

Lab assignment-6.4

Name: Neela.Sai shivathika

Enroll no: 2403A54112

Batch: 18

TASK 1:

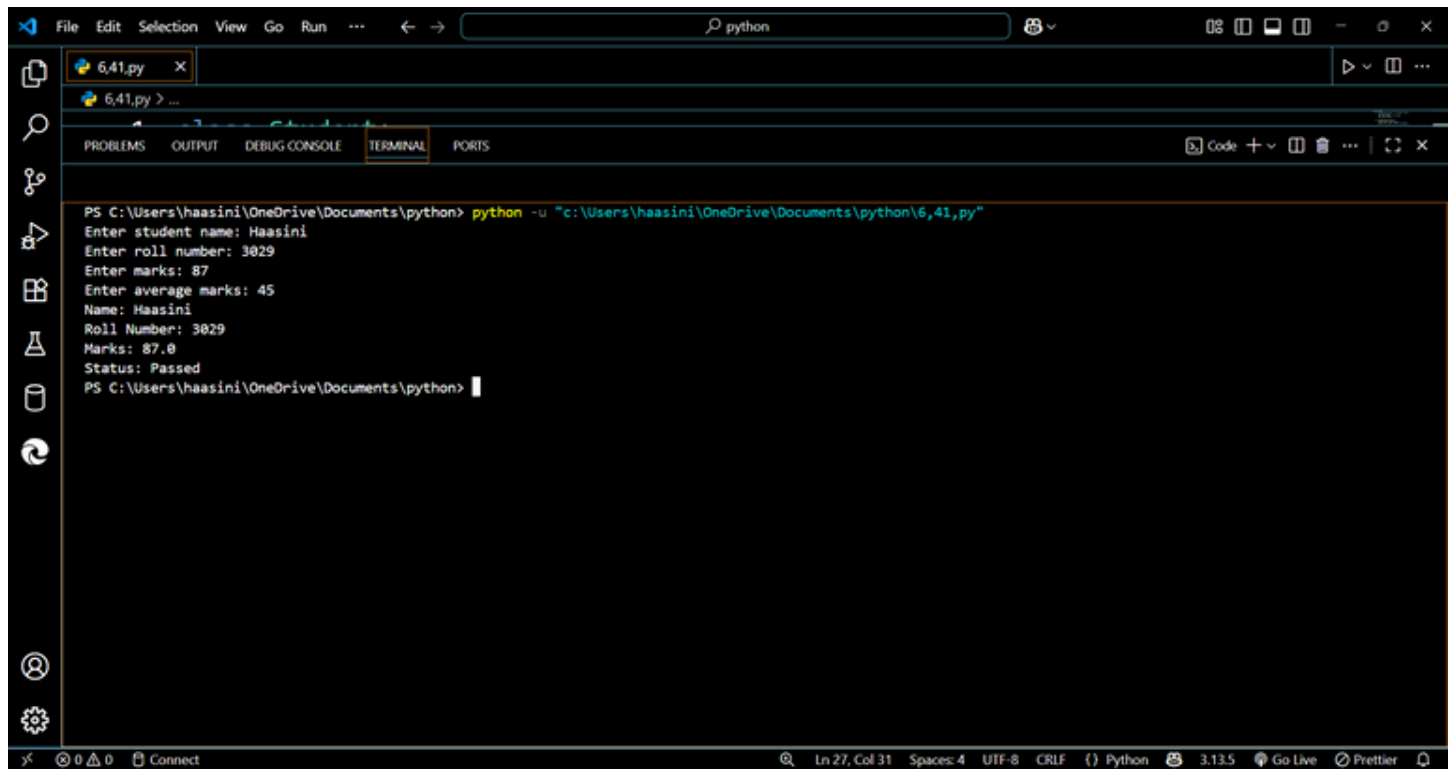
Prompt: Write a python program with attributes name, roll_number and marks using methods display_details() and is_passed() for demonstrating the marks and check whether the marks are greater than average.

