

AI Assisted Coding LAB:-6.4

NAME:- NIPUN.I

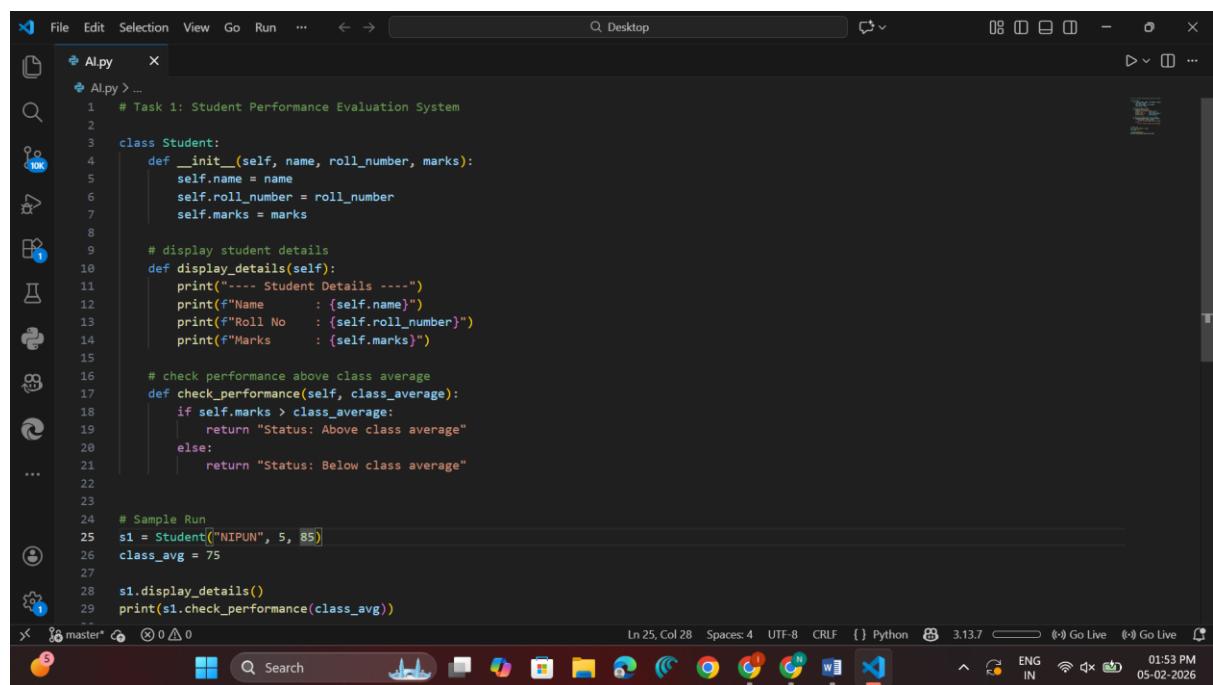
H.T.NO:- 2303A52208

BATCH – 43

=====

⇒ TASK 1:-

=====



The screenshot shows a code editor window titled "AI.py". The code is a Python script named "AI.py" which defines a class "Student". The class has an __init__ method to initialize name, roll_number, and marks. It has a display_details method to print student details. It also has a check_performance method to compare marks against a class average. A sample run is shown where a student with name "NIPUN", roll number 5, and marks 85 is created, and its details are printed along with a status message.

```
# Task 1: Student Performance Evaluation System

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

    # display student details
    def display_details(self):
        print("---- Student Details ----")
        print(f"Name : {self.name}")
        print(f"Roll No : {self.roll_number}")
        print(f"Marks : {self.marks}")

    # check performance above class average
    def check_performance(self, class_average):
        if self.marks > class_average:
            return "Status: Above class average"
        else:
            return "Status: Below class average"

# Sample Run
s1 = Student("NIPUN", 5, 85)
class_avg = 75

s1.display_details()
print(s1.check_performance(class_avg))
```

OUTPUT:-

```
PS C:\Users\NIPUN\OneDrive\Desktop> & C:/Python313/python.exe c:/Users/NIPUN/OneDrive/Desktop/AI.py
---- Student Details ----
Name      : NIPUN
Roll No   : 5
Marks     : 85
Status: Above class average
PS C:\Users\NIPUN\OneDrive\Desktop>
```

Summary :-

This task focused on building a basic academic evaluation system using a Python class. A Student class was created with attributes name, roll_number, and marks. AI-based code completion was used to generate methods for:

- Displaying student details
- Evaluating student performance using conditional logic

The AI successfully generated:

- Proper use of self for class attributes
- if-else conditions for performance checking
- Clean and readable method structures

This task demonstrated how AI can assist in building structured class definitions and logic-based methods efficiently.

