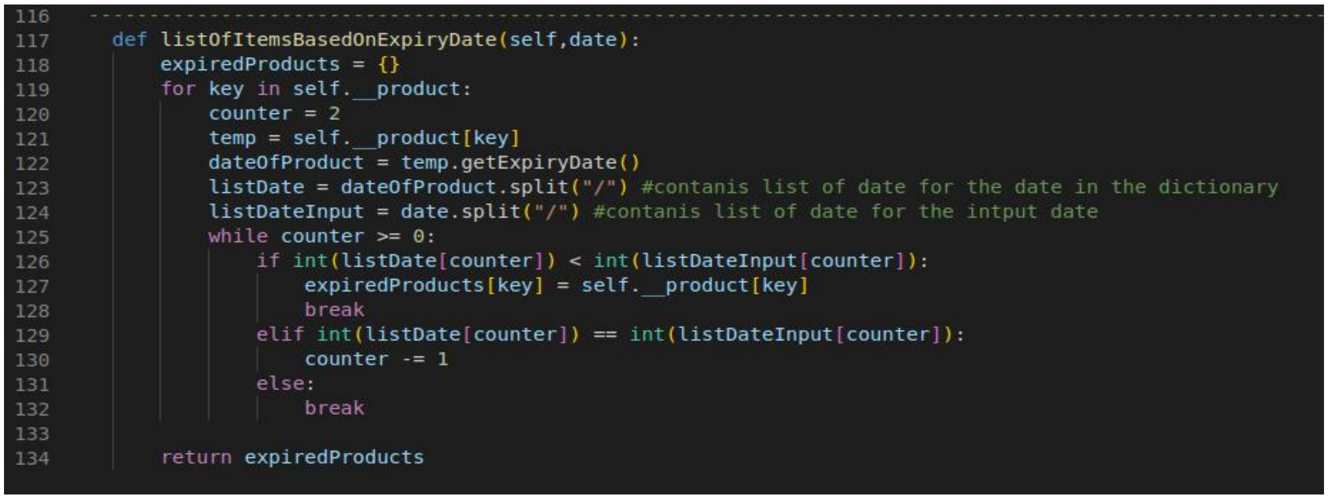
This method calls another method in the warehouse class called ***listOfItemsBasedOnExpiryDate(inputDate)***.



```python
def listItems_ExpiryDate(self,date):
    expiredProducts = self.__warehouseComp.listOfItemsBasedOnExpiryDate(date)
    totalWholeSaleCost = 0.0
    SaleCost = 0.0
    for key in expiredProducts:
        totalWholeSaleCost += float(expiredProducts[key].getwholesaleCost()) *float(expiredProducts[key].getQuantity())
        SaleCost += float(expiredProducts[key].getSalesCost())*float(expiredProducts[key].getQuantity())
        expiredProducts[key].toString()

    print("Whole Sale Cost: ",totalWholeSaleCost)
    print("Sales Cost: ",SaleCost)
```

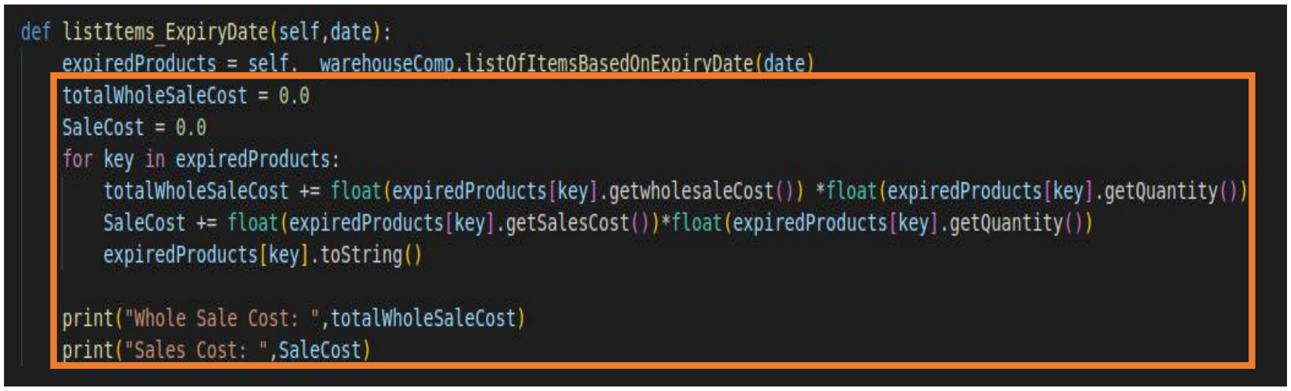
This method returns a dictionary that contains only the products which have an expiry date before the date entered from user.

First, the method will create an empty dictionary called ***expiredProducts.*** Using loop, the program will traverse each product in the warehouse, then it will get the date of the product and split numbers depending on (/) and store the result in a list called ***listDate***. Using inner while loop, we start comparing between the years, months and days to get the product which has an expiry date before the entered date.



