<python lang="en">

<head>

Shop 3000

– Comprehensive Documentation

</head>

<body>

1. Introduction

The Shop 3000 script is a comprehensive Python-based shop management system designed to facilitate product management, sales tracking, and customer interaction. The system utilizes a MySQL database for data storage, Matplotlib for graphical analysis, and pyttsx3 for text-to-speech feedback. This documentation aims to provide an in-depth understanding of the script's features, functionalities, potential improvements, and security considerations.

2. Features

2.1 Database Connection

The script establishes a connection to a MySQL database using the `mysql.connector` library. This connection is pivotal for storing and managing crucial information, including product details, purchase records, stock information, and cash balances.

2.2 Encryption

To enhance security, the script implements a Vigenere cipher for encrypting and decrypting passwords. This adds an additional layer of protection to sensitive information, particularly login credentials.

2.3 Graphical Analysis

Users, particularly administrators, have the capability to analyze sales data through graphical representations. This includes line graphs, pie charts, and bar graphs, providing visual insights into product performance and overall sales trends.

2.4 Administrator Controls

Administrators are empowered with a suite of controls to efficiently manage various aspects of the system:

- **Product Management:** Enables addition, removal, and price modification of products in the stock.
- **Cash Management:** Facilitates deposit and withdrawal operations to adjust the cash balance.
- **View Accounts:** Provides access to detailed accounts, including sales records and stock information.
- **Change Admin Password:** Allows the administrator to update their password for enhanced security.
- **Graphical Representations:** Visualizes sales data through graphical analysis tools.

2.5 Main Menu

The main menu serves as the central hub for users, offering several options:

- **Buy Products:** Customers can purchase items, adding them to their cart.
- **Administrator Login:** Grants access to the administrator controls for managing the system.
- **Miscellaneous Options:** Provides additional features, including a user guide, credits, and a placeholder for a project section.
- **Quit Program:** Allows users to exit the program gracefully.

2.6 Customer Interaction

Customers can interact with the system through a dedicated menu, performing actions such as adding products to their cart, checking out, viewing available items, and returning to the main menu.

3. Potential Improvements

To enhance the script's functionality, maintainability, and user experience, consider the following improvements:

3.1 Input Validation

Implement thorough input validation mechanisms to handle unexpected inputs gracefully and improve the overall robustness of the system.

3.2 Code Organization

Organize the code into functions or classes to improve readability, maintainability, and modularity.

3.3 Security measures

Explore more secure methods for storing and validating passwords, considering industry best practices for credential management.

3.4 Exception Handling

Enhance the script's robustness by incorporating proper exception handling throughout the codebase.

3.5 User Interface

Consider developing a graphical user interface (GUI) using a library like Tkinter to provide a more user-friendly and intuitive experience.

3.6 Documentation

Include comprehensive comments and documentation to elucidate the purpose and functionality of various code segments, aiding future maintenance and development efforts.

4. Security Considerations

Ensure the secure handling of database connection details and sensitive information, especially in a real-world application. Evaluate and implement security best practices to safeguard user data and system integrity.

5. Conclusion

The Shop 3000 script demonstrates a robust foundation for a shop management system, providing a range of features for both customers and administrators. By implementing the suggested improvements and adhering to security best practices, the script can evolve into a reliable and secure solution for small-scale retail operations.

6. Documentation

6.1 Usage

1. Run the script (`shop3000.py`).
2. Follow the on-screen prompts to navigate through the application.
3. Enjoy managing your shop with Shop 3000!

6.2 Notes

- This code assumes a basic understanding of Python and MySQL. Familiarize yourself with Python and MySQL concepts for better utilization.
- For more details on specific functionalities, refer to the comments within the code. Detailed comments have been provided to aid understanding.