=====

⇒ TASK 2:-

=====

```
# Task 2: Data Processing in a Monitoring System

sensor_readings = [10, 15, 20, 23, 30, 45, 60]

for reading in sensor_readings:
    # identify even numbers, calculate square, and print result
    if reading % 2 == 0:
        square = reading * reading
        print(f"Even Reading: {reading} | Square: {square}")
```

OUTPUT:-

```
PS C:\Users\NIPUN\OneDrive\Desktop> & C:/Python313/python.exe c:/Users/NIPUN/OneDrive/Desktop/AI.py
Even Reading: 10 | Square: 100
Even Reading: 20 | Square: 400
Even Reading: 30 | Square: 900
Even Reading: 60 | Square: 3600
PS C:\Users\NIPUN\OneDrive\Desktop>
```

Summary:-

This task involved processing numerical sensor data using loops and conditions. A for loop was initiated manually, and AI-based completion was guided using comments.

The AI generated logic to:

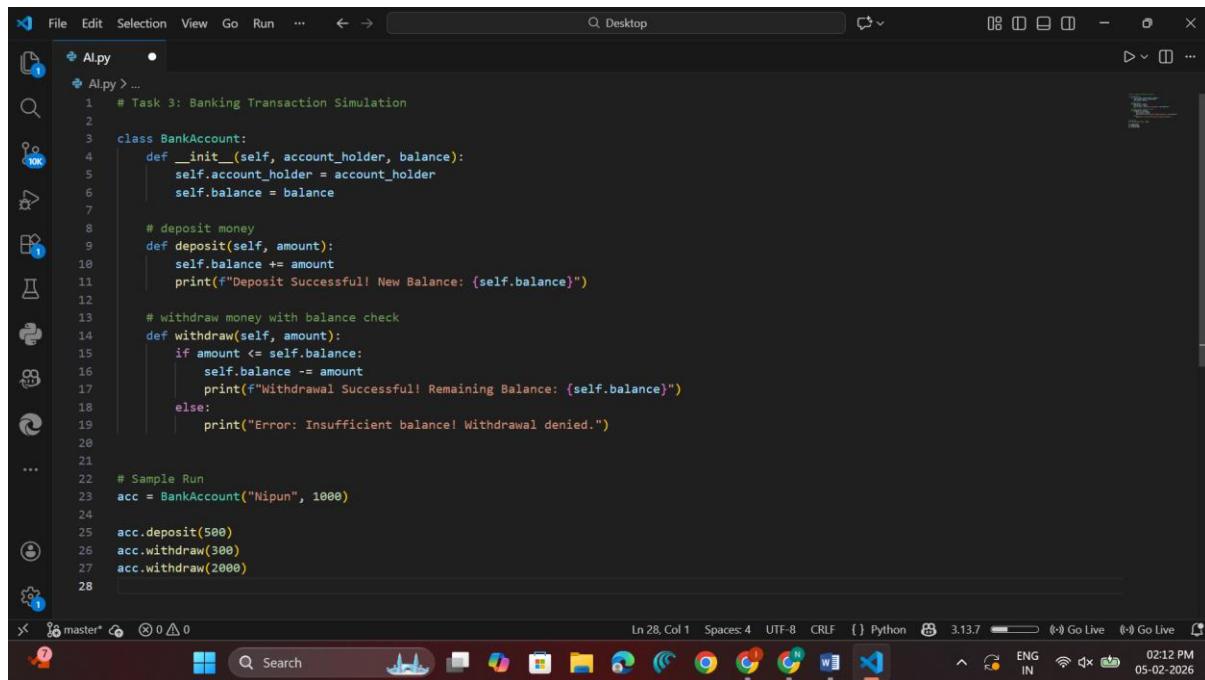
- Identify even numbers using the modulus (%) operator
- Calculate the square of even values
- Print results in a readable format

This task demonstrated AI's ability to understand loop structures, conditional logic, and mathematical operations through prompt-driven guidance.

=====

⇒ TASK 3:-

=====



```
# Task 3: Banking Transaction Simulation
class BankAccount:
    def __init__(self, account_holder, balance):
        self.account_holder = account_holder
        self.balance = balance

    # deposit money
    def deposit(self, amount):
        self.balance += amount
        print(f"Deposit Successful! New Balance: {self.balance}")

    # withdraw money with balance check
    def withdraw(self, amount):
        if amount <= self.balance:
            self.balance -= amount
            print(f"Withdrawal Successful! Remaining Balance: {self.balance}")
        else:
            print("Error: Insufficient balance! Withdrawal denied.")

# Sample Run
acc = BankAccount("Nipun", 1000)

acc.deposit(500)
acc.withdraw(300)
acc.withdraw(2000)
```

OUTPUT:-

```
PS C:\Users\NIPUN\OneDrive\Desktop> & C:/Python313/python.exe c:/Users/NIPUN/OneDrive/Desktop/AI.py
Deposit Successful! New Balance: 1500
Withdrawal Successful! Remaining Balance: 1200
Error: Insufficient balance! Withdrawal denied.
PS C:\Users\NIPUN\OneDrive\Desktop> 
```

Summary:-

In this task, a BankAccount class was created to simulate real-world banking operations. The class included attributes for account_holder and balance.

