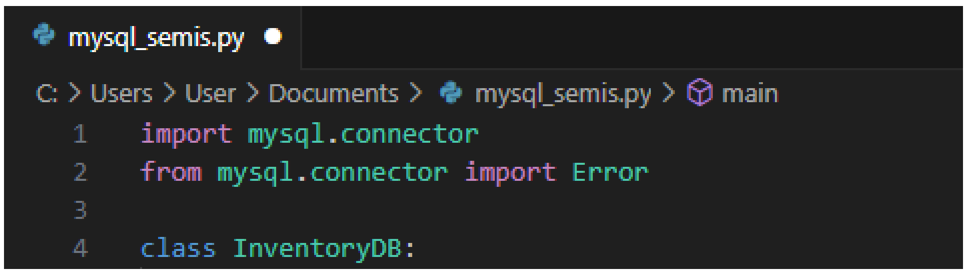
OUTPUT – AGUIRRE, PAUL VINCENT S.
COMPROG – CP102

PART 2: Python Application – CRUD Operations with MySQL

Develop a Python application that connects to your MySQL database and performs full CRUD (Create, Read, Update, Delete) operations using Object-Oriented Programming (OOP). Your application should meet the following requirements:

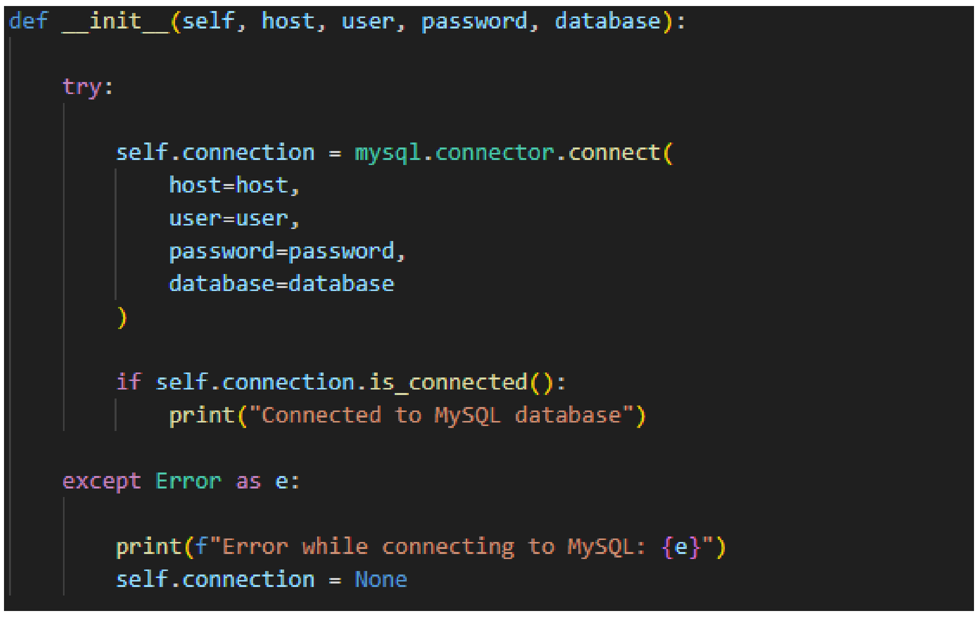
1. *Create a class that encapsulates methods for connecting to the database and performing add, edit, delete, and search operations on at least one table.*

➢ **CLASS;**



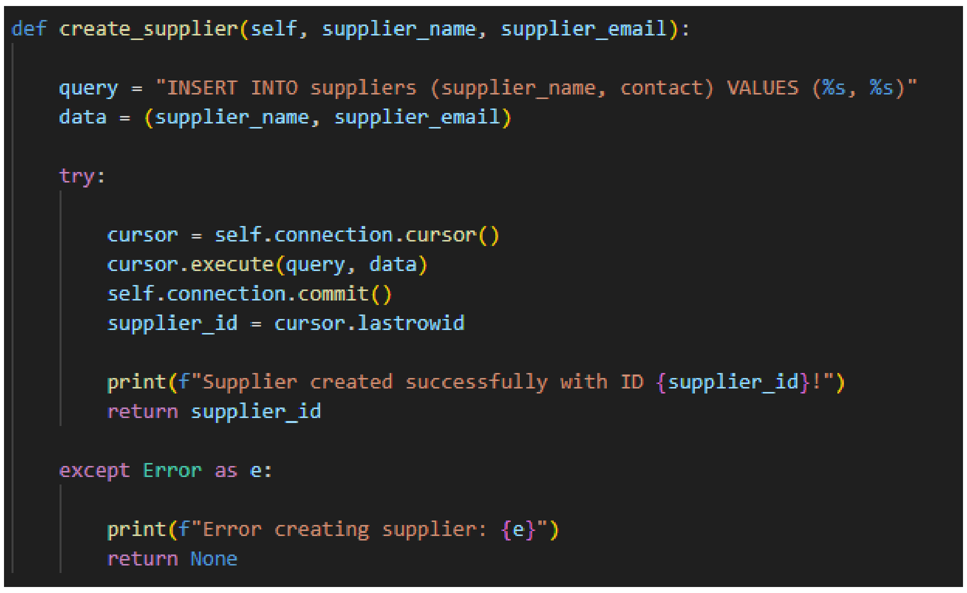
```
mysql_semis.py ●
C: > Users > User > Documents > mysql_semis.py > main
import mysql.connector
from mysql.connector import Error

class InventoryDB:
```

➢ **INITIALIZATION;**



```
def __init__(self, host, user, password, database):

    try:

        self.connection = mysql.connector.connect(
            host=host,
            user=user,
            password=password,
            database=database
        )

        if self.connection.is_connected():
            print("Connected to MySQL database")

    except Error as e:

        print(f"Error while connecting to MySQL: {e}")
        self.connection = None
```

