ASSIGNMENT – 6.4

2303A51355

Batch-10

Task-1

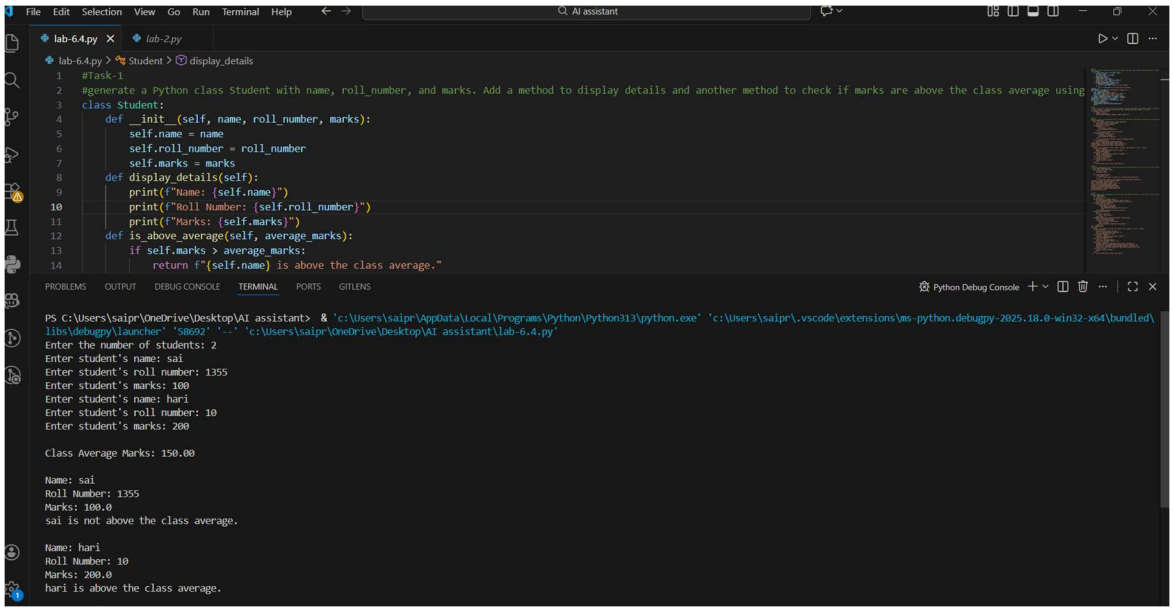
Prompt: generate a Python class Student with name, roll_number, and marks. Add a method to display details and another method to check if marks are above the class average using if-else. take user input

Code :

```
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_above_average(self, average_marks):
        if self.marks > average_marks:
            return f"{self.name} is above the class average."
        else:
            return f"{self.name} is not above the class average."
students = []
num_students = int(input("Enter the number of students: "))
for _ in range(num_students):
    name = input("Enter student's name: ")
```

```
    roll_number = input("Enter student's roll number: ")
    marks = float(input("Enter student's marks: "))
    students.append(Student(name, roll_number, marks))
total_marks = sum(student.marks for student in students)
average_marks = total_marks / num_students
print(f"\nClass Average Marks: {average_marks:.2f}\n")
for student in students:
    student.display_details()
    print(student.is_above_average(average_marks))
    print()
```

Output :



Code Analysis :

- Defines a Student class with name, roll number, and marks as attributes.
- Uses user input to create multiple student objects and store them in a list.
- Calculates the class average using a loop and total marks.
- Uses if-else to check whether a student's marks are above the average.
- Displays student details and result in a readable format.

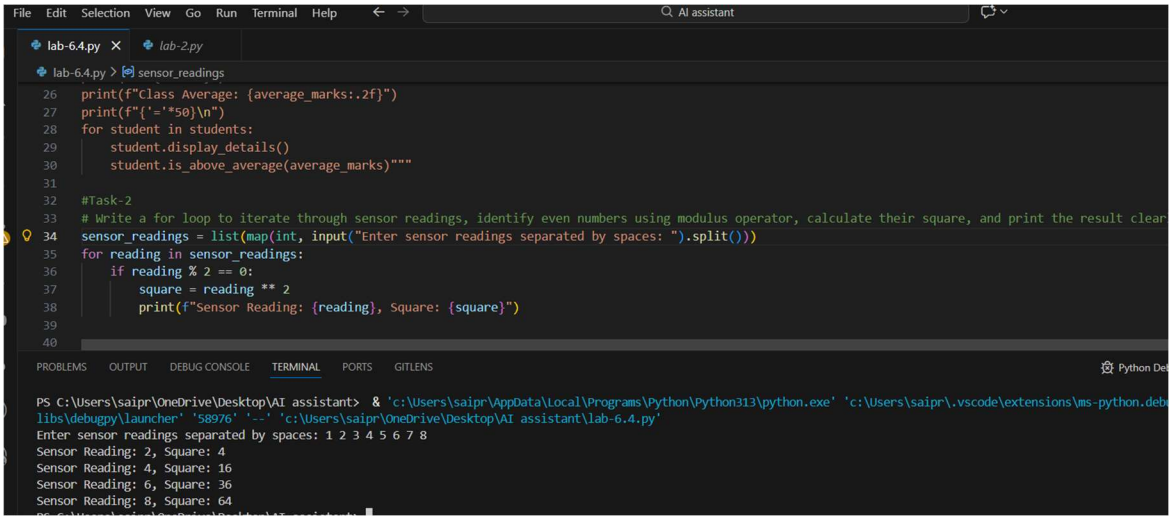
Task-2

Prompt: Write a for loop to iterate through sensor readings, identify even numbers using modulus operator, calculate their square, and print the result clearly and user input.

