PM SHRI KVSL MEERUT CANTT

Project report
Computer Science
SESSION – 2023-2024

Topic : "*Retail Operations*"

Submitted By : Kanishk Negi

Class : 12th A

Roll No : 21712310

Submitted To : Mr.Ajay Kumar
(PGT COMPUTER SCIENCE)

Certificate

This is to certify that this project is developed by Master Kanishk Negi of class XII Science for the fulfillment of CBSE Practical Examinations 2024 in the session 2023-24 and this is not copy of any other Project.

_____ _____

Examiner's Signature Teacher-In-Charge

_____ _____

Date Principal

_____

School Rubber Stamp

ACKNOWLEDGEMENT

With Condor & Pleasure I take opportunity to express my sincere thanks and obligation to esteemed guide of Mr. Ajay Kumar, PGT(CS) PM SHRI KVSL MEERUT CANTT. It is because of his able and mature guidance and co-operation without which it would not have been possible for me to complete my project.

It is my pleasant duty to thank all my class mates of school who never hesitated from me time during the project.

Finally, I gratefully acknowledge the support, encouragement & patience of my family, and as always, nothing in my life would be possible without God.

Thank You!

Class XII-A

OVERVIEW OF PYTHON LANGUAGE

Python, created by Guido van Rossum in 1991, is a widely used, readable, and adaptable programming language. Its simplicity and readability are emphasized, using indentation for code blocks. A general-purpose language, Python supports procedural and object-oriented programming. It's interpreted, executing code line by line, and offers an interactive mode for learning and quick prototyping.

Dynamic typing allows flexibility in variable types. Python boasts an extensive standard library, minimizing the need for code from scratch. The active community contributes to the Python Package Index (PyPI) with frameworks like Django and Flask. Python is cross-platform, running on various operating systems. Its applications span web development, automation, data analysis, machine learning, scientific computing, and more. As an open-source language, Python's collaborative nature contributes to its high demand in industry.

In summary, Python's popularity lies in its readability, versatility, and community support, making it a go-to language across diverse domains.

Summary & Need of the Project

In the world of technology everything needs to be automated. "*Retail Operations*" uses python language to automate the tasks that are manually managed. Using Retail Operations an individual can easily manage his shop in a more efficient way. The customers can use the interface easily and can use "help" if they are stuck in any situation, also owner can use the 'owner_login' to add an item, update stock of an item and get the latest orders. The application uses MySQL database to store information about items and also username and passwords of the customers for authentication. It also uses "mysql.connector" library to for interfacing the MySQL database and the python language. Management of any shop can be done easily using "RETAIL OPERATIONS". Though it has command line interface, still it is user-friendly as operations can be performed by just typing phrases in the prompt.

SYSTEM REQUIREMENTS

Processors:

Intel® Core™ i5 processor 4300M at 2.60 GHz or 2.59 GHz (1 socket, 2 cores, 2 threads per core), 8 GB of DRAM

Intel® Xeon® processor E5-2698 v3 at 2.30 GHz (2 sockets, 16 cores each, 1 thread per core), 64 GB of DRAM

Intel® Xeon Phi™ processor 7210 at 1.30 GHz (1 socket, 64 cores, 4 threads per core), 32 GB of DRAM, 16 GB of MCDRAM (flat mode enabled)

Disk space: 2 to 3 GB

Operating systems: Windows® 10, macOS*, and Linux*

Minimum System Requirements

Processors: Intel Atom® processor or Intel® Core™ i3 processor

Disk space: 1 GB

Operating systems: Windows* 7 or later, macOS, and Linux

Python* versions: 2.7.X, 3.6.X

Included development tools: conda*, conda-env, Jupyter Notebook* (IPython)

Compatible tools: Microsoft Visual Studio*, PyCharm*

Included Python packages: NumPy, SciPy, scikit-learn*, pandas, Matplotlib, Numba*, Intel® Threading Building Blocks, pyDAAL, Jupyter, mpi4py, PIP*, and others.

