SMS

**SALES MANAGEMENT SYSTEM**

COMPUTER SCIENCE CBSE PROJECT

**Prateek Vachher | XII – C | Amity International School, Saket**

Contents

[Acknowledgement 3](#_Toc460528755)

[Certificate 4](#_Toc460528756)

[Introduction 5](#_Toc460528757)

[Data Dictionary 6](#_Toc460528758)

[Product Catalog 12](#_Toc460528759)

[Pseudo Codes 13](#_Toc460528760)

[Code 24](#_Toc460528761)

[EMPLOYEE MANAGEMENT 25](#_Toc460528762)

[INVENTORY MANAGEMENT 32](#_Toc460528763)

[EMAIL ENGINE 55](#_Toc460528764)

[SAMPLE BILL GENERATED 57](#_Toc460528765)

[Output 59](#_Toc460528766)

Acknowledgement

I would like to thank my computer science teacher, Mrs. Sunita Dash ma’am, without whom this project would not have come forth. Her support, guidance and ever encouraging thoughts kept me on right track throughout the course of this project.

I would also like to thank my Family for constantly supporting me.

Certificate

This is to certify that **Prateek Vachher** and **Vishrut Jaipuria** of class **XII C** have carried out this project under my supervision and guidance. This project is their genuine work.

**Mrs. Sunita Dash**

Computer Science Teacher

Amity International School, Saket

Introduction

This project aims at developing software which can facilitate operations of any company which primarily deals in sales of computer hardware and Inventory Management. The project uses the concept of data file handling and other concepts of “object oriented programming”.

The prime objective is to keep record of all the employees and keep a track of every transaction thereby maintaining inventory of various types of laptops, printers, scanners and desktops available at “Machine Hardware Organization (MHO)”. Thus, a Bill is presented to the customer at the end when he has finished ordering a particular type of product.

Also, a key feature of the program is that it automatically provides bonuses to employees on every sales made by the employee. There is an in-built warning system which alarms the user about low stock quantity and does not proceed further with the program unless until stock is added.

Data Dictionary

Class: employee

**Data Members**

|  |  |  |  |
| --- | --- | --- | --- |
| S. No. | Name | Type | Remarks |
| 1. | id | Integer | Employee ID |
| 2. | First Name | String | First Name of Employee |
| 3. | Surname | String | Last Name of Employee |
| 4. | Address | String | Address of Employee |
| 5. | City | String | City of Employee |
| 6. | Product\_Division | String | Product Division of Employee |
| 7. | dobday | Integer | Date of Birth of Employee |
| 8. | dobmonth | Integer | Month of Birth of Employee |
| 9. | dobyear | Integer | Year of Birth of Employee |
| 10. | dojdate | Integer | Date of Joining of Employee |
| 11. | dojmonth | Integer | Month of Joining of Employee |
| 12. | dojyear | Integer | Year of Joining of Employee |
| 13. | targets | Integer | Targers of Employee |
| 14. | bonus | Integer | Bonus of Employee |

**Member Functions**

|  |  |  |  |
| --- | --- | --- | --- |
| S. No. | Name | Return Type | Remarks |
| 1. | valueinput | Void | Updates Dictionary with Employee Credentials |
| 2. | dobconsolidate | Void | Consolidates Employee Date of Birth |
| 3. | dojconsolidate | Void | Consolidates Employee Date of Joining |
| 4. | generaterecord | Generated Record | Generate Employee Records |
| 5. | patchspot | Predefined Values | Consolidates Employee Credentials |
| 6. | updatefirstname | Void | Updates Employee First Name |
| 7. | updatesurname | Void | Updates Employee Last Name |
| 8. | updateaddress | Void | Updates Employee Address |
| 9. | updateproduct | Void | Updates Product Division |
| 10. | sale | Void | Generates Bonus for Employee |

Class: inventory

**Data Members**

|  |  |  |  |
| --- | --- | --- | --- |
| S. No. | Name | Type | Remarks |
| 1 | product\_code | string | Product Code |
| 2 | product\_name | string | Product Name |
| 3 | product\_quantity | Integer | Product Quantity |
| 4 | count | Integer | Bill Number |
| 5 | product\_SP | Integer | Product Selling Price |
| 6 | product\_company | String | Product Company |
| 7 | product\_VAT | Integer | Product VAT |
| 8 | tempdict | Dictionary | Saves Product Information |
| 9 | usermainmenuinput\_inventory | Integer | Take main menu Input |
| 10 | salespersonid | Integer | Salesperson ID |
| 11 | today | String | To import the date |

**Member Functions**

|  |  |  |  |
| --- | --- | --- | --- |
| S. No. | Name | Return Type | Remarks |
| 1 | menu | void | Main Menu |
| 2 | addstocks | void | Add Stocks |
| 3 | removestocks | void | Remove Stocks |
| 4 | addnewproduct | void | Add A New Product |
| 5 | removeproduct | void | Remove a Product |
| 6 | savecustomerinformation | void | Add A New Customer |
| 7 | entersales | void | Enter Sales |
| 8 | billgeneration | void | Generate A Bill |
| 9 | editinformation | void | Edit Product Information |
| 10 | inventorysearch | void | Search For a Product |
| 11 | inventorylistout | void | View Full Inventory |
| 12 | billsearch | void | Search For A Bill |
| 13 | fullvalue | void | Get Full Value of Present Stocks |
| 14 | salesdates | void | Get Sales Figures Between 2 entered dates |
| 15 | salescheck | void | Get Sales Figures For a Specific Dates |
| 16 | viewcustomerinformation | void | View Customer Information |
| 17 | existancecheckername | List | Checks For existance of a product name |
| 18 | existancecheckercode | List | Checks For existance of a product code |
| 19 | reedit | Integer | Submenu |
| 20 | quantitychecker | String | Checks for quantity |

Product Catalog

|  |  |  |
| --- | --- | --- |
| Product Codes and Types | | |
| Product Type | **Product Code** | **Price (in Rupees)** |
| Laptops | L1 | 20,000 |
| L2 | 30,000 |
| L3 | 40,000 |
| Printers | P1 | 15,000 |
| P2 | 20,000 |
| P3 | 25,000 |
| Scanners | S1 | 5,000 |
| S2 | 10,000 |
| S3 | 15,000 |
| Desktops | D1 | 40,000 |
| D2 | 50,000 |
| D3 | 60,000 |

Pseudo Codes

1. Pseudo Code:
   * + Class: employee
     + Function Name: valueinput( )
     + Return Type: void
     + Objective: Updates Dictionary with Employee Credentials
     + Access Specifier: public
     + File Name: employeemanagement
2. Pseudo Code:
   * + Class: employee
     + Function Name: dobconsolidate( )
     + Return Type: void
     + Objective: Consolidates Employee Date of Birth
     + Access Specifier: public
     + File Name: employeemanagement
3. Pseudo Code:
   * + Class: employee
     + Function Name: dojconsolidate( )
     + Return Type: void
     + Objective: Consolidates Employee Date of Joining
     + Access Specifier: public
     + File Name: employeemanagement
4. Pseudo Code:
   * + Class: employee
     + Function Name: generaterecord()
     + Return Type: Generated Record
     + Objective: Generate Employee Records
     + Access Specifier: public
     + File Name: employeemanagement
5. Pseudo Code:
   * + Class: employee
     + Function Name: patchspot( )
     + Return Type: Predefined Values
     + Objective: Consolidates Employee Credentials
     + Access Specifier: public
     + File Name: employeemanagement
6. Pseudo Code:
   * + Class: employee
     + Function Name: updatefirstname(char[ ], char [ ])
     + Return Type: void
     + Objective: Updates Employee First Name
     + Access Specifier: public
     + File Name: employeemanagement
7. Pseudo Code:
   * + Class: employee
     + Function Name: updatesurname( )
     + Return Type: void
     + Objective: Updates Employee Last Name
     + Access Specifier: public
     + File Name: employeemanagement
8. Pseudo Code:
   * + Class: employee
     + Function Name: updateaddress( )
     + Return Type: void
     + Objective: Updates Employee Address
     + Access Specifier: public
     + File Name: employeemanagement
9. Pseudo Code:
   * + Class: employee
     + Function Name: updateproduct( )
     + Return Type: void
     + Objective: Updates Employee Product Division
     + Access Specifier: public
     + File Name: employeemanagement
10. Pseudo Code:
    * + Class: employee
      + Function Name: sale( )
      + Return Type: void
      + Objective: Generates Bonus for employees
      + Access Specifier: public
      + File Name: employeemanagement
11. Pseudo Code:
    * + Function Name: emailit( )
      + Return Type: Boolean
      + Objective: Sends Email to the Customer with Invoice
      + Access Specifier: public
      + File Name: emailengine
12. Pseudo Code:
    * + Function Name: mainmenu( )
      + Return Type: string
      + Objective: Provides the User with Menu Interface
      + Access Specifier: public
      + File Name: employeemanagement
13. Pseudo Code:
    * + Function Name: sync( )
      + Return Type: void
      + Objective: To read write records to the Databases
      + Access Specifier: public
      + File Name: employeemanagement
14. Pseudo Code:
    * + Function Name: searchid( )
      + Return Type: Record
      + Objective: To return employee credentials after searching from database
      + Access Specifier: public
      + File Name: employeemanagement
15. Pseudo Code:
    * + Function Name: getid( )
      + Return Type: Integer
      + Objective: Gets Upper bound Employee ID from web SQL Database hosted at AWS Server
      + Access Specifier: public
      + File Name: employeemanagement
16. Pseudo Code:
    * + Function Name: updateid( )
      + Return Type: void
      + Objective: Updates Upperbound of Employee ID from web SQL Database hosted at AWS Server
      + Access Specifier: public
      + File Name: employeemanagement
17. . Pseudo Code:
    * + Class: inventory
      + Function Name : menu()
      + Return Type: Void
      + Objective : Navigates to different function
      + Access Specifier : Public
      + File Name : inventorymanagement
18. Pseudo Code:
    * + Class  : inventory
      + Function Name : addstocks()
      + Return Type : Void
      + Objective :  To add stock to the current Product Database
      + Access Specifier : Public
      + File Name : inventorymanagement
19. Pseudo Code:
    * + Class : inventory
      + Function Name   : removestocks()
      + Return Type : Void
      + Objective  :  To remove stock from the current Product Database
      + Access Specifier : Public
      + File Name : inventorymanagement
20. Pseudo Code:
    * + Class  : inventory
      + Function Name : addproduct()
      + Return Type : Void
      + Objective :  To add a new product to the Product Database
      + Access Specifier : Public
      + File Name : inventorymanagement
21. Pseudo Code:
    * + Class : inventory
      + Function Name: removeproduct()
      + Return Type : Void
      + Objective :  To remove a product from the Product Database using product  code/name/company name
      + Access Specifier : Public
      + FileName:inventorymanagement
22. Pseudo Code:
    * + Class : inventory
      + Function Name   : savecustomerinformation()
      + Return Type : Void
      + Objective :  To add a new customer to the Customer Database
      + Access Specifier : Public
      + File Name: inventorymanagement
23. Pseudo Code:
    * + Class: inventory
      + Function Name   : entersales()
      + Return Type  : Void
      + Objective:  To make a new invoice/enter sales
      + Access Specifier : Public
      + File Name : inventorymanagement
24. Pseudo Code:
    * + Class : inventory
      + Function Name : billgenerates()
      + Return Type: Void
      + Objective:  Generates the bill in the backend
      + Access Specifier : Public
      + File Name: inventorymanagement
25. Pseudo Code:
    * + Class: inventory
      + Function Name : editinformation()
      + Return Type: Void
      + Objective:  To edit existing product information i.e
      + name/code/description/vat/selling price
      + Access Specifier : Public
      + File Name : inventorymanagement
26. Pseudo Code:
    * + Class : inventory
      + Function Name : Inventorysearch()
      + Return Type: Void
      + Objective:  To search the inventory for product information
      + Access Specifier : Public
      + File Name: inventorymanagement
27. Pseudo Code:
    * + Class : inventory
      + Function Name   : inventorylistout()
      + Return Type: Void
      + Objective:  To view complete inventory
      + Access Specifier : Public
      + File Name : inventorymanagement
28. Pseudo Code:
    * + Class : inventory
      + Function Name : billsearch()
      + Return Type: Void
      + Objective:  To search for information for a particular invoice
      + Access Specifier : Public
      + File Name : inventorymanagement
29. Pseudo Code:
    * + Class : inventory
      + Function Name   : fullvalue()
      + Return Type  : Void
      + Objective :  To get full value of stocks in the inventory
      + Access Specifier : Public
      + File Name : inventorymanagement
30. Pseudo Code:
    * + Class  : inventory
      + Function Name  : salesdates()
      + Return Type : Void
      + Objective :  To get sales figures between 2 dates
      + Access Specifier : Public
      + File Name : inventorymanagement
31. Pseudo Code:
    * + Class : inventory
      + Function Name   : salescheck()
      + Return Type : Void
      + Objective  :  To get sales figures for a particular date
      + Access Specifier : Public
      + File Name : inventorymanagement
32. Pseudo Code:
    * + Class : inventory
      + Function Name   : viewcustomerinformation()
      + Return Type : Void
      + Objective :  To get existing customer information
      + Access Specifier : Public
      + File Name : inventorymanagement
33. Pseudo Code:
    * + Class : inventory
      + Function Name   : existancecheckername()
      + Return Type : List of True/False
      + Objective  :  To check for existance of a product name
      + Access Specifier : Public
      + File Name : inventorymanagement
34. 2. Pseudo Code:
    * + Class : inventory
      + Function Name  : existancecheckercode()
      + Return Type : List of True/False and product name
      + Objective :  To check for existance of a product code
      + Access Specifier : Public
      + File Name : inventorymanagement
35. Pseudo Code:
    * + Class : inventory
      + Function Name : reedit()
      + Return Type : Integer
      + Objective :  Submenu
      + Access Specifier : Public
      + File Name  : inventorymanagement
36. Pseudo Code:
    * + Class : inventory
      + Function Name   : quantitychecker()
      + Return Type : List of True/False
      + Objective :  To check for existance quantity for a product
      + Access Specifier : Public
      + File Name : inventorymanagement

Code

EMPLOYEE MANAGEMENT

#IMPORT CALLS  
  
*import* pickle  
*from* inventorymanagement *import* \*  
  
# -----------------------------------------------------------------------------#  
# CLASSES DEFINATION #  
  
# EMPLOYEE DEFINATION #  
*class* employee(object):  
 *def \_\_init\_\_*(self):  
 self.id = 0  
 self.\_\_first\_name = ""  
 self.\_\_surname = ""  
 self.\_\_address = ""  
 self.\_\_city = ""  
 self.\_\_product\_division = ""  
 self.\_\_dobday = 0  
 self.\_\_dobmonth = 0  
 self.\_\_dobyear = 0  
 self.\_\_dojday = 0  
 self.\_\_dojmonth = 0  
 self.\_\_dojyear = 0  
 self.\_\_dojdate = ""  
 self.\_\_targets = 0  
 self.\_\_bonus = 0  
 self.\_\_salary = 60000  
 self.\_\_final1 = ""  
 self.\_\_final2 = ""  
  
 *def* valueinput(self):  
 self.\_\_first\_name = raw\_input("1. First Name : ")  
 self.\_\_surname = raw\_input("2. Surname : ")  
 self.\_\_address = raw\_input("3. Address : ")  
 self.\_\_city = raw\_input("4. City : ")  
 self.\_\_product\_division = raw\_input("5. Product Division : ").lower()  
 *print* "6. Date of Birth"  
 self.\_\_dobday = input(" Day : ")  
 self.\_\_dobmonth = input(" Month : ")  
 self.\_\_dobyear = input(" Year : ")  
 *print* "7. Date of Joining"  
 self.\_\_dojday = input(" Day : ")  
 self.\_\_dojmonth = input(" Month : ")  
 self.\_\_dojyear = input(" Year : ")  
  
 *def* dobconsolidate(self):  
 *if* len(self.\_\_final1) < 3:  
 self.\_\_final1 += str(self.\_\_dobday)  
 self.\_\_final1 += "/"  
 self.\_\_final1 += str(self.\_\_dobmonth)  
 self.\_\_final1 += "/"  
 self.\_\_final1 += str(self.\_\_dobyear)  
  
 *def* dojconsolidate(self):  
 *if* len(self.\_\_final2) < 3:  
 self.\_\_final2 += str(self.\_\_dojday)  
 self.\_\_final2 += "/"  
 self.\_\_final2 += str(self.\_\_dojmonth)  
 self.\_\_final2 += "/"  
 self.\_\_final2 += str(self.\_\_dojyear)  
  
 *def* generaterecord(self):  
 self.dobconsolidate()  
 self.dojconsolidate()  
 *print* "1. Name :", self.\_\_first\_name, self.\_\_surname  
 *print* "2. Address :", self.\_\_address  
 *print* "3. Product Division :", self.\_\_product\_division  
 *print* "4. Date of Birth :", self.\_\_final1  
 *print* "5. Date of Joining :", self.\_\_final2  
 *print* "6. Targets Achieved :", self.\_\_targets  
 *print* "7. Bonus :", "Rs.", self.\_\_bonus  
 *print* "8. Salary :", "Rs.", self.\_\_salary  
  
 *def* patchspot(self):  
 *return* [self.\_\_first\_name, self.\_\_first\_name + " " + self.\_\_surname, self.\_\_product\_division]  
  
 *def* updatefirstname(self, *new*):  
 self.\_\_first\_name = *new  
 def* updatesurname(self,*new*):  
 self.\_\_surname = *new  
 def* updateaddress(self,*new*):  
 self.\_\_address = *new  
 def* updateproduct(self,*new*):  
 self.\_\_product\_division = *new  
 def* sale(self,*amount*,*quantity*):  
 self.\_\_bonus += *amount* \* 0.02  
 self.\_\_targets += *quantity*# -----------------------------------------------------------------------------#  
# FUNCTION DEFINATION #  
  
