

ASSIGNMENT -6.4

2303a51343

B-10

TASK-1

Prompt: Generate student performance evaluation system with attributes like name

Rollno, marks and give message according to marks obtained by student with user input

Code: class Student:

```
def __init__(self, name, roll_number, marks):
```

```
    self.name = name
```

```
    self.roll_number = roll_number
```

```
    self.marks = marks
```

```
def evaluate_performance(self):
```

```
    if self.marks >= 90:
```

```
        return "Excellent"
```

```
    elif self.marks >= 75:
```

```
        return "Good"
```

```
    elif self.marks >= 60:
```

```
        return "Average"
```

```
    else:
```

```
        return "Needs Improvement"
```

```
# Taking user input
```

```
name = input("Enter student's name: ")
```

```
roll_number = input("Enter student's roll number: ")
```

```
marks = float(input("Enter student's marks: "))
```

```
# Creating a Student object
```

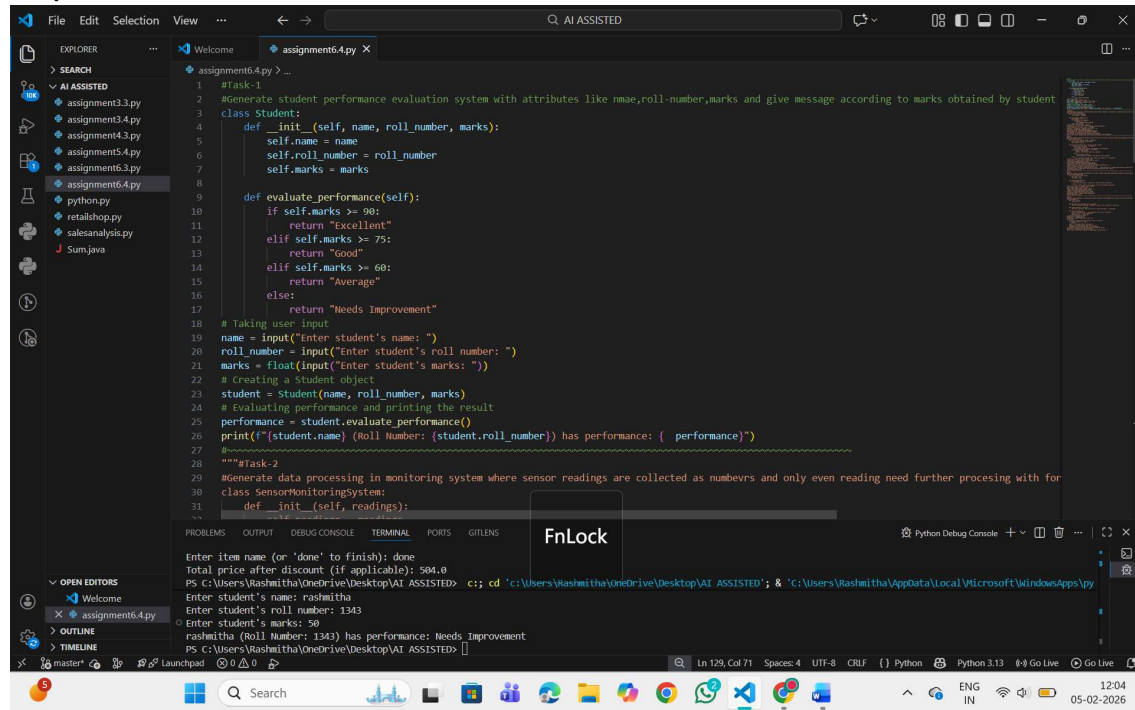
```
student = Student(name, roll_number, marks)
```

```
# Evaluating performance and printing the result
```

```
performance = student.evaluate_performance()
```

```
print(f"{student.name} (Roll Number: {student.roll_number}) has performance: { performance}")
```