Package Name & Application Info

Programming Language Used: 

Application Used : 

Libraries and Modules Included:

1. mysql.connector
2. datetime
3. os
4. tabulate

Name of File: *shop.py*

Topic of Project: Retail Operations

DOCUMENTATION

The app is used exactly similar to the terminal or MySQL. It has a prompt and predefined keyword that are used to perform specific operations.

SHOP>

(Prompt of the application)

The application have separate login and interface for owner and customers. For customers the list of keywords are as follows:

Keywords	Their Use
login	For user to log in to their account.
signup	For registering a new account.
list	To list all available items.
buy	To buy an item.
search	To search an item.
history	To see past orders.
EXIT	To exit the application.
clear	To clear the terminal.
help	To print the help menu

The output of “help”:

```
SHOP>help
              WELCOME TO OUR SHOP
            WHAT WOULD YOU LIKE TO DO?
TO LOGIN TYPE :- login
TO SIGNUP TYPE:- signup
TO LIST ALL ITEMS TYPE:- list
TO BUY ANYTHING TYPE:- buy
TO SEARCH A PRODUCT TYPE:- search
TO SEE YOUR ORDER HISTORY TYPE:- history
TO EXIT TYPE:- EXIT
TO PRINT THIS MENU:- help
TO CLEAR TERMINAL TYPE:- clear
```

The output of “list”:

```
SHOP>list
  ITEM ID  ITEM NAME          ITEM PRICE    ITEM QUANTITY
---------  ---------------  ------------  ---------------
        1  SOAP                       10              366
        2  DETERGENT 10 KG           600              999
        3  CHIPS 250g                 50             1289
        4  Biscuit                    80              500
        5  Toys                      500              274
```

The output of login:

```
ENTER YOUR USERNAME: test
ENTER YOUR PASSWORD: test
SHOP@test>
```

and output of signup:

```
ENTER YOUR USERNAME: TEST10
ENTER YOUR PASSWORD: TEST10
ACCOUNT CREATED
```

The owner’s interface can be accessed by the keyword owner_login which is not present in the normal help menu:

```
SHOP>owner_login
ENTER USERNAME: owner
ENTER PASSWROD: owner

TO ADD A ITEM TYPE:- add_item
TO UPDATE STOCK TYPE:- update_stock
TO GET RECENT ORDERS TYPE:- recent_orders

{___OWNER___}>>
```

Functions used and their Purpose

Function Name	Purpose
1). make_connection()	Defined to connect to the database.
2). login_user()	Defined to login the user.
3). signup_user()	Defined to create a new user.
4). add_item()	Defined to add a new item.
5). update_qty()	Defined to update the quantity of an item.
6). search_item()	Defined to search an item.
7) List()	Defined to arrange the output from MySQL in tabular form.
8) handle_commands()	Defined to handle the input from the prompt in terminal.
9) os.system()	Used to execute some os specific commands.