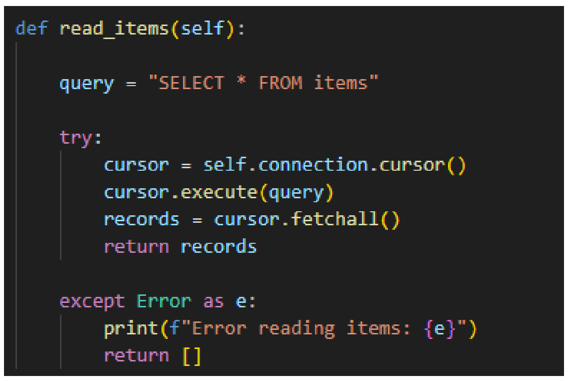
➢ **CREATING SUPLIER;**



```
def create_supplier(self, supplier_name, supplier_email):

    query = "INSERT INTO suppliers (supplier_name, contact) VALUES (%s, %s)"
    data = (supplier_name, supplier_email)

    try:

        cursor = self.connection.cursor()
        cursor.execute(query, data)
        self.connection.commit()
        supplier_id = cursor.lastrowid

        print(f"Supplier created successfully with ID {supplier_id}!")
        return supplier_id

    except Error as e:

        print(f"Error creating supplier: {e}")
        return None
```

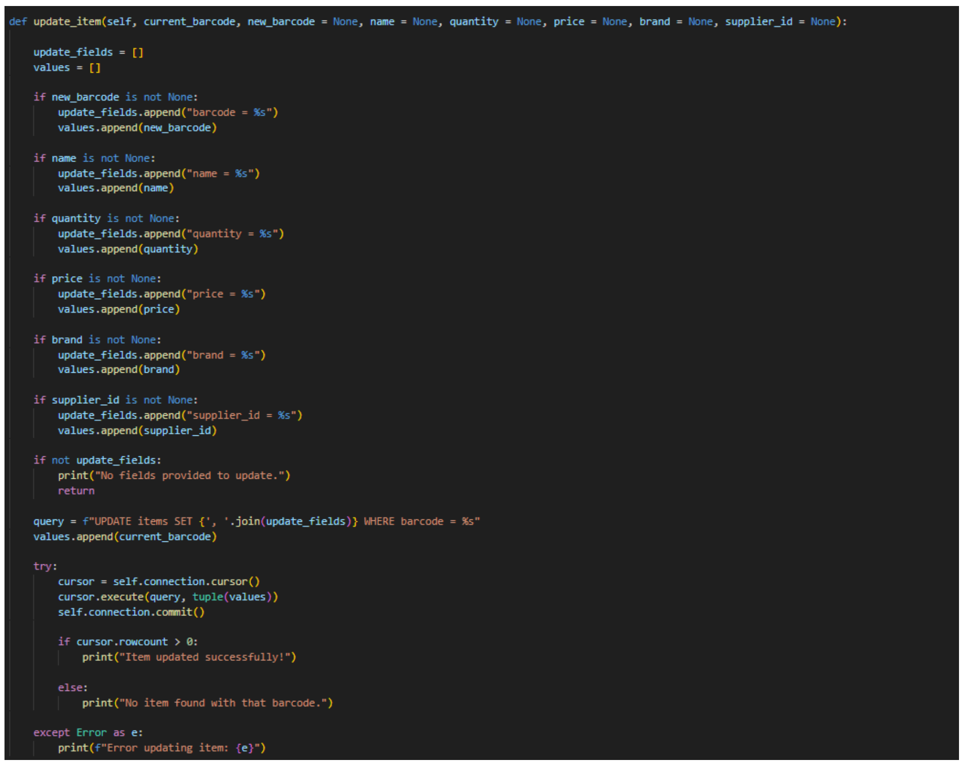
OUTPUT – AGUIRRE, PAUL VINCENT S.
COMPROG – CP102

- **CREATING AN ITEM;**



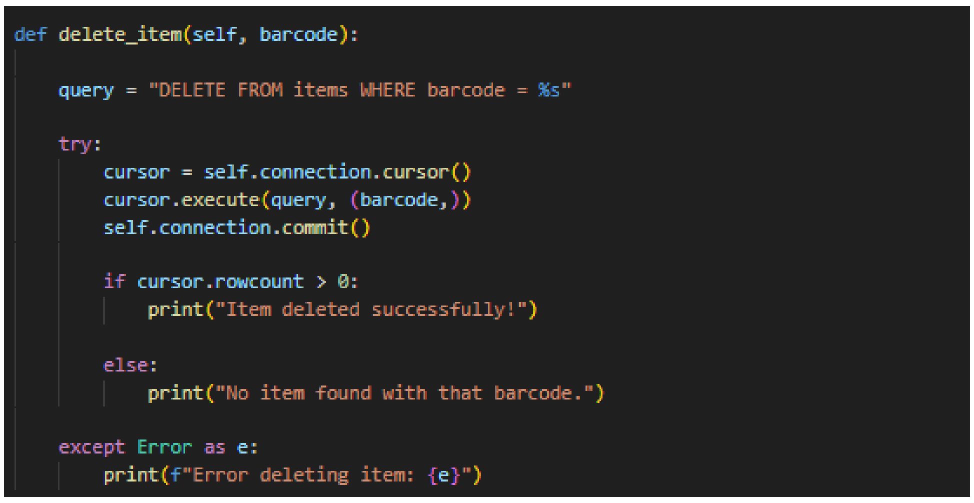
```python
def read_items(self):

    query = "SELECT * FROM items"

    try:
        cursor = self.connection.cursor()
        cursor.execute(query)
        records = cursor.fetchall()
        return records

    except Error as e:
        print(f"Error reading items: {e}")
        return []
```

- **READING / LIST ALL ITEM;**



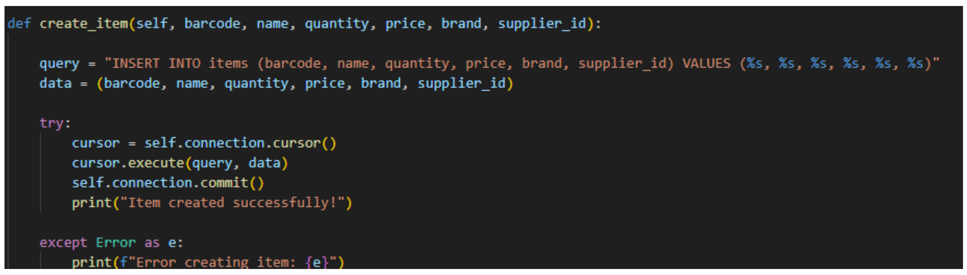
```python
def update_item(self, current_barcode, new_barcode = None, name = None, quantity = None, price = None, brand = None, supplier_id = None):

    update_fields = []
    values = []

    if new_barcode is not None:
        update_fields.append("barcode = %s")
        values.append(new_barcode)

    if name is not None:
        update_fields.append("name = %s")
        values.append(name)

    if quantity is not None:
        update_fields.append("quantity = %s")
        values.append(quantity)

    if price is not None:
        update_fields.append("price = %s")
        values.append(price)

    if brand is not None:
        update_fields.append("brand = %s")
        values.append(brand)

    if supplier_id is not None:
        update_fields.append("supplier_id = %s")
        values.append(supplier_id)

    if not update_fields:
        print("No fields provided to update.")
        return

    query = f"UPDATE items SET {', '.join(update_fields)} WHERE barcode = %s"
    values.append(current_barcode)

    try:
        cursor = self.connection.cursor()
        cursor.execute(query, tuple(values))
        self.connection.commit()

        if cursor.rowcount > 0:
            print("Item updated successfully!")

        else:
            print("No item found with that barcode.")

    except Error as e:
        print(f"Error updating item: {e}")
```