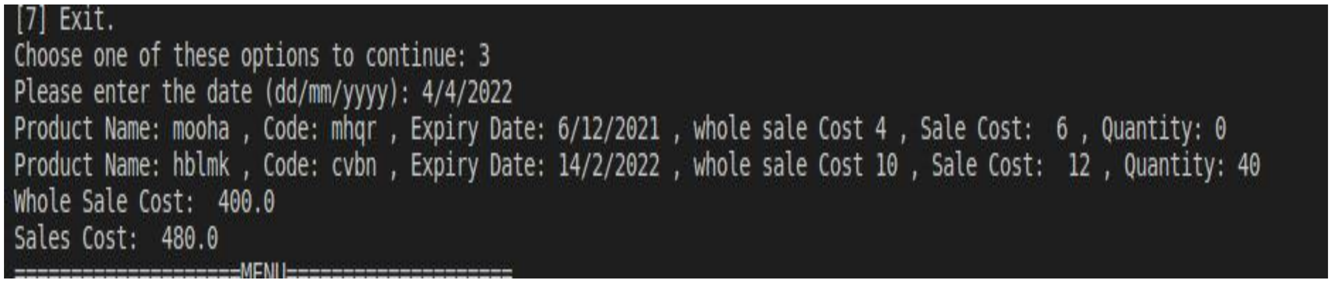
```python
    def listOfItemsBasedOnExpiryDate(self,date):
        expiredProducts = {}
        for key in self.__product:
            counter = 2
            temp = self.__product[key]
            dateOfProduct = temp.getExpiryDate()
            listDate = dateOfProduct.split("/") #contanis list of date for the date in the dictionary
            listDateInput = date.split("/") #contanis list of date for the intput date
            while counter >= 0:
                if int(listDate[counter]) < int(listDateInput[counter]):
                    expiredProducts[key] = self.__product[key]
                    break
                elif int(listDate[counter]) == int(listDateInput[counter]):
                    counter -= 1
                else:
                    break

        return expiredProducts
```

After returning the dictionary of expired products, the software will display the total wholesale cost of these items and the total sales cost of these items.



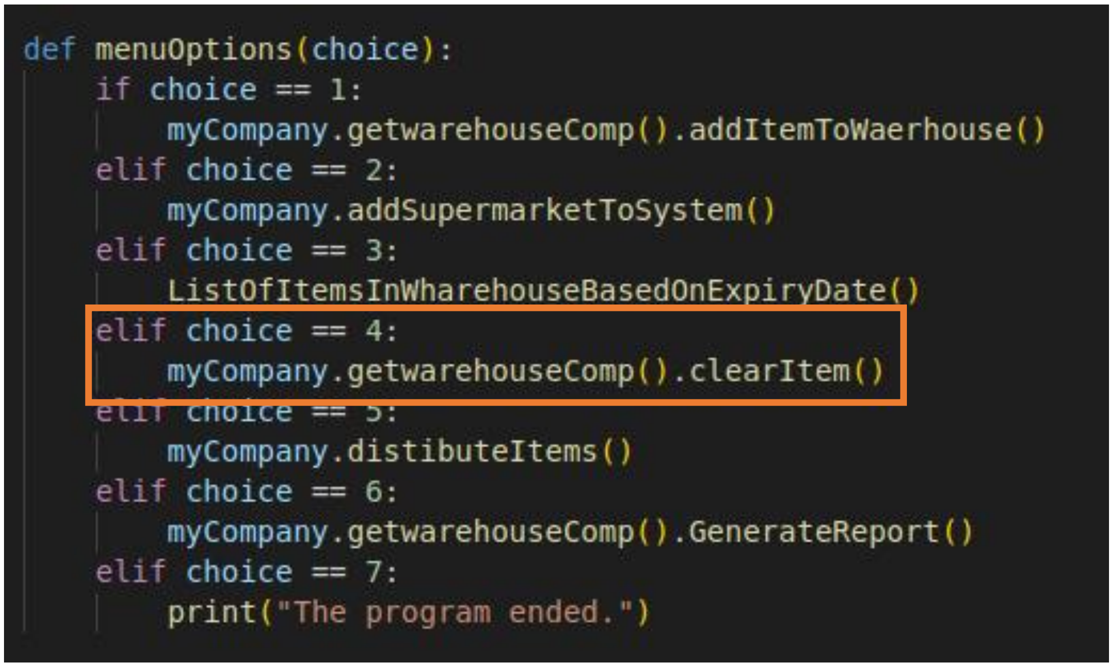
```python
def listItems_ExpiryDate(self,date):
    expiredProducts = self.__warehouseComp.listOfItemsBasedOnExpiryDate(date)
    totalWholeSaleCost = 0.0
    SaleCost = 0.0
    for key in expiredProducts:
        totalWholeSaleCost += float(expiredProducts[key].getwholesaleCost()) *float(expiredProducts[key].getQuantity())
        SaleCost += float(expiredProducts[key].getSalesCost())*float(expiredProducts[key].getQuantity())
        expiredProducts[key].toString()

    print("Whole Sale Cost: ",totalWholeSaleCost)
    print("Sales Cost: ",SaleCost)
```

User Interface:



```
[7] Exit.
Choose one of these options to continue: 3
Please enter the date (dd/mm/yyyy): 4/4/2022
Product Name: mooha , Code: mhqr , Expiry Date: 6/12/2021 , whole sale Cost 4 , Sale Cost:  6 , Quantity: 0
Product Name: hblmk , Code: cvbn , Expiry Date: 14/2/2022 , whole sale Cost 10 , Sale Cost:  12 , Quantity: 40
Whole Sale Cost:  400.0
Sales Cost:  480.0
===================MENU===================
```