Output:



```
1 #Task-1
2 #Generate student performance evaluation system with attributes like nmae,roll-number,marks and give message according to marks obtained by student
3 class Student:
4     def __init__(self, name, roll_number, marks):
5         self.name = name
6         self.roll_number = roll_number
7         self.marks = marks
8
9     def evaluate_performance(self):
10        if self.marks >= 90:
11            return "Excellent"
12        elif self.marks >= 75:
13            return "Good"
14        elif self.marks >= 60:
15            return "Average"
16        else:
17            return "Needs Improvement"
18
19 # Taking user input
20 name = input("Enter student's name: ")
21 roll_number = input("Enter student's roll number: ")
22 marks = float(input("Enter student's marks: "))
23 # Creating a Student object
24 student = Student(name, roll_number, marks)
25 # Evaluating performance and printing the result
26 performance = student.evaluate_performance()
27 print(f"{student.name} (Roll number: {student.roll_number}) has performance: { performance}")
28
29 """Task-2
30 #Generate data processing in monitoring system where sensor readings are collected as numbevrns and only even reading need further procesing with for
31 class SensorMonitoringSystem:
32     def __init__(self, readings):
```

Enter item name (or 'done' to finish): done
Total price after discount (if applicable): 504.0
PS C:\Users\Rashmitha\OneDrive\Desktop\AI ASSISTED> c:\cd 'C:\Users\Rashmitha\OneDrive\Desktop\AI ASSISTED' & 'C:\Users\Rashmitha\AppData\Local\Microsoft\WindowsApps\py

Enter student's name: rashmitha
Enter student's roll number: 1343
Enter student's marks: 50
rashmitha (Roll number: 1343) has performance: Needs Improvement
PS C:\Users\Rashmitha\OneDrive\Desktop\AI ASSISTED> []

Analysis: This program uses a Student class to store student details and marks. It evaluates performance using conditional statements based on marks. The result displays a performance message like Excellent, Good, or Average

TASK-2

Prompt: Generate data processing in monitoring system where sensor readings are collected as numbevrns and only even reading need further procesing with for loop to iterate over a list of intergers redaings.

Code: class SensorMonitoringSystem:

```
def __init__(self, readings):
    self.readings = readings

def process_even_readings(self):
    even_readings = []
    for reading in self.readings:
        if reading % 2 == 0:
            even_readings.append(reading)
    return even_readings
```

Taking user input for sensor readings

readings_input = input("Enter sensor readings (comma separated): ")