- **UPDATING AN ITEM;**



```python
def delete_item(self, barcode):

    query = "DELETE FROM items WHERE barcode = %s"

    try:
        cursor = self.connection.cursor()
        cursor.execute(query, (barcode,))
        self.connection.commit()

        if cursor.rowcount > 0:
            print("Item deleted successfully!")

        else:
            print("No item found with that barcode.")

    except Error as e:
        print(f"Error deleting item: {e}")
```

- **DELETING AN ITEM;**

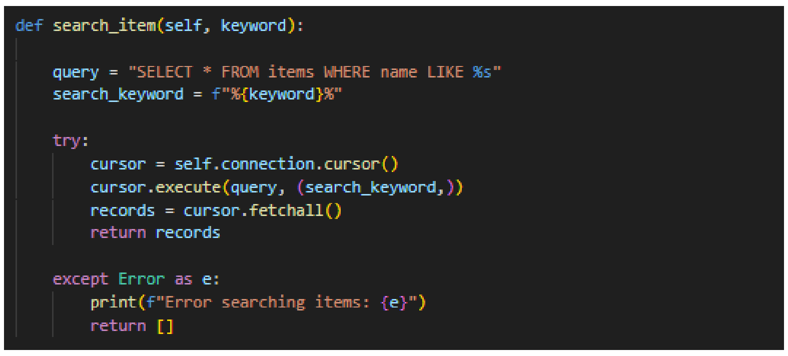


```python
def create_item(self, barcode, name, quantity, price, brand, supplier_id):

    query = "INSERT INTO items (barcode, name, quantity, price, brand, supplier_id) VALUES (%s, %s, %s, %s, %s, %s)"
    data = (barcode, name, quantity, price, brand, supplier_id)

    try:
        cursor = self.connection.cursor()
        cursor.execute(query, data)
        self.connection.commit()
        print("Item created successfully!")

    except Error as e:
        print(f"Error creating item: {e}")
```

-
-

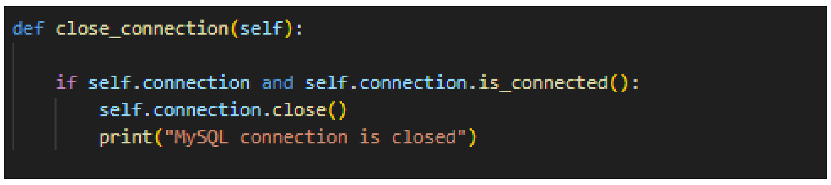
OUTPUT – AGUIRRE, PAUL VINCENT S.
COMPROG – CP102

- **SEARCHING AN ITEM;**



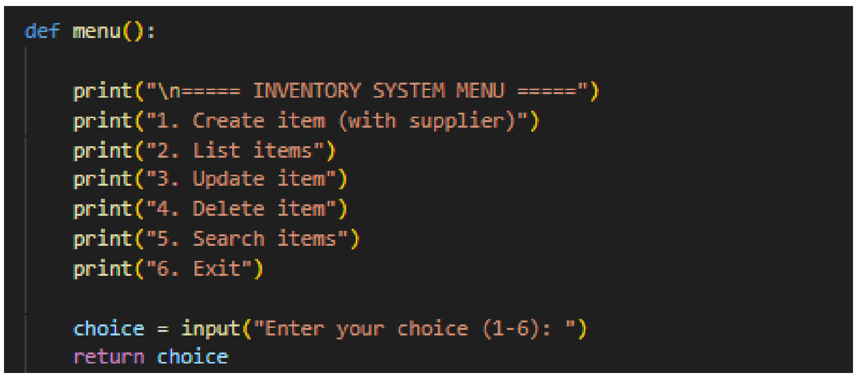
```python
def search_item(self, keyword):

    query = "SELECT * FROM items WHERE name LIKE %s"
    search_keyword = f"%{keyword}%"

    try:
        cursor = self.connection.cursor()
        cursor.execute(query, (search_keyword,))
        records = cursor.fetchall()
        return records

    except Error as e:
        print(f"Error searching items: {e}")
        return []
```

- **CLOSING THE CONNECTION;**



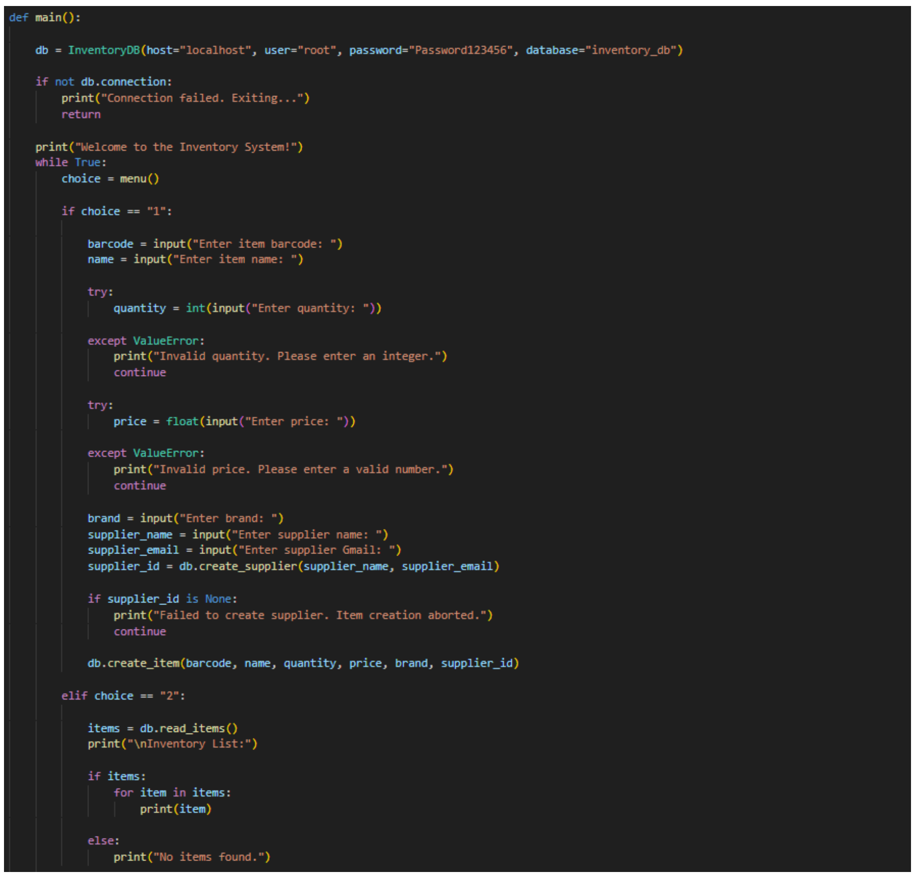
```python
def close_connection(self):

    if self.connection and self.connection.is_connected():
        self.connection.close()
        print("MySQL connection is closed")
```