AI-assisted code completion generated methods for:

- Depositing money
- Withdrawing money
- Preventing withdrawals when balance is insufficient

The AI used:

- Conditional statements (if-else)
- Class attributes via self
- User-friendly output messages

This task showed how AI can help implement real-life business logic safely and logically.

=====

⇒ TASK 4:-

=====

```
# Task 4: Student Scholarship Eligibility Check

students = [
    {"name": "Nipun", "score": 80},
    {"name": "Neha", "score": 70},
    {"name": "Rahul", "score": 90},
    {"name": "Priya", "score": 76}
]

i = 0
# iterate through list and print students with score > 75
print("Scholarship Eligible Students:")
while i < len(students):
    if students[i]["score"] > 75:
        print(students[i]["name"])
    i += 1
```

Output:-

```
PS C:\Users\NIPUN\OneDrive\Desktop> & C:/Python313/python.exe c:/Users/NIPUN/OneDrive/Desktop/AI.py
Scholarship Eligible Students:
Nipun
Rahul
Priya
PS C:\Users\NIPUN\OneDrive\Desktop>
```

Summary:-

This task focused on using loops and conditionals with structured data. A list of dictionaries was created to store student names and scores.

AI-based completion was used to generate a while loop that:

- Iterates through the list
- Checks eligibility using conditions
- Prints names of students scoring more than 75

The task demonstrated AI's ability to manage:

- Index-based iteration
- Data structure access
- Logical filtering using conditions

=====

⇒ TASK 5:-

=====

```
# Task 5: Online Shopping Cart Module
class ShoppingCart:
    def __init__(self):
        self.items = []
    # add item to cart
    def add_item(self, name, price, quantity):
        self.items.append({
            "name": name,
            "price": price,
            "quantity": quantity
        })
        print(f"{name} added to cart.")
    # remove item from cart
    def remove_item(self, name):
        self.items = [item for item in self.items if item["name"] != name]
        print(f"{name} removed from cart.")
    # calculate total bill
    def calculate_total(self):
        total = 0
        for item in self.items:
            total += item["price"] * item["quantity"]
        # apply discount condition
        if total > 1000:
            print("Discount Applied: 10%")

```

```
class ShoppingCart:
    def calculate_total(self):
        return total
    # display cart items
    def show_cart(self):
        print("\n--- Cart Items ---")
        for item in self.items:
            print(f"{item['name']} | Price: {item['price']} | Qty: {item['quantity']}")

# Sample Run
cart = ShoppingCart()

cart.add_item("Laptop", 60000, 1)
cart.add_item("Mouse", 500, 2)
cart.add_item("Keyboard", 1000, 1)

cart.show_cart()

total_bill = cart.calculate_total()
print(f"\nTotal Bill: {total_bill}")

cart.remove_item("Mouse")
cart.show_cart()

total_bill = cart.calculate_total()
print(f"\nUpdated Total Bill: {total_bill}")

```

Output:-

```
PS C:\Users\NIPUN\OneDrive\Desktop> & C:/Python313/python.exe c:/Users/NIPUN/OneDrive/Desktop/AI.py
Laptop added to cart.
Mouse added to cart.
Keyboard added to cart.

--- Cart Items ---
Laptop | Price: 60000 | Qty: 1
Mouse | Price: 500 | Qty: 2
Keyboard | Price: 1000 | Qty: 1
Discount Applied: 10%

Total Bill: 55800.0
Mouse removed from cart.

--- Cart Items ---
Laptop | Price: 60000 | Qty: 1
Keyboard | Price: 1000 | Qty: 1
Discount Applied: 10%

Updated Total Bill: 54900.0
PS C:\Users\NIPUN\OneDrive\Desktop> []
```

Summary:-

This task involved building a simplified e-commerce shopping cart system. A ShoppingCart class was created with an empty list to store cart items.

AI-assisted completion generated methods for:

- Adding items to the cart
- Removing items from the cart
- Calculating total bill using loops
- Applying conditional discounts

The AI correctly implemented:

- Loop-based total calculation
- Conditional discount logic
- Data structure handling
- Object-oriented design principles

This task demonstrated AI's effectiveness in building scalable, real-world application logic.