Converting input string to a list of integers

```

readings = list(map(int, readings_input.split(',')))

# Creating a SensorMonitoringSystem object

monitoring_system = SensorMonitoringSystem(readings)

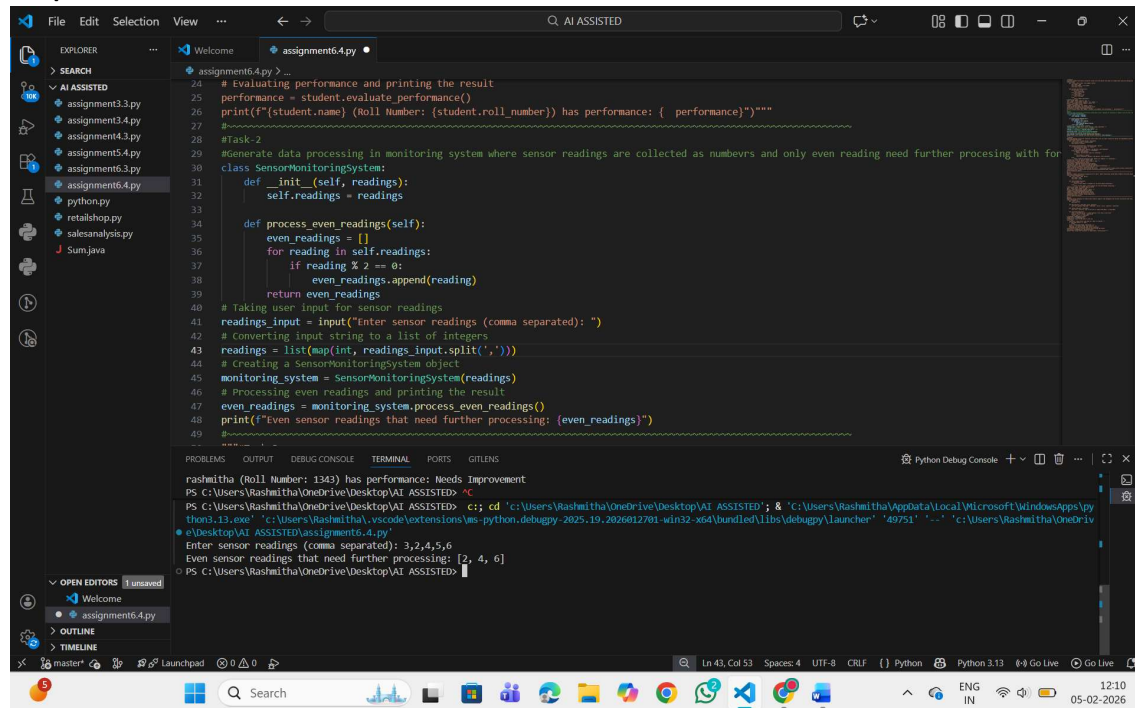
# Processing even readings and printing the result

even_readings = monitoring_system.process_even_readings()

print(f"Even sensor readings that need further processing: {even_readings}")

```

Output:



The screenshot shows a VS Code editor with a Python file named 'assignment6.4.py'. The code defines a 'SensorMonitoringSystem' class with methods for evaluating performance, processing even readings, and taking user input. The terminal output shows the execution of the script, including the user input '3,2,4,5,6' and the resulting even readings '[2, 4, 6]'. The terminal also shows the command used to run the script: 'python3.13.exe'.

```

24 # Evaluating performance and printing the result
25 performance = student.evaluate_performance()
26 print(f"{student.name} (Roll Number: {student.roll_number}) has performance: { performance}")
27
28 #Task-2
29 #Generate data processing in monitoring system where sensor readings are collected as numbevrns and only even reading need further procesing with for
30
31 class SensorMonitoringSystem:
32     def __init__(self, readings):
33         self.readings = readings
34
35     def process_even_readings(self):
36         even_readings = []
37         for reading in self.readings:
38             if reading % 2 == 0:
39                 even_readings.append(reading)
40         return even_readings
41
42 # Taking user input for sensor readings
43 readings_input = input("Enter sensor readings (comma separated): ")
44 # converting input string to a list of integers
45 readings = list(map(int, readings_input.split(',')))
46 # Creating a SensorMonitoringSystem object
47 monitoring_system = SensorMonitoringSystem(readings)
48 # Processing even readings and printing the result
49 even_readings = monitoring_system.process_even_readings()
50 print(f"Even sensor readings that need further processing: {even_readings}")
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

```

```

rashmitha (Roll Number: 1343) has performance: Needs Improvement
PS C:\Users\Rashmitha\OneDrive\Desktop\VAI ASSISTED> cd 'c:\Users\Rashmitha\OneDrive\Desktop\VAI ASSISTED' & 'C:\Users\Rashmitha\AppData\Local\Microsoft\WindowsApps\python3.13.exe' 'c:\Users\Rashmitha\vscode\extensions\ms-python.debugpy-2025.19.2026012701-win32-x64\bundle\libs\debugpy\launcher' '49751' '-' 'c:\Users\Rashmitha\OneDrive\Desktop\VAI ASSISTED\assignment6.4.py'
Enter sensor readings (comma separated): 3,2,4,5,6
Even sensor readings that need further processing: [2, 4, 6]
PS C:\Users\Rashmitha\OneDrive\Desktop\VAI ASSISTED>