- **MENU;**

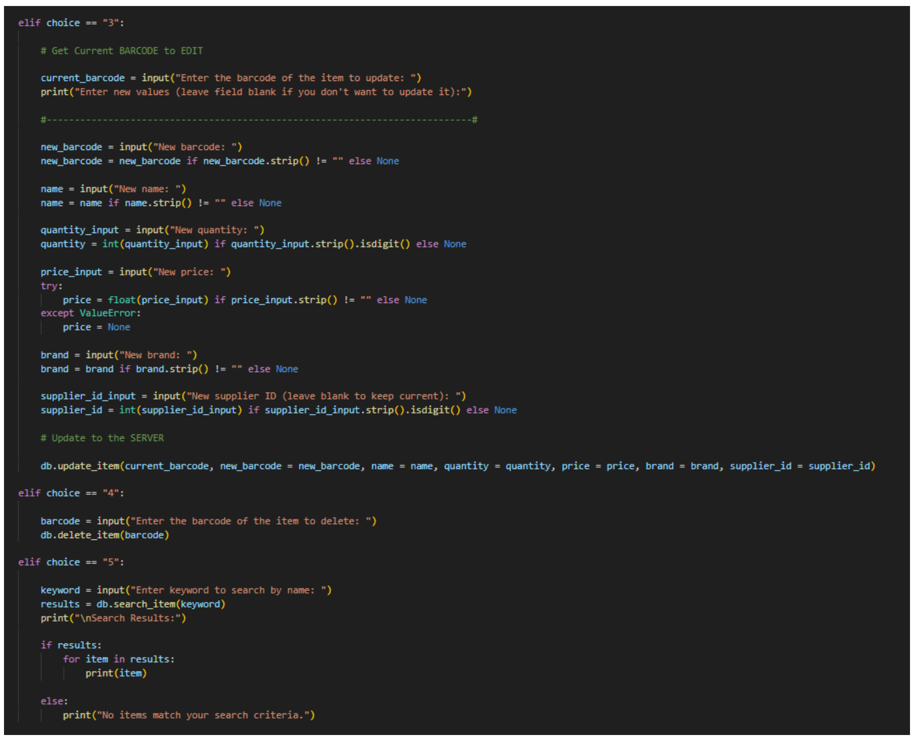


```python
def menu():

    print("\n===== INVENTORY SYSTEM MENU =====")
    print("1. Create item (with supplier)")
    print("2. List items")
    print("3. Update item")
    print("4. Delete item")
    print("5. Search items")
    print("6. Exit")

    choice = input("Enter your choice (1-6): ")
    return choice
```

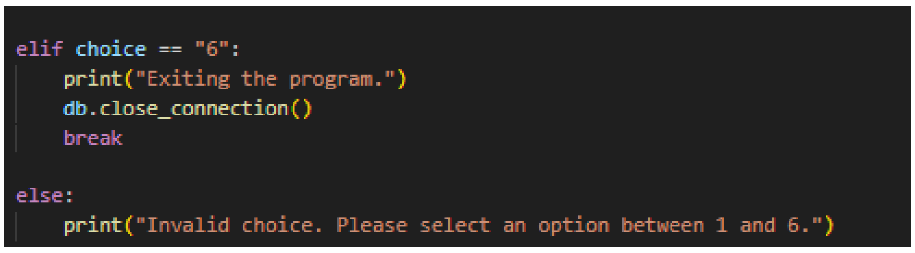
- **MAIN PROGRAM or UI;**



```python
def main():

    db = InventoryDB(host="localhost", user="root", password="Password123456", database="inventory_db")

    if not db.connection:
        print("Connection failed. Exiting...")
        return

    print("Welcome to the Inventory System!")
    while True:
        choice = menu()

        if choice == "1":

            barcode = input("Enter item barcode: ")
            name = input("Enter item name: ")

            try:
                quantity = int(input("Enter quantity: "))

            except ValueError:
                print("Invalid quantity. Please enter an integer.")
                continue

            try:
                price = float(input("Enter price: "))

            except ValueError:
                print("Invalid price. Please enter a valid number.")
                continue

            brand = input("Enter brand: ")
            supplier_name = input("Enter supplier name: ")
            supplier_email = input("Enter supplier Gmail: ")
            supplier_id = db.create_supplier(supplier_name, supplier_email)

            if supplier_id is None:
                print("Failed to create supplier. Item creation aborted.")
                continue

            db.create_item(barcode, name, quantity, price, brand, supplier_id)

        elif choice == "2":

            items = db.read_items()
            print("\nInventory List:")

            if items:
                for item in items:
                    print(item)

            else:
                print("No items found.")
```

