K . Laxmi Prasanna

2303A52310

BATCH-42

Task 1: Student Performance Evaluation System

PROMPT:

Write a Python code for class student performance evaluation create a method to display student details, create another method to compare marks with class average.

CODE:

```
class Student:
    """
    A class to represent a student.
    """

    def __init__(self, name, roll_number, marks):
        self.name = name
        self.roll_number = roll_number
        self.marks = marks

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

    def check_performance(self, class_average):
        if self.marks > class_average:
            return f"{self.name} is above class average."
        elif self.marks < class_average:
            return f"{self.name} is below class average."
        else:
            return f"{self.name} is exactly at class average."


# -------- MAIN PROGRAM --------
if __name__ == "__main__":

    # Taking user input
    name = input("Enter student name: ")
    roll_number = int(input("Enter roll number: "))
    marks = float(input("Enter marks: "))
```

```python
class_average = float(input("Enter class average: "))

# Creating Student object
student = Student(name, roll_number, marks)

# Display details and performance
print("\nStudent Details")
student.display_details()

print("\nPerformance Result")
print(student.check_performance(class_average))
```

OUTPUT:

```
Name: Bob
Roll Number: 102
Marks: 65
Performance: Below Average
PS C:\Users\kurim>
```

JUSTIFICATION:

The code stores student details and marks in a class.
It prints student information and checks if marks are above or below the class average.

Task 2: Data Processing in a Monitoring System

PROMPT:

Write a python code for monitoring system ask user for the number of sensor readings

Take each sensor reading as integer input from the user Identify even numbers Calculate their square Print the result in a readable format

CODE:

```python
# Take sensor readings from the user
n = int(input("Enter number of sensor readings: "))

sensor_readings = []

for i in range(n):
    reading = int(input(f"Enter reading {i + 1}: "))
    sensor_readings.append(reading)

# Process only even readings
print("\nEven Sensor Readings and Their Squares:")
for reading in sensor_readings:
    if reading % 2 == 0:
        square = reading ** 2
        print("Even Reading:", reading, "| Square:", square)

```

OUTPUT:

```
Enter number of sensor readings: 5
Enter reading 1: 10
Enter reading 2: 12
Enter reading 3: 24
Enter reading 4: 33
Enter reading 5: 45

Even Sensor Readings and Their Squares:
Even Reading: 10 | Square: 100
Even Reading: 12 | Square: 144
Even Reading: 24 | Square: 576
```

JUSTIFICATION:

The loop iterates through sensor readings and selects even numbers using % 2 == 0.
Squares are calculated and printed in a readable format for monitoring.

Task 3: Banking Transaction Simulation

PROMPT:

Write a python code to Create a BankAccount class with deposit and withdraw methods that update balance and prevent withdrawals if insufficient

CODE:

C: > Users > sdnaz > OneDrive > Desktop > ai assiant coding > Untitled-1.py > BankAccount

```python
# Ask the user for account holder name and initial balance

# Create a method deposit(amount)
# Take deposit amount from user input
# Add it to balance using self and print confirmation

# Create a method withdraw(amount)
# Take withdrawal amount from user input
# Use if-else to check sufficient balance
# Prevent withdrawal if balance is insufficient
# Print user-friendly messages
class BankAccount:
    def __init__(self, account_holder, balance):
        self.account_holder = account_holder
        self.balance = balance

    def deposit(self, amount):
        self.balance += amount
        print("Deposit successful.")
        print("Current Balance:", self.balance)

    def withdraw(self, amount):
        if amount <= self.balance:
            self.balance -= amount
            print("Withdrawal successful.")
            print("Current Balance:", self.balance)
        else:
            print("Withdrawal failed: Insufficient balance.")


# -------- MAIN PROGRAM --------
name = input("Enter account holder name: ")
```

```python
# -------- MAIN PROGRAM --------
name = input("Enter account holder name: ")
balance = float(input("Enter initial balance: "))

account = BankAccount(name, balance)

deposit_amount = float(input("Enter deposit amount: "))
account.deposit(deposit_amount)

withdraw_amount = float(input("Enter withdrawal amount: "))
account.withdraw(withdraw_amount)
```

OUTPUT:

```
Enter account holder name: PRASANNA
Enter initial balance: 5000
Enter deposit amount: 2000
Deposit successful.
Current Balance: 7000.0
Enter withdrawal amount: 2000
Withdrawal successful.
Current Balance: 5000.0
```

JUSTIFICATION:

The code manages a bank account with deposit and withdraw functions.
It ensures withdrawals do not exceed the available balance.