SOURCE CODE

1. shop.py

```
import os
from tabulate import tabulate
from datetime import datetime
import mysql.connector as mycn
from credintials import host , username , password , database,
owner_name, owner_password
#-----------------------------FUNCTIONS-------------------------------
#
def make_connection():
    global connection
    connection =
mycn.connect(host=host,username=username,passwd=password,database=data
base)
    cursor = connection.cursor()
    return cursor
def login_user(username,password):
    cursor = make_connection()
    cursor.execute(f"SELECT uname,passwd FROM users where
uname='{username}';")
    data = cursor.fetchone()
    connection.close()
    if data != None:
        if password == data[1]:
            return True,username
        else:
            print("WRONG PASSWORD")
            return False
    else:
        print("WRONG USERNAME")
        return False
def signup_user(username,password):
    cursor = make_connection()
    try:
        cursor.execute(f"INSERT INTO users(uname,passwd)
VALUES('{username}','{password}')")
        connection.commit()
        connection.close()
        print("ACCOUNT CREATED")
        global PROMPT
        PROMPT="SHOP>"
```

```
    except Exception as e:
        print(e)
        print("Your account cant be created")
def add_item(itm_name , itm_price , itm_qty ):
    cursor = make_connection()
    try:
        cursor.execute(f"INSERT INTO items(itm_name,itm_price,itm_qty)
VALUES('{itm_name}','{itm_price}','{itm_qty}')")
        connection.commit()
        connection.close()
        print(f"ADDED ITEM {itm_name}")

    except Exception as e:
        print("ITEM COULDNT BE ADDED")
        print(e)
def update_qty(iod,magnitude,itm_id):
    cursor = make_connection()
    try:
        cursor.execute(f"UPDATE items SET itm_qty=itm_qty{iod}
{magnitude} where iid='{itm_id}'")
        connection.commit()
        connection.close()
    except Exception as e:
        print(e)
        print("THE REQUESTED ACTION CANT BE DONE RIGHT NOW")
def search_item(itm_name):
    cursor = make_connection()
    cursor.execute(f"SELECT * FROM items WHERE itm_name like '%
{itm_name}%'")
    data = cursor.fetchall()
    cols = ['ITEM ID','ITEM NAME','ITEM PRICE','ITEM QUANTITY']
    print(tabulate(data, cols))
    return True
def List():
        cursor = make_connection()
        cursor.execute("SELECT * FROM items")
        data = cursor.fetchall()
        columns = ['ITEM ID','ITEM NAME','ITEM PRICE','ITEM QUANTITY']
        print(tabulate(data, headers=columns))
def handle_commands(command):
    if command.lower() == "login":
        result = login_user(input("ENTER YOUR USERNAME:
"),input("ENTER YOUR PASSWORD: "))
        if result:
            global PROMPT
            global USER
            USER = result[1]
```