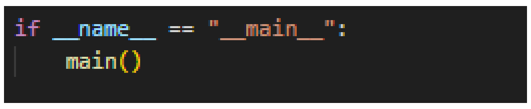


```python
elif choice == "3":

    # Get Current BARCODE to EDIT

    current_barcode = input("Enter the barcode of the item to update: ")
    print("Enter new values (leave field blank if you don't want to update it):")

    #-----------------------------------------------------------------------#

    new_barcode = input("New barcode: ")
    new_barcode = new_barcode if new_barcode.strip() != "" else None

    name = input("New name: ")
    name = name if name.strip() != "" else None

    quantity_input = input("New quantity: ")
    quantity = int(quantity_input) if quantity_input.strip().isdigit() else None

    price_input = input("New price: ")
    try:
        price = float(price_input) if price_input.strip() != "" else None
    except ValueError:
        price = None

    brand = input("New brand: ")
    brand = brand if brand.strip() != "" else None

    supplier_id_input = input("New supplier ID (leave blank to keep current): ")
    supplier_id = int(supplier_id_input) if supplier_id_input.strip().isdigit() else None

    # Update to the SERVER

    db.update_item(current_barcode, new_barcode = new_barcode, name = name, quantity = quantity, price = price, brand = brand, supplier_id = supplier_id)

elif choice == "4":

    barcode = input("Enter the barcode of the item to delete: ")
    db.delete_item(barcode)

elif choice == "5":

    keyword = input("Enter keyword to search by name: ")
    results = db.search_item(keyword)
    print("\nSearch Results:")

    if results:
        for item in results:
            print(item)

    else:
        print("No items match your search criteria.")
```



```python
elif choice == "6":
    print("Exiting the program.")
    db.close_connection()
    break

else:
    print("Invalid choice. Please select an option between 1 and 6.")
```

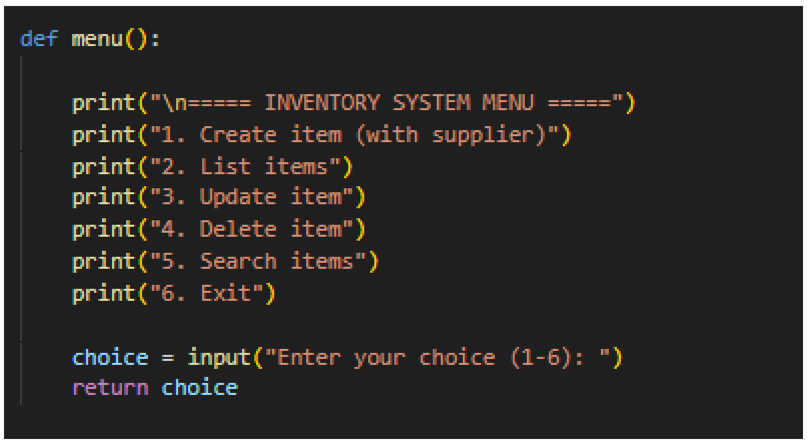
➢ **START;**



```python
if __name__ == "__main__":
    main()
```

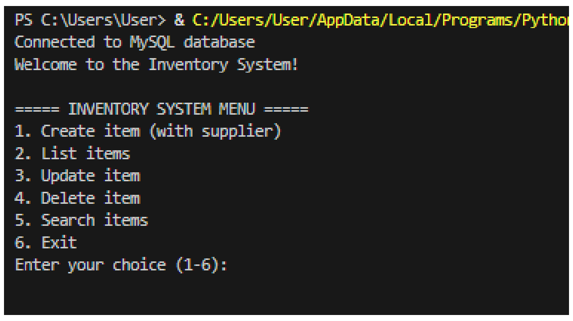
2. *Implement a main program that presents a text-based menu to the user and invokes the appropriate class methods based on the selected option.*

- **MENU;**



```python
def menu():

    print("\n===== INVENTORY SYSTEM MENU =====")
    print("1. Create item (with supplier)")
    print("2. List items")
    print("3. Update item")
    print("4. Delete item")
    print("5. Search items")
    print("6. Exit")

    choice = input("Enter your choice (1-6): ")
    return choice
```

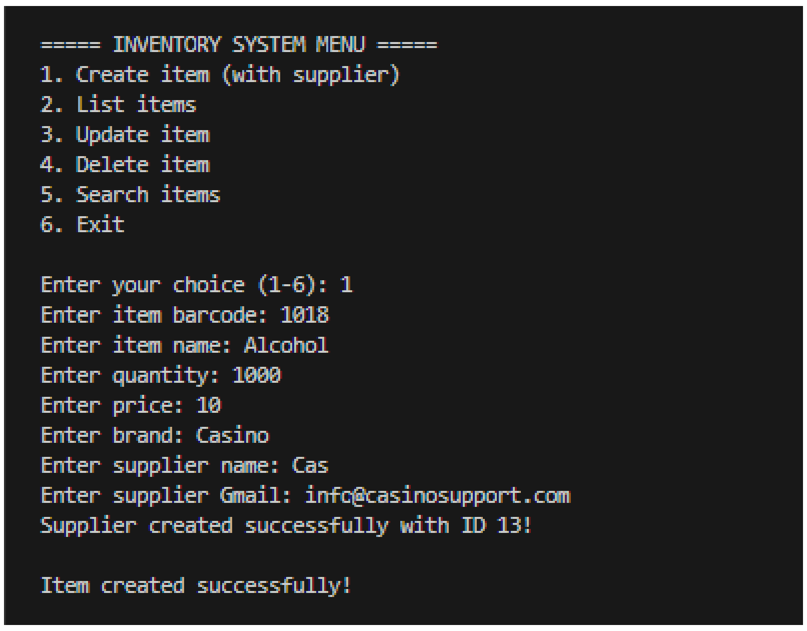
- **OUTPUT;**



```
PS C:\Users\User> & C:/Users/User/AppData/Local/Programs/Pytho
Connected to MySQL database
Welcome to the Inventory System!

===== INVENTORY SYSTEM MENU =====
1. Create item (with supplier)
2. List items
3. Update item
4. Delete item
5. Search items
6. Exit
Enter your choice (1-6):
```