# MAIN MENU - 1  
*def* mainmenu1():  
 *print* "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_", "Main Menu", "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"  
 *print* ""  
 *print* "1. Employee Management"  
 *print* "2. Inventory Management"  
 *print* "0. Exit"  
 *print* ""  
 a = input("Choice(1/2/0): ")  
 *return* a  
  
# MAIN MENU - 2  
*def* mainmenu2():  
 *print* "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_", "Employee Directory", "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"  
 *print* ""  
 *print* "1. Add New Employee"  
 *print* "2. Show Employee Records"  
 *print* "3. Search Employee Directory"  
 *print* "4. Delete Employee Record"  
 *print* "5. Modify Employee Record"  
 *print* "0. Return to Main Menu"  
 *print* ""  
 a = input("Choice(1/2/3/4/5/0): ")  
 *return* a  
  
# MAIN MENU - 3  
*def* mainmenu3():  
 *print* "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_", "Employee Search", "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"  
 *print* ""  
 *print* "1. Search By User ID"  
 *print* "2. Search By Name"  
 *print* "3. Search By Product Division "  
 *print* "0. Return to Previous Menu "  
 *print* ""  
 a = input("Choice(1/2/3/0): ")  
 *return* a  
  
# SUB MENU - 1  
*def* submenu1():  
 *print* "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_", "Modifying Employee Credentials", "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"  
 *print* ""  
 *print* "Enter 1 to modify First Name"  
 *print* "Enter 2 to modify Surname"  
 *print* "Enter 3 to modify Address"  
 *print* "Enter 4 to modify Product Division"  
 *print* "0. Return to Previous Menu "  
 *print* ""  
 a = raw\_input("Choice(1,2,3,4/0): ").split()  
 *return* a  
  
# SYNC WITH MASTER DB  
*def* sync(*data*):  
 pickle.dump(*data*, open("Databases/employeedatabase.db", "wb"))  
  
#SEARCH  
*def* searchid(*userid*):  
 details = masterdatabase[*userid*]  
 details.generaterecord()  
  
*def* getid():  
 *import* pymysql  
 db = pymysql.connect("52.66.46.128", "root", "Welcome123", "sms")  
 cursor = db.cursor()  
 sql = "SELECT \* FROM employee\_id"  
 *try*:  
 cursor.execute(sql)  
 results = cursor.fetchall()  
 *return* results[0][0]  
 *except*:  
 *print* "ERROR CHECKING Auth"  
  
*def* updateid():  
 *import* pymysql  
 db = pymysql.connect("52.66.46.128", "root", "Welcome123", "sms")  
 cursor = db.cursor()  
 sql = "UPDATE employee\_id set id=id+1"  
 *try*:  
 cursor.execute(sql)  
 db.commit()  
 *except*:  
 *print* "ERROR CHECKING Auth"  
 db.rollback()  
  
# -----------------------------------------------------------------------------#  
# MASTER WORKSPACE #  
  
  
*while* True:  
 *print* ""  
 mainmenu1var = mainmenu1()  
 # OPTION 1 - MAIN MENU 1  
 *if* mainmenu1var == 1:  
 *while* True:  
 masterdatabase = pickle.load(open("Databases/employeedatabase.db","rb"))  
 *print* ""  
 mainmenu2var = mainmenu2()  
  
  
 # OPTION 1 - MAIN MENU 2  
 *if* mainmenu2var == 1:  
 a = input("How many Employees do you wish to add? ")  
 *while* a != 0:  
 a -= 1  
 *print* ""  
 b = getid()  
 *print* "Employee ID: ", b  
 *if* b *in* masterdatabase :  
 *print* "Error, The Employee ID is already taken"  
 a += 1  
 *continue* c = employee()  
 c.valueinput()  
 masterdatabase[b] = c  
 sync(masterdatabase)  
 *print* "Employee Credentials synced with Database"  
 updateid()  
  
 # OPTION 2 - MAIN MENU 2  
 *elif* mainmenu2var == 2:  
 *for* counter *in* masterdatabase:  
 *print* ""  
 *print* ".............................................................................."  
 *print* ""  
 *print* "Employee ID:", counter  
 (masterdatabase[counter]).generaterecord()  
  
 # OPTION 3 - MAIN MENU 2  
 *elif* mainmenu2var == 3:  
 *while* True:  
 mainmenu3var = mainmenu3()  
 # OPTION 1 - MAIN MENU 3  
 *if* mainmenu3var == 1:  
 userid = input("Enter User ID of the Employee: ")  
 *if* userid *not in* masterdatabase:  
 *print* ""  
 *print* ".............................................................................."  
 *print* " No Employee with this User ID"  
 *print* ".............................................................................."  
 *print* ""  
 *else*:  
 *print* ""  
 *print* ".............................................................................."  
 *print* "Employee ID:", userid  
 searchid(userid)  
 *print* ".............................................................................."  
 *print* ""  
  
 # OPTION 2 - MAIN MENU 3  
 *elif* mainmenu3var == 2:  
 username = raw\_input("Enter Name of the Employee: ")  
 flag = 0  
 *for* counter *in* masterdatabase:  
  
 *if* (masterdatabase[counter]).patchspot()[0].lower() == username.lower() *or* (masterdatabase[counter]).patchspot()[1].lower() == username.lower():  
 *print* ".............................................................................."  
 *print* ""  
 *print* "User ID:", counter  
 (masterdatabase[counter]).generaterecord()  
 *print* ""  
 flag = 1  
 *if* flag == 0:  
 *print* ""  
 *print* "The Employee with name", username, "does not exist in the master database"  
 *print* ""  
  
  
 # OPTION 3 - MAIN MENU 3  
 *elif* mainmenu3var == 3:  
 productdiv = raw\_input("Enter Product Division: ")  
 flag = 0  
 *for* counter *in* masterdatabase:  
  
 *if* (masterdatabase[counter]).patchspot()[2].lower() == productdiv.lower():  
 *print* ".............................................................................."  
 *print* ""  
 *print* "User ID:", counter  
 (masterdatabase[counter]).generaterecord()  
 *print* ""  
 flag = 1  
 *if* flag == 0:  
 *print* ""  
 *print* "The Product Division", productdiv, "does not exist in the master database"  
 *print* ""  
  
 # OPTION 0 - MAIN MENU 3  
 *elif* mainmenu3var == 0:  
 *break* # Else Condition - Main  
 *else*:  
 *print* ""  
 *print* "Incorrect Option entered!"  
 *print* ""  
  
 # OPTION 4 - MAIN MENU 2  
 *elif* mainmenu2var == 4:  
 userid = input("Enter User ID of Employee to be Deleted: ")  
 *if* userid *not in* masterdatabase:  
 *print* ""  
 *print* ".............................................................................."  
 *print* " No Employee with this User ID"  
 *print* ".............................................................................."  
 *print* ""  
 *else*:  
 *print* ""  
 *print* "Found Employee with ID:", userid  
 query = raw\_input("Are you sure that you want to delete the following Employee? ")  
 *if* query == 'Y' *or* query == 'y' *or* query == "Yes" *or* query == "yes":  
 *del* masterdatabase[userid]  
 *print* ""  
 sync(masterdatabase)  
  
 *elif* mainmenu2var == 5:  
 userid = input("Enter User ID of the Employee: ")  
 *if* userid *not in* masterdatabase:  
 *print* ""  
 *print* ".............................................................................."  
 *print* " No Employee with this User ID"  
 *print* ".............................................................................."  
 *print* ""  
 *else*:  
 *print* ""  
 submenu1var = submenu1()  
 *if* submenu1var == '0':  
 *break  
 for* counter *in* submenu1var:  
 *if* counter == '1':  
 updatedname = raw\_input("Enter updated First Name: ")  
 masterdatabase[userid].updatefirstname(updatedname)  
 *elif* counter == '2':  
 updatedsurname = raw\_input("Enter updated Surname: ")  
 masterdatabase[userid].updatesurname(updatedsurname)  
 *elif* counter == '3':  
 updatedaddress = raw\_input("Enter updated Address: ")  
 masterdatabase[userid].updateaddress(updatedaddress)  
 *elif* counter == '4':  
 updatedproduct = raw\_input("Enter updated Product Division: ")  
 masterdatabase[userid].updateproduct(updatedproduct)  
  
 sync(masterdatabase)  
 *print* "Employee Credentials updated in Database"  
 *elif* mainmenu2var == 0:  
 *break  
 else*:  
 *print* ""  
 *print* "Incorrect Option entered!"  
 *print* ""  
  
 *elif* mainmenu1var == 2:  
 a = inventory()  
 a.menu()

INVENTORY MANAGEMENT

#### IMPORT CALLS  
*import* pickle  
*import* os  
*import* time  
*from* datetime *import* datetime  
*from* emailengine *import* emailit  
#-------------------- CLASS DEFINATION --------------------------#  
  
*class* inventory(object):  
  
 #############################################################################################  
  
 *def \_\_init\_\_*(self):  
 self.product\_code = ''  
 self.product\_name = ''  
 self.product\_quantity = 0  
 self.count = 0  
 self.product\_SP = 0  
 self.product\_company = ''  
 self.product\_VAT = 0  
 self.tempdict={}  
 self.usermainmenuinput\_inventory = 0  
 self.salespersonid = 0  
 self.today = ''  
  
 #############################################################################################  
 #############################################################################################  
  
 *def* menu(self):  
 mainmenu = ['1. Add Stocks','2. Remove Stocks','3. Add New Product','4. Remove Product','5. To add new customer information','6. Make Sales','7. Edit Information',  
 '8. Search Inventory','9. View Inventory','10. Search Sales Records',  
 '11. View full value of stocks','12. To check sales between 2 dates','13. To check sales for a specific date',  
 '14. To view customer information','15. Go Back']  
 *print  
 print* '\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Inventory Menu \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_'  
 *print* ""  
 *for* choice *in* mainmenu:  
 *print* choice  
 *print* self.usermainmenuinput\_inventory = input('Choice(1/2/3/4/5/6/7/8/9/10/11/12/13/14/15): ')  
 *while* self.usermainmenuinput\_inventory *not in* [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]:  
 self.usermainmenuinput\_inventory = input('Choice(1/2/3/4/5/6/7/8/9/10/11/12/13/14/15): ')  
 *print  
 if* self.usermainmenuinput\_inventory == 1:  
 self.addstocks()  
 *elif* self.usermainmenuinput\_inventory == 2:  
 self.removestocks()  
 *elif* self.usermainmenuinput\_inventory == 3:  
 self.addnewproduct()  
 *elif* self.usermainmenuinput\_inventory == 4:  
 self.removeproduct()  
 *elif* self.usermainmenuinput\_inventory == 5:  
 self.savecustomerinformation()  
 *elif* self.usermainmenuinput\_inventory == 6:  
 self.entersales()  
 *elif* self.usermainmenuinput\_inventory == 7:  
 self.editinformation()  
 *elif* self.usermainmenuinput\_inventory == 8:  
 self.inventorysearch()  
 *elif* self.usermainmenuinput\_inventory == 9:  
 self.inventorylistout()  
 *elif* self.usermainmenuinput\_inventory == 10:  
 self.billsearch()  
 *elif* self.usermainmenuinput\_inventory == 11:  
 self.fullvalue()  
 *elif* self.usermainmenuinput\_inventory == 12:  
 self.salesdates()  
 *elif* self.usermainmenuinput\_inventory == 13:  
 self.salescheck()  
 *elif* self.usermainmenuinput\_inventory == 14:  
 self.viewcustomerinformation()  
 *else*:  
 *pass* #############################################################################################  
 #############################################################################################  
  
 *def* addstocks(self):  
 globalinventorydatabase = pickle.load(open("Databases/inventorydatabase.db", "rb"))  
 *print* "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Add Stocks \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"  
 *print* response = 'y'  
 *while* response == 'y':  
 code = raw\_input('Product Code: ')  
 d = self.existancecheckercode(code)  
 d1 = d[1]  
 *if* d[0] == True:  
 *print* "Product Name :",d[1]  
 *print* 'Current Stock :', globalinventorydatabase[d1]['Quantity']  
 *print* tempstock = input('Enter Stocks to be Added : ')  
 tempstock1 = input('Re-Enter Stocks to be Added: ')  
 *if* tempstock == tempstock1:  
 globalinventorydatabase[d1]['Quantity'] += tempstock  
 *print* 'Stock Successfully Synced'  
 *print* 'New stocks :',globalinventorydatabase[d1]['Quantity']  
 response = 'n'  
 *else*:  
 *print* 'Please Re-Enter'  
 *print  
 else*:  
 *print* 'Incorrect Product Code'  
 *print* pickle.dump(globalinventorydatabase, open("Databases/inventorydatabase.db", "wb"))  
 self.menu()  
  
 #############################################################################################

#############################################################################################  
  
 *def* removestocks(self):  
 globalinventorydatabase = pickle.load(open("Databases/inventorydatabase.db", "rb"))  
 *print* "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Remove Stocks \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"  
 *print  
 print* '1. Product Code\n2. Product Name\n3. Main Menu'  
 *print* choice1 = input('Choice(1/2/3) : ')  
 *print* ""  
 *while* choice1 *not in* [1,2,3]:  
 *print* 'Incorrect option'  
 choice1 = input('Choice(1/2/3) : ')  
 *if* choice1 == 1:  
 *while* True:  
 code = raw\_input('Product Code: ')  
 d = self.existancecheckercode(code)  
 *if* d[0] == True:  
 d1 = d[1]  
 *print* "Product Name :",d[1]  
 *print* 'Current Stock :', globalinventorydatabase[d1]['Quantity']  
 tempstock = input('Stock to be Removed: ')  
 tempstock1 = input('Re-Enter Stock to be Removed: ')  
 *if* tempstock == tempstock1:  
 globalinventorydatabase[d1]['Quantity'] -= tempstock  
 *print* 'Inventory has Succesfully been Updated'  
 *print* 'New stocks :',globalinventorydatabase[d1]['Quantity']  
 *break  
 else*:  
 *print* "Incorrect Product Code"  
 *print* ""  
 *elif* choice1 == 2:  
 *while* True:  
 name = raw\_input('Product Name: ')  
 *if* name *in* globalinventorydatabase:  
 *print* 'Current Stock :', globalinventorydatabase[name]['Quantity']  
 adstock = input('Stock to be Removed: ')  
 adstock1 = input('Re-Enter Stock to be Removed: ')  
 *if* adstock == adstock1:  
 globalinventorydatabase[name]['Quantity'] -= adstock  
 *print* 'Inventory has Succesfully been Updated'  
 *print* 'New stocks :',globalinventorydatabase[name]['Quantity']  
 *break  
 else*:  
 *print* 'Incorrect Product Name'  
 *print* ""  
 *elif* choice1 == 3:  
 self.menu()  
 pickle.dump(globalinventorydatabase, open("Databases/inventorydatabase.db", "wb"))  
 self.menu()  
  
 #############################################################################################  
 #############################################################################################  
  
 *def* addnewproduct(self):  
  
 *print* '\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Add New Product \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_'  
 *print* globalinventorydatabase = pickle.load(open("Databases/inventorydatabase.db", "rb"))  
 self.tempdict = {}  
 self.product\_code = raw\_input('Product Code: ')  
 self.product\_name = raw\_input('Product Name: ')  
 flag = 0  
 *if* self.existancecheckercode(self.product\_code)[0] == False:  
 flag = 1  
 *if* self.existancecheckername(self.product\_name)[0] == False:  
 flag = 2  
 self.product\_company = raw\_input('Product Company Name: ')  
 self.product\_description = raw\_input('Product Description :')  
 self.product\_quantity = input('Product Quantity: ')  
 self.product\_SP = input('Product Selling Price: ')  
 self.product\_VAT = input('Product VAT: ')  
 self.tempdict['Code']=self.product\_code  
 self.tempdict['Description'] = self.product\_description  
 self.tempdict['Company'] = self.product\_company  
 self.tempdict['Quantity'] = self.product\_quantity  
 self.tempdict['SP'] = self.product\_SP  
 self.tempdict['VAT'] = self.product\_VAT  
 self.globalinventorydatabase1[self.product\_name] = self.tempdict  
 choice3 = self.product\_name  
 *for* x *in* globalinventorydatabase:  
 tedict = globalinventorydatabase[x]  
 *if* x == choice3:  
 *print  
 print* 'Name :',x  
 *for* y *in* tedict:  
 *print* y,':',tedict[y]  
 *print  
 print* '1. To confirm addition\n2. To reenter product information'  
 choice = input('Choice(1/2): ')  
 *if* choice == 1:  
 globalinventorydatabase.update(self.globalinventorydatabase1)  
 pickle.dump(globalinventorydatabase, open("Databases/inventorydatabase.db", "wb"))  
 *print  
 print* 'Product successfully added'  
  
 *elif* choice == 2:  
 self.addnewproduct()  
 *else*:  
 *print* 'Wrong choice inputted'  
 self.reedit()  
 *if* self.reedit() == 2:  
 self.addnewproduct()  
  
 *if* flag == 0:  
 *print* 'The product code you entered already exists'  
 self.reedit()  
 *if* self.reedit() == 2:  
 self.addnewproduct()  
  
 *elif* flag == 1:  
 *print* 'The product name you entered already exists'  
 self.reedit()  
 *if* self.reedit() == 2:  
 self.addnewproduct()  
  
 self.menu()  
  
 #############################################################################################

#############################################################################################  
  