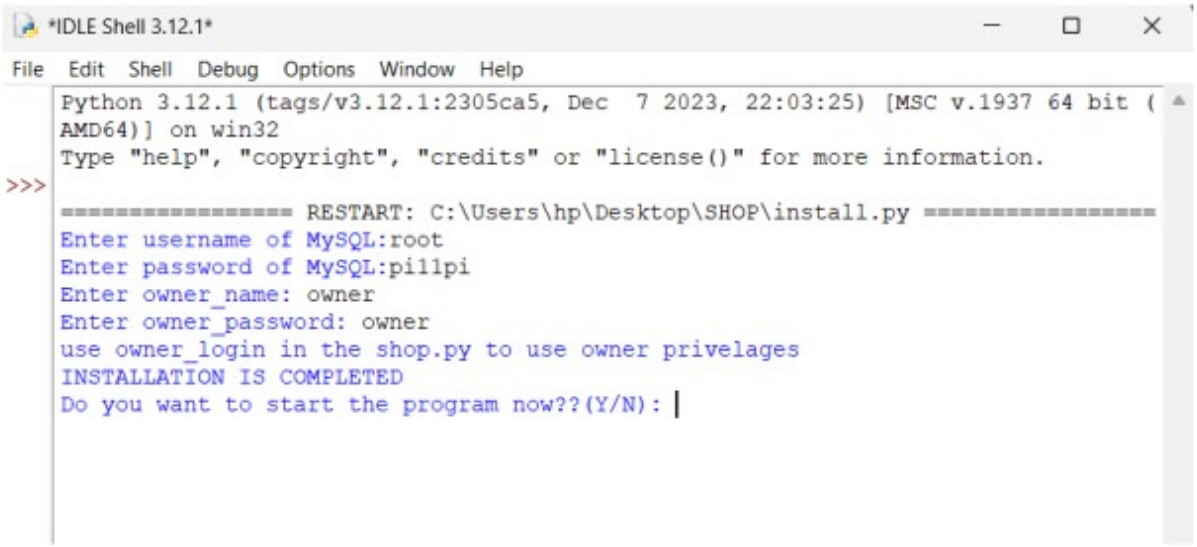
```
            PROMPT = f"SHOP@{result[1]}>"
    elif command.lower() == "signup":
        signup_user(input("ENTER YOUR USERNAME: "),input("ENTER YOUR
PASSWORD: "))
    elif command.lower() == "buy":
        if USER:
            cursor = make_connection()
            PROMPT=f"{USER}/buying>"
            print(PROMPT+"ENTER THE ITEM NAME YOU WANT TO BUY: ")
            item=input()
            items = search_item(item)
            connection.close()
            if items:
                cursor = make_connection()
                print(PROMPT+"NOW ENTER ITEM ID: ")
                itid = input()
                qty = input(PROMPT+"ENTER THE AMOUNT OF THE ITEM YOU
WANT: ")
                cursor.execute(f"SELECT iid,itm_name,itm_price from
items where iid='{itid}'")
                data = cursor.fetchone()
                try:
                    update_qty('-',qty,itid)
                    with open(f"logs/{USER}.log",'a') as log:
                        log.write(f"[ {datetime.now()} ] BOUGHT {qty}
{data[1]} AT THE PRICE {data[2]} \n")
                        log.close()
                    with open(f"logs/owner_notify.log",'a') as log:
                        log.write(f"[ {datetime.now()} ] {USER} BOUGHT
{qty} {data[1]} AT THE PRICE {data[2]} ITEM_ID is {data[0]} \n")
                    print("ITEM HAS BEEN BOOKED FOR YOU")
                except Exception as e:
                    print("INVALID ITEM ID")
        else:
            print("LOGIN FIRST")

    elif command.lower() == "search":
        if USER:
            PROMPT = f"{USER}/searching>"
            search_item(itm_name=input(f"{PROMPT}ENTER THE ITEM NAME:
"))
        else:
            print("LOGIN FIRST")
    elif command.lower() == "owner_login":
        username_and_password = input("ENTER USERNAME: "),input("ENTER
PASSWROD: ")
```

```
        if username_and_password[0] == owner_name and
username_and_password[1] == owner_password:
            USER = "OWNER"
            PROMPT = "{___OWNER___}>>"
            print(owner_menu)
        else:
            print("WRONG USENAME OR PASSWORD")

    elif command.lower() == "help":
        if USER == "OWNER":
            print(owner_menu)
        else:
            print(menu)
    elif command.lower() == "clear":
        os.system("clear")
    elif command.lower() == "add_item":
        if USER == "OWNER":
            tprompt = PROMPT
            PROMPT = "ADDING_ITEM>"
            add_item(input(PROMPT+"ENTER ITEM NAME:
"),input(PROMPT+"ENTER ITEM PRICE:"),input(PROMPT+"ENTER ITEM
QUANTITY: "))
            PROMPT = tprompt
        else:
            print(f"INVALID COMMAND '{command}'")
    elif command.lower() == "update_stock":
        List()
        if USER == 'OWNER':
            iod = input(PROMPT+"INCREASE( I ) or DECREASE
( D )").lower()
            if iod.upper() == 'I':
                update_qty('+',input("ENTER AMOUNT INCREASED FOR THIS
ITEM: "),input("ENTER ITEM ID: "))
            else:
                update_qty('-',input("ENTER AMOUNT DECREASED FOR THIS
ITEM: "),input("ENTER ITEM ID: "))
        else:
            print(f"INVALID COMMAND '{command}'")
    elif command.lower() == "recent_orders":
        with open("logs/owner_notify.log") as log:
            data = log.readlines()
            for x in data:
                print(x)
    elif command.lower() == "history":
        try:
            with open(f"logs/{USER}.log") as log:
                data = log.readlines()
```