6.3 Credits

The Shop 3000 project was collaboratively developed by:

- **Insaf**

- **Aamir**
- **Davidson**

6.4 License

This project is open source under no license

6.5 Acknowledgments

- Special thanks to all libraries used
- The project is inspired by the need for a simple and effective sales management system for small businesses.
- Thanks to our informatics mentor for helpful support

6.6 Version History

- **Version 1.0:** Initial release (provide date).
- **Version 1.1 :** added customer and admin controls
- **Version 2.0 :** added cipher , matpoltlib graphs
- **Version 3.0 :** added pttsx3

</body>

Source code ----------

This section imports necessary libraries:

- mysql.connector for connecting to MySQL database.
- matplotlib.pyplot for creating plots.
- pyttsx3 for text-to-speech functionality.

```
import mysql.connector as mysq
import matplotlib.pyplot as plt
def establish_connection(password):
    try:
        mydb = mysq.connect(host="localhost", user='root', passwd=password)
        c = mydb.cursor()
        return mydb, c
    except mysq.Error as e:
        print(f"Error connecting to MySQL: {e}")
        exit()
```

Establish connection
Encrypting Text using Vigenere Cipher

```
def encrypt(plain_text, key):
    encrypted_text = ""
    key_index = 0

    for char in plain_text:
        if char.isalpha():
            key_char = key[key_index % len(key)]
            key_index += 1

            if char.isupper():
                base = ord('A')
            else:
                base = ord('a')

            encrypted_char = chr((ord(char) - base + ord(key_char.upper()) - ord('A')) % 26 + base)
            encrypted_text += encrypted_char
        else:
            encrypted_text += char

    return encrypted_text
```

Updating cash in database

```
def cash(x, c):
    c.execute("SELECT * FROM cash")
    cash, = c.fetchone()
    cash += x
    c.execute("UPDATE cash SET cash = %s",(cash,))
    mydb.commit()
```

Display guide information

```
def guide():
    print('''
    ====== Guide ======

    ==== How to use program ====
    All libraries used:
      1. matplotlib
      2. mysql-connector
      3. pyttsx3

    === Bibliography ===

    1. Vigenere cipher
    2. Sales management project tutorial
    ''')
    engine.say('''
        How to use program
        All libraries used
          1. matplotlib
          2. mysql-connector
          3. pyttsx3

        Bibliography
        1. Vigenere cipher
        2. Sales management project tutorial
        '''
    )
    engine.runAndWait()
```

Creating tables if not exist

```
def pretables(c,mydb):
    c.execute("CREATE TABLE IF NOT EXISTS login(username VARCHAR(30) DEFAULT 'admin', password VARCHAR(40) DEFAULT %s)", (defpass,))
    c.execute("CREATE TABLE IF NOT EXISTS purchases(order_date DATE, name VARCHAR(30) NOT NULL, product_code INT NOT NULL, amount INT)")
    c.execute("CREATE TABLE IF NOT EXISTS stock(product_code INT NOT NULL, product_name VARCHAR(40), quantity INT, price INT)")
    c.execute("CREATE TABLE IF NOT EXISTS cash(cash INT DEFAULT 0)")
    mydb.commit()
```

Main menu : always looping

```
def mainmenu():
    while True:
        print('''
    -------- Welcome to the Shop 3000

        1. Buy a product
        2. Administrator login
        3. Miscellaneous
        0. Quit program?

        ''')
        engine.say('''welcome to the shop

        1. buy something
        2. administration login
        3. miscellaneous
        0. quit program

        ''')
        engine.runAndWait()

        choice1 = int(input("Enter your choice: "))
        if choice1 == 2:
            login(c,mydb)
        elif choice1 == 1:
            customer(c,mydb)
        elif choice1 == 3:
            other(c,mydb)
        elif choice1 == 0:
            print("Quitting...")
            exit()
        else:
            print("Invalid choice! Please try again.")
```