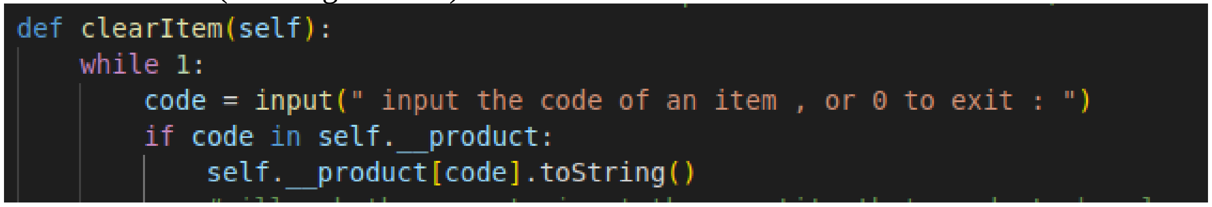
Clear an item from the warehouse:



```
def menuOptions(choice):
    if choice == 1:
        myCompany.getwarehouseComp().addItemToWaerhouse()
    elif choice == 2:
        myCompany.addSupermarketToSystem()
    elif choice == 3:
        ListOfItemsInWharehouseBasedOnExpiryDate()
    elif choice == 4:
        myCompany.getwarehouseComp().clearItem()
    elif choice == 5:
        myCompany.distibuteItems()
    elif choice == 6:
        myCompany.getwarehouseComp().GenerateReport()
    elif choice == 7:
        print("The program ended.")
```

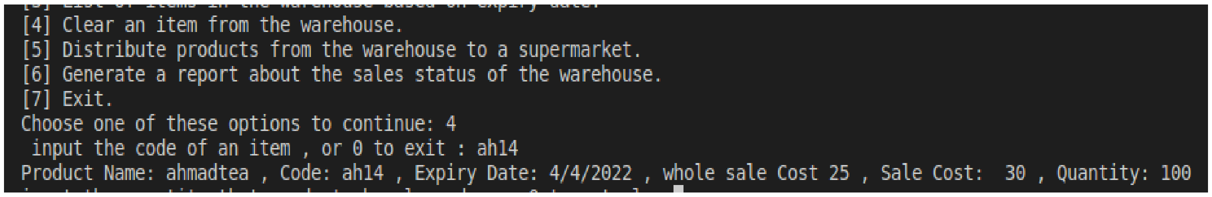
The process of removing items from the warehouse takes place in several stages, since this operation is specific to the warehouse, the code for this operation will be in the warehouse code.

- The software should ask the user to input the code of an item, then we look for the presence of this code in the warehouse. The user can perform this operation sequentially until he enters 0 (meaning finished)



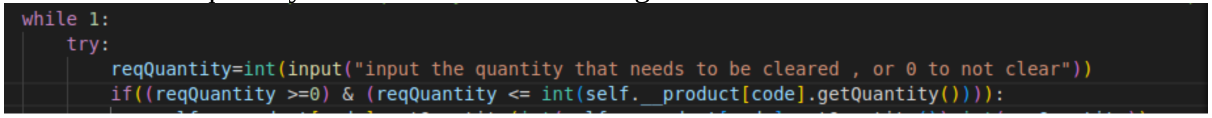
```
def clearItem(self):
    while 1:
        code = input(" input the code of an item , or 0 to exit : ")
        if code in self.__product:
            self.__product[code].toString()
```

user Interface:



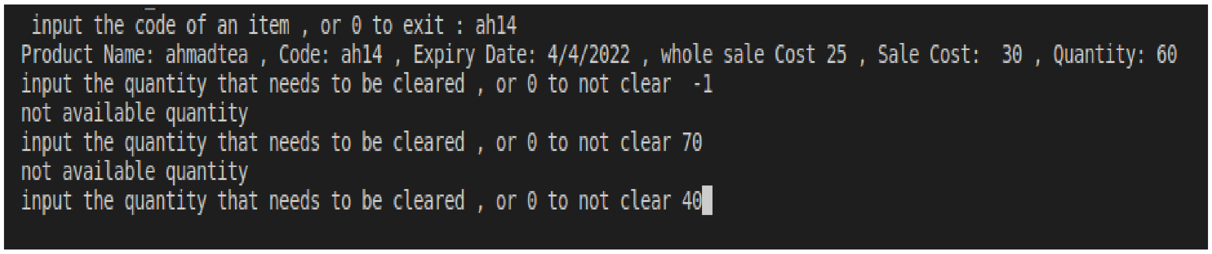
```
[4] Clear an item from the warehouse.
[5] Distribute products from the warehouse to a supermarket.
[6] Generate a report about the sales status of the warehouse.
[7] Exit.
Choose one of these options to continue: 4
 input the code of an item , or 0 to exit : ah14
Product Name: ahmadtea , Code: ah14 , Expiry Date: 4/4/2022 , whole sale Cost 25 , Sale Cost:  30 , Quantity: 100
```

- After that, we make the user enter the quantity he wants to delete, provided that the value is less than the quantity in the store and not be negative.



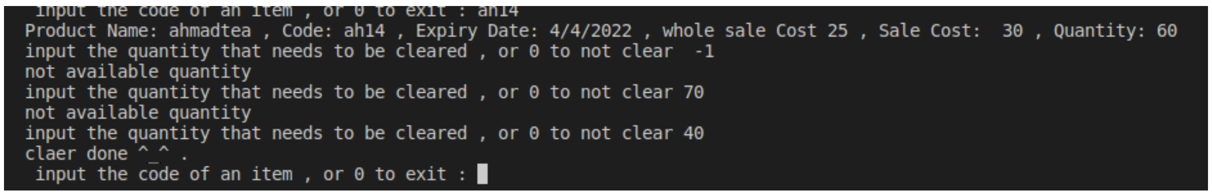
```
while 1:
    try:
        reqQuantity=int(input("input the quantity that needs to be cleared , or 0 to not clear"))
        if((reqQuantity >=0) & (reqQuantity <= int(self.__product[code].getQuantity()))):
```

User Interface:



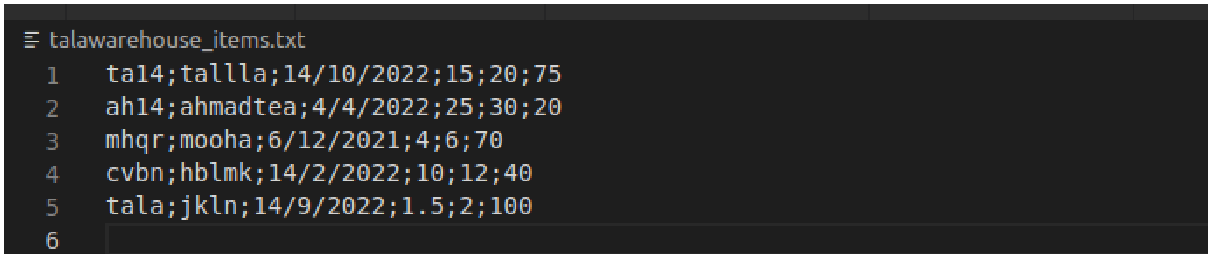
```
 input the code of an item , or 0 to exit : ah14
Product Name: ahmadtea , Code: ah14 , Expiry Date: 4/4/2022 , whole sale Cost 25 , Sale Cost:  30 , Quantity: 60
input the quantity that needs to be cleared , or 0 to not clear  -1
not available quantity
input the quantity that needs to be cleared , or 0 to not clear 70
not available quantity
input the quantity that needs to be cleared , or 0 to not clear 40
```

- If the value is allowed, we work to subtract the quantity entered by the user from the quantity inside the warehouse, and after completing that, we work to modify the data in the file.



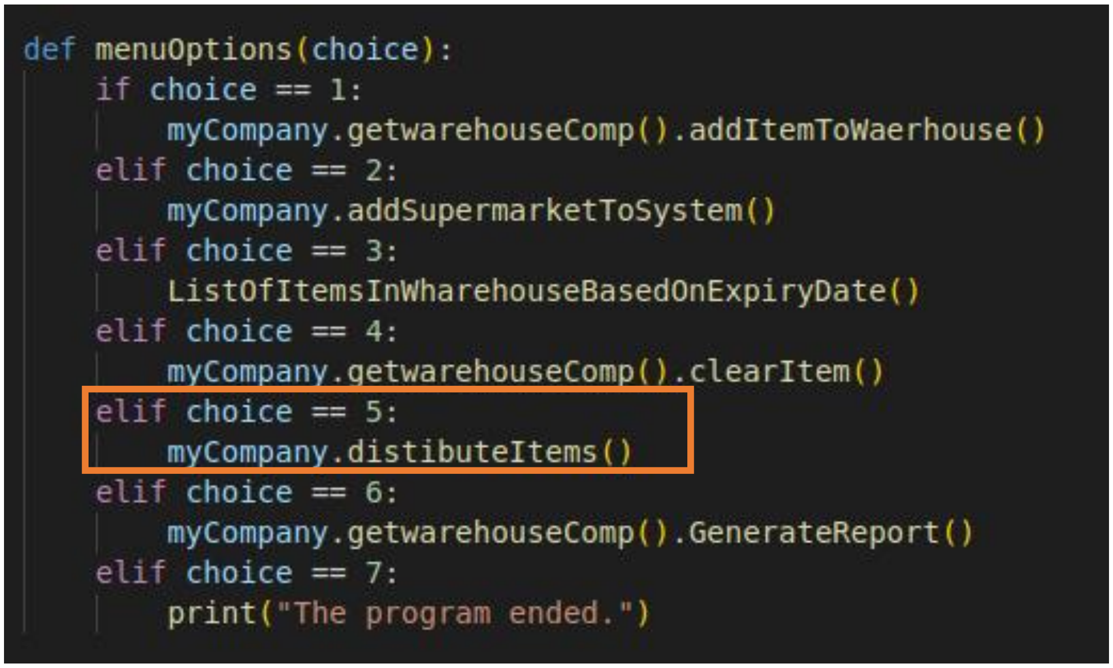
```
 input the code of an item , or 0 to exit : ah14
Product Name: ahmadtea , Code: ah14 , Expiry Date: 4/4/2022 , whole sale Cost 25 , Sale Cost:  30 , Quantity: 60
input the quantity that needs to be cleared , or 0 to not clear  -1
not available quantity
input the quantity that needs to be cleared , or 0 to not clear 70
not available quantity
input the quantity that needs to be cleared , or 0 to not clear 40
claer done ^_^ .
 input the code of an item , or 0 to exit :
```

If we look for this code inside the file:



```
ta14;tallla;14/10/2022;15;20;75
ah14;ahmadtea;4/4/2022;25;30;20
mhqr;mooha;6/12/2021;4;6;70
cvbn;hblmk;14/2/2022;10;12;40
tala;jkln;14/9/2022;1.5;2;100
```

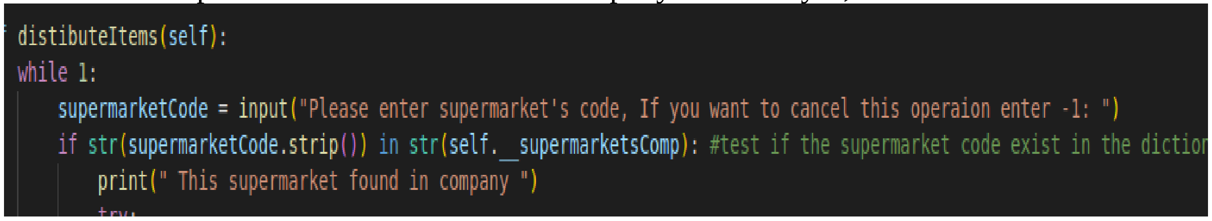
Distribute products from the Warehouse to a Supermarket:



```python
def menuOptions(choice):
    if choice == 1:
        myCompany.getwarehouseComp().addItemToWaerhouse()
    elif choice == 2:
        myCompany.addSupermarketToSystem()
    elif choice == 3:
        ListOfItemsInWharehouseBasedOnExpiryDate()
    elif choice == 4:
        myCompany.getwarehouseComp().clearItem()
    elif choice == 5:
        myCompany.distibuteItems()
    elif choice == 6:
        myCompany.getwarehouseComp().GenerateReport()
    elif choice == 7:
        print("The program ended.")
```