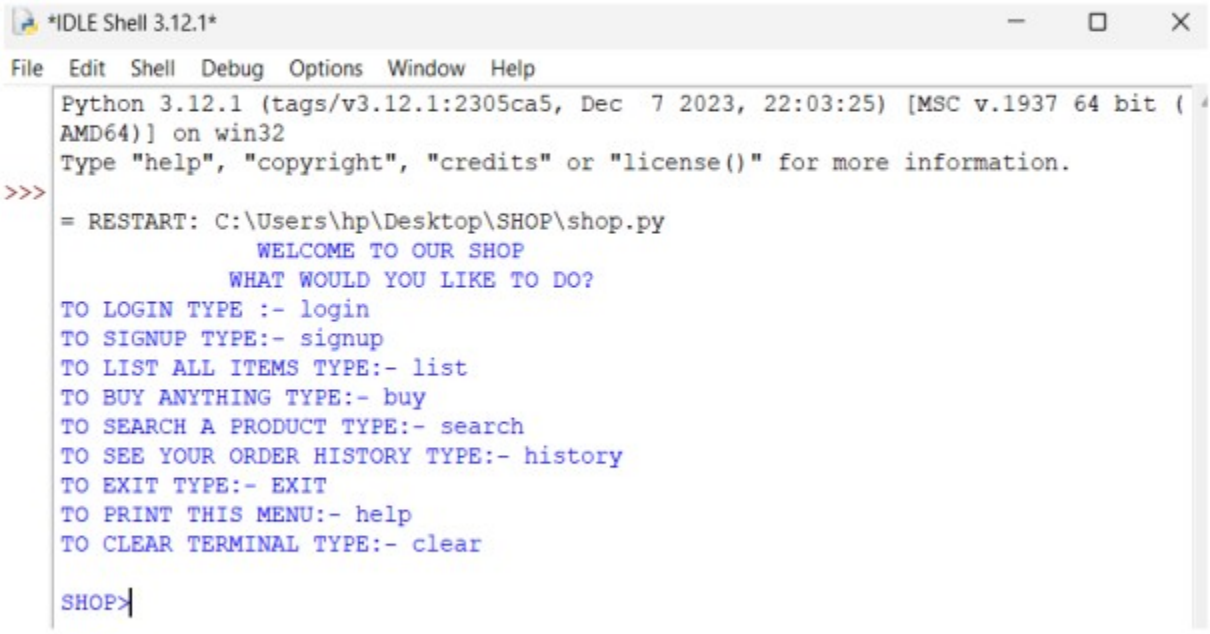
```
                for x in data:
                    print(x)
        except:
            print("YOU BOUGHT NOTHING FOR NOW")
    elif command.lower() == "list":
        List()
    elif command.lower() == '' or ' ' :
        pass

    else:
        print(f"INVALID COMMAND '{command}'")
#-------------------------------------------------------------------------
--------------------------#
#_____________________________________MAIN__PROGRAM_____________________
_____________________________#
menu = """               WELCOME TO OUR SHOP
            WHAT WOULD YOU LIKE TO DO?
TO LOGIN TYPE :- login
TO SIGNUP TYPE:- signup
TO LIST ALL ITEMS TYPE:- list
TO BUY ANYTHING TYPE:- buy
TO SEARCH A PRODUCT TYPE:- search
TO SEE YOUR ORDER HISTORY TYPE:- history
TO EXIT TYPE:- EXIT
TO PRINT THIS MENU:- help
TO CLEAR TERMINAL TYPE:- clear
            """
owner_menu = """
TO ADD A ITEM TYPE:- add_item
TO UPDATE STOCK TYPE:- update_stock
TO GET RECENT ORDERS TYPE:- recent_orders
"""
USER = False
print(menu)
PROMPT="SHOP>"
while True:
    command = input(PROMPT)
    if command.lower() == "exit":
        break
    else:
        handle_commands(command)
```