 *def* removeproduct(self):  
 *print* '\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Remove Product \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_'  
 *print* globalinventorydatabase = pickle.load(open("Databases/inventorydatabase.db", "rb"))  
 *print* '1. to remove with product name\n2. to remove with product code\n3. to remove a company'  
 *print* choice6 = input('Choice(1/2/3): ')  
 *if* choice6 == 1:  
 chr = 'y'  
 *while* chr == 'y':  
 name1 = raw\_input('Enter product name: ')  
 qwerty = self.existancecheckername(name1)  
  
 *if* qwerty[0] == True:  
 name2 = raw\_input('Reenter product name: ')  
 *if* name1 == name2:  
 *print* '1. To confirm deletion\n2. To reenter product information\n3. to go to main menu'  
 choice = input('Choice(1/2/3): ')  
 *if* choice == 1:  
 *del* globalinventorydatabase[name1]  
 pickle.dump(globalinventorydatabase, open("Databases/inventorydatabase.db", "wb"))  
 *print* 'Product with name',name1,'successfully deleted'  
 chr = 'n'  
 *elif* choice == 2:  
 self.removeproduct()  
 *elif* choice == 3:  
 self.menu()  
 *else*:  
 *print* 'Wrong choice inputted'  
 self.reedit()  
 *if* self.reedit() == 2:  
 self.removeproduct()  
 *else*:  
 *print* 'Entries do not match'  
 self.reedit()  
 *if* self.reedit() == 2:  
 self.removeproduct()  
 *else*:  
 *print* 'Product Name does not exist'  
 *print* '1. To enter product name again\n2. To go back to main menu'  
 asd = input('Choice(1/2) : ')  
 *if* asd == 1:  
 chr = 'y'  
 *else*:  
 self.menu()  
 *elif* choice6 == 2:  
 code1 = raw\_input('Enter product code: ')  
 code2 = raw\_input('Reenter product code: ')  
 *if* code1 == code2:  
 d = self.existancecheckercode(code1)  
 *if* d[0] == True:  
 *print* '1. To confirm deletion\n2. To reenter product information\n3. to go to main menu'  
 choice = input('Choice(1/2/3)')  
 *if* choice == 1:  
 *del* globalinventorydatabase[d[1]]  
 pickle.dump(globalinventorydatabase, open("Databases/inventorydatabase.db", "wb"))  
 *print* 'Product with code',code1,'successfully deleted'  
 *elif* choice == 2:  
 self.removeproduct()  
 *elif* choice == 3:  
 self.menu()  
 *else*:  
 *print* 'Wrong choice inputted'  
 self.reedit()  
 *if* self.reedit() == 2:  
 self.removeproduct()  
  
 *else*:  
 *print* 'Entries do not match'  
 *elif* choice6 == 3:  
 company1 = raw\_input('Enter company name: ')  
 company2 = raw\_input('Reenter company name: ')  
 *if* company1 == company2:  
 *for* x *in* globalinventorydatabase:  
 tdict=globalinventorydatabase[x]  
 *if* tdict['Company'] == company1:  
 *print* '1. To confirm deletion\n2. To reenter product information\n3. to go to main menu'  
 choice = input('Choice(1/2/3)')  
 *if* choice == 1:  
 *del* globalinventorydatabase[x]  
 pickle.dump(globalinventorydatabase, open("Databases/inventorydatabase.db", "wb"))  
 *print* 'All products with company name',company1,'successfully deleted'  
 *elif* choice == 2:  
 self.removeproduct()  
 *elif* choice == 3:  
 self.menu()  
 *else*:  
 *print* 'Wrong choice inputted'  
 self.reedit()  
 *if* self.reedit() == 2:  
 self.removeproduct()  
 *break  
 else*:  
 *print* 'Entries do not match'  
 *else*:  
 *print* 'Wrong submenu choice entered'  
 self.reedit()  
 *if* self.reedit() == 2:  
 self.removeproduct()  
 *print* self.menu()  
  
 #############################################################################################

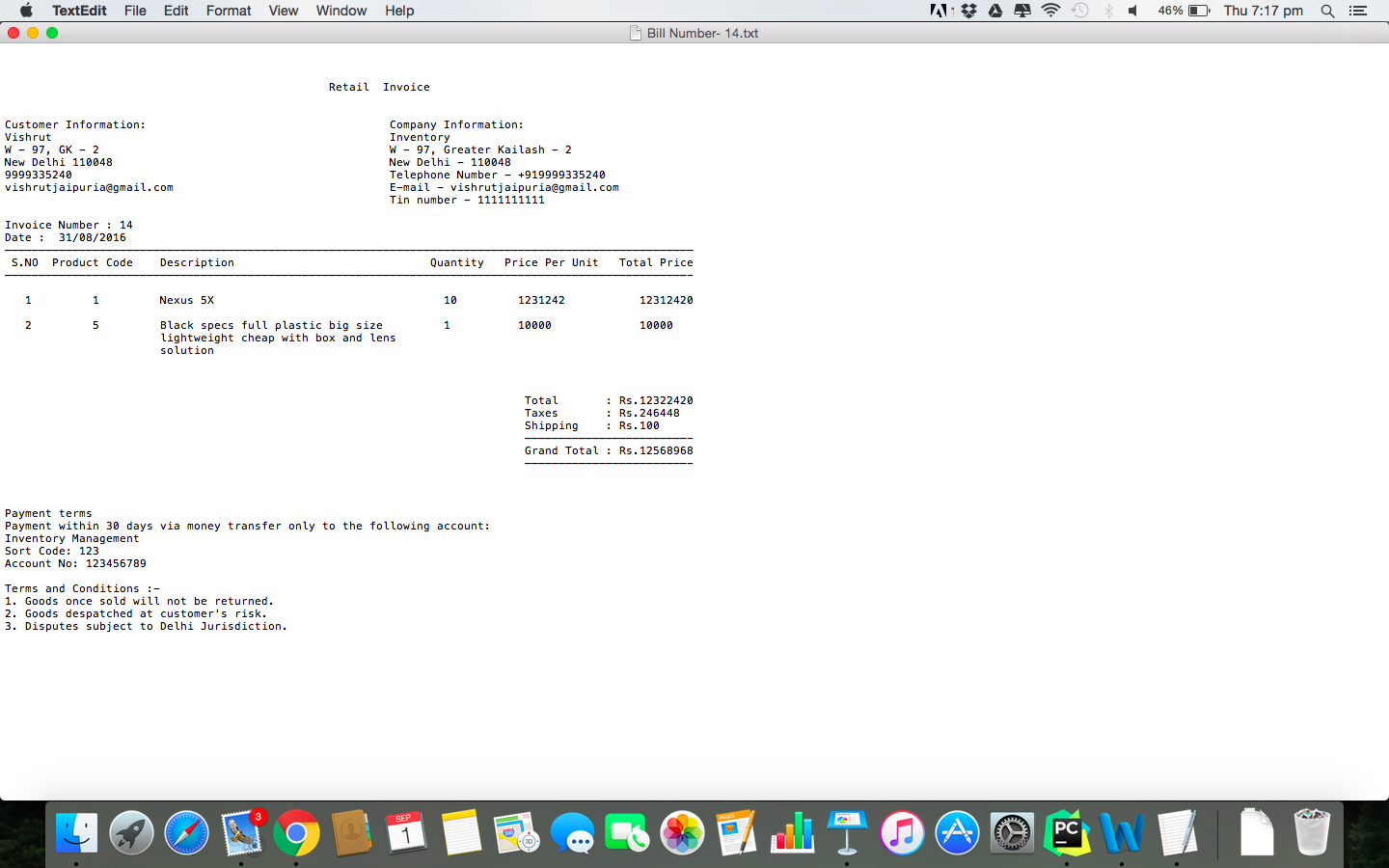
#############################################################################################  
  
 *def* savecustomerinformation(self):  
 *print* '\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ New Customer \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_'  
 *print* globalcompanydatabase = pickle.load(open("Databases/companymanagement.db", "rb"))  
 customername = raw\_input('Customer Name :')  
 customeradd1 = raw\_input('Customer Address(Line 1) :')  
 customeradd2 = raw\_input('Customer Address(Line 2) :')  
 customercity = raw\_input('Customer City :')  
 customerno = raw\_input('Customer Phone Number :')  
 customeremail = raw\_input('Customer Email :')  
 tempdict = {}  
 tempdict['address1'] = customeradd1  
 tempdict['address2'] = customeradd2  
 tempdict['City'] = customercity  
 tempdict['Phone Number'] = customerno  
 tempdict['email'] = customeremail  
 tempdict['Salesrecord'] = {}  
 globalcompanydatabase['Customers'][customername] = tempdict  
 *print  
 print* 'Successfully added new customer'  
 pickle.dump(globalcompanydatabase, open("Databases/companymanagement.db", "wb"))  
 self.menu()  
  
 #############################################################################################

#############################################################################################  
  
 *def* entersales(self):  
 *print* '\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Enter Sales \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_'  
 *print* globalemployeedatabase = pickle.load(open("Databases/employeedatabase.db", "rb"))  
 globalinventorydatabase = pickle.load(open("Databases/inventorydatabase.db", "rb"))  
 globalcompanydatabase = pickle.load(open("Databases/companymanagement.db", "rb"))  
 d = globalcompanydatabase['sales']  
 self.count = 1  
 *for* x *in* d:  
 self.count = x+1  
 *print* 'Bill Number :',self.count  
 *print* '1. To use existing customer information\n2. To add new customer information'  
 choice2 = input('1/2 : ')  
 *while* choice2 *not in* [1,2]:  
 choice2 = input('Choice(1/2) : ')  
 *if* choice2 == 2:  
 *print* "Bill Number: ", self.count  
 customername = raw\_input('Customer Name :')  
 customeradd1 = raw\_input('Customer Address(Line 1) :')  
 customeradd2 = raw\_input('Customer Address(Line 2) :')  
 customercity = raw\_input('Customer City :')  
 customerno = raw\_input('Customer Phone Number :')  
 customeremail = raw\_input('Customer Email :')  
 *print  
 print* '1. To save this customer information\n2. To continue'  
 choice3 = input('Choice(1/2) : ')  
 *if* choice3 == 1:  
 tempdict = {}  
 tempdict['address1'] = customeradd1  
 tempdict['address2'] = customeradd2  
 tempdict['City'] = customercity  
 tempdict['Phone Number'] = customerno  
 tempdict['email'] = customeremail  
 globalcompanydatabase['Customers'][customername] = tempdict  
 *print* 'Customer Information saved'  
 *print  
  
 elif* choice2 == 1:  
 customername = raw\_input('Customer Name :')  
 *if* customername *in* globalcompanydatabase['Customers']:  
 d1 = globalcompanydatabase['Customers'][customername]  
 customeradd1 = d1['address1']  
 customeradd2 = d1['address2']  
 customerno = d1['Phone Number']  
 customeremail = d1['email']  
 customercity = d1['City']  
  
 tdict = {}  
 self.today = time.strftime("%d/%m/%Y")  
 tdict['Name'] = customername  
 tdict['Address'] = customeradd1 +', ' + customeradd2  
 tdict['City'] = customercity  
 tdict['Phone Number'] = customerno  
 tdict['Email'] = customeremail  
 tempdict = {}  
 chr = 'Y'  
 l1,l2,l3,l4,l5,l6 = [],[],[],[],[],[]  
 self.salespersonid = (input("Employee ID: "))  
  
 *if* self.salespersonid *in* globalemployeedatabase:  
 tempdict['Salesperson id'] = self.salespersonid  
 *while* self.salespersonid *not in* globalemployeedatabase:  
 *print* 'Employee id does not exist'  
 self.salespersonid = (input("Employee ID: "))  
 tempdict['Salesperson id'] = self.salespersonid  
 *while* chr == 'Y':  
 prcode = raw\_input('Product Code: ')  
 d1 = self.existancecheckercode(prcode)  
 *if* d1[0] == True:  
 *print* "Price: ", globalinventorydatabase[d1[1]]['SP']  
 quantity = input('Quantity: ')  
 *if* self.quantitychecker(d1[1],quantity) == True:  
 l5.append(d1[1])  
 price = globalinventorydatabase[d1[1]]['SP']  
 l4.append(d1[1])  
 l1.append(prcode)  
 l2.append(quantity)  
 l3.append(price)  
 l6.append(globalinventorydatabase[d1[1]]['Description'])  
 *else*:  
 *print* 'Stock going negative'  
 self.reedit()  
 *if* self.reedit() == 2:  
 self.entersales()  
 *else*:  
 *print* 'The product does not exist'  
 *print* '1. To go to main menu\n2. To go to enter customer information again\n3. To continue entering product names'  
 choice = input('Choice(1/2/3) : ')  
 flag = 0  
 *while* choice != 1 *or* choice != 2 *or* choice != 3:  
 *if* choice == 1:  
 self.menu()  
 *elif* choice == 2:  
 self.entersales()  
 *elif* choice == 3:  
 flag = 1  
 *break* choice = input()  
 *if* flag == 1:  
 *continue* chr = raw\_input('Enter more sales? (Y/N): ')  
 *print* self.shipping = 0  
 *print* '1. To Enter Shipping and Handling Charges\n2. To not enter Shipping and Handling Charges'  
 choice1 = input('Choice(1/2): ')  
 *if* choice1 == 1:  
 self.shipping = input('Enter shipping and handling charges: ')  
 tempdict['Customer Information'] = tdict  
 tempdict['Product Code'] = l1  
 tempdict['Product Name'] = l5  
 tempdict['Description'] = l6  
 tempdict['Quantity'] = l2  
 tempdict['Price'] = l3  
 totalprice = 0  
 temp = globalcompanydatabase['Customers'][customername]['Salesrecord']  
 temp['Bills'].append(self.count)  
 *for* x *in* range(len(l1)):  
 totalprice += l3[x] \* l2[x]  
 tempdict['Total amount of sales'] = totalprice  
 d[self.count] = tempdict  
 count1 = 0  
 temp['Total sales'] += totalprice + (totalprice \* 2 / 100) + self.shipping  
 *for* i *in* l4:  
 globalinventorydatabase[i]['Quantity'] -= l2[count1]  
 count1 += 1  
 tempdict['Date'] = self.today  
 pickle.dump(globalinventorydatabase, open("Databases/inventorydatabase.db", "wb"))  
 pickle.dump(globalcompanydatabase, open("Databases/companymanagement.db", "wb"))  
 *print* 'Database Synced'  
 *print  
 print* 'Stocks left after Sales :-'  
 *for* x *in* range(len(l4)):  
 *print* 'Product Name:',l4[x]  
 *print* 'Stocks:',globalinventorydatabase[l4[x]]['Quantity']  
 *print* self.billgeneration()  
  
 ############################################################################################  
 #############################################################################################  
  
 *def* billgeneration(self):  
 globalcompanydatabase = pickle.load(open("Databases/companymanagement.db", "rb"))  
 d = globalcompanydatabase['sales'][self.count]  
 ifile = open("InvoiceTemplate/InvoiceTemplate.txt","r")  
 str1 = ifile.readlines()  
 ifile.close()  
 string = ''  
 *for* x *in* range(3):  
 string += str(str1[x])  
 string += d['Customer Information']['Name']  
 *for* x *in* range(57 - len(globalcompanydatabase['sales'][self.count]['Customer Information']['Name'])):  
 string += ' '  
 string += str1[3][57:len(str1[3])+1]  
  
 string += d['Customer Information']['Address']  
 *for* x *in* range(57 - len(d['Customer Information']['Address'])):  
 string += ' '  
 string += str1[4][57:len(str1[4])+1]  
  
 string += d['Customer Information']['City']  
 *for* x *in* range(57 - len(globalcompanydatabase['sales'][self.count]['Customer Information']['City'])):  
 string += ' '  
 string += str1[5][57:len(str1[5])+1]  
  
 string += d['Customer Information']['Phone Number']  
 *for* x *in* range(57 - len(globalcompanydatabase['sales'][self.count]['Customer Information']['Phone Number'])):  
 string += ' '  
 string += str1[6][57:len(str1[6])+1]  
 string += d['Customer Information']['Email']  
 *for* x *in* range(57 - len(d['Customer Information']['Email'])):  
 string += ' '  
 string += str1[7][57:len(str1[7])+1]  
 string += str1[8]  
 string += str1[9]  
 string += 'Invoice Number : ' + str(self.count) + '\n'  
 string += 'Date : ' + self.today + '\n'#str[11]  
 *for* x *in* range(12,16):  
 string += str1[x]  
 l = []  
 l.append(string)  
 totalamount = 0  
 *for* x *in* range(len(d['Price'])):  
 flag = 0  
 string = ''  
 string += ' '  
 string += str(x+1)  
 *for* y *in* range(3):  
 string += ' '  
 *for* z *in* range(6):  
 string += ' '  
 string += str(d['Product Code'][x])  
 *for* c *in* range(10 - len(str(d['Product Code'][x])) ):  
 string += ' '  
  
 string100 = []  
 *if* len(d['Description'][x]) <= 36:  
  
 string += str(d['Description'][x])  
 *for* abcd *in* range(42 - len(str(d['Description'][x]))):  
 string += ' '  
  
 *else*:  
 string100 = d['Description'][x].split()  
 string102 = string100[:]  
 string101 = ''  
 *for* xy *in* range(len(string102)):  
 *if* len(string101 + string102[xy]) >= 36:  
 string += string101  
 string101 = ''  
 *break  
 else*:  
  
 string101 += string102[xy] + ' '  
 string100.remove(string102[xy])  
 string += ' ' \* 7  
 flag = 1  
  
 string += str(d['Quantity'][x])  
 *for* ab *in* range(3 - len(str(d['Quantity'][x])) + 8):  
 string += ' '  
 string += str(d['Price'][x])  
 *for* abc *in* range(14 - len(str(d['Price'][x])) + 4):  
 string += ' '  
 totalprice = d['Quantity'][x] \* d['Price'][x]  
 string += str(totalprice)  
 totalamount += totalprice  
 string += '\n'  
 *if* flag != 1:  
 l.append(string)  
 l.append('\n')  
 *else*:  
 l.append(string)  
 string101 = ''  
 *for* x *in* string100:  
 *if* len(string101 + x) <= 36:  
 string101 += x + ' '  
 *else*:  
 l.append(' '\*23)  
 l.append(string101 + '\n')  
 string101 = x + ' '  
 l.append(' '\*23)  
 l.append(string100[-1] + '\n')  
 l.append('\n')  
  