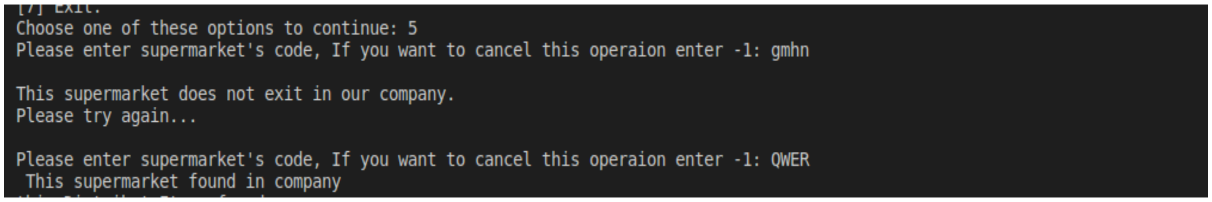
It is considered one of the most important processes here, through which items are distributed from stores to supermarkets, with a special file containing the required items (DistributeItems_<SupermarketCode>.txt).
All code will be in **company class**.

- The software should ask for the code of the supermarket to distribute products to. and check if this supermarket was found in the company or not. If yes, continuous work.



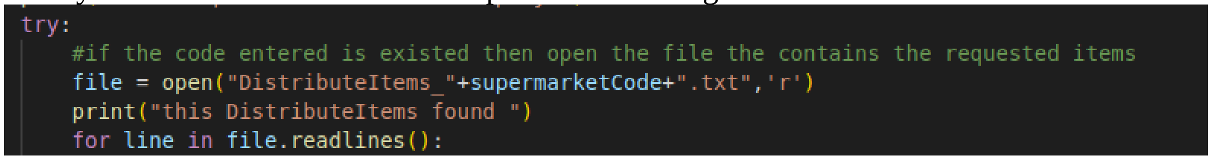
```
distibuteItems(self):
while 1:
    supermarketCode = input("Please enter supermarket's code, If you want to cancel this operaion enter -1: ")
    if str(supermarketCode.strip()) in str(self.__supermarketsComp): #test if the supermarket code exist in the diction
        print(" This supermarket found in company ")
        try:
```

use Interface:

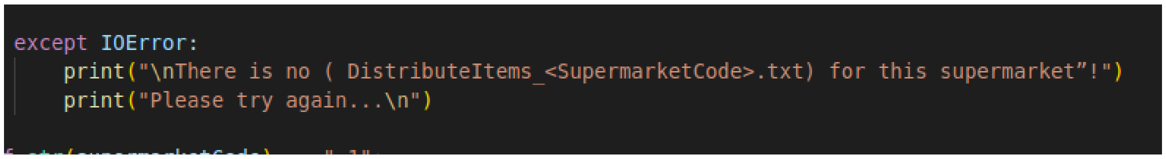


```
[7] Exit.
Choose one of these options to continue: 5
Please enter supermarket's code, If you want to cancel this operaion enter -1: gmhn

This supermarket does not exit in our company.
Please try again...

Please enter supermarket's code, If you want to cancel this operaion enter -1: QWER
 This supermarket found in company
```

- Then, it will load a text file ("DistributeItems_<SupermarketCode>.txt") that includes a list of item codes and quantity for each item requested by that supermarket, and start read line by line. if this file not found will print error message for user.

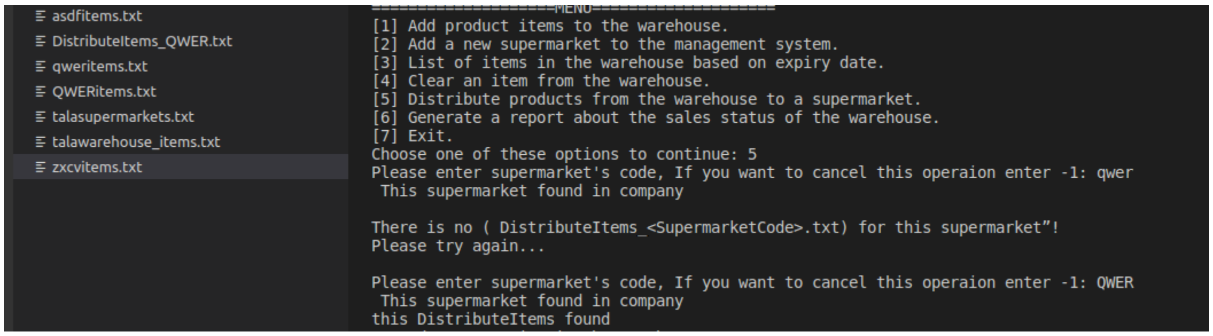


```
try:
    #if the code entered is existed then open the file the contains the requested items
    file = open("DistributeItems_"+supermarketCode+".txt",'r')
    print("this DistributeItems found ")
    for line in file.readlines():
```



```
except IOError:
    print("\nThere is no ( DistributeItems_<SupermarketCode>.txt) for this supermarket"!")
    print("Please try again...\n")
```