OUTPUT

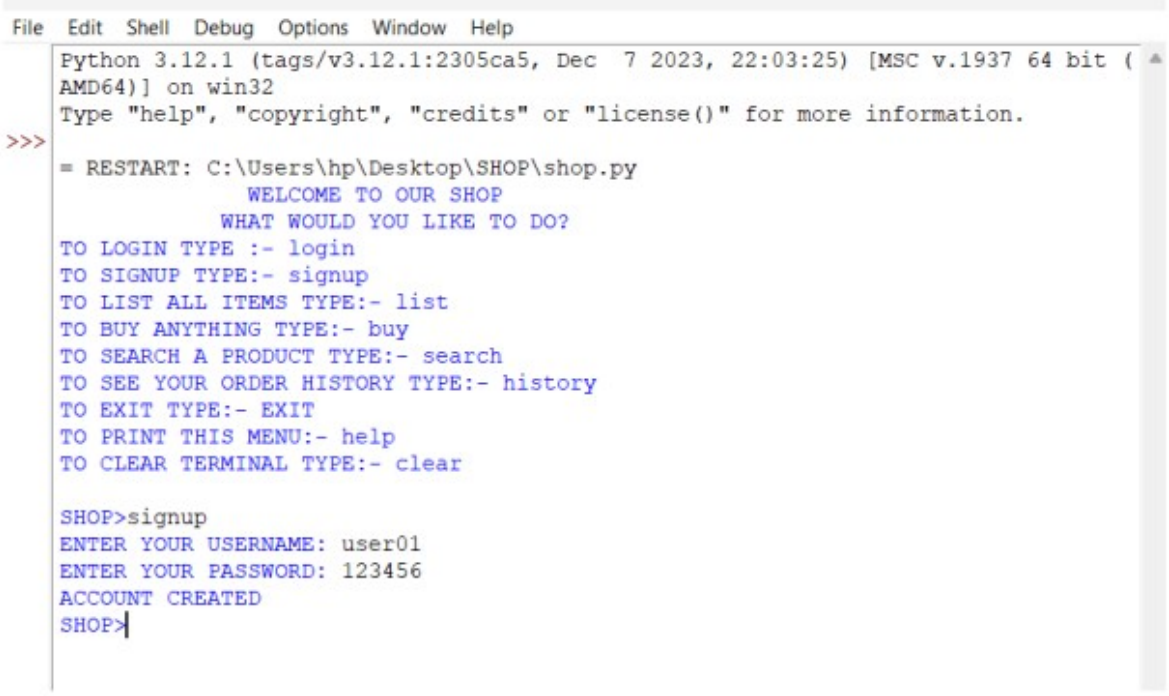
The Output of “install.py”:



The Output of “shop.py”:



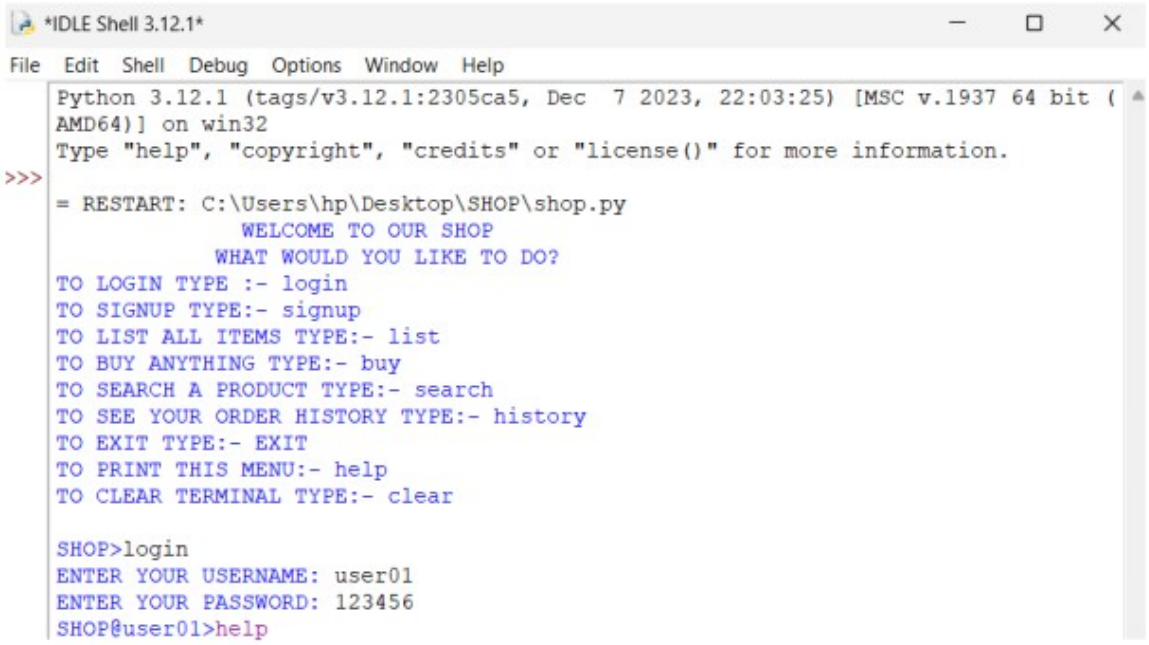
The Output of “SHOP>signup”:



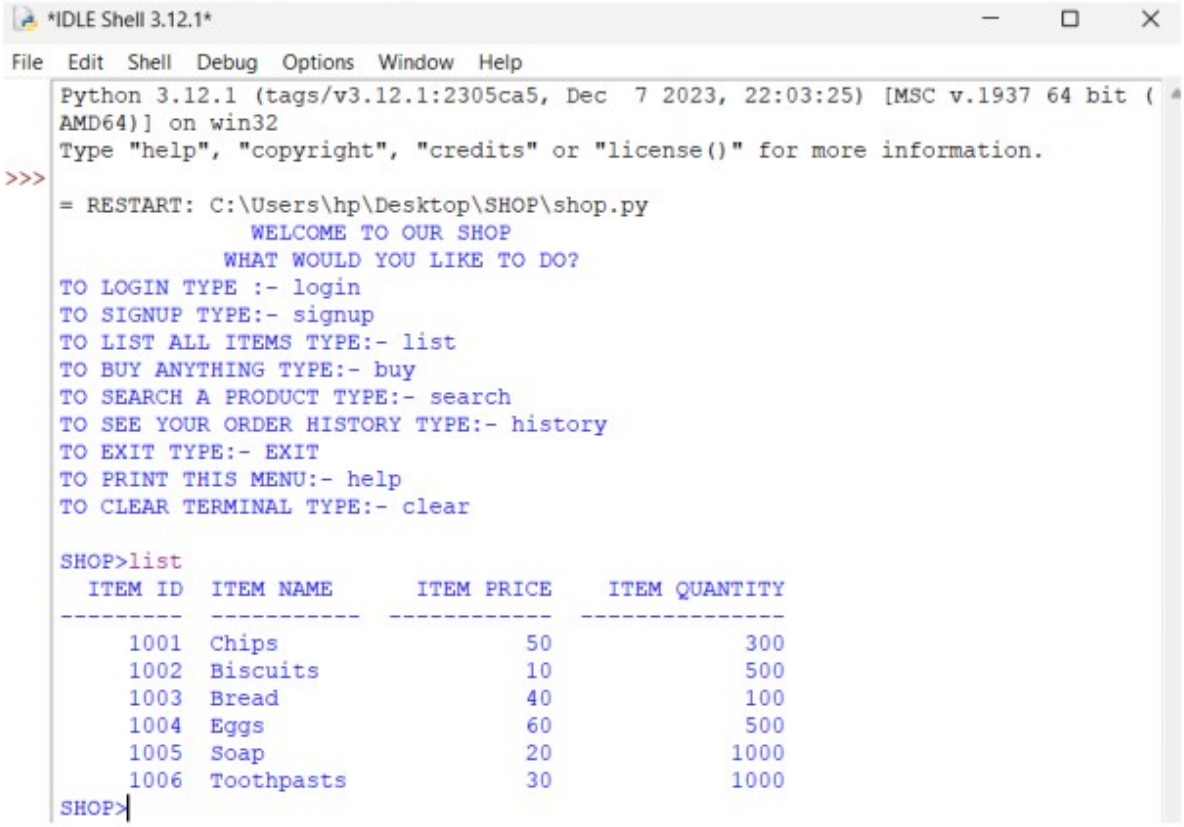
The Output of “help”:

```
SHOP>help
              WELCOME TO OUR SHOP
            WHAT WOULD YOU LIKE TO DO?
TO LOGIN TYPE :- login
TO SIGNUP TYPE:- signup
TO LIST ALL ITEMS TYPE:- list
TO BUY ANYTHING TYPE:- buy
TO SEARCH A PRODUCT TYPE:- search
TO SEE YOUR ORDER HISTORY TYPE:- history
TO EXIT TYPE:- EXIT
TO PRINT THIS MENU:- help
TO CLEAR TERMINAL TYPE:- clear

SHOP>
```

