

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:

The screenshot shows the Microsoft Visual Studio Code interface. The left sidebar displays a file tree with several Python files: assignment3.3.py, assignment3.4.py, assignment4.3.py, assignment5.4.py, assignment6.3.py, assignment6.4.py, python.py, retailshop.py, salesanalysis.py, and Sum.java. The main editor area contains the following Python code:

```
#Task-1
#Generate student performance evaluation system with attributes like name,roll_number,marks and give message according to marks obtained by student
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}")

"""
#Task-2
#Generate data processing in monitoring system where sensor readings are collected as numbers and only even reading need further processing with for loop to iterate over a list of integers readings
def __init__(self, readings):
    self.readings = readings

    Enter item name (or 'done' to finish): done
    Total price after discount (if applicable): 500.0
PS C:\Users\rashmita\OneDrive\Desktop\AI ASSISTED> c:; cd 'c:\Users\rashmita\OneDrive\Desktop\AI ASSISTED'; & 'C:\Users\rashmita\AppData\Local\Microsoft\WindowsApps\py
Enter student's name: rashmita
Enter student's roll number: 1343
Enter student's marks: 50
rashmita (Roll Number: 1343) has performance: Needs Improvement
PS C:\Users\rashmita\OneDrive\Desktop\AI ASSISTED> [
```

The terminal window at the bottom shows the execution of the code. It prompts for student details and then prints the performance message. The status bar at the bottom right indicates the date and time as 05-02-2026.

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 numbers and only even reading need further processing with for loop to iterate over a list of integers readings.

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 Visual Studio Code (VS Code) interface with the following details:

- File Explorer:** Shows files like `assignment3.3.py`, `assignment3.4.py`, `assignment4.3.py`, `assignment5.4.py`, `assignment6.3.py`, and `assignment6.4.py`.
- Search Bar:** Contains the text "AI ASSISTED".
- Code Editor:** Displays the content of `assignment6.4.py`. The code is a Python script for a SensorMonitoringSystem that processes sensor readings. It includes a function to evaluate student performance based on even-numbered sensor readings.
- Terminal:** Shows the command-line output of running the script:

```
rashmita (Roll Number: 1343) has performance: Needs Improvement
PS C:\Users\Rashmita\OneDrive\Desktop\AI ASSISTED> cd 'C:\Users\Rashmita\OneDrive\Desktop\AI ASSISTED' & & cd 'C:\Users\Rashmita\AppData\Local\Microsoft\WindowsApps\pythond3.10.exe' & c:\Users\Rashmita\vscode\extensions\ms-python.python.debug-2025.10.2026012701-win32-x64\bundled\lits\debug\launcher`49751' --> 'c:\Users\Rashmita\OneDrive\Desktop\AI 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\Rashmita\OneDrive\Desktop\AI ASSISTED>
```
- Bottom Status Bar:** Shows file paths like "master", "Launchpad", and "Timeline".
- Bottom Icons:** Includes icons for search, file operations, and various extensions like Python and Go Live.

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)

transaction_result = bank_account.process_transaction(transaction_type, amount)

print(transaction_result)
```

Output:

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:
```

re

lse:

return f"{{self.name}} is not eligible for the

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

```
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 the Visual Studio Code interface with the following details:

- File Explorer:** Shows files like `assignment64.py`, `assignment33.py`, `assignment43.py`, `assignments4.py`, `assignments3.py`, and `assignments43.py`.
- Search Bar:** Contains the text `assignment64.py`.
- Code Editor:** Displays the `assignment64.py` file content, which includes importing `bank_account` and `transaction_type`, processing a transaction, and generating student scholarship eligibility based on marks.
- Terminal:** Shows the execution of the script in a Python debugger. The terminal output is as follows:
 - Python version: 2025.19.2026012701
 - Input: `name = input('Enter student's name: ')`
 - Input: `marks = float(input('Enter student's marks: '))`
 - Output: `student = ScholarshipEligibility(name, marks)`
 - Output: `# Checking eligibility and printing the result`
 - Output: `eligibility_result = student.check_eligibility()`
 - Output: `print(eligibility_result)`
- Status Bar:** Shows the current file is `assignment64.py`, and the status bar indicates the file has 111 lines and 111 characters.

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}")

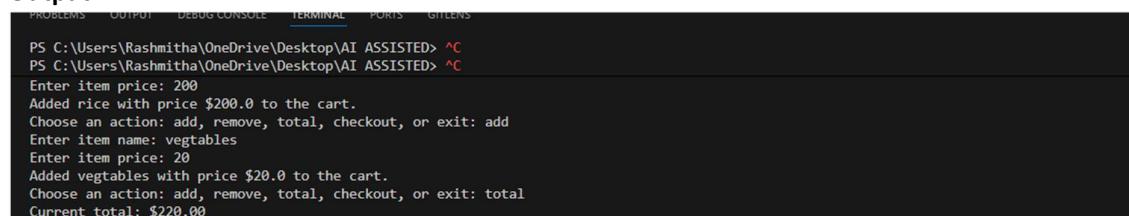
elif action == "checkout":
    discount_threshold = float(input("Enter discount threshold: "))
    discount_rate = float(input("Enter discount rate (as a decimal): "))
    final_total = cart.apply_discount(discount_threshold, discount_rate)
    print(f"Final total after discount (if applicable): ${final_total:.2f}")

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

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

```

Output:



```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITHUB
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