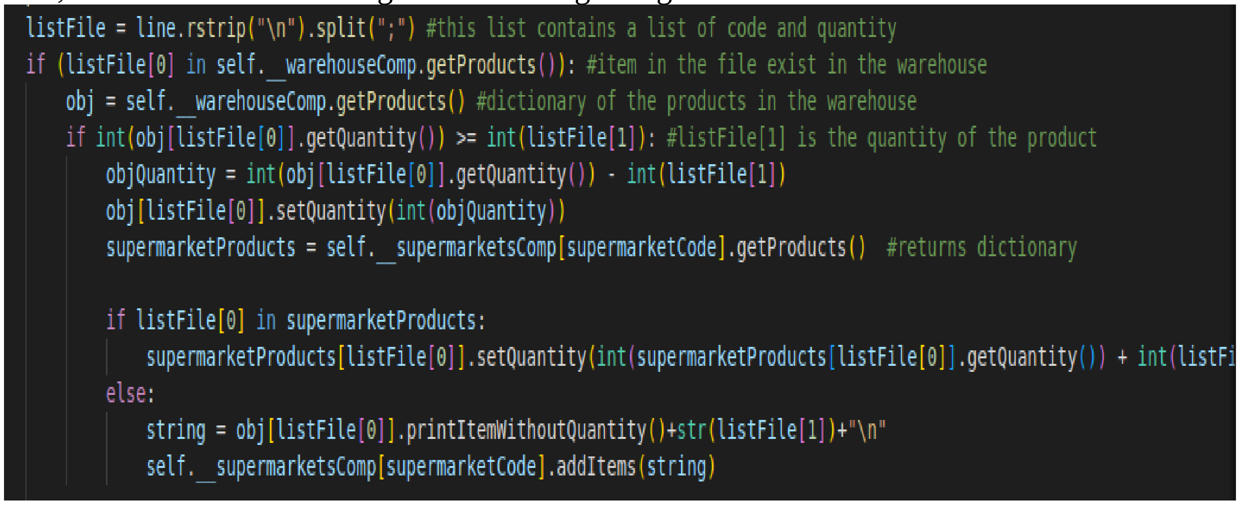
User Interface:



```
asdfitems.txt
DistributeItems_QWER.txt
qweritems.txt
QWERitems.txt
talasupermarkets.txt
talawarehouse_items.txt
zxcvitems.txt
```

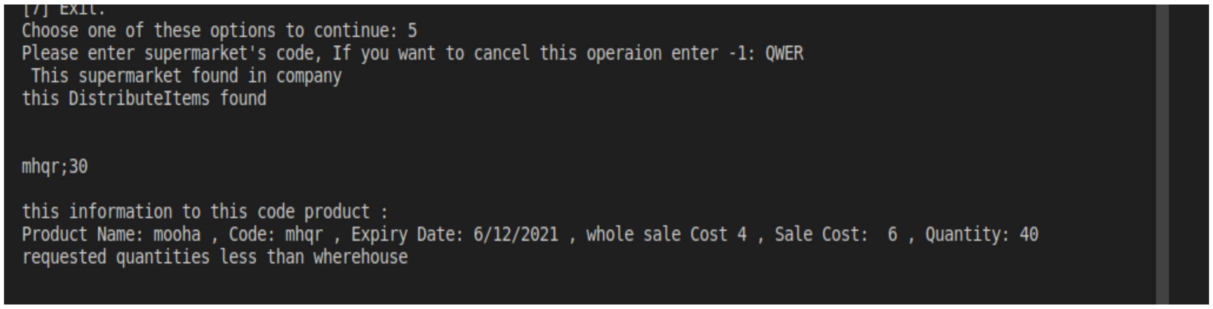
```
=====================MENU=====================
[1] Add product items to the warehouse.
[2] Add a new supermarket to the management system.
[3] List of items in the warehouse based on expiry date.
[4] Clear an item from the warehouse.
[5] Distribute products from the warehouse to a supermarket.
[6] Generate a report about the sales status of the warehouse.
[7] Exit.
Choose one of these options to continue: 5
Please enter supermarket's code, If you want to cancel this operaion enter -1: qwer
 This supermarket found in company

There is no ( DistributeItems_<SupermarketCode>.txt) for this supermarket"!
Please try again...

Please enter supermarket's code, If you want to cancel this operaion enter -1: QWER
 This supermarket found in company
this DistributeItems found
```

- The software, will then check the warehouse and distribute the requested quantities of each item. if the requested quantities less than warehouse will add these items to the list of items available at the supermarket and remove them from the warehouse.
  If the item is already in the store, we will clean the previous quantity, but if the item is new, we will work on creating it from the beginning.