```

Analysis: This program collects sensor readings from the user as a list. A for loop checks each reading and selects only even numbers. Even readings are stored and displayed for further processing

TASK-3

Prompt: Generate banking transaction simulation system where user can input transaction amount and type(deposit/withdrawal), with attributes account_holder,balance

Code: def __init__(self, account_holder, balance=0):

```

self.account_holder = account_holder

```

```

self.balance = balance

```

```

def process_transaction(self, transaction_type, amount):

```

```

    if transaction_type.lower() == 'deposit':

```

```

        self.balance += amount

        return f"Deposited: {amount}. New Balance: {self.balance}"

    elif transaction_type.lower() == 'withdrawal':

        if amount > self.balance:

            return "Insufficient funds for withdrawal."

        else:

            self.balance -= amount

            return f"Withdrew: {amount}. New Balance: {self.balance}"

    else:

        return "Invalid transaction type. Please use 'deposit' or 'withdrawal'."

# Taking user input for account holder name
account_holder = input("Enter account holder's name: ")

# Creating a BankingTransaction object
bank_account = BankingTransaction(account_holder)

# Taking user input for transaction type and amount
transaction_type = input("Enter transaction type (deposit/withdrawal): ")

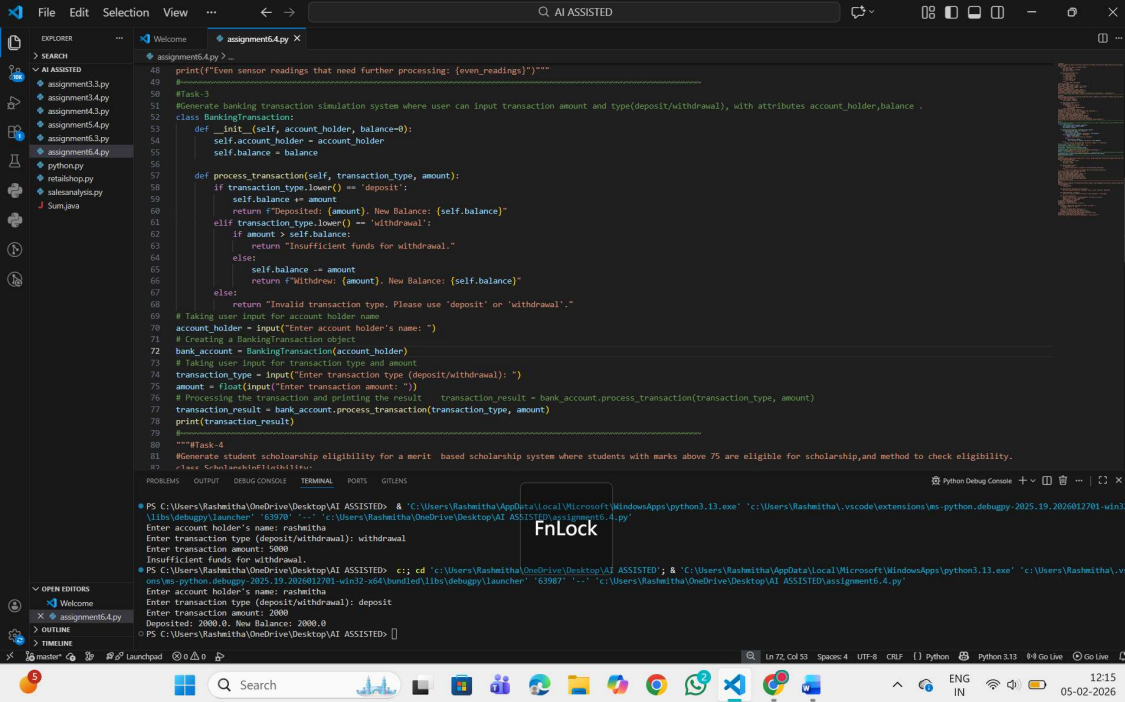
amount = float(input("Enter transaction amount: "))

# Processing the transaction and printing the result
transaction_result = bank_account.process_transaction(transaction_type, amount)

print(transaction_result)