Code:

A screenshot of a Python IDE window titled 'python'. The code defines a 'Student' class with attributes 'name', 'roll_number', and 'marks'. It includes two methods: 'display_details()' which prints the student's information, and 'is_passed(average)' which checks if the student's marks are greater than a provided average. An example usage is shown at the bottom, where user input is used to create a student object and call these methods.

```
1 class Student:
2     def __init__(self, name, roll_number, marks):
3         self.name = name
4         self.roll_number = roll_number
5         self.marks = marks
6
7     def display_details(self):
8         print(f"Name: {self.name}")
9         print(f"Roll Number: {self.roll_number}")
10        print(f"Marks: {self.marks}")
11
12    def is_passed(self, average):
13        if self.marks > average:
14            print("Status: Passed")
15        else:
16            print("Status: Failed")
17
18 # Example usage
19 if __name__ == "__main__":
20     name = input("Enter student name: ")
21     roll_number = input("Enter roll number: ")
22     marks = float(input("Enter marks: "))
23     average = float(input("Enter average marks: "))
24
25     student = Student(name, roll_number, marks)
26     student.display_details()
27     student.is_passed(average)
```

Output:



```
PS C:\Users\haasini\OneDrive\Documents\python> python -u "c:\Users\haasini\OneDrive\Documents\python\6,41.py"
Enter student name: Haasini
Enter roll number: 3029
Enter marks: 87
Enter average marks: 45
Name: Haasini
Roll Number: 3029
Marks: 87.0
Status: Passed
PS C:\Users\haasini\OneDrive\Documents\python>
```

Explanation

This program defines a **Student class** with attributes:

- name → student's name
- roll_number → student's roll number
- marks → student's marks

Methods inside the class:

1. display_details(self) → Prints the student's details (name, roll number, and marks).
2. is_passed(self, average) → Checks if the student's marks are greater than or equal to the given average.
 - If yes → prints **"Status: Passed"**
 - Else → prints **"Status: Failed"**

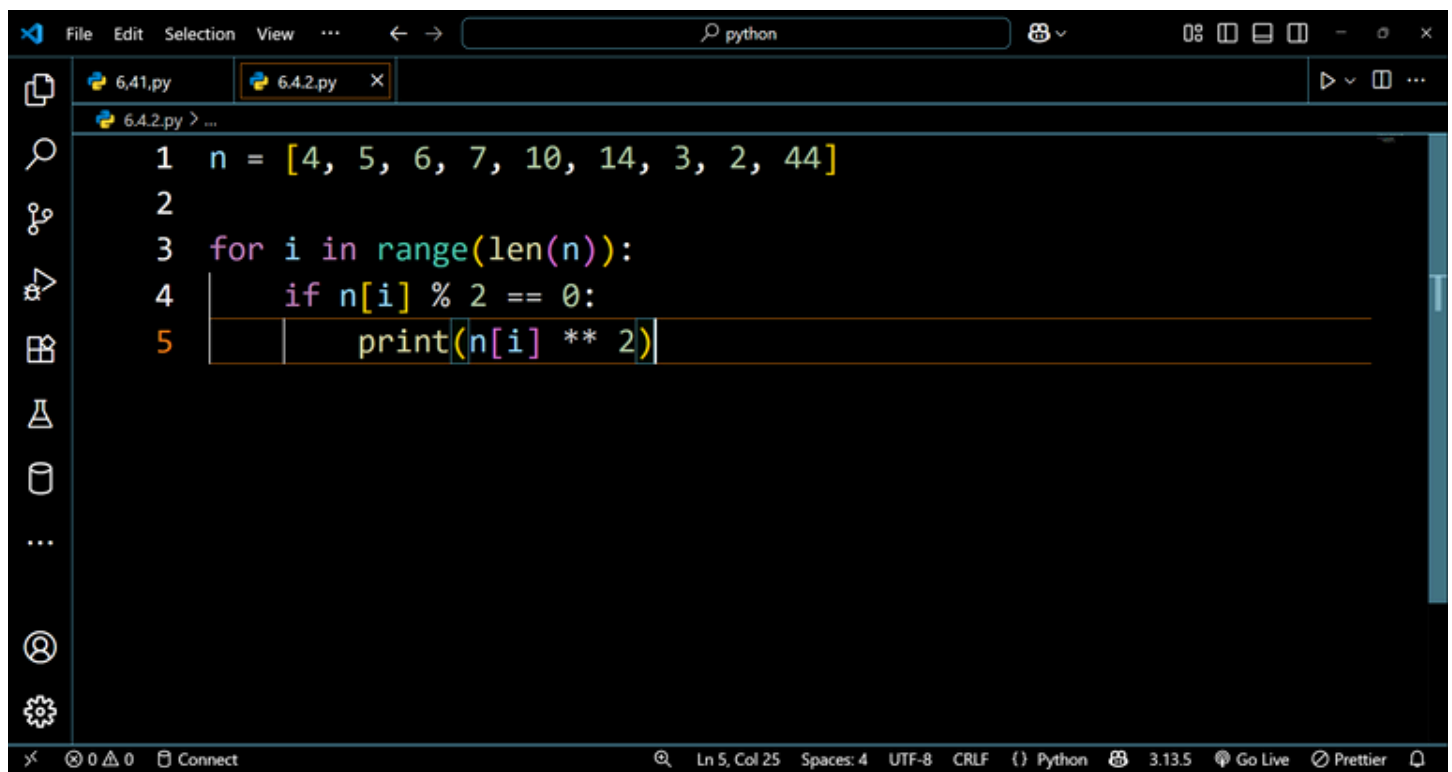
Main program flow:

- Takes user input for student's name, roll number, marks, and the average marks.
- Creates a Student object with this data.
- Calls display_details() to show student information.
- Calls is_passed(average) to check whether the student passed or failed.

TASK 2:

Prompt: n=[4,5,6,7,10,14,3,2,44] for(i=0;i<n;i++) calculate and print only squares of even numbers only using conditional logic if (n%2==0)

Code:

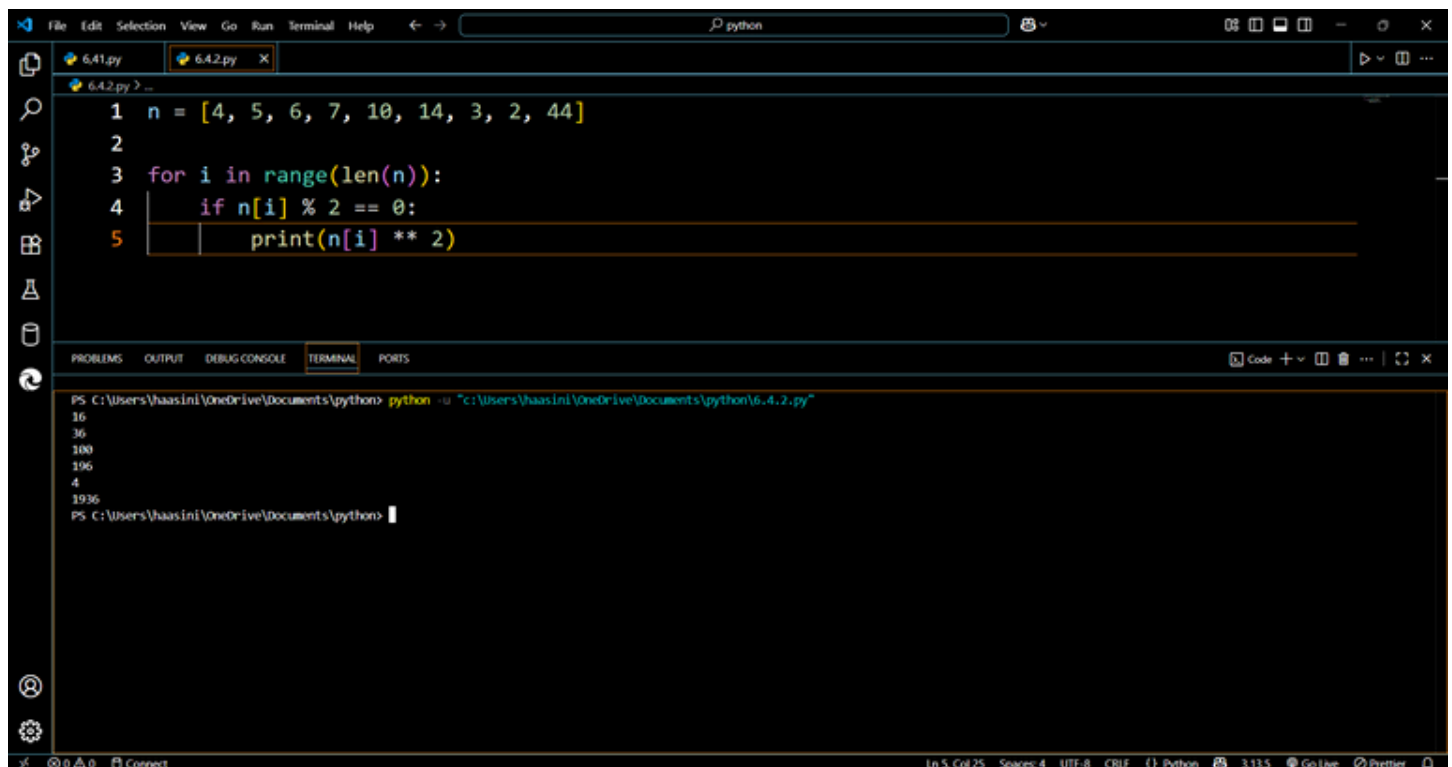


The screenshot shows the Visual Studio Code editor with a Python file named 6.4.2.py. The code is as follows:

```
1 n = [4, 5, 6, 7, 10, 14, 3, 2, 44]
2
3 for i in range(len(n)):
4     if n[i] % 2 == 0:
5         print(n[i] ** 2)
```

The status bar at the bottom indicates the current position is Line 5, Column 25, with 4 spaces, UTF-8 encoding, CRLF line endings, Python 3.13.5 interpreter, and Prettier formatting.

Output:



The screenshot shows the same VS Code editor with the terminal window open. The command executed is `python -u "c:\Users\haasini\OneDrive\Documents\python\6.4.2.py"`. The output of the script is displayed in the terminal:

```
16
36
100
196
4
1936
```

The status bar at the bottom is identical to the first screenshot.

Explanation:

- A list `n` of numbers are taken.
- The loop goes through each index of the list.
- The if condition checks if the element is **even** (`n[i] % 2 == 0`).
- If true → it prints the **square** of that number (`n[i] ** 2`).

TASK 3:

Prompt:

write a python program with class called bank account having attributes account_holder and balance with methods for deposit(), withdraw(), and check for insufficient balance prevent from overdrawing.

Code:

A screenshot of a Python IDE window titled 'python'. The editor shows a file named '6.6.3.py' containing the following code:

```
1 class BankAccount:
2     def __init__(self, account_holder, balance=0.0):
3         self.account_holder = account_holder
4         self.balance = balance
5
6     def deposit(self, amount):
7         if amount > 0:
8             self.balance += amount
9             print(f"Deposited ${amount:.2f}. New balance: ${self.balance:.2f}")
10        else:
11            print("Deposit amount must be positive.")
12
13    def withdraw(self, amount):
14        if amount <= 0:
15            print("Withdrawal amount must be positive.")
16        elif amount > self.balance:
17            print("Insufficient balance. Withdrawal denied.")
18        else:
19            self.balance -= amount
20            print(f"Withdrew ${amount:.2f}. New balance: ${self.balance:.2f}")
21
22    def get_balance(self):
23        return self.balance
24
25 # Example usage:
26 if __name__ == "__main__":
27     account = BankAccount("Alice", 100)
28     account.deposit(50)
29     account.withdraw(30)
30     account.withdraw(150)
31     print(f"Final balance: ${account.get_balance():.2f}")
```

Output:

```
1 class BankAccount:
2     def __init__(self, account_holder, balance=0.0):
3         self.account_holder = account_holder
4         self.balance = balance
5
6     def deposit(self, amount):
7         if amount > 0:
8             self.balance += amount
```

Terminal Output:

```
PS C:\Users\haasini\OneDrive\Documents\python> python -u "c:\Users\haasini\OneDrive\Documents\python\6.4.3.py"
Deposited $50.00, New balance: $150.00
Withdraw $30.00, New balance: $120.00
Insufficient balance. Withdrawal denied.
Final balance: $120.00
PS C:\Users\haasini\OneDrive\Documents\python>
```

Explanation

This program defines a **BankAccount class** that simulates a simple bank account.