 *for* hello *in* range(15,17):  
 l.append(str1[hello])  
  
 string = ''  
 string += str1[17][0:94]  
 string += str(totalamount) + '\n'  
 l.append(string)  
 string = ''  
 string += str(str1[18][0:94])  
 string += str(totalamount \* 2/100)  
 string += '\n'  
 l.append(string)  
 string = ''  
 choice1 = 0  
 *if* self.shipping != 0:  
 string += ' ' \* 77 + 'Shipping : Rs.'  
 string += str(self.shipping) + '\n'  
 l.append(string)  
 l.append(str1[19])  
 string = ''  
 string += str(str1[20][0:94])  
 string += str(totalamount + (totalamount \* 2/100) + self.shipping)  
 string += '\n'  
 l.append(string)  
 l.append(str1[21])  
 *for* x *in* range(22,len(str1)):  
 l.append(str1[x])  
 ifile = open("BillsGenerated/hello2.txt","w")  
 ifile.writelines(l)  
 ifile.close()  
 os.rename('BillsGenerated/hello2.txt',"BillsGenerated/Bill Number- "+ str(self.count)+".txt")  
 emailit(globalcompanydatabase['sales'][self.count]['Customer Information']['Email'],self.count)  
 *print  
 print* 'Bill Generated'  
 self.menu()  
  
 #############################################################################################  
 ############################################################################################  
  
 *def* ç(self):  
 *print* '\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Edit Formation \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_'  
 *print* globalinventorydatabase = pickle.load(open("Databases/inventorydatabase.db", "rb"))  
 *print* '1. to edit code\n2. to edit product name\n3. to edit selling price\n4. to edit VAT\n5. to edit product description'  
 *print* choice8 = input('Choice(1/2/3/4/5): ')  
 *if* choice8 == 1:  
 tempcode = raw\_input('Enter original product code: ')  
 d = self.existancecheckercode(tempcode)  
 *if* d[0] == True:  
 newtempcode = raw\_input('Enter the new code: ')  
 newtempcode1 = raw\_input('Enter the new code: ')  
 d1 = self.existancecheckercode(newtempcode)  
 *if* d1[0] == False:  
 *if* newtempcode == newtempcode1:  
 d2 = globalinventorydatabase[d[1]]  
 d2['Code'] = newtempcode  
 *print* 'Code successfully changed'  
 pickle.dump(globalinventorydatabase, open("Databases/inventorydatabase.db", "wb"))  
 *else*:  
 *print* 'Two entries do not match'  
  
 *if* self.reedit() == 2:  
 self.editinformation()  
 *else*:  
 *print* 'Entries do not match'  
  
 *if* self.reedit() == 2:  
 self.editinformation()  
  
 *else*:  
 *print* 'Product code does not exist'  
  
 *if* self.reedit() == 2:  
 self.editinformation()  
  
 *elif* choice8 == 2:  
 tempname = raw\_input('Enter original product name: ')  
 d = self.existancecheckername(tempname)  
 *if* d[0] == True:  
 newtempname = raw\_input('Enter new name: ')  
 newtempname1 = raw\_input('Reenter new name: ')  
 *if* newtempname1 == newtempname:  
 i = d[1]  
 tempdict = {}  
 tempdict = globalinventorydatabase[i]  
 *del* globalinventorydatabase[i]  
 globalinventorydatabase.update(tempdict)  
 *print* 'Name successfully changed'  
 pickle.dump(globalinventorydatabase, open("Databases/inventorydatabase.db", "wb"))  
 *else*:  
 *print* 'Entries do not match'  
  
 *if* self.reedit() == 2:  
 self.editinformation()  
  
 *else*:  
 *print* 'Name entered does not exist'  
  
 *if* self.reedit() == 2:  
 self.editinformation()  
  
 *elif* choice8 == 3:  
 globalinventorydatabase = pickle.load(open("Databases/inventorydatabase.db", "rb"))  
 *print* '1. to edit using code\n2. to edit using product name'  
 *print* choice9 = input('Choice(1/2): ')  
 *if* choice9 == 1:  
 tempcode = raw\_input('Enter original product code: ')  
 d = self.existancecheckercode(tempcode)  
 *if* d[0] == True:  
 tempsp = input('Enter new selling price: ')  
 tempsp1 = input('Reenter new selling price: ')  
 *print  
 if* tempsp == tempsp1:  
 globalinventorydatabase[d[1]]['SP'] = tempsp  
 pickle.dump(globalinventorydatabase, open("Databases/inventorydatabase.db", "wb"))  
 *print* 'Selling price successfully changed'  
 *print* "New Selling Price: Rs.",tempsp  
 *else*:  
 *print* "Entries do not match"  
  
 *if* self.reedit() == 2:  
 self.editinformation()  
 *else*:  
 'Product with entered code does not exist'  
  
 *if* self.reedit() == 2:  
 self.editinformation()  
  
 *elif* choice9 == 2:  
 tempname = raw\_input('Enter product name: ')  
 *if* tempname *in* globalinventorydatabase:  
 *print* 'Current selling price :',globalinventorydatabase[tempname]['SP']  
 tempsp = input('Enter new selling price: ')  
 tempsp1 = input('Reenter new selling price: ')  
 *if* tempsp == tempsp1:  
 globalinventorydatabase[tempname]['SP'] = tempsp  
 pickle.dump(globalinventorydatabase, open("Databases/inventorydatabase.db", "wb"))  
 *print* 'Selling price successfully changed'  
 *else*:  
 *print* 'Entries do not match'  
  
 *if* self.reedit() == 2:  
 self.editinformation()  
 *else*:  
 *print* 'The entered product name does not exist'  
 self.reedit()  
 *if* self.reedit() == 2:  
 self.editinformation()  
 *else*:  
 *print* 'Wrong choice entered'  
 self.reedit()  
 *if* self.reedit() == 2:  
 self.editinformation()  
  
 *elif* choice8 == 4:  
 globalinventorydatabase = pickle.load(open("Databases/inventorydatabase.db", "rb"))  
 *print* '1. to edit using code\n2. to edit using product name'  
 *print* choice9 = input('Choice(1/2): ')  
 *if* choice9 == 1:  
 tempcode = raw\_input('Enter original product code: ')  
 d = self.existancecheckercode(tempcode)  
 *if* d[0] == True:  
 *print* 'Current VAT :',globalinventorydatabase[d[1]]['VAT'],'%'  
 tempvat = input('Enter new VAT: ')  
 tempvat1 = input('Reenter new VAT: ')  
 *print  
 if* tempvat == tempvat1:  
 globalinventorydatabase[d[1]]['VAT'] = tempvat  
 pickle.dump(globalinventorydatabase, open("Databases/inventorydatabase.db", "wb"))  
 *print* 'VAT successfully changed'  
 *print* "New VAT",tempvat,'%'  
 *else*:  
 *print* "Entries do not match"  
  
 *if* self.reedit() == 2:  
 self.editinformation()  
 *else*:  
 'Product with entered code does not exist'  
  
 *if* self.reedit() == 2:  
 self.editinformation()  
 *elif* choice9 == 2:  
 tempname = raw\_input('Enter product name: ')  
 *if* tempname *in* globalinventorydatabase:  
 *print* 'VAT :',globalinventorydatabase[tempname]['VAT']  
 tempvat = input('Enter new VAT: ')  
 tempvat1 = input('Reenter new VAT: ')  
 *if* tempvat == tempvat1:  
 globalinventorydatabase[tempname]['VAT'] = tempvat  
 pickle.dump(globalinventorydatabase, open("Databases/inventorydatabase.db", "wb"))  
 *print* 'VAT successfully changed'  
 *else*:  
 *print* 'Entries do not match'  
  
 *if* self.reedit() == 2:  
 self.editinformation()  
 *else*:  
 *print* 'The entered product name does not exist'  
  
 *if* self.reedit() == 2:  
 self.editinformation()  
 *else*:  
 *print* 'Wrong choice entered'  
  
 *if* self.reedit() == 2:  
 self.editinformation()  
  
 *elif* choice8 == 5:  
 globalinventorydatabase = pickle.load(open("Databases/inventorydatabase.db", "rb"))  
 *print* '1. to edit using code\n2. to edit using product name'  
 *print* choice9 = input('Choice(1/2): ')  
 *if* choice9 == 1:  
 tempcode = raw\_input('Enter original product code: ')  
 d = self.existancecheckercode(tempcode)  
 *if* d[0] == True:  
 *print* 'Current Description :',globalinventorydatabase[d[1]]['Description']  
 tempdes = raw\_input('Enter new description: ')  
 tempdes1 = raw\_input('Reenter new description: ')  
 *print  
 if* tempdes == tempdes1:  
 globalinventorydatabase[d[1]]['Description'] = tempdes  
 pickle.dump(globalinventorydatabase, open("Databases/inventorydatabase.db", "wb"))  
 *print* 'Descriptione successfully changed'  
 *print* "New Selling Price : Rs.",tempdes  
 *else*:  
 *print* "Entries do not match"  
  
 *if* self.reedit() == 2:  
 self.editinformation()  
 *else*:  
 'Product with entered code does not exist'  
  
 *if* self.reedit() == 2:  
 self.editinformation()  
  
 *elif* choice9 == 2:  
 tempname = raw\_input('Enter product name: ')  
 *if* tempname *in* globalinventorydatabase:  
 *print* 'Current Description :',globalinventorydatabase[tempname]['Description']  
 tempdes = raw\_input('Enter new Description: ')  
 tempdes1 = raw\_input('Reenter new Description: ')  
 *if* tempdes == tempdes1:  
 globalinventorydatabase[tempname]['Description'] = tempdes  
 pickle.dump(globalinventorydatabase, open("Databases/inventorydatabase.db", "wb"))  
 *print* 'Description successfully changed'  
 *else*:  
 *print* 'Entries do not match'  
  
 *if* self.reedit() == 2:  
 self.editinformation()  
 *else*:  
 *print* 'The entered product name does not exist'  
  
 *if* self.reedit() == 2:  
 self.editinformation()  
 *else*:  
 *print* 'Wrong choice entered'  
  
 *if* self.reedit() == 2:  
 self.editinformation()  
 *else*:  
 *print* 'Wrong submenu choice entered'  
  
 *if* self.reedit() == 2:  
 self.editinformation()  
  
 pickle.dump(globalinventorydatabase, open("Databases/inventorydatabase.db", "wb"))  
 self.menu()  
  
 ############################################################################################  
 #############################################################################################  
  
 *def* inventorysearch(self):  
 *print* '\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Inventory Search \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_'  
 *print* globalinventorydatabase = pickle.load(open("Databases/inventorydatabase.db", "rb"))  
 *print* '1. to search with product name\n2. to search with product code\n3. to search with company name'  
 choice2 = input('Choice(1/2/3) : ')  
 *print  
 if* choice2 == 1:  
 choice3 = raw\_input('Enter name of product: ')  
 *print  
 if* choice3 *in* globalinventorydatabase:  
 *print* 'Name :',choice3  
 *for* x *in* globalinventorydatabase[choice3] :  
 *print* x,' ' \* (11 - len(x)),':', globalinventorydatabase[choice3][x]  
 *else*:  
 *print* 'Product does not exist'  
  
 *elif* choice2 == 2:  
 choice4 = raw\_input('Enter the product code: ')  
 d = self.existancecheckercode(choice4)  
 *if* d[0] == True:  
 *print* 'Name :',d[1]  
 *for* x *in* globalinventorydatabase[d[1]]:  
 *print* x,' ' \* (11 - len(x)),':',globalinventorydatabase[d[1]][x]  
  
 *elif* choice2 == 3:  
 choice7 = raw\_input('Enter name of the company: ')  
 *print* flag = 0  
 *for* x *in* globalinventorydatabase:  
 tedict = globalinventorydatabase[x]  
 *if* tedict['Company'] == choice7:  
 flag = 1  
 *print  
 print* 'Name :',x  
 *for* y *in* tedict:  
 *print* y,' ' \* (11 - len(y)),':',tedict[y]  
 *if* flag == 0:  
 *print* "Company does not exist"  
 *if* self.reedit() == 2:  
 self.inventorysearch()  
 *else*:  
 *print* 'Wrong submenu choice entered'  
 *if* self.reedit() == 2:  
 self.inventorysearch()  
 self.menu()  
  
 ############################################################################################  
 #############################################################################################  
  
 *def* inventorylistout(self):  
 *print* '\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Full Inventory \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_'  
 *print* globalinventorydatabase = pickle.load(open("Databases/inventorydatabase.db", "rb"))  
 *for* x *in* globalinventorydatabase:  
 tdict = globalinventorydatabase[x]  
 *print  
 print* 'Name :',x  
 *for* y *in* tdict:  
 *print* y,':',tdict[y]  
  
 self.menu()  
 #############################################################################################  
 #############################################################################################  
  
 *def* billsearch(self):  
 *print* '\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Bill Search \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_'  
 *print* globalcompanydatabase = pickle.load(open("Databases/companymanagement.db", "rb"))  
 billnumber = input('Enter Bill Number : ')  
 *print* d = globalcompanydatabase['sales']  
 *if* billnumber *in* d:  
 *print* 'Bill Number :',billnumber  
 *print* 'Bill Date :',d[billnumber]['Date']  
 *print* 'Salesperson id :',d[billnumber]['Salesperson id']  
 *print  
 print* 'Customer Information :-'  
 *print  
 print* ' Name :',d[billnumber]['Customer Information']['Name']  
 *print* ' Address :',d[billnumber]['Customer Information']['Address']  
 *print* ' City-Pincode :',d[billnumber]['Customer Information']['City']  
 *print* ' Phone Number :',d[billnumber]['Customer Information']['Phone Number']  
 *print* ' Email Address :',d[billnumber]['Customer Information']['Email']  
 *print  
 print* 'Products Sold :-'  
 *print  
 for* x *in* range(len(d[billnumber]['Price'])):  
 *print* ' Product Code :',d[billnumber]['Product Code'][x]  
 *print* ' Product Name :',d[billnumber]['Product Name'][x]  
 *print* ' Product Description :',d[billnumber]['Description'][x]  
 *print* ' Quantity Sold :',d[billnumber]['Quantity'][x]  
 *print* ' Selling Price :','Rs.',d[billnumber]['Price'][x]  
 *print* ' Total price :','Rs.',d[billnumber]['Price'][x] \* d[billnumber]['Quantity'][x]  
 *print  
 print* 'Total amount of sales :','Rs.',d[billnumber]['Total amount of sales'] + (d[billnumber]['Total amount of sales'] \* 2 / 100 )  
 *else*:  
 *print* 'Bill number does not exist'  
 *print* '1. to go back to main menu\n2. Enter bill number again'  
 choice = input('Choice(1/2): ')  
 *if* choice == 1:  
 self.menu()  
 *elif* choice == 2:  
 self.billsearch()  
 *else*:  
 *while* choice *not in* [1,2]:  
 *print* 'Reenter menu choice'  
 choice = input('Choice(1/2): ')  
 self.menu()  
  
 #############################################################################################  
 #############################################################################################  
  
 *def* fullvalue(self):  
 globalinventorydatabase = pickle.load(open("Databases/inventorydatabase.db", "rb"))  
 fullquantity = 0  
 fullprice = 0  
 *for* x *in* globalinventorydatabase:  
 fullquantity += globalinventorydatabase[x]['Quantity']  
 fullprice += (globalinventorydatabase[x]['Quantity'] \* globalinventorydatabase[x]['SP'])  
 *print* 'Value of all stocks is : Rs.',fullprice  
 *print* 'Quantity of stocks is :',fullquantity,'piece[s]'  
  
 #############################################################################################  
 #############################################################################################  
  