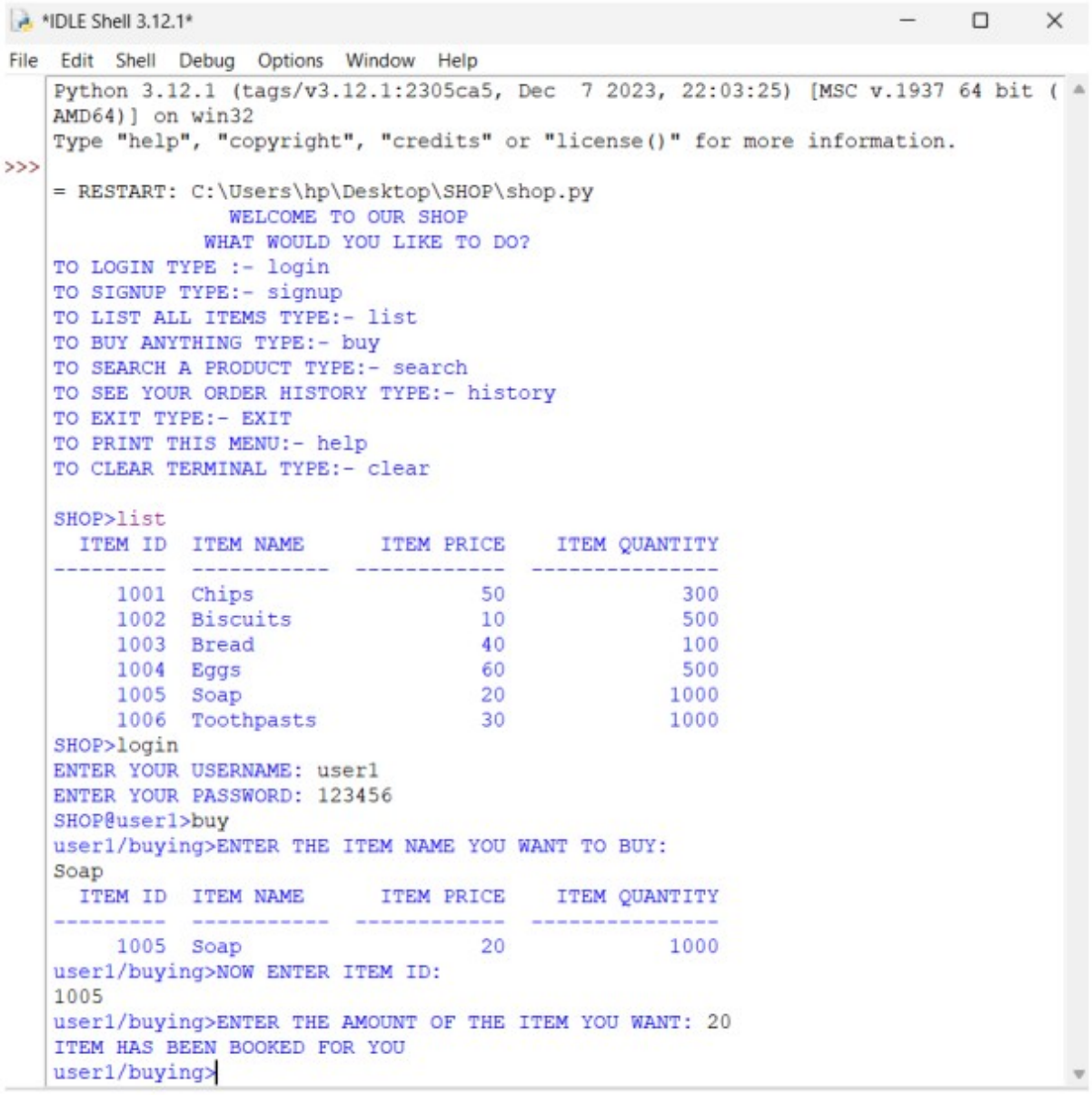
The Output of “SHOP>login”:



The Output of “list”:



The Output of “buy”:



BIBLIOGRAPHY

The source of my information is always been my teacher Mr. Ajay Kumar (PGT C.S). But some other help is also been taken from other sources such as:-

PyPI

Python Documentation

MySQL Documentation

At last I want to thank my classmates for supporting me.

–