Buying side

```
def customer(c,mydb):
    while True:
        print("""
    1. Add a product to cart.
    2. Check out.
    3. View Available Items.
    4. Go to menu.
    """)
        engine.say("""
    1. add item to cart
    2. check out
    3. View Available Items
    4. Go to menu
    """)
        engine.runAndWait()

        ch2 = int(input("Enter your choice:"))

        if ch2 == 1:
            name = input("Enter your name:")
            pcode = int(input("Enter product code:"))
            c.execute("SELECT product_code FROM stock")
            existing_product_codes = [row[0] for row in c.fetchall()]

            if pcode in existing_product_codes:
                pc = str(pcode)
                quantity = int(input("Enter product quantity:"))
                c.execute("SELECT * FROM stock WHERE product_code = %s", (pc,))
                row = c.fetchone()
                if row:
                    t_code, t_name, t_quan, t_price = row
                    net_quan = t_quan - quantity
                    amount1 = t_price * quantity
                    global amount
                    amount+=amount1
                    q = str(net_quan)
                    codee = str(pcode)
                    amm = str(amount)

                    c.execute("UPDATE stock SET quantity = %s WHERE product_code = %s", (q, codee))


                    print("-------  Added to cart successfully!!. Returning to main menu.. ----------  ")
                    engine.say("""
    added to cart successfully
    """)
                    engine.runAndWait()
                else:
                    print("Product not found. Try again!")
                    customer()
            else:
                print("Wrong product code. Try again!")
                customer(c,mydb)
```

```
        elif ch2 == 2:
            print(f"Amount to be paid -- {amount}")
            ch15 = input("Are you sure you want to buy? (y/n): ")
            if ch15.lower() == "y":
                c.execute("INSERT INTO purchases VALUES (NOW(), %s, %s, %s)", (name, codee, amm))
                print(" --------  Thank you for shopping with us!! Come again soon.  ----------")
                engine.say("""
    thank you shopping with us come again soon
    """)
                engine.runAndWait()
                cash(amount, c)
                mydb.commit()
                customer()
            else:
                c.execute("UPDATE stock SET quantity = %s WHERE product_code = %s", (t_quan, codee))
                customer(c,mydb)
        elif ch2 == 3:
            c.execute("SELECT * FROM stock")
            F = "%15s %15s %15s %15s"
            print(F % ("Product code", "  Product name", "Items left", "Price"))
            print("-" * 65)
            for row in c:
                for item in row:
                    print("%14s" % item, end=' ')
                print()
            print("-" * 65)

        elif ch2 == 4:
            mainmenu()

        else:
            print("Error: Invalid choice. Try again.")
            customer()
```

Miscellinious

```
def other():
    print('''
    ====== More Options ====
    1. Guide
    2. Credits
    ''')
    choice15 = int(input("Enter your choice: "))
    if choice15 == 1:
        guide()
    elif choice15 == 2:
        print('''
    ===== Credits =====
    1. Insaf
    2. Aamir
    3. Davidson
    ''')
```

Login

```
def login(c,mydb):
    login_attempts = 3

    while login_attempts > 0:
        login_user = input("Enter administrator username: ")
        login_pass = input("Enter your password: ")

        c.execute("SELECT * FROM login")
        for row in c:
            username, hashed_password = row

        if login_user == username and encrypt(encrypt(encrypt(login_pass, mysql),mysql),mysql) == hashed_password:
            print("Login successful! Opening Administrator Controls...")
            admin(c,mydb)
            break
        else:
            print("Login failed. attempts remaining =",login_attempts-1)
            login_attempts -= 1

        if login_attempts == 0:
            print("Login attempts exhausted. Exiting...")
            exit()

def admin(c,mydb):
```

Cash withdrew or add

```
def cash_config(c,mydb):
    c.execute("SELECT * FROM cash")
    cash_balance, = c.fetchone()
    print("Balance =", cash_balance)

    withdraw = int(input("How much cash to be withdrawn? (0 to skip): "))
    deposit = int(input("How much cash to be deposited? : "))

    cash_balance -= withdraw
    cash_balance += deposit

    print("Balance of Cash available =", cash_balance)


    c.execute("UPDATE cash SET cash = %s",(cash_balance,))
    mydb.commit()
    admin(c,mydb)
```