 *def* salesdates(self):  
 *print* '\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Sales \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_'  
 *print* d1 = pickle.load(open("Databases/companymanagement.db", "rb"))  
 chr1 = 'y'  
 chr2 = 'y'  
 chr3 = 'y'  
 todate = ''  
 *while* chr3 == 'y':  
 *while* chr1 == 'y':  
 fromdate = raw\_input('Enter date in the format(From) : DD/MM/YYYY : ')  
 *if* fromdate[0:2].isdigit() *and* fromdate[2] == '/' *and* fromdate[3:5].isdigit() *and* fromdate[5] == '/' *and* fromdate[6:10].isdigit():  
 *if* fromdate[0:2] <= '31' *and* fromdate[3:5] <= '12' *and* fromdate[6:-1] <= '2016':  
 chr1 = 'n'  
 *else*:  
 chr1 = 'y'  
 *print* 'Invalid date'  
 *print  
 else*:  
 chr1 = 'y'  
 *print* 'Invalid Date'  
 *print  
  
 while* chr2 == 'y':  
 todate = raw\_input('Enter date in the format(To) : DD/MM/YYYY : ')  
 *if* todate[0:2].isdigit() *and* todate[2] == '/' *and* todate[3:5].isdigit() *and* todate[5] == '/' *and* todate[6:10].isdigit():  
 *if* todate[0:2] <= '31' *and* todate[3:5] <= '12' *and* todate[6:10] <= '2016':  
 *if* todate[6:10] > fromdate[6:10]:  
 chr2 = 'n'  
 chr3 = 'n'  
 *break  
 elif* todate[6:10] == fromdate[6:10]:  
 *if* todate[3:5] > fromdate[3:5]:  
 chr2 = 'n'  
 chr3 = 'n'  
 *break  
 elif* todate[3:5] == fromdate[3:5]:  
 *if* todate[0:2] > fromdate[0:2]:  
 chr2 = 'n'  
 chr3 = 'n'  
 *break  
 else*:  
 *print* 'To date entered if before the after date'  
 *print* '1. To enter from date again\n2. To enter to date again'  
 choice = input('1/2 : ')  
 *if* choice == 1:  
 chr3 = 'y'  
 chr2 = 'y'  
 chr1 = 'y'  
 *elif* choice == 2:  
 chr2 = 'y'  
 *else*:  
 *print* 'To date entered if before the after date'  
 *print* '1. To enter from date again\n2. To enter to date again'  
 choice = input('1/2 : ')  
 *if* choice == 1:  
 chr3 = 'y'  
 chr2 = 'y'  
 chr1 = 'y'  
 *elif* choice == 2:  
 chr2 = 'y'  
 *else*:  
 *print* 'To date entered if before the after date'  
 *print* '1. To enter from date again\n2. To enter to date again'  
 choice = input('1/2 : ')  
 *if* choice == 1:  
 chr3 = 'y'  
 chr2 = 'y'  
 chr1 = 'y'  
 *elif* choice == 2:  
 chr2 = 'y'  
 *else*:  
 chr2 = 'y'  
 *print* 'Invalid date'  
 *print  
 else*:  
 chr2 = 'y'  
 *print* 'Invalid Date'  
 *print* d = d1['sales']  
 totalquantity = 0  
 totalsales = 0  
 billno = []  
 date\_format = "%d/%m/%Y"  
 a = datetime.strptime(fromdate, date\_format)  
 b = datetime.strptime(todate, date\_format)  
 e = datetime.strptime(fromdate, date\_format)  
 delta = b - a  
 count = 0  
 l = []  
 *for* x *in* d:  
 temp = d[x]['Date']  
 c = datetime.strptime(temp, date\_format)  
 delta1 = c - a  
 delta2 = b - c  
 delta3 = e - a  
 *if* delta1.days >= delta3.days *and* delta2.days >= delta3.days:  
 totalsales += (d[x]['Total amount of sales'] \* 102 / 100)  
 totalquantity += sum(d[x]['Quantity'])  
 billno.append(x)  
 *if* d[x]['Total amount of sales'] \* 102 / 100 > count:  
 temp = []  
 l = []  
 temp.append(d[x]['Customer Information']['Name'])  
 temp.append(d[x]['Total amount of sales'] \* 102 / 100)  
 temp.append(x)  
 l.append(temp)  
 *elif* d[x]['Total amount of sales'] \* 102 / 100 == count:  
 temp = []  
 temp.append(d[x]['Customer Information']['Name'])  
 temp.append(d[x]['Total amount of sales'] \* 102 / 100)  
 temp.append(x)  
 l.append(temp)  
 *print* 'Total Quantity Sold :',totalquantity,'piece[s]'  
 *print* 'Total Amount of sales :','Rs.',totalsales  
 *print* 'Average sales per day :','Rs.',totalsales / delta.days  
 *print* 'Bill numbers :',billno[0] , '-' , billno[-1]  
 *print  
 if* len(l) >= 1:  
 *print* 'Highest Purchase[s] :-'  
 *for* x *in* l:  
 *print* ' Bill Number :',x[2]  
 *print* ' Customer Name :',x[0]  
 *print* ' Total Amount : Rs.',x[1]  
 *print* self.menu()  
  
 #############################################################################################  
  
 ############################################################################################  
  
 *def* salescheck(self):  
 *print* '\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Sales \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_'  
 *print* d1 = pickle.load(open("Databases/companymanagement.db", "rb"))  
 chr1 = 'y'  
 date = ''  
 *while* chr1 == 'y':  
 date = raw\_input('Enter date in the format : DD/MM/YYYY : ')  
 *if* date[0:2].isdigit() *and* date[2] == '/' *and* date[3:5].isdigit() *and* date[5] == '/' *and* date[6:10].isdigit():  
 *if* date[0:2] <= '31' *and* date[3:5] <= '12' *and* date[6:-1] <= '2016':  
 chr1 = 'n'  
 *else*:  
 chr1 = 'y'  
 *print* 'Invalid date'  
 *print  
 else*:  
 chr1 = 'y'  
 *print* 'Invalid Date'  
 *print* d = d1['sales']  
 totalquantity = 0  
 totalsales = 0  
 billno = []  
 count = 0  
 l = []  
 *for* x *in* d:  
 *if* d[x]['Date'] == date:  
 totalsales += (d[x]['Total amount of sales'] \* 102 / 100)  
 totalquantity += sum(d[x]['Quantity'])  
 billno.append(x)  
 *if* d[x]['Total amount of sales'] \* 102 / 100 > count:  
 temp = []  
 l = []  
 temp.append(d[x]['Customer Information']['Name'])  
 temp.append(d[x]['Total amount of sales'] \* 102 / 100)  
 temp.append(x)  
 l.append(temp)  
 *elif* d[x]['Total amount of sales'] \* 102 / 100 == count:  
 temp = []  
 temp.append(d[x]['Customer Information']['Name'])  
 temp.append(d[x]['Total amount of sales'] \* 102 / 100)  
 temp.append(x)  
 l.append(temp)  
 *print* 'Total Quantity Sold :',totalquantity,'piece[s]'  
 *print* 'Total Amount of sales :','Rs.',totalsales  
 *if* len(billno) > 1:  
 *print* 'Bill numbers :',billno[0] , '-' , billno[-1]  
 *elif* len(billno) == 1:  
 *print* 'Bill number :',billno[0]  
 *else*:  
 *print* 'Bill numbers : None'  
 *print  
 if* len(l) >= 1:  
 *print* 'Highest Purchase[s] :-'  
 *for* x *in* l:  
 *print* ' Bill Number :',x[2]  
 *print* ' Customer Name :',x[0]  
 *print* ' Total Amount : Rs.',x[1]  
 *print* self.menu()  
 #############################################################################################  
  
 ############################################################################################  
  
 *def* viewcustomerinformation(self):  
 *print* '\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Customer Information\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_'  
 *print* globalcompanydatabase = pickle.load(open("Databases/companymanagement.db", "rb"))  
 name = raw\_input('Enter customer name: ')  
 tempdict = globalcompanydatabase['Customers'][name]  
 *print* 'Name :',name  
 *print* 'Address :',tempdict['address1'] + tempdict['address2']  
 *print* 'City :',tempdict['City']  
 *print* 'Phone Number :',tempdict['Phone Number']  
 *print* 'Email Id :',tempdict['email']  
 *print  
 print* 'Sales Record :-'  
 *print* ' Bill Numbers :',  
 *for* x *in* tempdict['Salesrecord']['Bills']:  
 *print* x,',',  
 *print  
 print* ' Total Purchase :',tempdict['Salesrecord']['Total sales']  
 self.menu()  
  
 #############################################################################################  
#############################################################################################  
  
 *def* existancecheckername(self, *id*):  
 globalinventorydatabase = pickle.load(open("Databases/inventorydatabase.db", "rb"))  
  
 *if id in* globalinventorydatabase:  
 l = []  
 l.append(True)  
 l.append(*id*)  
 *return* l  
 *else*:  
 l = []  
 l.append(False)  
 l.append('Nonexistant')  
 *return* l  
  
 #############################################################################################  
 #############################################################################################  
  
 *def* existancecheckercode(self,*id*):  
 globalinventorydatabase = pickle.load(open("Databases/inventorydatabase.db", "rb"))  
 flag = 0  
 *for* i *in* globalinventorydatabase:  
 id1 = globalinventorydatabase[i]  
 *if id* == id1['Code']:  
 flag = 1  
 l = []  
 l.append(True)  
 l.append(i)  
 *return* l  
  
 *if* flag != 1:  
 l = []  
 l.append(False)  
 l.append('Nonexistant')  
 *return* l  
   
 ############################################################################################  
 *def* reedit(self):  
 *print* '1. to go back to main menu\n2. to go back to submenu'  
 *print* choic = input('Choice(1/2): ')  
 *if* choic == 1:  
 self.menu()  
 *elif* choic == 2:  
 *return* 2  
 ######################################################################################  
 #############################################################################################  
  
 *def* quantitychecker(self,*name*,*quantity*):  
 globalinventorydatabase = pickle.load(open("Databases/inventorydatabase.db", "rb"))  
 *if name in* globalinventorydatabase:  
 *if* globalinventorydatabase[*name*]['Quantity'] >= *quantity*:  
 *return* True  
 *else*:  
 *print* 'Not enough quantity'

EMAIL ENGINE

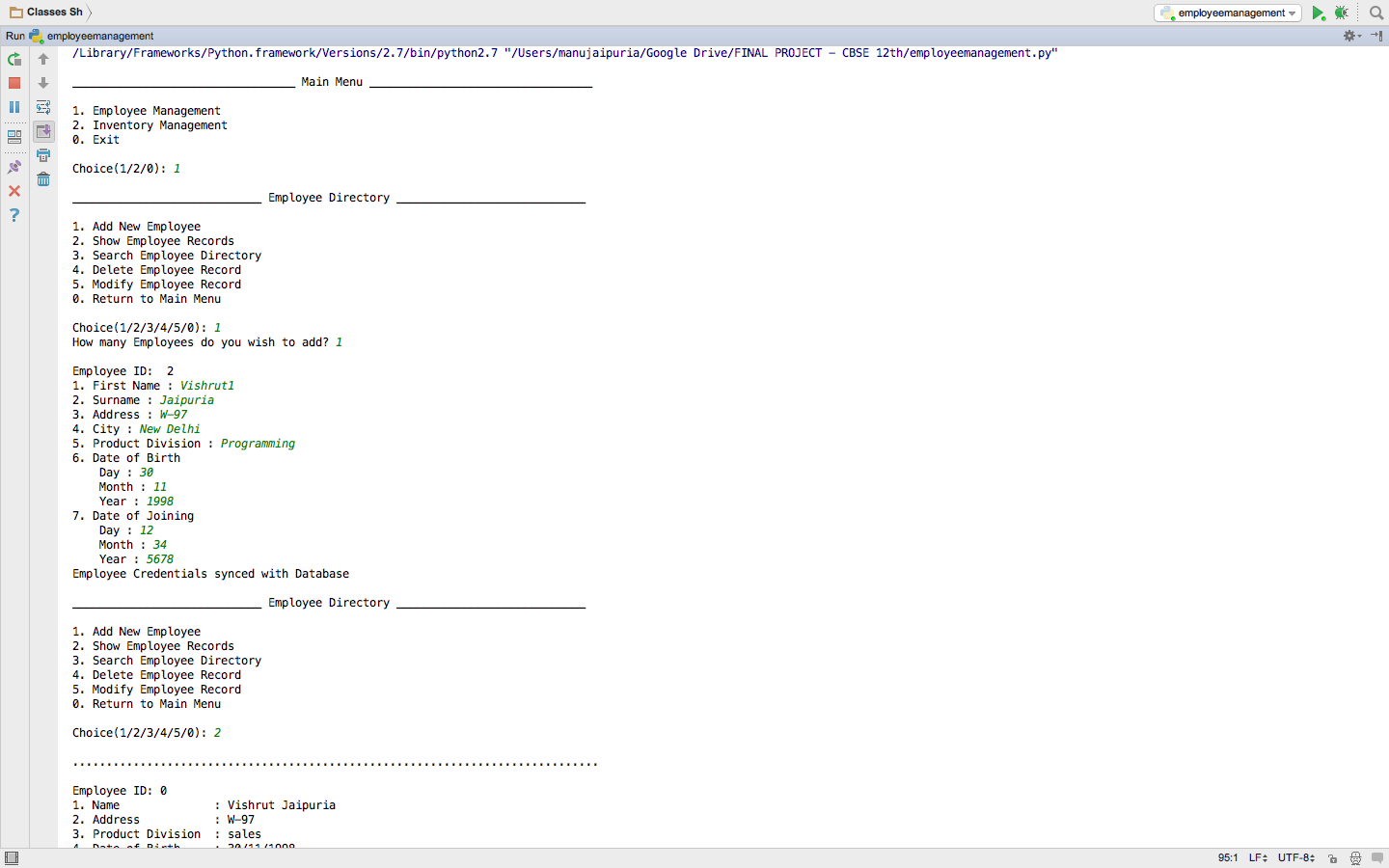
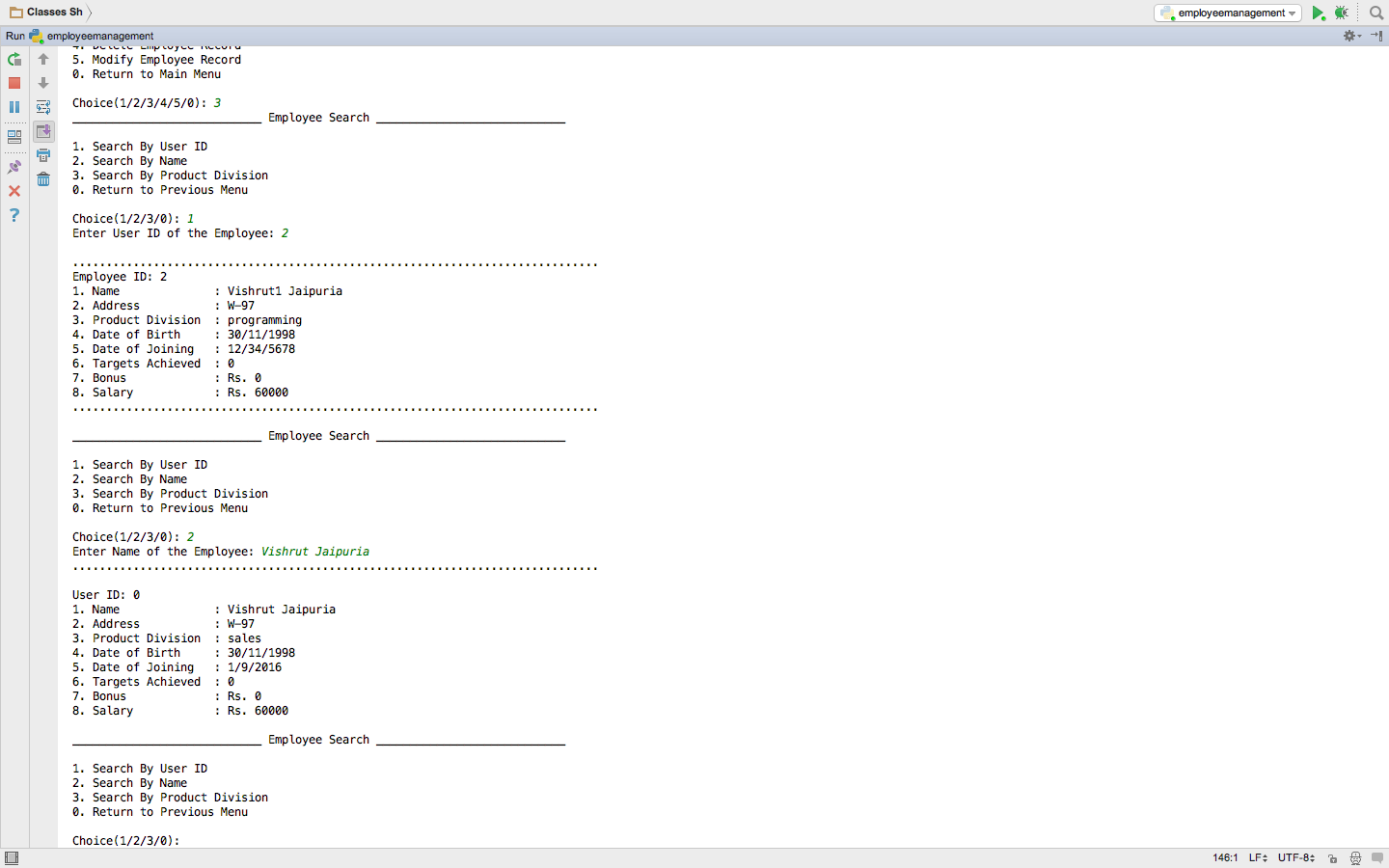
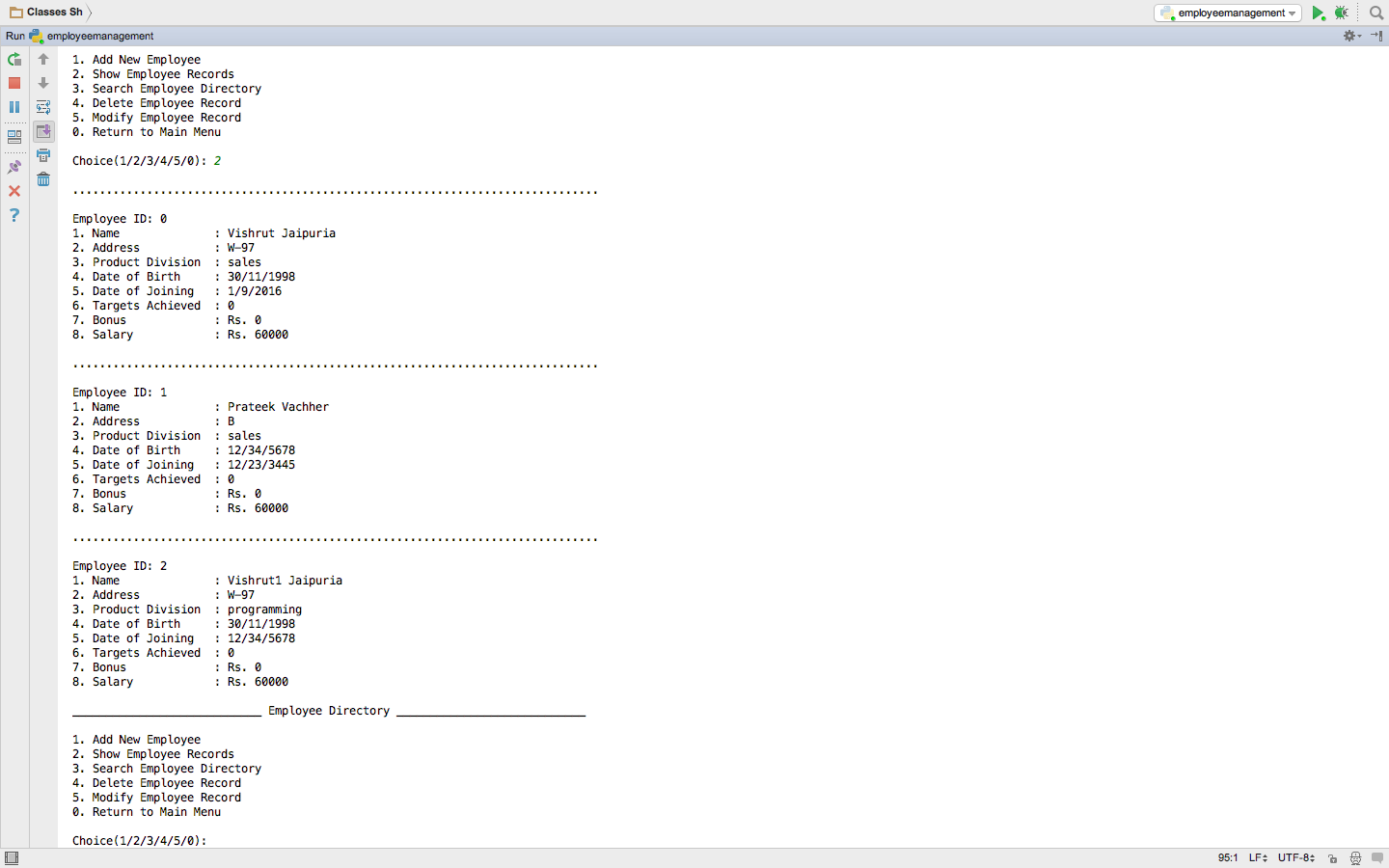
# -\*- coding: iso-8859-1 -\*-  
*def* emailit(*to*,*filename*):  
 *from* email.mime.text *import* MIMEText  
 *from* email.mime.application *import* MIMEApplication  
 *from* email.mime.multipart *import* MIMEMultipart  
 *from* smtplib *import* SMTP  
  