- **Attributes:**
 - `account_holder` → Name of the account holder.
 - `balance` → Stores the account balance (default is 0.0).
- **Methods:**
 - `deposit(amount)` → Adds money if the amount is positive.
 - `withdraw(amount)` → Subtracts money if amount is valid and balance is enough.
 - `get_balance()` → Returns the current balance.
- **Example Usage:**
 - Creates an account for **Alice** with \$100.
 - Deposits \$50 → balance becomes \$150.
 - Withdraws \$30 → balance becomes \$120.
 - Attempts to withdraw \$150 → denied because of insufficient funds.
 - Prints final balance = **\$120.00**.

TASK 4:

Prompt:

```
students = [  
    {"name": "Haasini", "score": 92},  
  
    {"name": "Rahul", "score": 85},  
  
    {"name": "Priya", "score": 78},  
  
    {"name": "Arjun", "score": 88},  
  
    {"name": "Meena", "score": 65},
```

```
{ "name": "Vikram", "score": 55},  
  
{ "name": "Sita", "score": 72},  
  
{ "name": "Ravi", "score": 60}  
  
]
```

Print the students who scored more than 75 using while loop

Code:



```
1 students = [  
2     { "name": "Haasini", "score": 92},  
3     { "name": "Rahul", "score": 85},  
4     { "name": "Priya", "score": 78},  
5     { "name": "Arjun", "score": 88},  
6     { "name": "Meena", "score": 65},  
7     { "name": "Vikram", "score": 55},  
8     { "name": "Sita", "score": 72},  
9     { "name": "Ravi", "score": 60}  
10 ]  
11  
12 i = 0  
13 while i < len(students):  
14     if students[i]["score"] > 75:  
15         print(students[i]["name"])  
16     i += 1
```

Output:

The screenshot shows a VS Code editor with a Python file named `6.4.4.py`. The code defines a list of student dictionaries:

```
1 students = [  
2     {"name": "Haasini", "score": 92},  
3     {"name": "Rahul", "score": 85},  
4     {"name": "Priya", "score": 78},  
5     {"name": "Arjun", "score": 88},  
6     {"name": "Meena", "score": 65},  
7     {"name": "Vikram", "score": 55},  
8     {"name": "Sita", "score": 72}.  
]
```

The terminal shows the command `python -u "c:\Users\haasini\OneDrive\Documents\python\6.4.4.py"` being executed, resulting in the output:

```
Haasini  
Rahul  
Priya  
Arjun  
PS C:\Users\haasini\OneDrive\Documents\python>
```

Explanation:

The screenshot shows a VS Code editor with a Python file named `6.4.4.py`. The code defines a list of student dictionaries and a loop that iterates over them:

```
4 {"name": "Priya", "score": 78},  
5 {"name": "Arjun", "score": 88},  
6 {"name": "Meena", "score": 65},  
7 {"name": "Vikram", "score": 55},  
8 {"name": "Sita", "score": 72},  
9 {"name": "Rahul", "score": 85},  
10 ]  
11  
12 i = 0  
13 while i < len(students):  
14     if students[i]:  
15         print(students[i])  
16     i += 1
```

The chat window on the right explains the `NameError` that occurs when running the code. It states that the variable `i` is not defined before it is used in the `while` loop. The chat window also provides a summary of the error and an example of proper usage:

This tells Python you intend to use the global variable, not create a new local one.

short explanation also

used /explain (rerun without)

It looks like your code only contains the single variable name `i`.

This by itself does nothing in Python—it just references the variable `i`, but since it's not assigned a value or used in any operation, it will cause a `NameError` if you try to run it.