Admin

```
def admin(c,mydb):
    print('''
    === Administrator controls ===

    a. Show graphs (LineGraph/PieChart..)
    b. Configure products (Add/Remove...) .
    c. Configure cash (Deposit/Withdrawal...).
    d. View accounts.
    e. Change admin password.
    f. Back to main menu.
    ''')

    engine.say('''Administrator controls:

    a. Show graphs (LineGraph/PieChart..)
    b. Configure products (Add/Remove...) .
    c. Configure cash (Deposit/Withdrawal...).
    d. View accounts.
    e. Change admin password.
    f. Back to main menu.
    ''')
    engine.runAndWait()

    choice2 = input("Enter your choice: ")

    if choice2 == "a":
        graph(c,mydb)
    elif choice2 == "b":
        product_config(c,mydb)
    elif choice2 == "c":
        cash_config(c,mydb)
    elif choice2 == "d":
        view_accounts(c,mydb)
    elif choice2 == "e":
        change_admin_password(c,mydb)
    elif choice2 == "f":
        mainmenu()
    else:
        print("Invalid choice. Please try again.")
        admin(c,mydb)
```

Changing admin password

```
def change_admin_password(c,mydb):
    old_password = input("Enter the old password: ")
    c.execute("SELECT * FROM login")
    username, hashed_old_password = c.fetchone()

    if encrypt(encrypt(encrypt(old_password, mysql), mysql), mysql) == hashed_old_password:
        new_password = input("Enter your new password: ")
        hashed_new_password = encrypt(encrypt(encrypt(new_password, mysql), mysql), mysql)

        c.execute("UPDATE login SET password = %s", (hashed_new_password,))
        mydb.commit()
        mainmenu()
    else:
        print("Wrong old password! Try again.")
        admin(c,mydb)
```

Graphs

```
def graph(c,mydb):
    gr_code = []
    gr_name = []
    gr_quantity = []
    gr_price = []

    print('''Which graph do you wish to see
        a. Line graph (quantity)
        b. Pie chart (items remaining)
        c. Price comparison graph
        ''')

    engine.say('''
Accessing data regarding sales ...
    ''')
    engine.runAndWait()

    choice4 = input("Enter your choice: ")

    if choice4 == "a":
        c.execute("SELECT * FROM stock")
        for row in c:
            code, name, quantity, price = row
            gr_code.append(code)
            gr_name.append(name)
            gr_quantity.append(quantity)
            gr_price.append(price)

        plt.plot(gr_name, gr_quantity)
        plt.show()
        admin(c,mydb)

    elif choice4 == "b":
        c.execute("SELECT * FROM stock")
        for row in c:
            code, name, quantity, price = row
            gr_code.append(code)
            gr_name.append(name)
            gr_quantity.append(quantity)
            gr_price.append(price)

        plt.pie(gr_quantity, labels=gr_name)
        plt.show()
        admin(c,mydb)

    elif choice4 == "c":
        c.execute("SELECT * FROM stock")
        for row in c:
            code, name, quantity, price = row
            gr_code.append(code)
            gr_name.append(name)
            gr_quantity.append(quantity)
            gr_price.append(price)

        plt.bar(gr_name, gr_price)
        plt.show()
        engine.say('''
Great sales
    ''')
        engine.runAndWait()
        admin(c,mydb)
    else:
        print("Invalid choice. Returning to admin menu...")
        admin(c,mydb)
```

```
def product_config(c,mydb):
    print('''Select from below:
        a. Add a new product.
        b. Change price of a product.
        c. Show all products.
        d. Remove product from stock.
        e. Back to main menu.
        ''')

    engine.say('''choose what to do
        a. Add new product
        b. Change price of product
        c. Show all products
        d. Remove product from stock
        e. Back to main menu
        ''')
    engine.runAndWait()

    choice3 = input("Enter your choice: ")

    if choice3 == "a":
        pcode = int(input("Enter product code:"))
        c.execute("SELECT product_code FROM stock")
        existing_product_codes = [row[0] for row in c.fetchall()]

        if pcode not in existing_product_codes:
            pname = input("Enter product name:")
            quantity = int(input("Enter quantity:"))
            price = int(input("Enter the price:"))

            c.execute("INSERT INTO stock VALUES (%s, %s, %s, %s)", (pcode, pname, quantity, price))
            mydb.commit()

            print("Product added successfully.")
            product_cost = int(input("How much did this product cost? (each):"))
            total_cost = product_cost * quantity

            c.execute("SELECT * FROM cash")
            cash_balance, = c.fetchone()
            cash_balance -= total_cost

            c.execute("UPDATE cash SET cash = %s",(cash_balance,))
            mydb.commit()

            print("Cash updated successfully.")
            admin(c,mydb)
        else:
            print("Product code already exists. Returning to main menu...")
            admin(c,mydb)


    if choice3=="b":
        print("")
        pcode=input("Enter product code :")
        newprice=input("Enter new price :")
        c.execute("UPDATE stock SET price = %s WHERE product_code = %s", (newprice, pcode))
        mydb.commit()
        print("Price changed successfully! Returning to main menu ....  ")
        admin(c,mydb)

    if choice3=="c":
        c.execute("select * from stock")
        F="%15s %15s %15s %15s"
        print(F % ("product code", "  product name",  "quantity", "price"))
        print("="*125)
        for i in c:
            for j in i:
                print("%14s" % j, end=' ')
            print()
        print("="*125)
        admin(c,mydb)


    if choice3=="d":
        pcode=input("Enter product code of the product u want to delete :")
        y=input("Confirm? (Y/N) :")
        if y=="y"or"Y":
            c.execute("delete from stock where product_code= %s",(pcode,))
            mydb.commit()
            admin(c,mydb)
        else:
            print("Returning to main menu ...")
            admin(c,mydb)
    if choice3=="e":
        mainmenu()
```