 msg = MIMEMultipart()  
 msg['Subject'] = 'Invoice - Sales Management System'  
 msg['From'] = 'sms@prateekvachher.in'  
 msg['Reply-to'] = 'sms@prateekvachher.in'  
 msg['To'] = *to* # That is what u see if dont have an email reader:  
 msg.preamble = 'Multipart massage.\n'  
  
 # This is the textual part:  
 part = MIMEText("Hello,\nPlease find attached your Invoice for the recent purchase with us. \n\nRegards\nTeam SMS")  
 msg.attach(part)  
  
 # This is the binary part(The Attachment):  
 part = MIMEApplication(open("BillsGenerated/Bill Number- "+str(*filename*)+".txt", "rb").read())  
 part.add\_header('Content-Disposition', 'attachment', filename="Bill Number- "+str(*filename*)+".txt")  
 msg.attach(part)  
  
 # Create an instance in SMTP server  
 smtp = SMTP("host5.dnsinweb.com")  
 # Start the server:  
 smtp.ehlo()  
 smtp.login("sms@prateekvachher.in", "Welcome123")  
  
 # Send the email  
 smtp.sendmail(msg['From'], msg['To'], msg.as\_string())

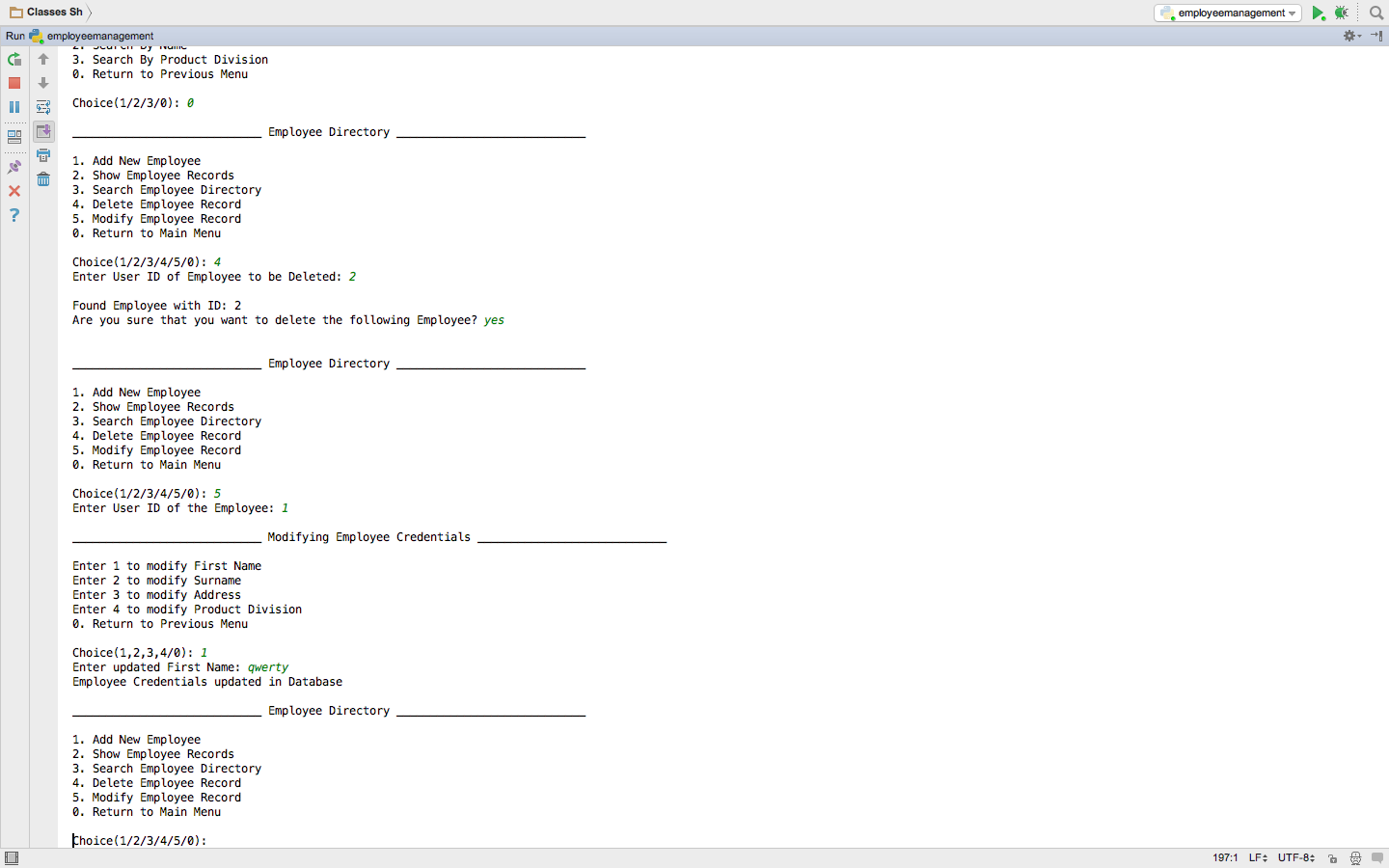
SAMPLE BILL GENERATED

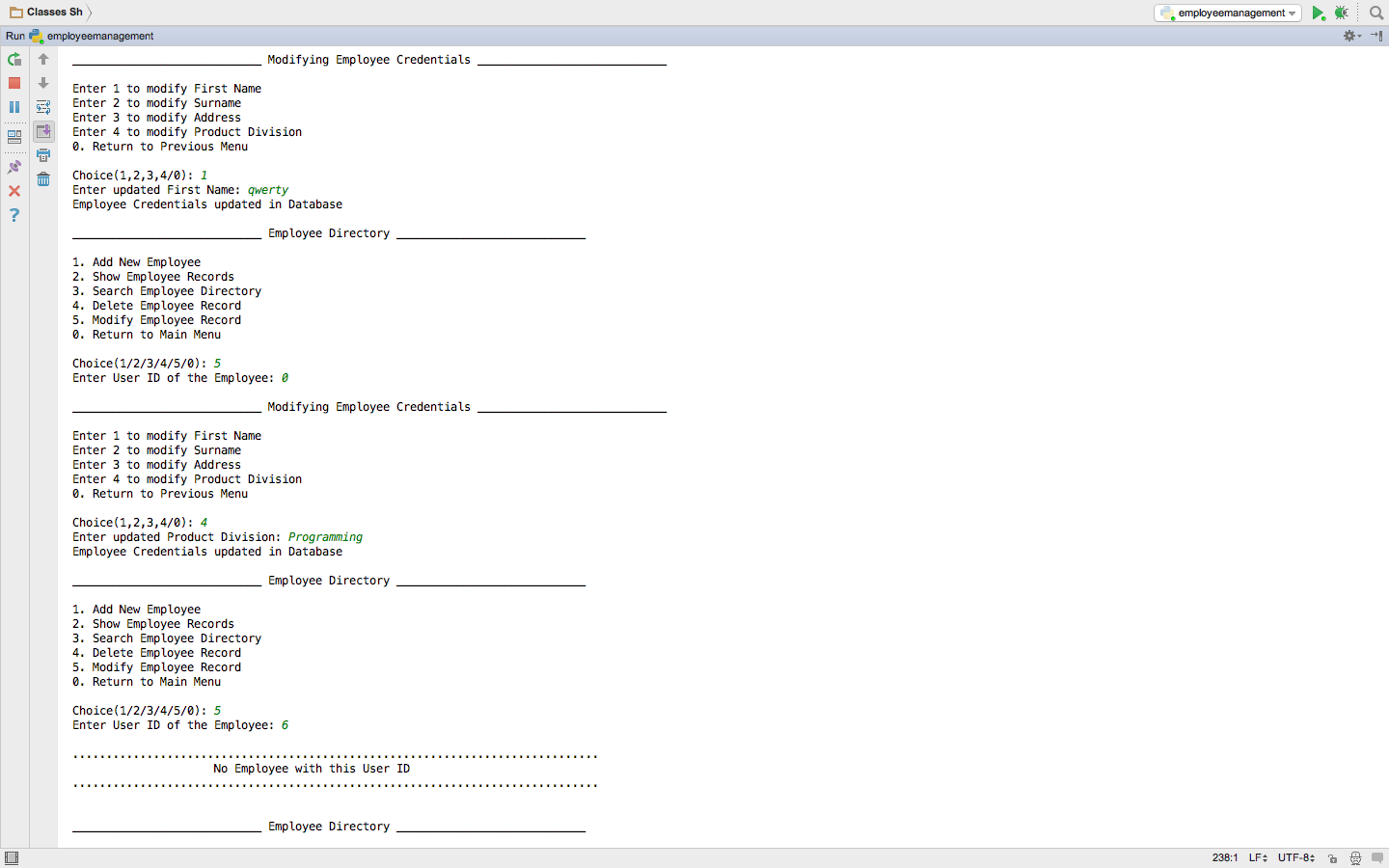