Summary:

- `i` is just a variable name.
- Without assignment (like `i = 0`) or usage, it has no effect.
- If you run this, Python will say `NameError: name 'i' is not defined` unless `i` was defined earlier.

Example of proper usage:

```
# Assign a value to i  
i = 5  
print(i) # This will print 5
```

TASK 5:

Prompt: Begin writing a class ShoppingCart with an empty items list. Add methods to `add_item`, `remove_item`, and use a loop to calculate the total bill with conditional discounts.

Code:

```
File Edit Selection View Go Run Terminal Help python
class ShoppingCart:
    def __init__(self):
        # Start with an empty list
        self.items = []

    def add_item(self, name, price):
        """Add an item with name and price to the cart."""
        self.items.append({"name": name, "price": price})
        print(f"{name} added to cart.")

    def remove_item(self, name):
        """Remove an item if it exists."""
        for item in self.items:
            if item["name"] == name:
                self.items.remove(item)
                print(f"{name} removed from cart.")
                return
        print(f"{name} not found in cart.")

    def calculate_total(self):
        """Calculate total with discount if applicable."""
        total = 0
        for item in self.items:
            total += item["price"]

        if total > 100: # Conditional discount
            discount = total * 0.1
            total -= discount
            print(f"Discount of 10% applied: -(discount)")

        return total

    def view_cart(self):
        """View items in the cart."""
        if not self.items:
            print("Cart is empty.")
        else:
```

```
File Edit Selection View Go Run Terminal Help python
class ShoppingCart:
    def view_cart(self):
        else:
            print("Items in cart:")
            for i, item in enumerate(self.items, 1):
                print(f"{i}. {item['name']} - ${item['price']}")

# --- Main Program with User Input ---
cart = ShoppingCart()

while True:
    print("\n--- Shopping Cart Menu ---")
    print("1. Add item")
    print("2. Remove item")
    print("3. View cart")
    print("4. Calculate total")
    print("5. Exit")

    choice = input("Enter your choice: ")

    if choice == "1":
        name = input("Enter item name: ")
        price = float(input("Enter item price: "))
        cart.add_item(name, price)

    elif choice == "2":
        name = input("Enter item name to remove: ")
        cart.remove_item(name)

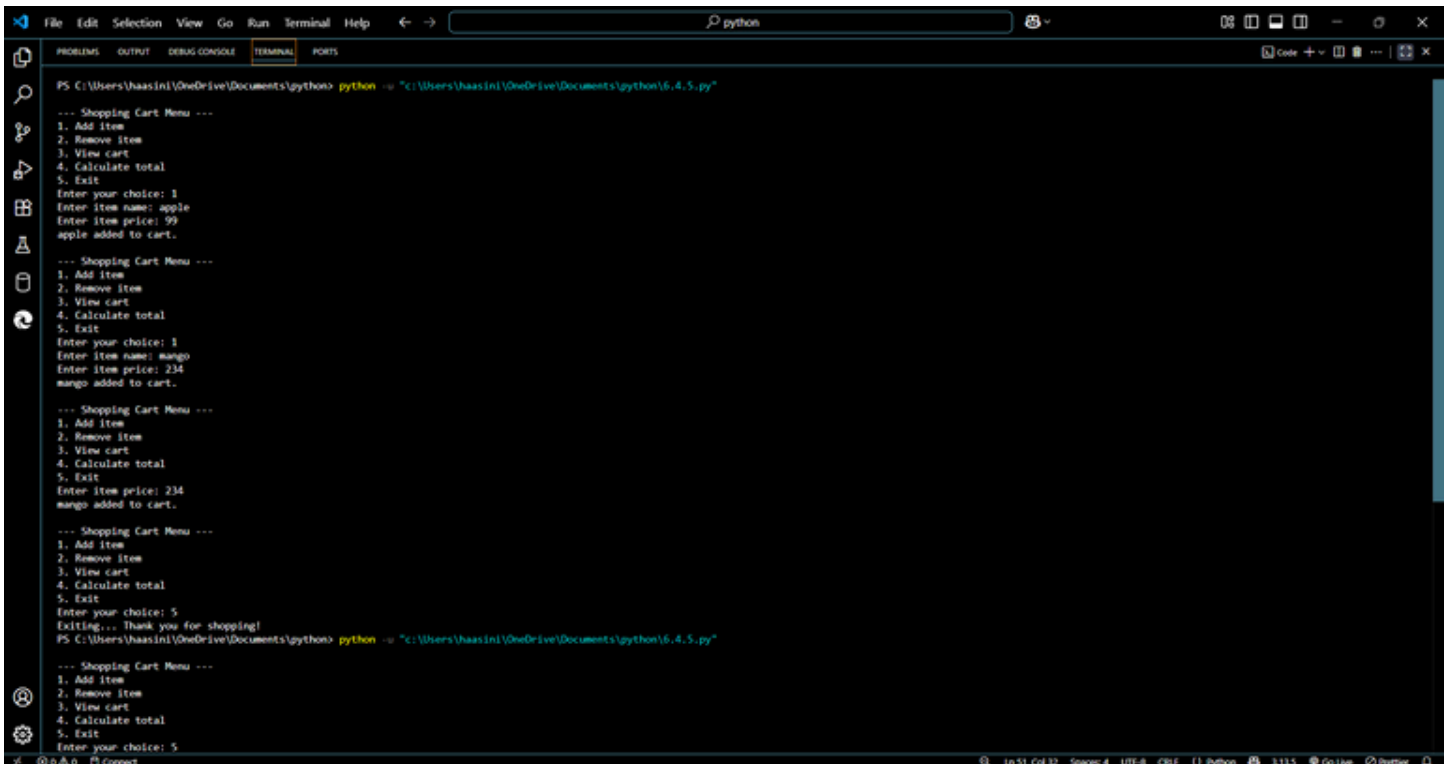
    elif choice == "3":
        cart.view_cart()

    elif choice == "4":
        total = cart.calculate_total()
        print("Total Bill: $", total)
```




```
File Edit Selection View Go Run Terminal Help python
72 elif choice == "5":
73     print("Exiting... Thank you for shopping!")
74     break
75
76 else:
77     print("Invalid choice. Please try again.")
78
```