Veiw accounts

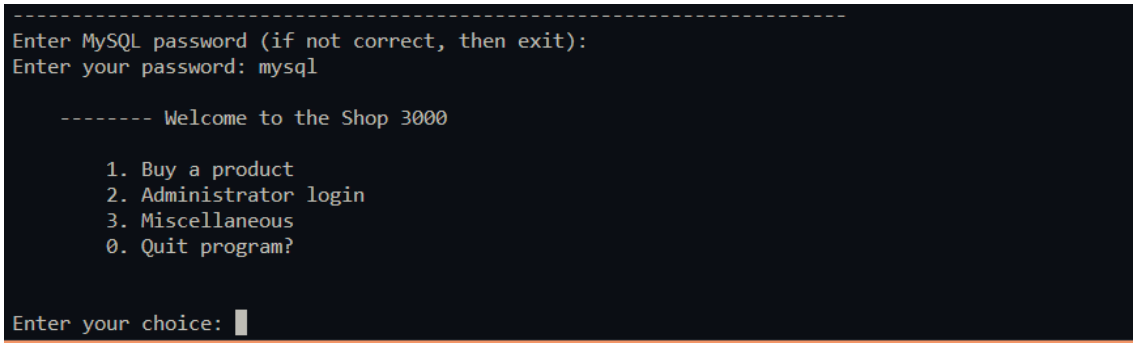
```
def view_accounts(c,mydb):
    print('''Select the Account
        a. Sales
        b. Products ''')

    choice4 = input("Enter your choice: ")

    if choice4 == "a":
        c.execute("SELECT * FROM purchases")
        print("Sales Account:")
        print("%15s %15s %15s %15s" % ("Date", "Customer name", "Product code", "Amount"))
        print("=" * 65)
        for row in c:
            print("%14s %14s %14s %14s" % row)
        print("=" * 65)

    elif choice4 == "b":
        c.execute("SELECT * FROM stock")
        print("Products Account:")
        print("%15s %15s %15s %15s" % ("Product code", "Product name", "Quantity", "Price"))
        print("=" * 65)
        for row in c:
            print("%14s %14s %14s %14s" % row)
        print("=" * 65)

    else:
        print("Invalid choice! Please try again.")
    admin(c,mydb)
```

__ main __ (actual program starts here)

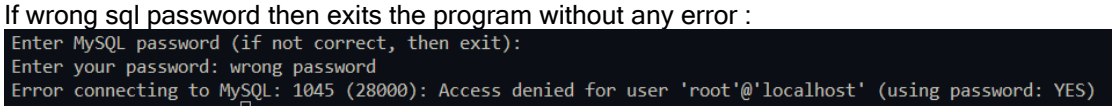
```
if __name__ == "__main__":
    engine = pyttsx3.init()
    engine.setProperty("rate", 5000)
    print("------------------------------------------------------------------------")
    print("Enter MySQL password (if not correct, then exit):")
    mysql=input("Enter your password: ")
    mydb, c = establish_connection(mysql)
    defpass = encrypt(encrypt(encrypt('mysql', mysql), mysql), mysql)
    c.execute("CREATE DATABASE IF NOT EXISTS shop3000")
    c.execute("USE shop3000")
    c.execute("SHOW TABLES")
    tables = c.fetchall()
    if (login,cash) not in tables:
        pretables(c,mydb)
        c.execute("insert into cash values()")
        c.execute("insert into login values()")
    mainmenu()
    aa=5
```

