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:

```python
class Student:
    def __init__(self, name, roll_number, marks):
        self.name = name
        self.roll_number = roll_number
        self.marks = marks

    def display_details(self):
        print(f"Name: {self.name}")
        print(f"Roll Number: {self.roll_number}")
        print(f"Marks: {self.marks}")

    def is_passed(self, average):
        if self.marks > average:
            print("Status: Passed")
        else:
            print("Status: Failed")

# Example usage
if __name__ == "__main__":
    name = input("Enter student name: ")
    roll_number = input("Enter roll number: ")
    marks = float(input("Enter marks: "))
    average = float(input("Enter average marks: "))

    student = Student(name, roll_number, marks)
    student.display_details()
    student.is_passed(average)
```

Output:

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:

```python
n = [4, 5, 6, 7, 10, 14, 3, 2, 44]

for i in range(len(n)):
    if n[i] % 2 == 0:
        print(n[i] ** 2)
```

Output:

```python
n = [4, 5, 6, 7, 10, 14, 3, 2, 44]

for i in range(len(n)):
    if n[i] % 2 == 0:
        print(n[i] ** 2)
```

```
PS C:\Users\haasini\OneDrive\Documents\python> python -u "c:\Users\haasini\OneDrive\Documents\python\6.4.2.py"
16
36
100
196
4
1936
PS C:\Users\haasini\OneDrive\Documents\python>
```

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:

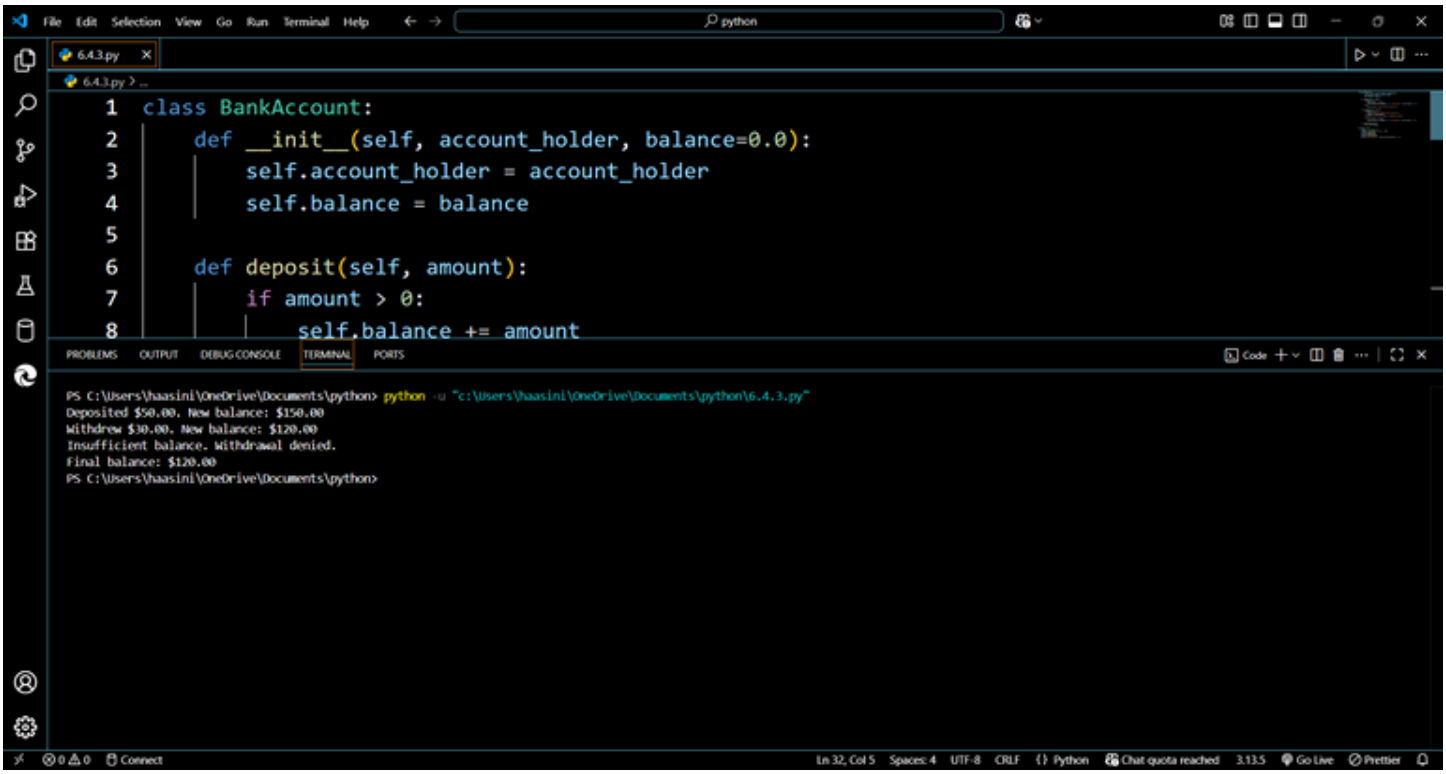
Prompt:

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:

```python
class BankAccount:
    def __init__(self, account_holder, balance=0.0):
        self.account_holder = account_holder
        self.balance = balance

    def deposit(self, amount):
        if amount > 0:
            self.balance += amount
            print(f"Deposited ${amount:.2f}. New balance: ${self.balance:.2f}")
        else:
            print("Deposit amount must be positive.")

    def withdraw(self, amount):
        if amount <= 0:
            print("Withdrawal amount must be positive.")
        elif amount > self.balance:
            print("Insufficient balance. Withdrawal denied.")
        else:
            self.balance -= amount
            print(f"Withdrew ${amount:.2f}. New balance: ${self.balance:.2f}")

    def get_balance(self):
        return self.balance

# Example usage:
if __name__ == "__main__":
    account = BankAccount("Alice", 100)
    account.deposit(50)
    account.withdraw(30)
    account.withdraw(150)
    print(f"Final balance: ${account.get_balance():.2f}")
```

Output:



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}

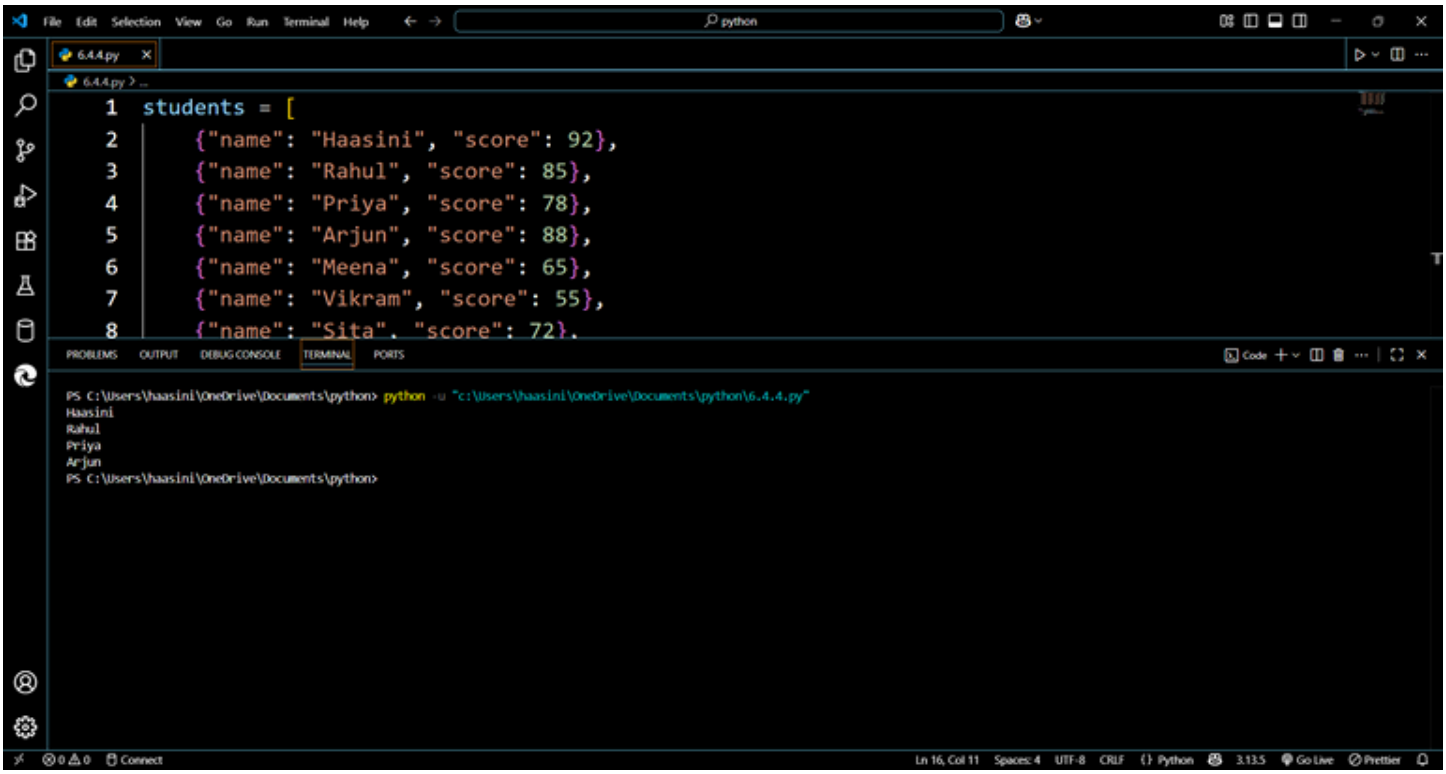
]