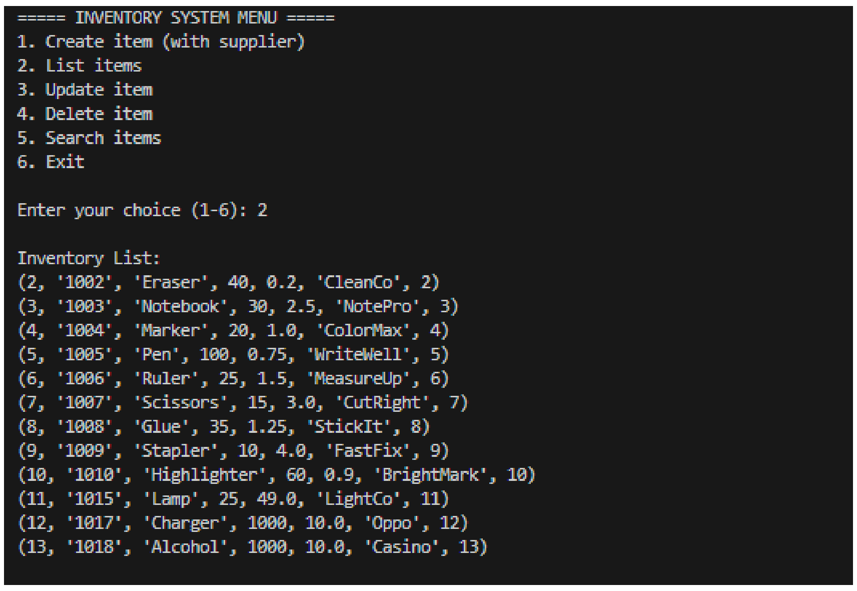
OUTPUT – AGUIRRE, PAUL VINCENT S.
COMPROG – CP102

- **CREATING AN ITEM;**



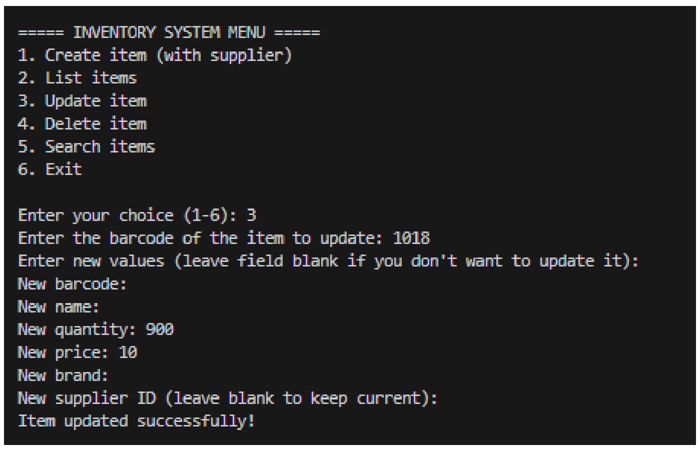
```
===== INVENTORY SYSTEM MENU =====
1. Create item (with supplier)
2. List items
3. Update item
4. Delete item
5. Search items
6. Exit

Enter your choice (1-6): 1
Enter item barcode: 1018
Enter item name: Alcohol
Enter quantity: 1000
Enter price: 10
Enter brand: Casino
Enter supplier name: Cas
Enter supplier Gmail: info@casinosupport.com
Supplier created successfully with ID 13!

Item created successfully!
```

- **LISTING OUT ALL ITEMS;**

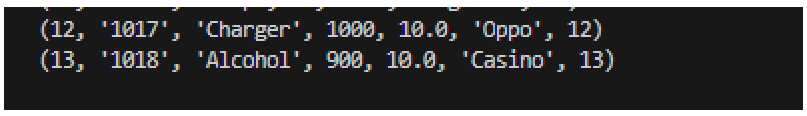


```
===== INVENTORY SYSTEM MENU =====
1. Create item (with supplier)
2. List items
3. Update item
4. Delete item
5. Search items
6. Exit

Enter your choice (1-6): 2

Inventory List:
(2, '1002', 'Eraser', 40, 0.2, 'CleanCo', 2)
(3, '1003', 'Notebook', 30, 2.5, 'NotePro', 3)
(4, '1004', 'Marker', 20, 1.0, 'ColorMax', 4)
(5, '1005', 'Pen', 100, 0.75, 'WriteWell', 5)
(6, '1006', 'Ruler', 25, 1.5, 'MeasureUp', 6)
(7, '1007', 'Scissors', 15, 3.0, 'CutRight', 7)
(8, '1008', 'Glue', 35, 1.25, 'StickIt', 8)
(9, '1009', 'Stapler', 10, 4.0, 'FastFix', 9)
(10, '1010', 'Highlighter', 60, 0.9, 'BrightMark', 10)
(11, '1015', 'Lamp', 25, 49.0, 'LightCo', 11)
(12, '1017', 'Charger', 1000, 10.0, 'Oppo', 12)
(13, '1018', 'Alcohol', 1000, 10.0, 'Casino', 13)
```

- **UPDATING AN ITEM;**



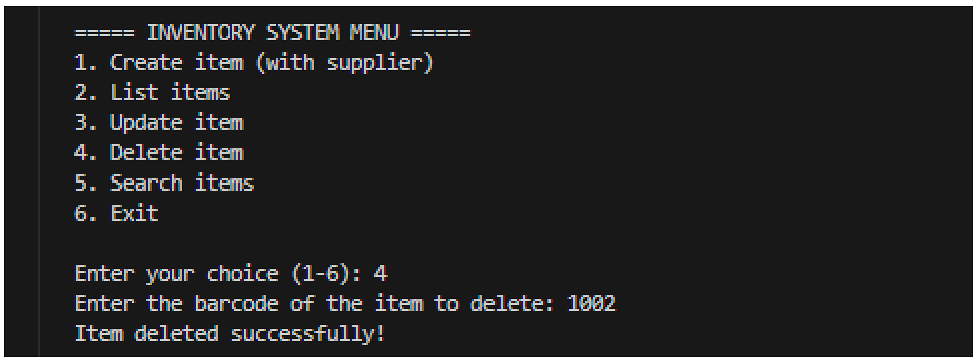
```
===== INVENTORY SYSTEM MENU =====
1. Create item (with supplier)
2. List items
3. Update item
4. Delete item
5. Search items
6. Exit

Enter your choice (1-6): 3
Enter the barcode of the item to update: 1018
Enter new values (leave field blank if you don't want to update it):
New barcode:
New name:
New quantity: 900
New price: 10
New brand:
New supplier ID (leave blank to keep current):
Item updated successfully!
```