OUTPUT ----------

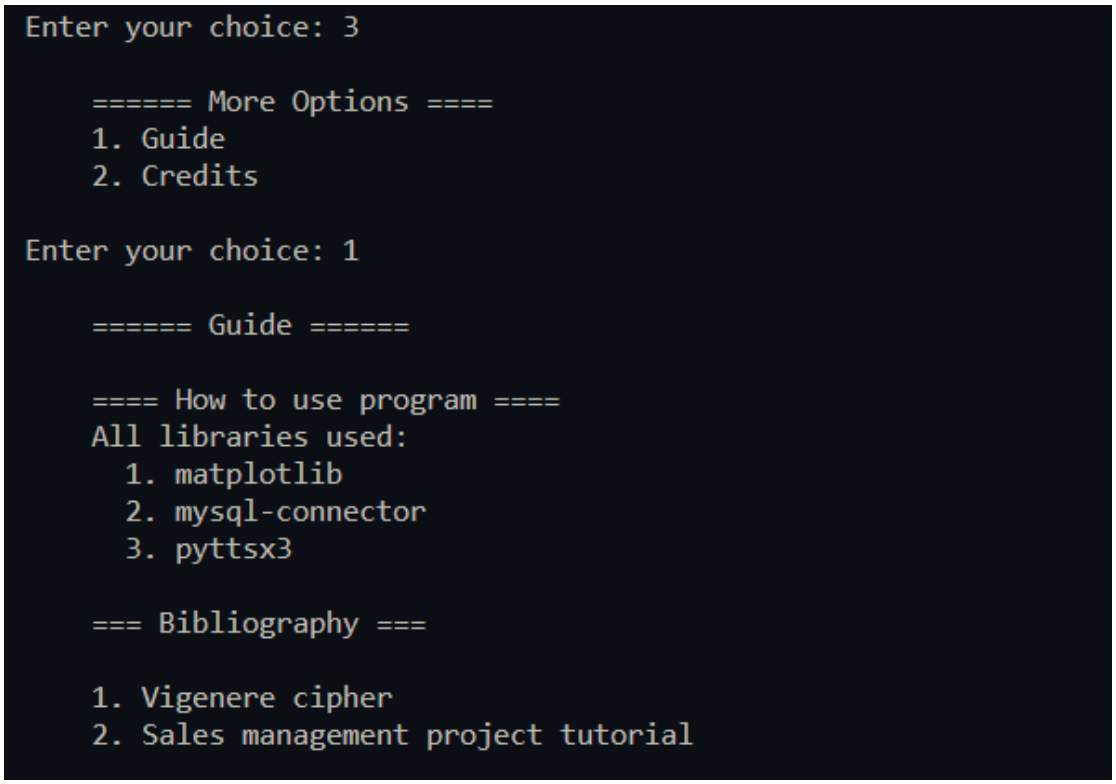
Mysql password to enter program and connect



If wrong sql password then exits the program without any error :



1.Miscelleanious :



Admin controls ----------

Password required to enter admin

If wrong password :

```
Enter your choice: 2
Enter administrator username: admin
Enter your password: wrongpassword
Login failed. attempts remaining = 2
Enter administrator username: admin
Enter your password: wrongpassword
Login failed. attempts remaining = 1
Enter administrator username: admin
Enter your password: wrongpassword
Login failed. attempts remaining = 0
Login attempts exhausted. Exiting...
```