```

Output:



```
48 print(f"Even sensor readings that need further processing: {even_readings}")"""
49
50 #Task-3
51 #Generate banking transaction simulation system where user can input transaction amount and type(deposit/withdrawal), with attributes account_holder,balance .
52 class BankingTransaction:
53     def __init__(self, account_holder, balance=0):
54         self.account_holder = account_holder
55         self.balance = balance
56
57     def process_transaction(self, transaction_type, amount):
58         if transaction_type.lower() == 'deposit':
59             self.balance += amount
60             return f'Deposited: (amount). New Balance: {self.balance}"
61         elif transaction_type.lower() == 'withdrawal':
62             if amount > self.balance:
63                 return "Insufficient funds for withdrawal."
64             else:
65                 self.balance -= amount
66                 return f'Withdrew: (amount). New Balance: {self.balance}"
67         else:
68             return "Invalid transaction type. Please use 'deposit' or 'withdrawal'."
69
70 # Taking user input for account holder name
71 account_holder = input("Enter account holder's name: ")
72 # Creating a BankingTransaction object
73 bank_account = BankingTransaction(account_holder)
74 # Taking user input for transaction type and amount
75 transaction_type = input("Enter transaction type (deposit/withdrawal): ")
76 amount = float(input("Enter transaction amount: "))
77 # Processing the transaction and printing the result
78 transaction_result = bank_account.process_transaction(transaction_type, amount)
79 print(transaction_result)
80
81 """Task-4
82 #Generate student scholarship eligibility for a merit based scholarship system where students with marks above 75 are eligible for scholarship,and method to check eligibility.
83 class ScholarshipEligibility:
84     def __init__(self, name, marks):
85         self.name = name
86         self.marks = marks
87
88     def check_eligibility(self):
89         if self.marks > 75:
90             return f'{self.name} is eligible for the merit-based scholarship."
91         else:
92             return f'{self.name} is not eligible for the merit-based scholarship."
93
94 # Taking user input for student name and marks
95 name = input("Enter student's name: ")
96 marks = int(input("Enter student's marks: "))
97 eligibility = ScholarshipEligibility(name, marks)
98 result = eligibility.check_eligibility()
99 print(result)
```

Analysis: this system simulates deposit and withdrawal operations using a class.

It updates account balance based on transaction type and amount.

It also prevents withdrawal when funds are insufficient.

TASK-4

Prompt: Generate student scholarship eligibility for a merit based scholarship system where students with marks above 75 are eligible for scholarship,and method to check eligibility.

Code: class ScholarshipEligibility:

```
def __init__(self, name, marks):
```

```
    self.name = name
```

```
    self.marks = marks
```

```
def check_eligibility(self):
```

```
    if self.marks > 75:
```

```
        return f'{self.name} is eligible for the merit-based scholarship."
```

```
    else:
```

```
        return f'{self.name} is not eligible for the merit-based scholarship."
```

```
# Taking user input for student name and marks
```

```
name = input("Enter student's name: ")
```

```
marks = float(input("Enter student's marks: "))

# Creating a ScholarshipEligibility object

student = ScholarshipEligibility(name, marks)

# Checking eligibility and printing the result

eligibility_result = student.check_eligibility()

print(eligibility_result)
```

Output:

The screenshot shows a VS Code editor with a file named `assignment6.4.py`. The code defines a `ScholarshipEligibility` class with an `__init__` method and a `check_eligibility` method. The `check_eligibility` method checks if the student's marks are greater than 75. If yes, it returns a message stating the student is eligible for a merit-based scholarship. If no, it returns a message stating the student is not eligible. The script then takes user input for the student's name and marks, creates a `ScholarshipEligibility` object, and prints the result of the `check_eligibility` method.