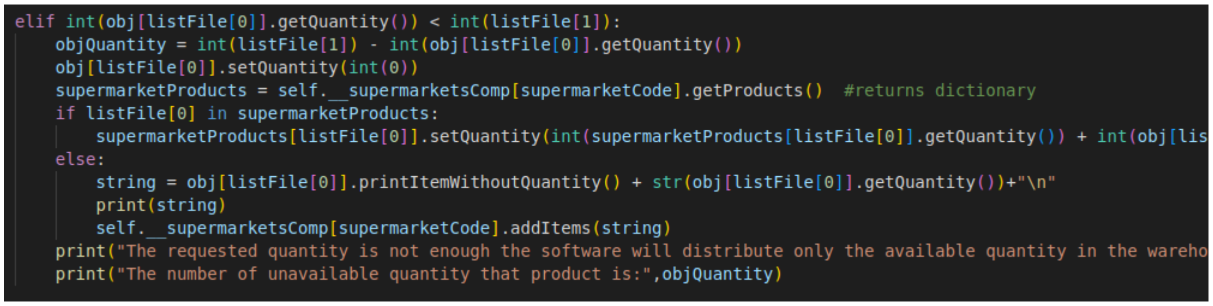
```
listFile = line.rstrip("\n").split(";") #this list contains a list of code and quantity
if (listFile[0] in self._warehouseComp.getProducts()): #item in the file exist in the warehouse
    obj = self._warehouseComp.getProducts() #dictionary of the products in the warehouse
    if int(obj[listFile[0]].getQuantity()) >= int(listFile[1]): #listFile[1] is the quantity of the product
        objQuantity = int(obj[listFile[0]].getQuantity()) - int(listFile[1])
        obj[listFile[0]].setQuantity(int(objQuantity))
        supermarketProducts = self._supermarketsComp[supermarketCode].getProducts()  #returns dictionary

        if listFile[0] in supermarketProducts:
            supermarketProducts[listFile[0]].setQuantity(int(supermarketProducts[listFile[0]].getQuantity()) + int(listFi
        else:
            string = obj[listFile[0]].printItemWithoutQuantity()+str(listFile[1])+"\n"
            self._supermarketsComp[supermarketCode].addItems(string)
```

user Interface:



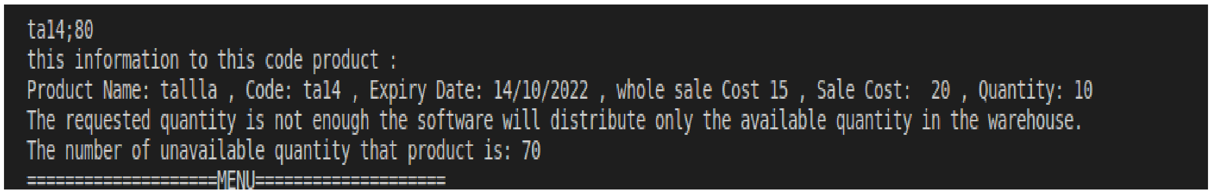
```
[7] Exit.
Choose one of these options to continue: 5
Please enter supermarket's code, If you want to cancel this operaion enter -1: QWER
 This supermarket found in company
this DistributeItems found


mhqr;30

this information to this code product :
Product Name: mooha , Code: mhqr , Expiry Date: 6/12/2021 , whole sale Cost 4 , Sale Cost:  6 , Quantity: 40
requested quantities less than wherehouse
```

- If the requested quantity of any item is not enough, the software will distribute only the available quantity. It will also print on the screen, a message about the item and the number of requested but not distributed quantities of this item.
  code:



```
elif int(obj[listFile[0]].getQuantity()) < int(listFile[1]):
    objQuantity = int(listFile[1]) - int(obj[listFile[0]].getQuantity())
    obj[listFile[0]].setQuantity(int(0))
    supermarketProducts = self.__supermarketsComp[supermarketCode].getProducts()  #returns dictionary
    if listFile[0] in supermarketProducts:
        supermarketProducts[listFile[0]].setQuantity(int(supermarketProducts[listFile[0]].getQuantity()) + int(obj[lis
    else:
        string = obj[listFile[0]].printItemWithoutQuantity() + str(obj[listFile[0]].getQuantity())+"\n"
        print(string)
        self.__supermarketsComp[supermarketCode].addItems(string)
    print("The requested quantity is not enough the software will distribute only the available quantity in the wareho
    print("The number of unavailable quantity that product is:",objQuantity)
```

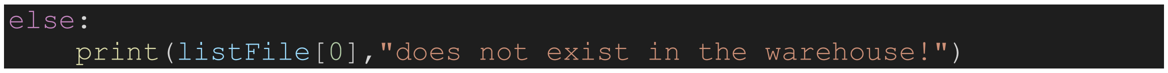
user Interface:



```
ta14;80
this information to this code product :
Product Name: tallla , Code: ta14 , Expiry Date: 14/10/2022 , whole sale Cost 15 , Sale Cost:  20 , Quantity: 10
The requested quantity is not enough the software will distribute only the available quantity in the warehouse.
The number of unavailable quantity that product is: 70
===================MENU===================
```

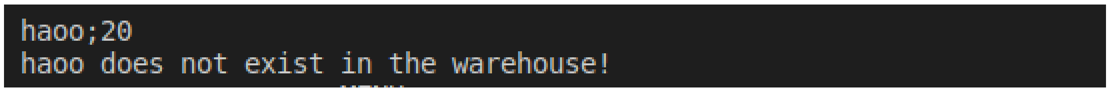
- If an item is not available at the warehouse or the code is wrong, the software should print a message on the screen with the code of this item and the requested amount on screen.

code:



```
else:
    print(listFile[0],"does not exist in the warehouse!")
```

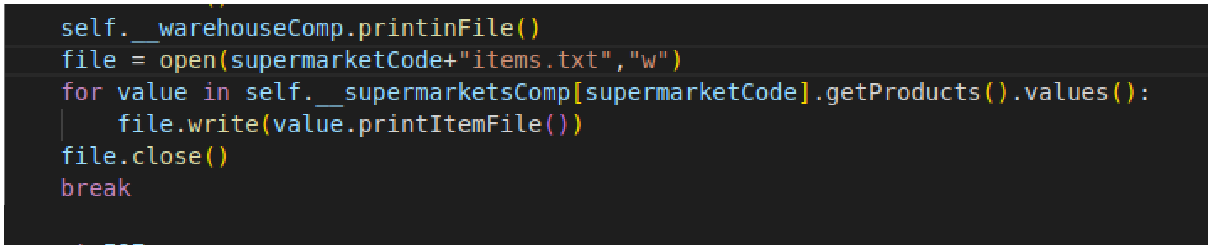
user Interface:



```
haoo;20
haoo does not exist in the warehouse!
```