Task 4: Student Scholarship Eligibility Check

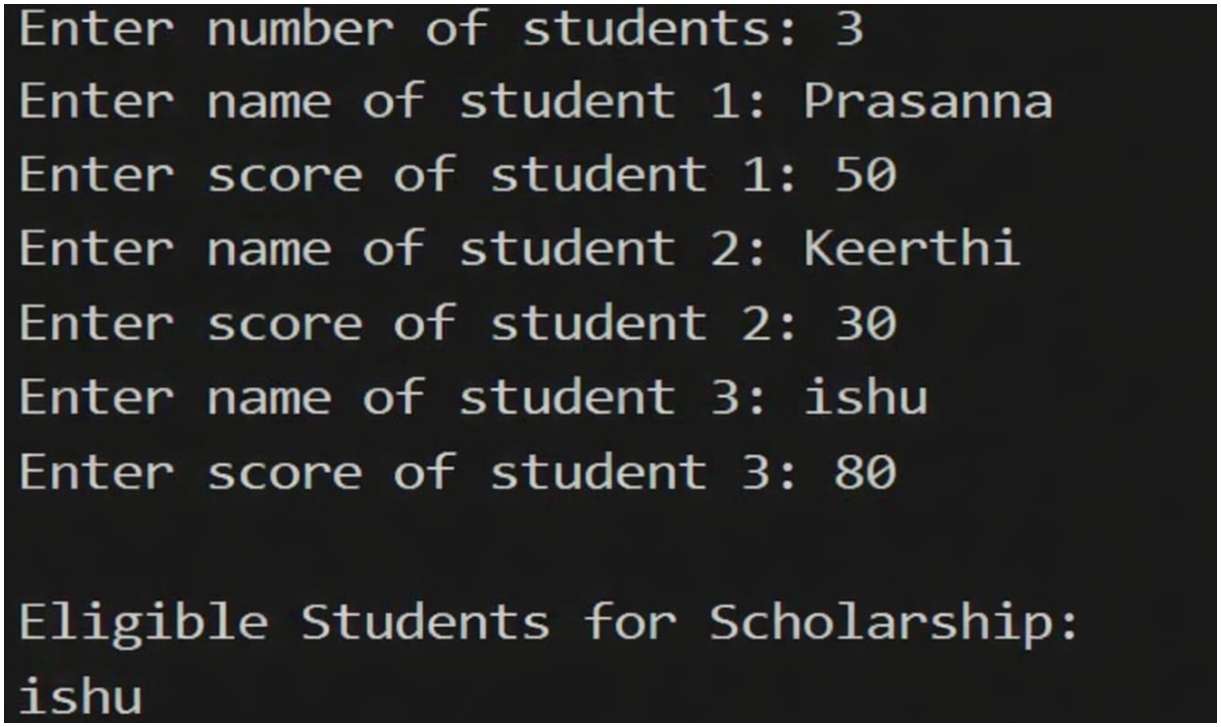
PROMPT:

Write a python code to Student Scholarship Eligibility Check Use a while loop to iterate through a list of students and print names of those with scores greater than 75

CODE:

```python
# write a python code for student scholarship eligibility # Use a while loop to iterate through the students list
n = int(input("Enter number of students: "))

students = []

# Take student details as input
for i in range(n):
    name = input(f"Enter name of student {i + 1}: ")
    score = int(input(f"Enter score of student {i + 1}: "))
    students.append({"name": name, "score": score})

print("\nEligible Students for Scholarship:")

# Use while loop to check eligibility
i = 0
while i < len(students):
    if students[i]["score"] > 75:
        print(students[i]["name"])
    i += 1
```

OUTPUT:



JUSTIFICATION:

The program checks each student's score in a list.
It prints the names of students who scored more than 75.