The terminal output shows the execution of the script. It prompts the user to enter the student's name and marks. The user enters 'rashmitha' and '75'. The output shows that 'rashmitha' is not eligible for the merit-based scholarship.

Analysis: This program checks student eligibility for a merit-based scholarship. Students with marks above 75 are considered eligible. The result is displayed using a class method

TASK-5

Prompt: Create a Python class `ShoppingCart` that stores items. Add methods to add items, remove items, calculate total using a loop, and apply discount if total exceeds a limit user input.

Code: class `ShoppingCart`:

```
def __init__(self):

    self.items = []

def add_item(self, item_name, price):

    """Add an item with its price to the shopping cart."""

    self.items.append({"name": item_name, "price": price})
```

```

        print(f"Added {item_name} with price ${price} to the cart.")

def remove_item(self, item_name):
    """Remove an item from the shopping cart by name."""
    for item in self.items:
        if item["name"] == item_name:
            self.items.remove(item)
            print(f"Removed {item_name} from the cart.")

    return

    print(f"Item {item_name} not found in the cart.")

def calculate_total(self):
    """Calculate the total price of items in the cart."""
    total = 0
    for item in self.items:
        total += item["price"]
    return total

def apply_discount(self, discount_threshold, discount_rate):
    """Apply a discount if the total exceeds a certain threshold."""
    total = self.calculate_total()
    if total > discount_threshold:
        discount = total * discount_rate
        total -= discount
        print(f"Discount of ${discount:.2f} applied.")
    return total

# Demonstration of the ShoppingCart class
cart = ShoppingCart()

while True:
    action = input("Choose an action: add, remove, total, checkout, or exit: ").lower()

```

```

if action == "add":
    item_name = input("Enter item name: ")
    price = float(input("Enter item price: "))
    cart.add_item(item_name, price)

elif action == "remove":
    item_name = input("Enter item name to remove: ")
    cart.remove_item(item_name)

elif action == "total":
    total = cart.calculate_total()
    print(f"Current total: ${total:.2f}")

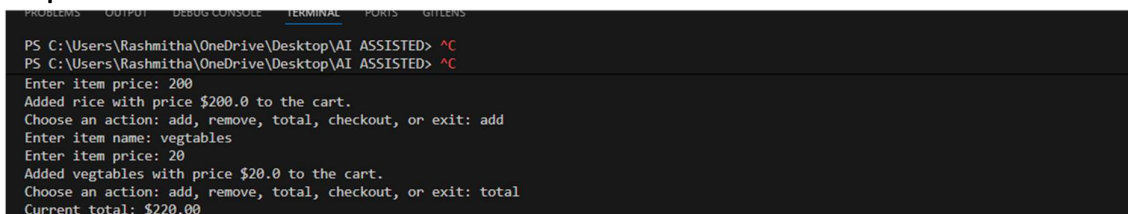
elif action == "checkout":
    discount_threshold = float(input("Enter discount threshold: "))
    discount_rate = float(input("Enter discount rate (as a decimal): "))
    final_total = cart.apply_discount(discount_threshold, discount_rate)
    print(f"Final total after discount (if applicable): ${final_total:.2f}")

elif action == "exit":
    print("Exiting the program.")
    break

else:
    print("Invalid action. Please choose again.")

```

Output:



```

PS C:\Users\Rashmitha\OneDrive\Desktop\AI ASSISTED> ^C
PS C:\Users\Rashmitha\OneDrive\Desktop\AI ASSISTED> ^C
Enter item price: 200
Added rice with price $200.0 to the cart.
Choose an action: add, remove, total, checkout, or exit: add
Enter item name: vegetables
Enter item price: 20
Added vegetables with price $20.0 to the cart.
Choose an action: add, remove, total, checkout, or exit: total
Current total: $220.00

```

Analysis: This program manages a shopping cart with add and remove options. It calculates the total price using a loop. A discount is applied if the total exceeds a user-defined limit