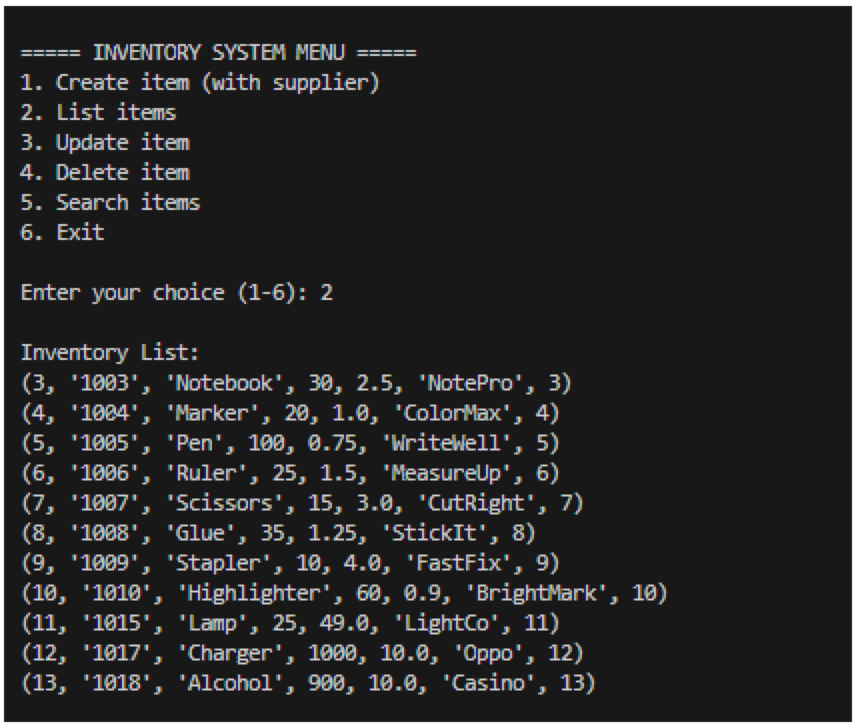
```
(12, '1017', 'Charger', 1000, 10.0, 'Oppo', 12)
(13, '1018', 'Alcohol', 900, 10.0, 'Casino', 13)
```

OUTPUT – AGUIRRE, PAUL VINCENT S.
COMPROG – CP102

- **DELETING AN ITEM;**

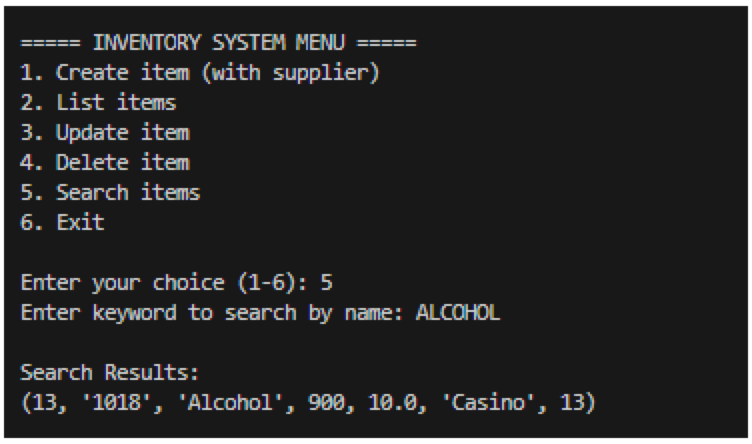


```
===== INVENTORY SYSTEM MENU =====
1. Create item (with supplier)
2. List items
3. Update item
4. Delete item
5. Search items
6. Exit

Enter your choice (1-6): 4
Enter the barcode of the item to delete: 1002
Item deleted successfully!
```



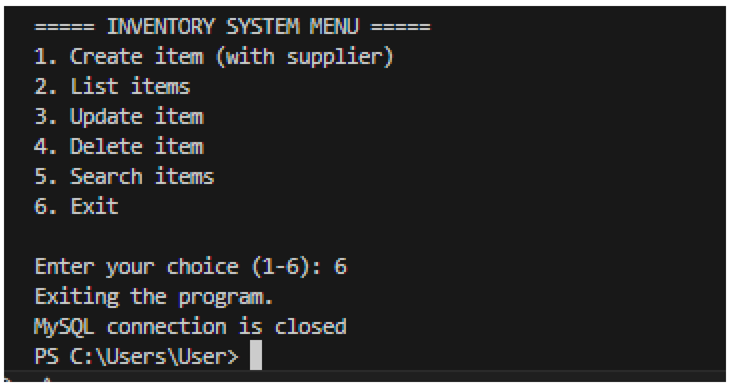
```
===== INVENTORY SYSTEM MENU =====
1. Create item (with supplier)
2. List items
3. Update item
4. Delete item
5. Search items
6. Exit

Enter your choice (1-6): 2

Inventory List:
(3, '1003', 'Notebook', 30, 2.5, 'NotePro', 3)
(4, '1004', 'Marker', 20, 1.0, 'ColorMax', 4)
(5, '1005', 'Pen', 100, 0.75, 'WriteWell', 5)
(6, '1006', 'Ruler', 25, 1.5, 'MeasureUp', 6)
(7, '1007', 'Scissors', 15, 3.0, 'CutRight', 7)
(8, '1008', 'Glue', 35, 1.25, 'StickIt', 8)
(9, '1009', 'Stapler', 10, 4.0, 'FastFix', 9)
(10, '1010', 'Highlighter', 60, 0.9, 'BrightMark', 10)
(11, '1015', 'Lamp', 25, 49.0, 'LightCo', 11)
(12, '1017', 'Charger', 1000, 10.0, 'Oppo', 12)
(13, '1018', 'Alcohol', 900, 10.0, 'Casino', 13)
```

- **SEARCHING AN ITEM;**



```
===== INVENTORY SYSTEM MENU =====
1. Create item (with supplier)
2. List items
3. Update item
4. Delete item
5. Search items
6. Exit

Enter your choice (1-6): 5
Enter keyword to search by name: ALCOHOL

Search Results:
(13, '1018', 'Alcohol', 900, 10.0, 'Casino', 13)
```

- **EXITING;**



```
===== INVENTORY SYSTEM MENU =====
1. Create item (with supplier)
2. List items
3. Update item
4. Delete item
5. Search items
6. Exit

Enter your choice (1-6): 6
Exiting the program.
MySQL connection is closed
PS C:\Users\User>
```