If right password : encrypted password

```
Enter administrator username: admin
Enter your password: mysql
Login successful! Opening Administrator Controls...

    === Administrator controls ===

    a. Show graphs (LineGraph/PieChart..)
    b. Configure products (Add/Remove...) .
    c. Configure cash (Deposit/Withdrawal...).
    d. View accounts.
    e. Change admin password.
    f. Back to main menu.
```

```
mysql> select * from
+----------+---------
| username | password
+----------+---------
| admin    | wsums
```

Configure cash

```
Enter your choice: c
Balance = 0
How much cash to be withdrawn? (0 to skip): 0
How much cash to be deposited? : 1000
Balance of Cash available = 1000
```

```
mysql> select * from cash;
+------+
| cash |
+------+
| 1000 |
```

Configure products

Add product

```
Select from below:
        a. Add a new product.
        b. Change price of a product.
        c. Show all products.
        d. Remove product from stock.
        e. Back to main menu.

Enter your choice: a
Enter product code:101
Enter product name:producta
Enter quantity:10
Enter the price:2
Product added successfully.
How much did this product cost? (each):1
```

Change price of product

```
 Enter your choice: b

 Enter product code :101
 Enter new price :5
```

Show all products

```
Enter your choice: c
   product code    product name        quantity          price
=================================================================================
          101       producta             10             5
=================================================================================
```

Remove products

```
Enter your choice: d
Enter product code of the product u want to delete :101
Confirm? (Y/N) :y
```

Return to main menu

```
Enter your choice: e

    -------- Welcome to the Shop 3000

        1. Buy a product
        2. Administrator login
        3. Miscellaneous
        0. Quit program?
```

Change admin password

```
Enter the old password: mysql
Enter your new password: cat
```

See accounts

Sales :

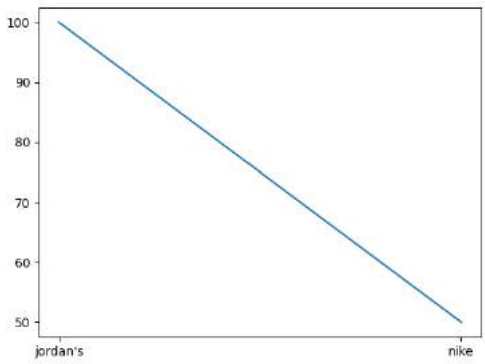
```
Enter your choice: d
Select the Account
        a. Sales
        b. Products
Enter your choice: a
Sales Account:
           Date    Customer name     Product code          Amount
=====================================================================
    2024-01-15         jenifer            102              10
    2024-01-28           insaf            103              10
    2024-01-28          dawood            102               3
=====================================================================
```

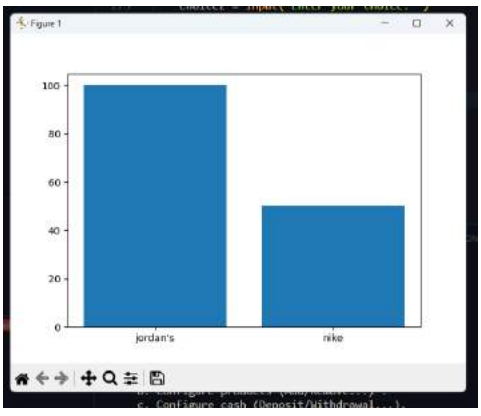
Stock :

```
Products Account:
   Product code     Product name        Quantity          Price
=====================================================================
           102        jordan's            100             100
           103            nike             50              50
=====================================================================
```

Show graph\

```
Which graph do you wish to see
        a. Line graph (quantity)
        b. Pie chart (items remaining)
        c. Price comparison graph
```





---------- buy something ----------

```
Enter your choice: 1

    1. Add a product to cart.
    2. Check out.
    3. View Available Items.
    4. Go to menu.
```

View available :

```
Enter your choice:3
   Product code    Product name      Items left           Price
=================================================================
          102       jordan's            100             100
          103           nike             50              50
=================================================================
```

Add item to cart

```
Enter your choice:1
Enter your name:jenifer
Enter product code:102
Enter product quantity:10
-------  Added to cart successfully!!. Returning to main menu.. ----------
```

Check out

```
Enter your choice:2
Amount to be paid -- 1000
Are you sure you want to buy? (y/n): y
```