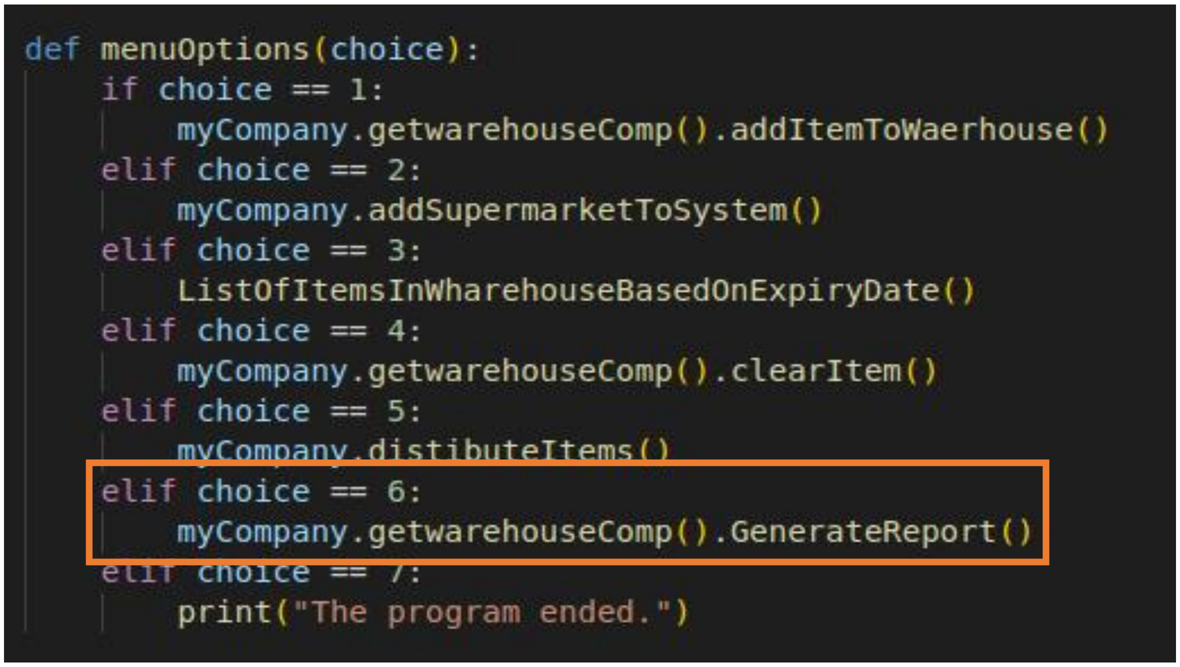
- After basically completing the basic process and setting the appropriate values in the item supermarket as well as the warehouse, we should update the files we have.

code:



```
    self.__warehouseComp.printinFile()
    file = open(supermarketCode+"items.txt","w")
    for value in self.__supermarketsComp[supermarketCode].getProducts().values():
        file.write(value.printItemFile())
    file.close()
    break
```

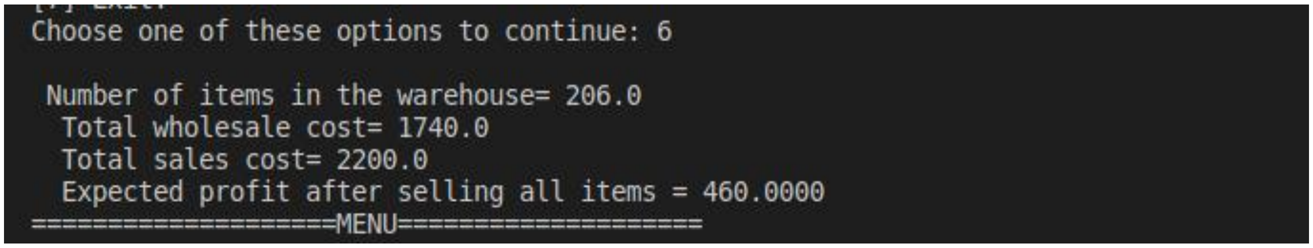
Generate a report about the sales status of the warehouse



```
def menuOptions(choice):
    if choice == 1:
        myCompany.getwarehouseComp().addItemToWaerhouse()
    elif choice == 2:
        myCompany.addSupermarketToSystem()
    elif choice == 3:
        ListOfItemsInWharehouseBasedOnExpiryDate()
    elif choice == 4:
        myCompany.getwarehouseComp().clearItem()
    elif choice == 5:
        myCompany.distibuteItems()
    elif choice == 6:
        myCompany.getwarehouseComp().GenerateReport()
    elif choice == 7:
        print("The program ended.")
```

We call ***getwarehouseComp()*** method from the company class the will return the products in the warehouse then we call from the same class ***GenerateReport()*** method that will print for the user the report about the warehouse for the company.

User Interface:



```
Choose one of these options to continue: 6

 Number of items in the warehouse= 206.0
  Total wholesale cost= 1740.0
  Total sales cost= 2200.0
  Expected profit after selling all items = 460.0000
====================MENU====================
```

Conclusion:

In this project we became more familiar with python programming. We succeed in implementing a software system that manages the product items in a warehouse and the distribution of these products to the supermarkets. This project encouraged us to do some researches for different the sectors that uses python programming such as AI and Machine learning.

References:

1- Linux Lab Manual

Access Date: 22-30/8/2022.

2- W3school for Python Programming

https://www.w3schools.com/python/default.asp

Access Date: 25-30/8/2022.