OUTPUT – AGUIRRE, PAUL VINCENT S.
COMPROG – CP102

SOURCES CODE;

```python
import mysql.connector
from mysql.connector import Error

class InventoryDB:
    def __init__(self, host, user, password, database):

        try:

            self.connection = mysql.connector.connect(
                host=host,
                user=user,
                password=password,
                database=database
            )

            if self.connection.is_connected():
                print("Connected to MySQL database")

        except Error as e:

            print(f"Error while connecting to MySQL: {e}")
            self.connection = None

    def create_supplier(self, supplier_name, supplier_email):

        query = "INSERT INTO suppliers (supplier_name, contact) VALUES (%s, %s)"
        data = (supplier_name, supplier_email)

        try:

            cursor = self.connection.cursor()
            cursor.execute(query, data)
            self.connection.commit()
            supplier_id = cursor.lastrowid

            print(f"Supplier created successfully with ID {supplier_id}!")
            return supplier_id

        except Error as e:

            print(f"Error creating supplier: {e}")
            return None

    def create_item(self, barcode, name, quantity, price, brand, supplier_id):

        query = "INSERT INTO items (barcode, name, quantity, price, brand, supplier_id) VALUES (%s, %s, %s, %s, %s, %s)"
        data = (barcode, name, quantity, price, brand, supplier_id)

        try:
            cursor = self.connection.cursor()
            cursor.execute(query, data)
            self.connection.commit()
            print("\nItem created successfully!")

        except Error as e:
            print(f"Error creating item: {e}")
    def read_items(self):

        query = "SELECT * FROM items"

        try:
            cursor = self.connection.cursor()
            cursor.execute(query)
            records = cursor.fetchall()
            return records

        except Error as e:
            print(f"Error reading items: {e}")
            return []

    def update_item(self, current_barcode, new_barcode = None, name = None, quantity = None,
price = None, brand = None, supplier_id = None):

        update_fields = []
        values = []

        if new_barcode is not None:
            update_fields.append("barcode = %s")
            values.append(new_barcode)

        if name is not None:
            update_fields.append("name = %s")
            values.append(name)

        if quantity is not None:
            update_fields.append("quantity = %s")
            values.append(quantity)

        if price is not None:
            update_fields.append("price = %s")
            values.append(price)

        if brand is not None:
            update_fields.append("brand = %s")
            values.append(brand)

        if supplier_id is not None:
            update_fields.append("supplier_id = %s")
            values.append(supplier_id)

        if not update_fields:
            print("No fields provided to update.")
            return

        query = f"UPDATE items SET {', '.join(update_fields)} WHERE barcode = %s"
        values.append(current_barcode)

        try:
            cursor = self.connection.cursor()
            cursor.execute(query, tuple(values))
            self.connection.commit()

            if cursor.rowcount > 0:
                print("Item updated successfully!")

            else:
                print("No item found with that barcode.")

        except Error as e:
            print(f"Error updating item: {e}")

    def delete_item(self, barcode):

        query = "DELETE FROM items WHERE barcode = %s"

        try:
            cursor = self.connection.cursor()
            cursor.execute(query, (barcode,))
            self.connection.commit()

            if cursor.rowcount > 0:
                print("Item deleted successfully!")

            else:
                print("No item found with that barcode.")

        except Error as e:
            print(f"Error deleting item: {e}")

    def search_item(self, keyword):

        query = "SELECT * FROM items WHERE name LIKE %s"
        search_keyword = f"%{keyword}%"

        try:
            cursor = self.connection.cursor()
            cursor.execute(query, (search_keyword,))
            records = cursor.fetchall()
            return records

        except Error as e:
            print(f"Error searching items: {e}")
            return []

    def close_connection(self):

        if self.connection and self.connection.is_connected():
            self.connection.close()
            print("MySQL connection is closed")


def menu():

    print("\n===== INVENTORY SYSTEM MENU =====")
    print("1. Create item (with supplier)")
    print("2. List items")
    print("3. Update item")
    print("4. Delete item")
    print("5. Search items")
    print("6. Exit")

    choice = input("\nEnter your choice (1-6): ")

    return choice
```

OUTPUT – AGUIRRE, PAUL VINCENT S.
COMPROG – CP102

```python
def main():

    db = InventoryDB(host="localhost", user="root", password="Password123456",
database="inventory_db")

    if not db.connection:
        print("Connection failed. Exiting...")
        return

    print("Welcome to the Inventory System!")
    while True:
        choice = menu()

        if choice == "1":

            barcode = input("Enter item barcode: ")
            name = input("Enter item name: ")

            try:
                quantity = int(input("Enter quantity: "))

            except ValueError:
                print("Invalid quantity. Please enter an integer.")
                continue

            try:
                price = float(input("Enter price: "))

            except ValueError:
                print("Invalid price. Please enter a valid number.")
                continue

            brand = input("Enter brand: ")
            supplier_name = input("Enter supplier name: ")
            supplier_email = input("Enter supplier Gmail: ")
            supplier_id = db.create_supplier(supplier_name, supplier_email)

            if supplier_id is None:
                print("Failed to create supplier. Item creation aborted.")
                continue

            db.create_item(barcode, name, quantity, price, brand, supplier_id)

        elif choice == "2":

            items = db.read_items()
            print("\nInventory List:")

            if items:
                for item in items:
                    print(item)

            else:
                print("No items found.")

        elif choice == "3":

            # Get Current BARCODE to EDIT

            current_barcode = input("Enter the barcode of the item to update: ")
```

OUTPUT – AGUIRRE, PAUL VINCENT S.
COMPROG – CP102

```
            print("Enter new values (leave field blank if you don't want to update it):")

            #-----------------------------------------------------------------------------#

            new_barcode = input("New barcode: ")
            new_barcode = new_barcode if new_barcode.strip() != "" else None

            name = input("New name: ")
            name = name if name.strip() != "" else None

            quantity_input = input("New quantity: ")
            quantity = int(quantity_input) if quantity_input.strip().isdigit() else None

            price_input = input("New price: ")
            try:
                price = float(price_input) if price_input.strip() != "" else None
            except ValueError:
                price = None

            brand = input("New brand: ")
            brand = brand if brand.strip() != "" else None

            supplier_id_input = input("New supplier ID (leave blank to keep current): ")
            supplier_id = int(supplier_id_input) if supplier_id_input.strip().isdigit() else None

            # Update to the SERVER

            db.update_item(current_barcode, new_barcode = new_barcode, name = name, quantity = quantity, price = price, brand = brand, supplier_id = supplier_id)

        elif choice == "4":

            barcode = input("Enter the barcode of the item to delete: ")
            db.delete_item(barcode)

        elif choice == "5":

            keyword = input("Enter keyword to search by name: ")
            results = db.search_item(keyword)
            print("\nSearch Results:")

            if results:
                for item in results:
                    print(item)

            else:
                print("No items match your search criteria.")


        elif choice == "6":
            print("Exiting the program.")
            db.close_connection()
            break

        else:
            print("Invalid choice. Please select an option between 1 and 6.")

if __name__ == "__main__":
    main()
```