Print the students who scored more than 75 using while loop

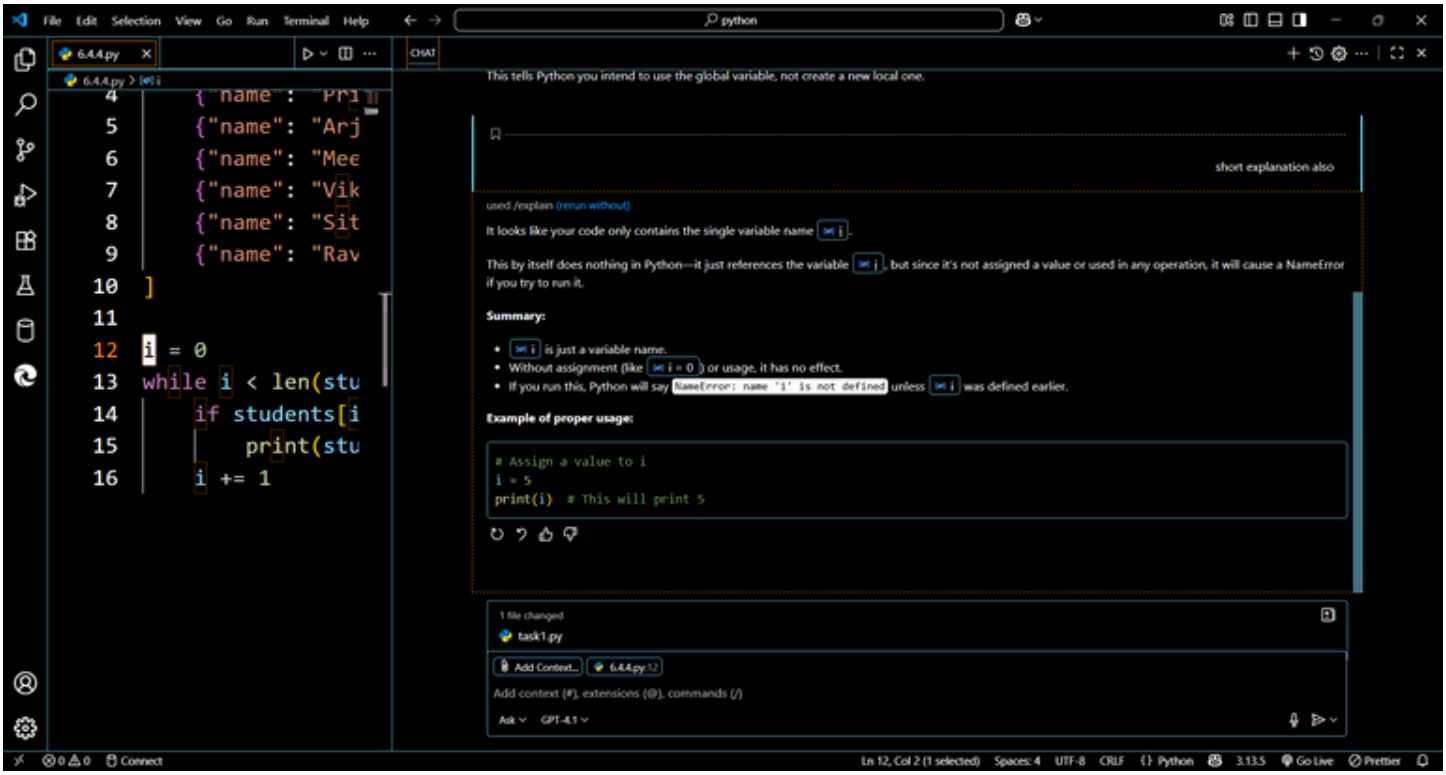
Code:

```python
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}
]

i = 0
while i < len(students):
    if students[i]["score"] > 75:
        print(students[i]["name"])
    i += 1
```

Output:



Explanation:



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:

```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:
```

```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)
```

```python
    elif choice == "5":
        print("Exiting... Thank you for shopping!")
        break

    else:
        print("Invalid choice. Please try again.")
```

Output:

```
PS C:\Users\haasini\OneDrive\Documents\python> python -u "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 -u "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.