Output:



```
PS C:\Users\haasini\OneDrive\Documents\python> python -p "c:\Users\haasini\OneDrive\Documents\python\6.4.5.py"

--- Shopping Cart Menu ---
1. Add item
2. Remove item
3. View cart
4. Calculate total
5. Exit
Enter your choice: 1
Enter item name: apple
Enter item price: 99
apple added to cart.

--- Shopping Cart Menu ---
1. Add item
2. Remove item
3. View cart
4. Calculate total
5. Exit
Enter your choice: 1
Enter item name: mango
Enter item price: 234
mango added to cart.

--- Shopping Cart Menu ---
1. Add item
2. Remove item
3. View cart
4. Calculate total
5. Exit
Enter item price: 234
mango added to cart.

--- Shopping Cart Menu ---
1. Add item
2. Remove item
3. View cart
4. Calculate total
5. Exit
Enter your choice: 5
Exiting... Thank you for shopping!
PS C:\Users\haasini\OneDrive\Documents\python> python -p "c:\Users\haasini\OneDrive\Documents\python\6.4.5.py"

--- Shopping Cart Menu ---
1. Add item
2. Remove item
3. View cart
4. Calculate total
5. Exit
Enter your choice: 5
```

Explanation:

This program creates a **ShoppingCart** class that lets users manage items in a cart using a menu-driven system.

- **Class Methods:**
 - `add_item(name, price)`: Adds an item (name + price) to the cart.
 - `remove_item(name)`: Removes an item if it exists in the cart.

- view_cart(): Displays all items in the cart.
- calculate_total(): Loops through items, calculates the total bill, and applies a **10% discount if total > 100**.
- **Main Program (Menu):** Uses a **while loop** to repeatedly show a menu with options:
 - Add item
 - Remove item
 - View cart
 - Calculate total
 - Exit

The user gives input, and the program calls the appropriate method.

- **Conditional logic:** If the total bill is above 100, a discount is applied. Otherwise, the total is shown without discount.