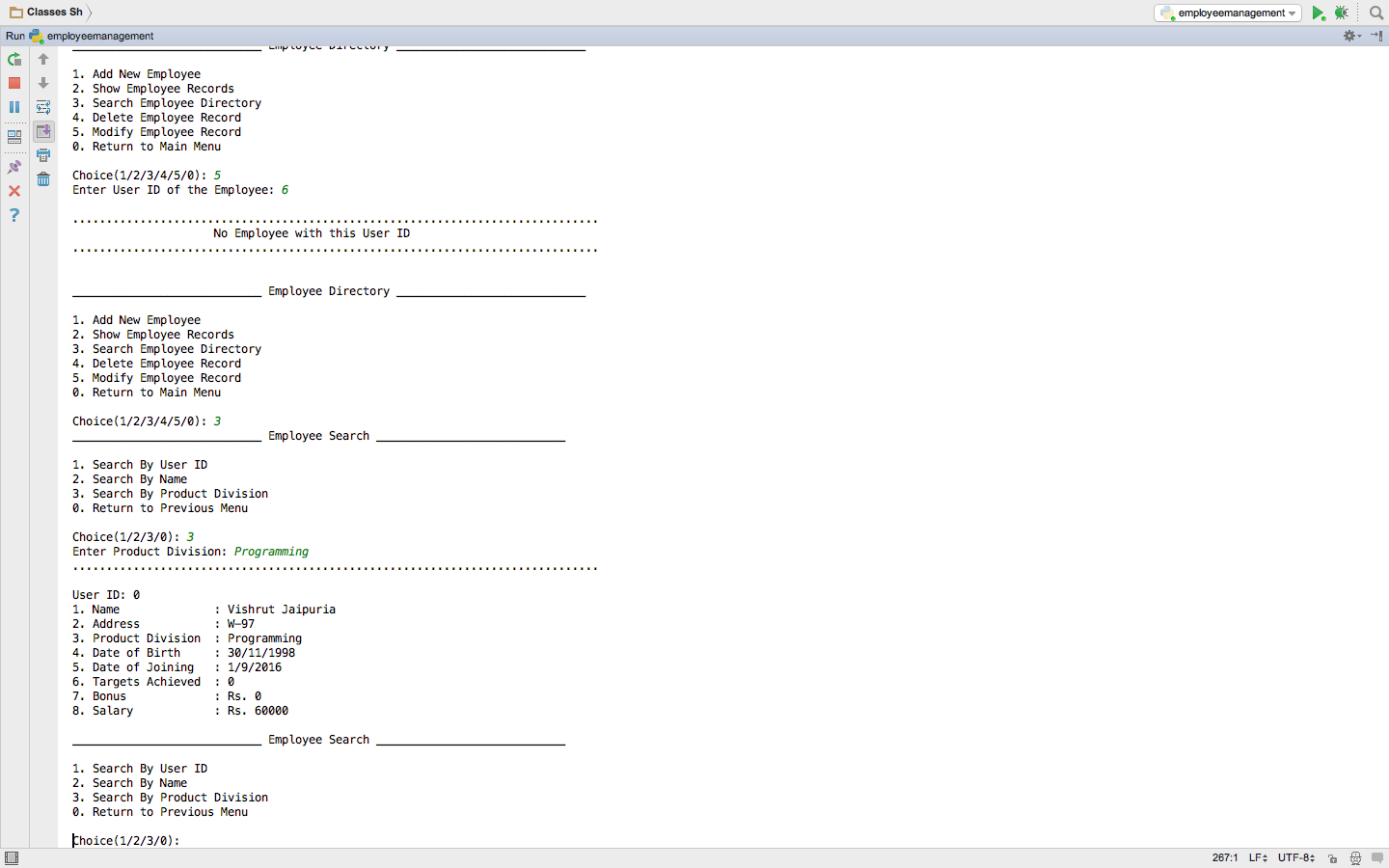
Output

EMPLOYEE MANAGEMENT

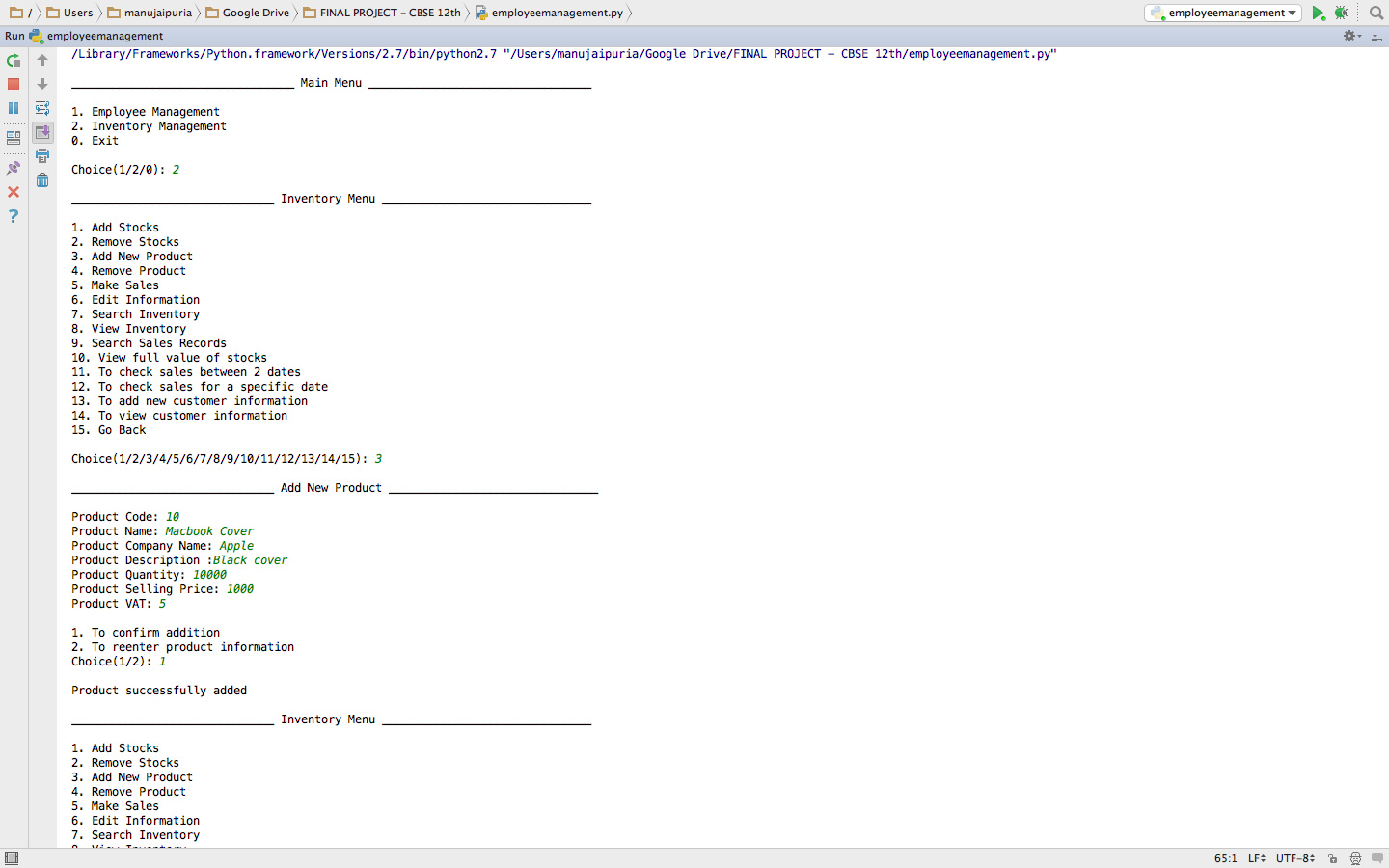
****

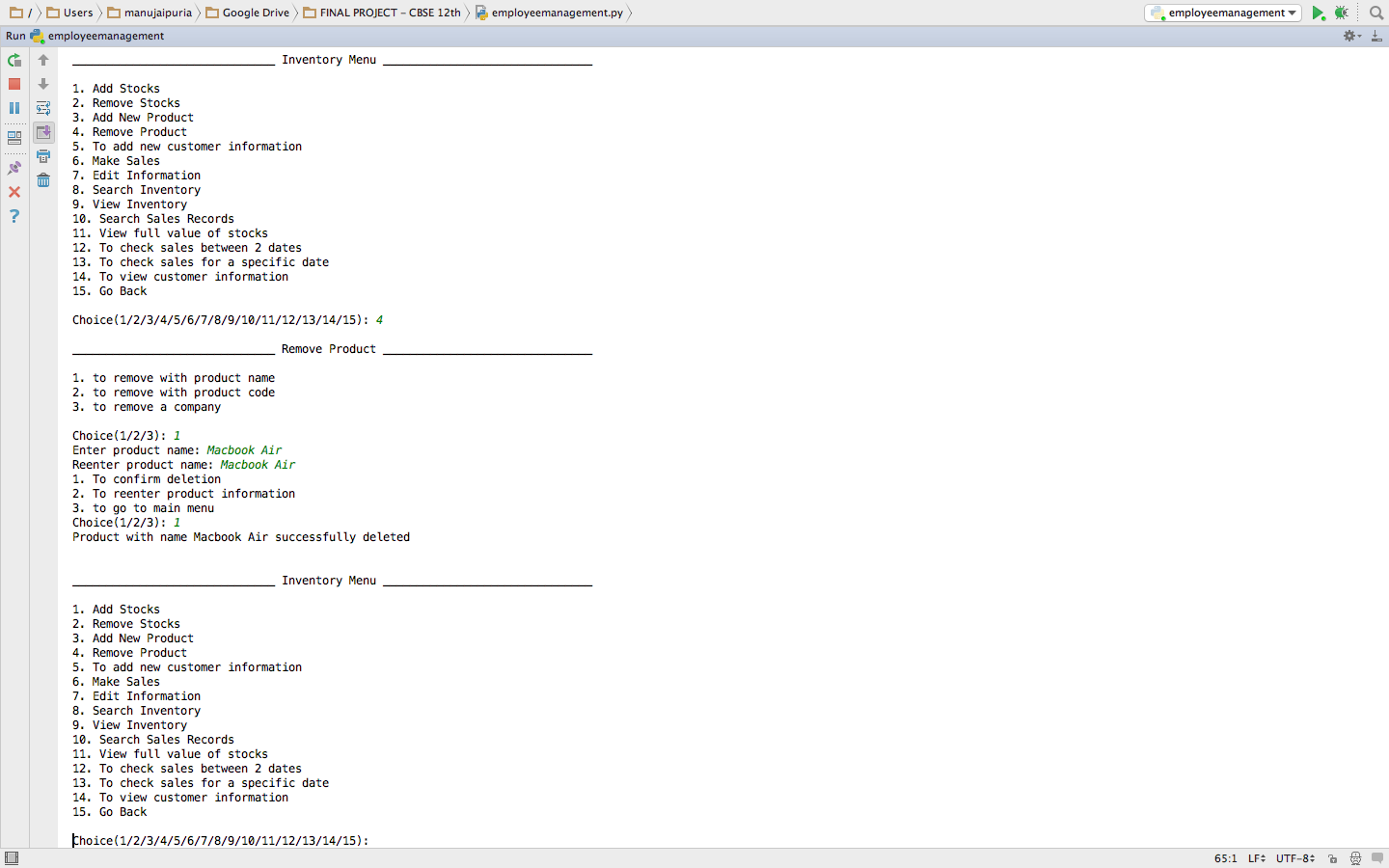


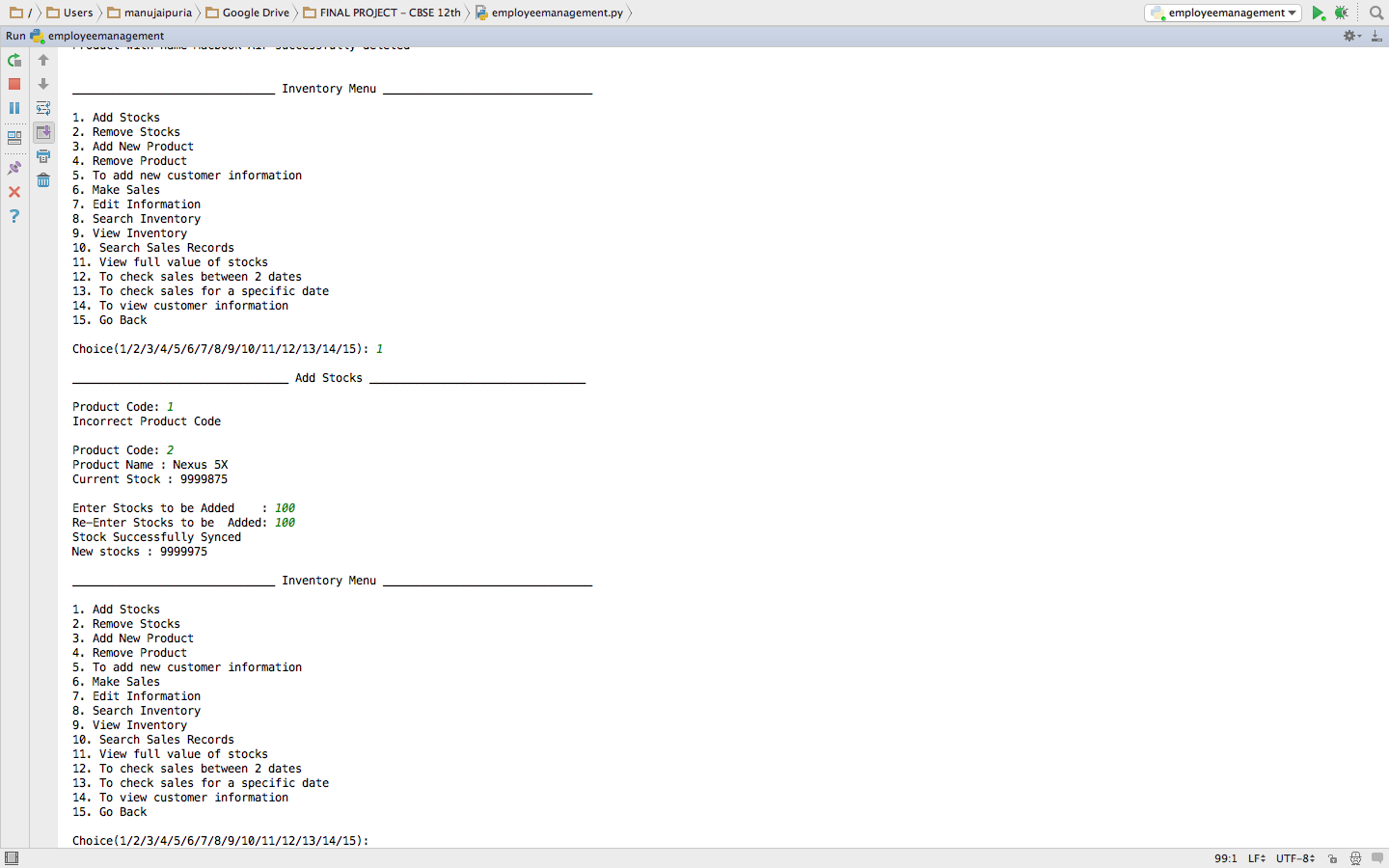


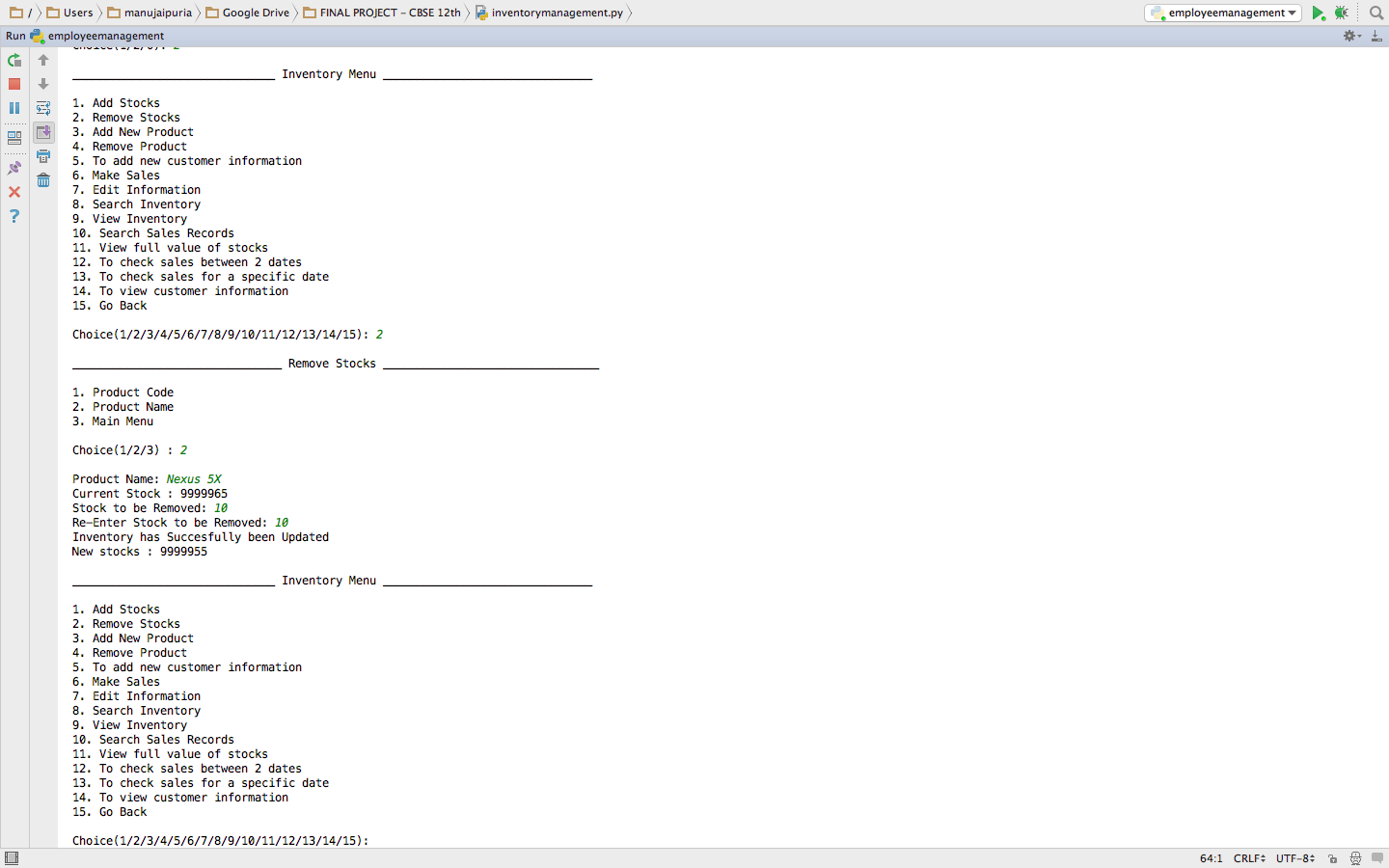


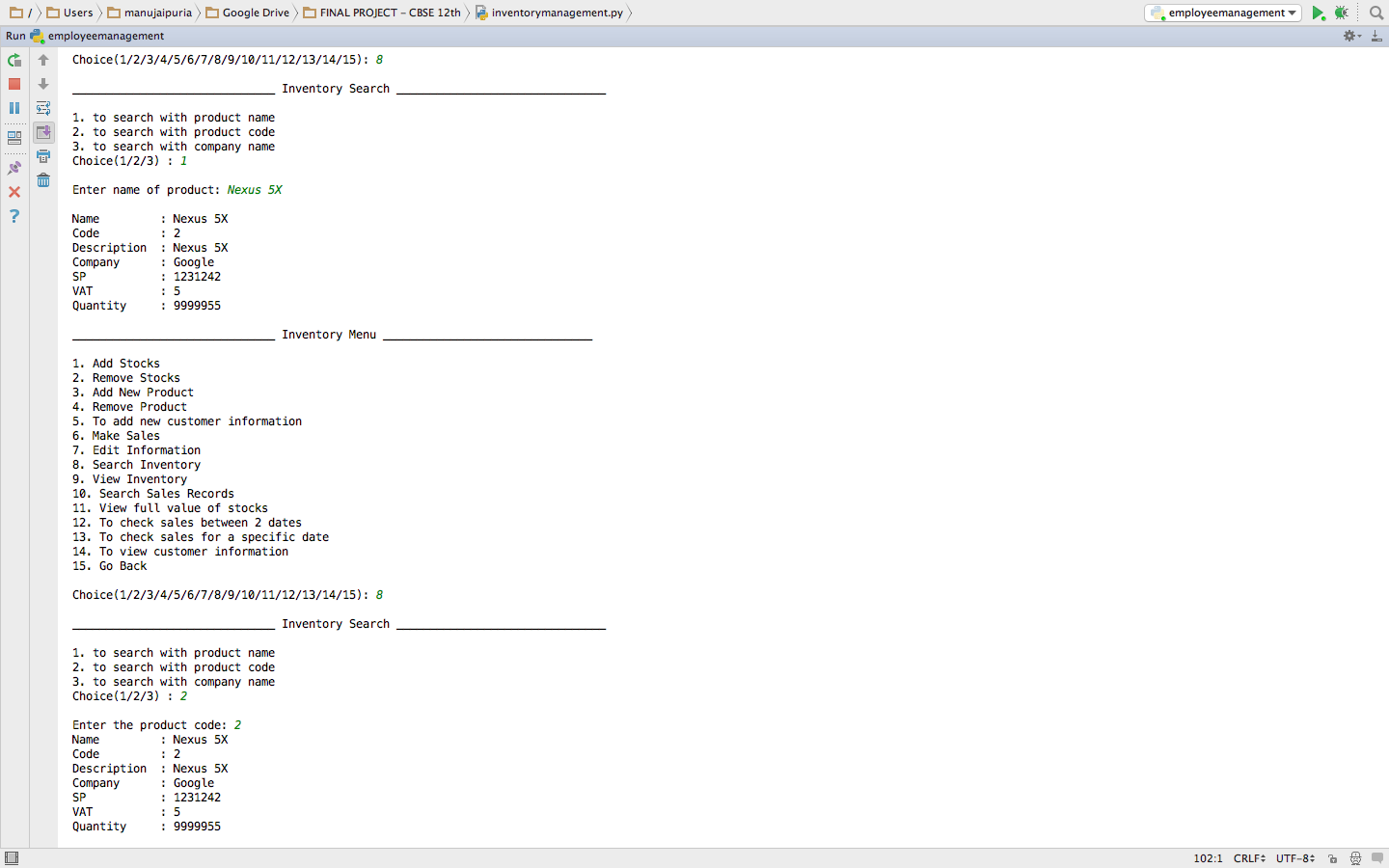
INVENTORY MANAGEMENT

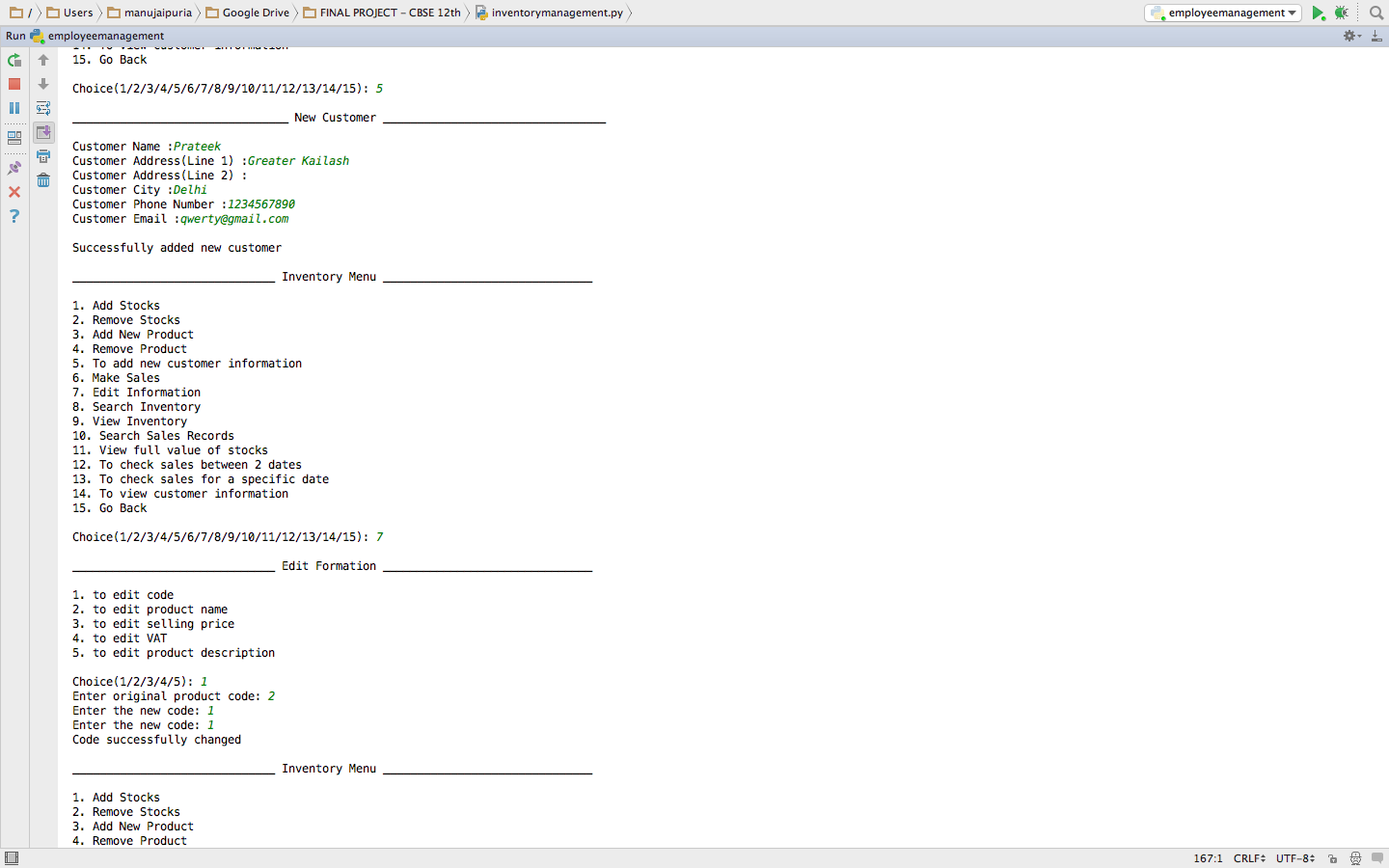


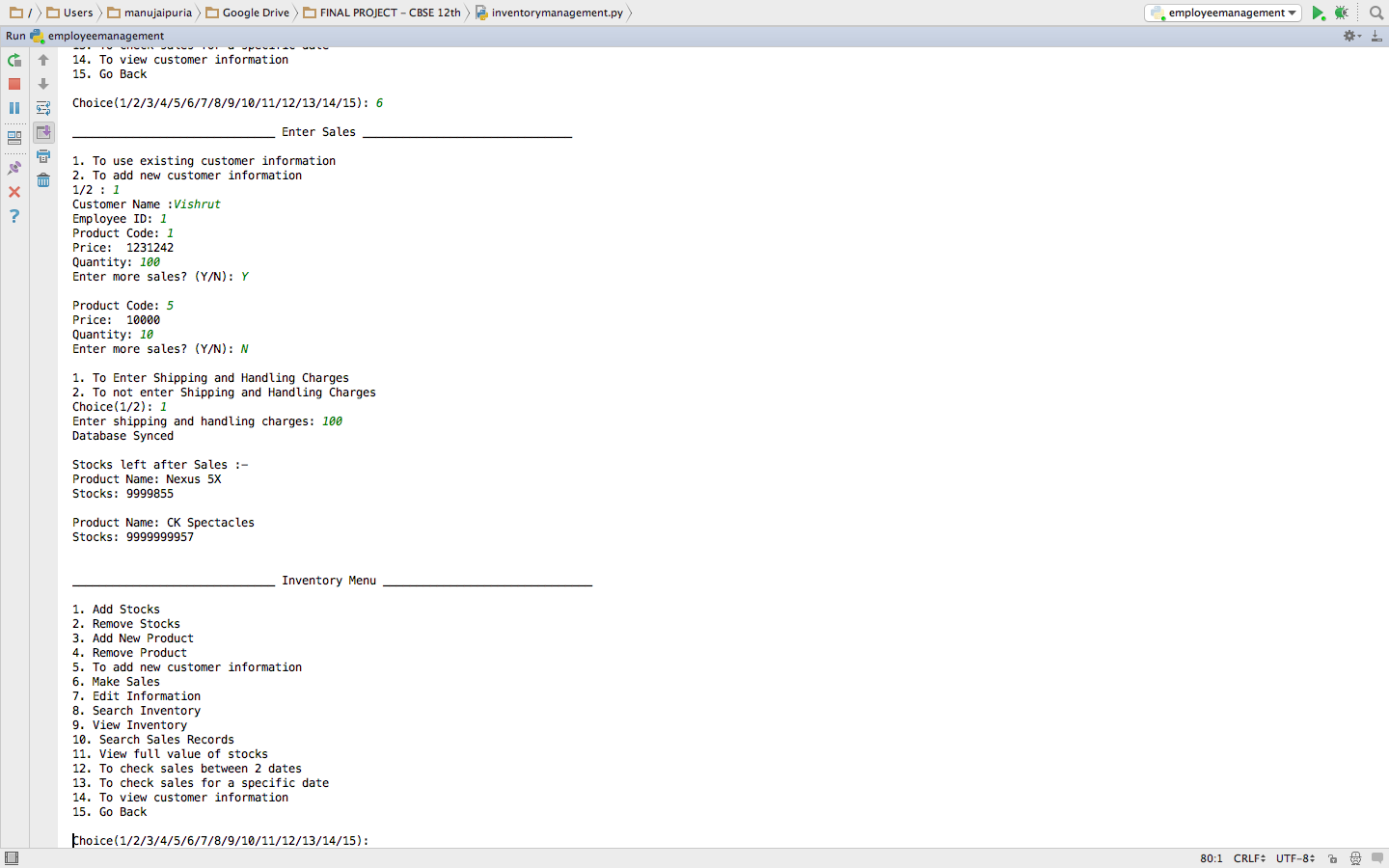


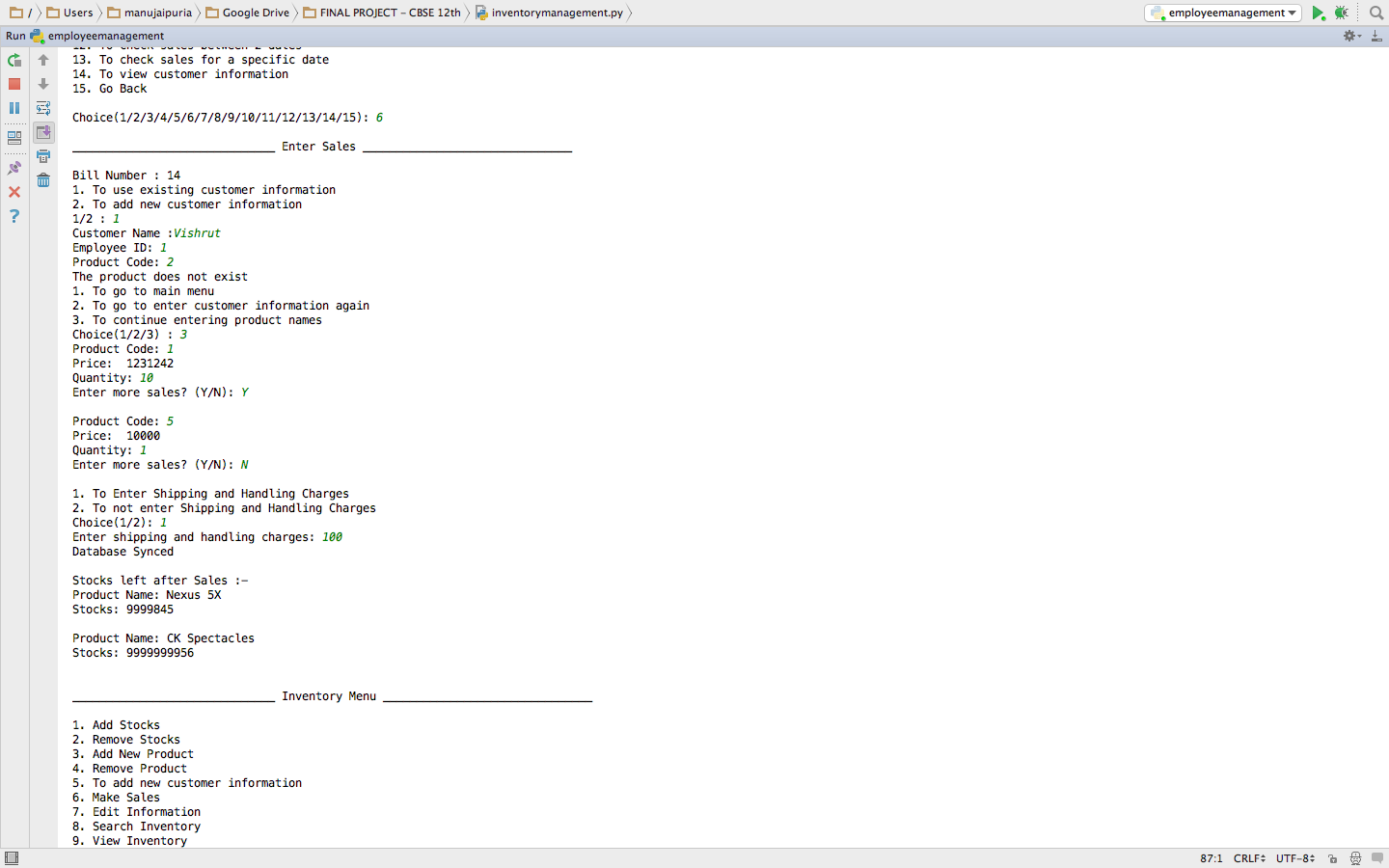


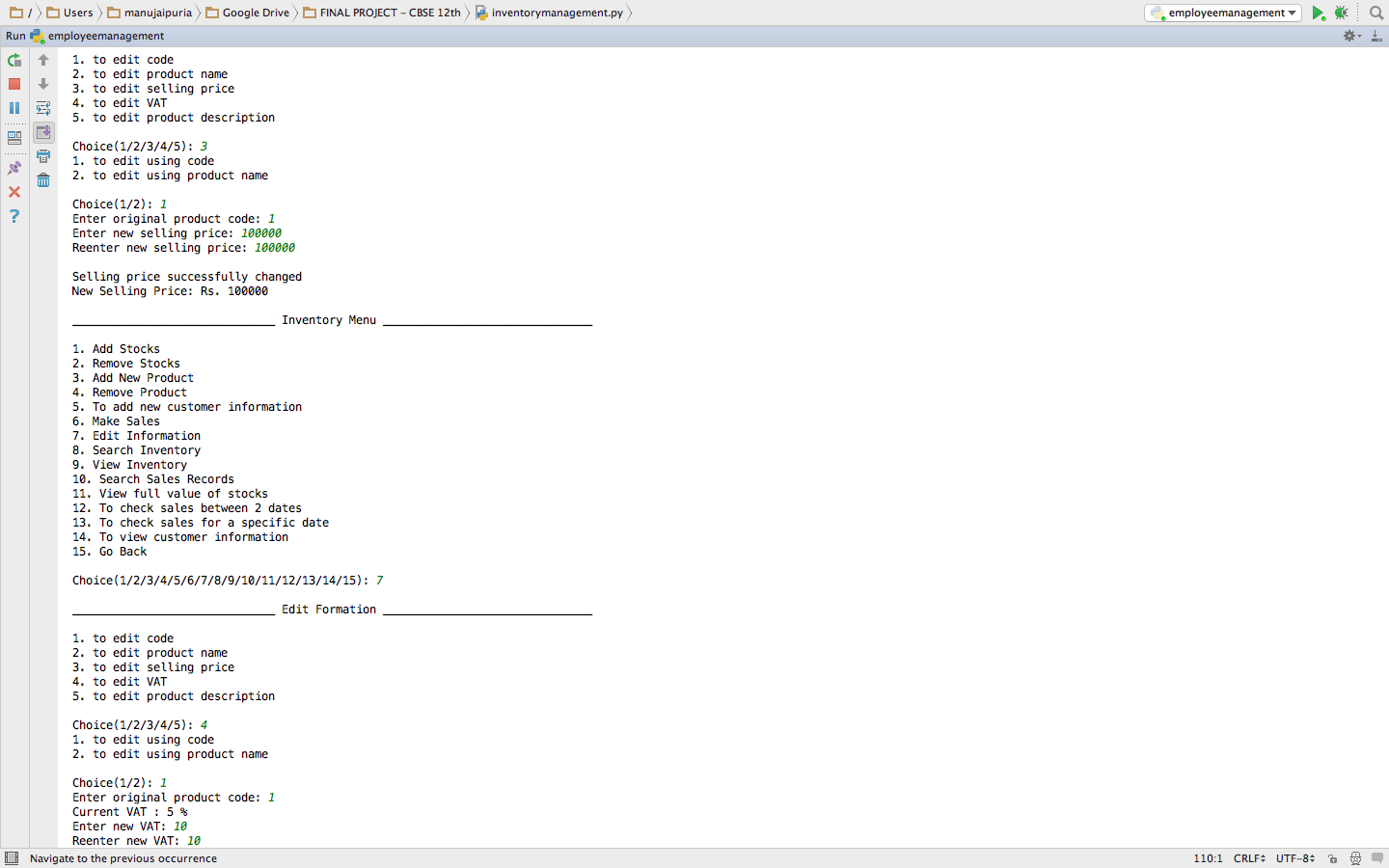
****

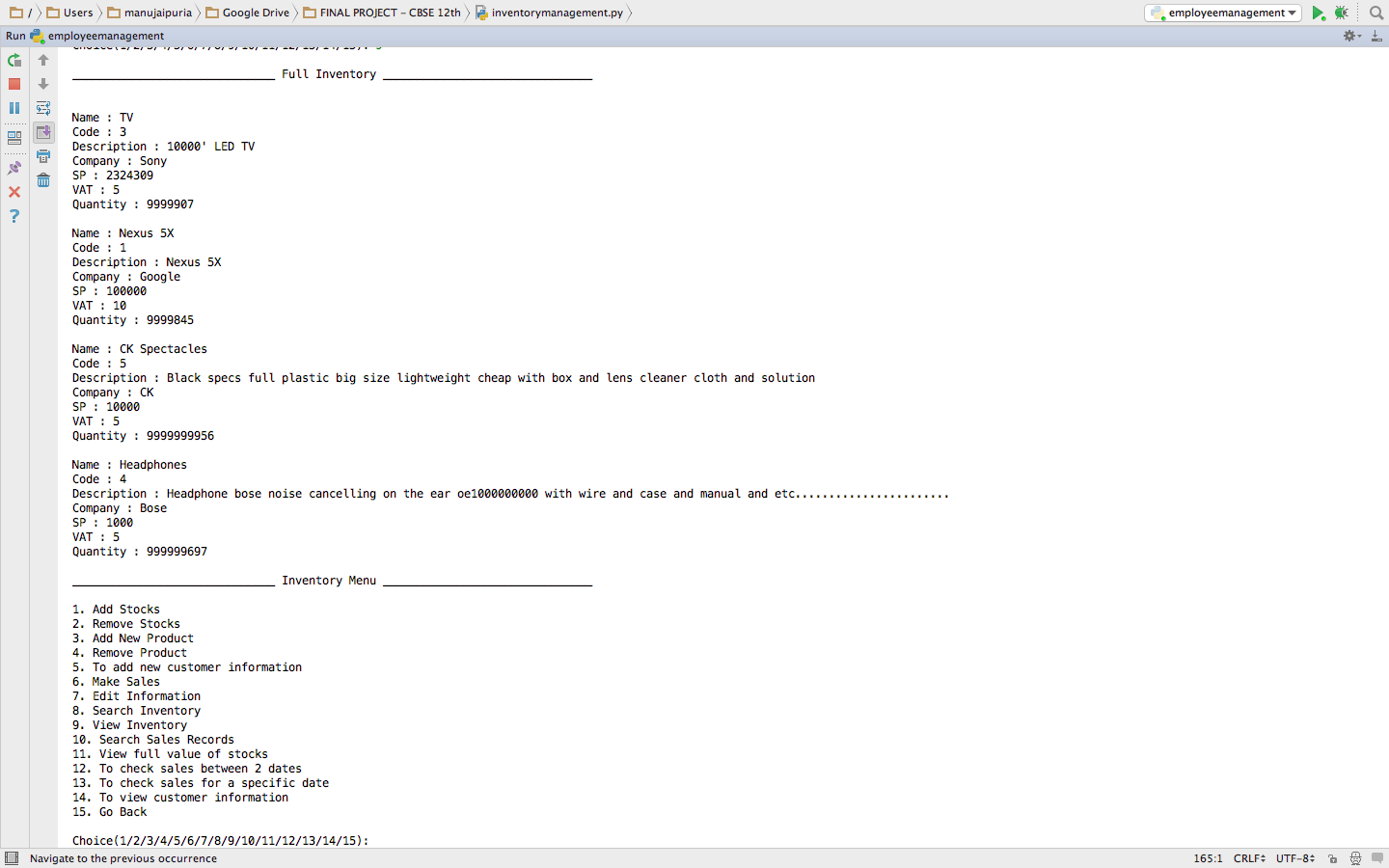
****











EMAIL ENGINE