TASK 5: Online Shopping Cart Module

PROMPT

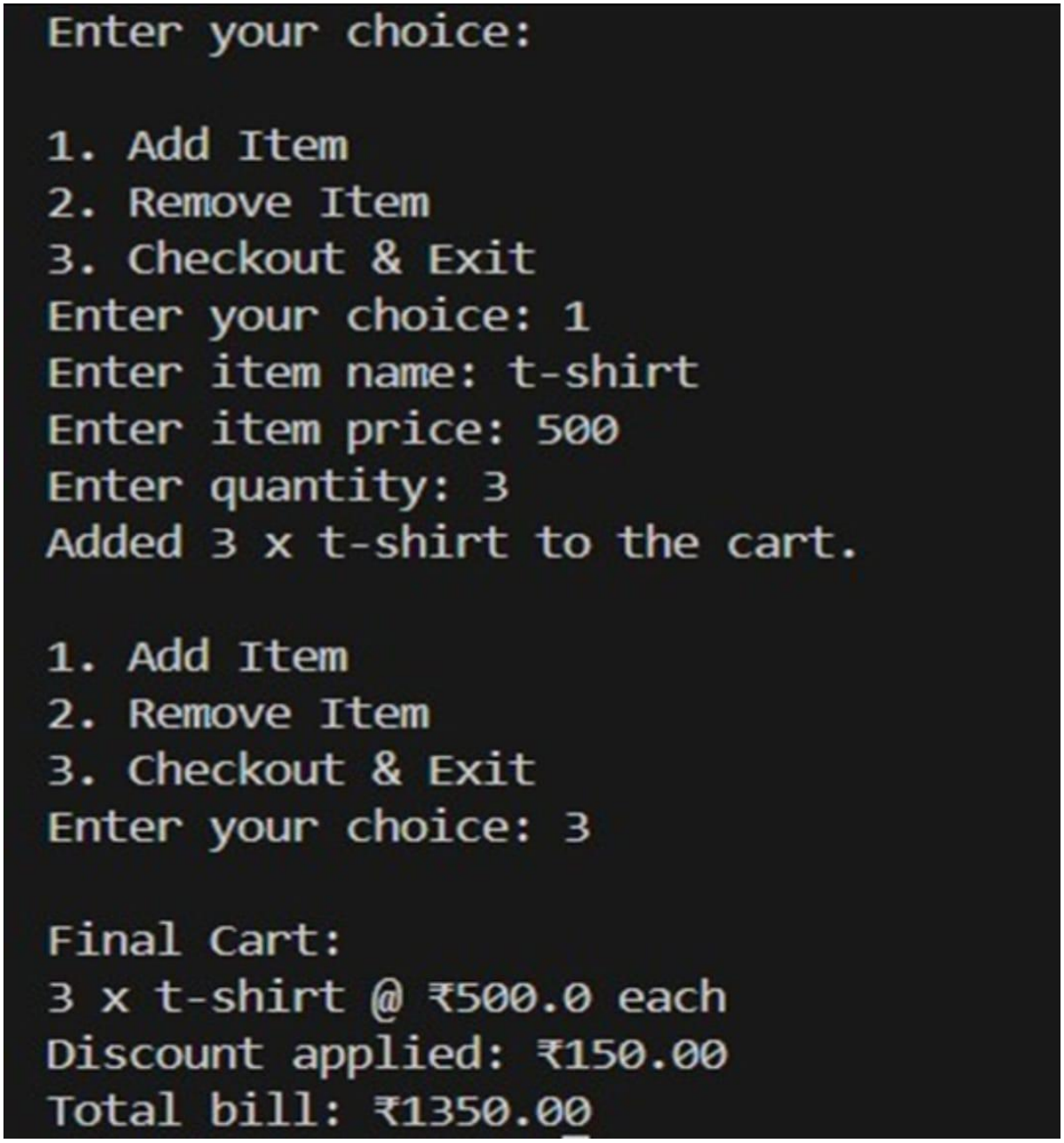
Write a python code to Create a Shopping Cart class with methods to add/remove items, calculate total, and apply discount if total > 50

CODE

```python
class ShoppingCart:
    def __init__(self):
        self.cart = []
    def add_item(self, name, price, quantity):
        self.cart.append({"name": name, "price": price, "quantity": quantity})
        print(f"Added {quantity} x {name} to the cart.")
    def remove_item(self, name):
        found = False
        for item in self.cart:
            if item["name"] == name:
                self.cart.remove(item)
                print(f"Removed {name} from the cart.")
                found = True
                break
        if not found:
            print(f"Item {name} not found in the cart.")
    def calculate_total(self):
        total = 0
        for item in self.cart:
            total += item["price"] * item["quantity"]
        if total > 500:
            discount = total * 0.10
            total -= discount
            print(f"Discount applied: ₹{discount:.2f}")
        print(f"Total bill: ₹{total:.2f}")
```

```python
# -------- MAIN PROGRAM --------
cart = ShoppingCart()

while True:
    print("\n1. Add Item")
    print("2. Remove Item")
    print("3. Checkout & Exit")
    choice = input("Enter your choice: ")
    if choice == "1":
        name = input("Enter item name: ")
        price = float(input("Enter item price: "))
        quantity = int(input("Enter quantity: "))
        cart.add_item(name, price, quantity)
    elif choice == "2":
        name = input("Enter item name to remove: ")
        cart.remove_item(name)
    elif choice == "3":
        print("\nFinal Cart:")
        for item in cart.cart:
            print(f"{item['quantity']} x {item['name']} @ ₹{item['price']} each")
        cart.calculate_total()
        break
    else:
        print("Invalid choice! Please try again.")
```

OUTPUT:



JUSTIFICATION:

The class manages items in a shopping cart and calculates the total bill.
It applies a discount if the total exceeds a certain amount and allows adding/removing items.