Code :

```
sensor_readings = list(map(int, input("Enter sensor readings separated by spaces: ").split()))
for reading in sensor_readings:
    if reading % 2 == 0:
        square = reading ** 2
        print(f"Sensor Reading: {reading}, Square: {square}")
```

Output :



Code Analysis :

- Takes sensor readings from the user as a list of integers.
- Uses a for loop to iterate through each reading.
- Identifies even numbers using the modulus (%) operator.
- Calculates the square of even readings using the power operator (**).
- Prints the sensor reading and its square clearly.

Task-3

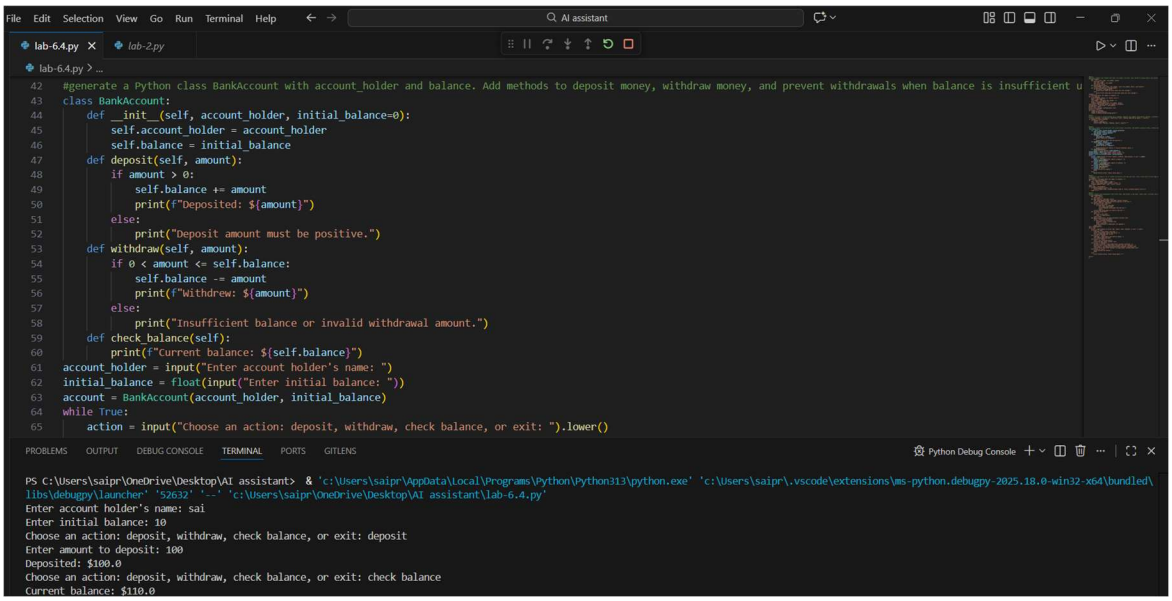
Prompt: #generate a Python class BankAccount with account_holder and balance. Add methods to deposit money, withdraw money, and prevent withdrawals when balance is insufficient using if-else user input.

Code : class BankAccount:

```
    def __init__(self, account_holder, initial_balance=0):
        self.account_holder = account_holder
        self.balance = initial_balance
    def deposit(self, amount):
        if amount > 0:
            self.balance += amount
            print(f"Deposited: ${amount}")
        else:
            print("Deposit amount must be positive.")
    def withdraw(self, amount):
        if 0 < amount <= self.balance:
            self.balance -= amount
            print(f"Withdrew: ${amount}")
        else:
            print("Insufficient balance or invalid withdrawal amount.")
    def check_balance(self):
        print(f"Current balance: ${self.balance}")
account_holder = input("Enter account holder's name: ")
initial_balance = float(input("Enter initial balance: "))
account = BankAccount(account_holder, initial_balance)
while True:
    action = input("Choose an action: deposit, withdraw, check balance, or exit: ").lower()
    if action == "deposit":
        amount = float(input("Enter amount to deposit: "))
        account.deposit(amount)
    elif action == "withdraw":
        amount = float(input("Enter amount to withdraw: "))
        account.withdraw(amount)
```

```
elif action == "check balance":
    account.check_balance()
elif action == "exit":
    print("Exiting the program.")
    break
else:
    print("Invalid action. Please choose again.")
```

Output :



Code Analysis :

- Creates a BankAccount class with account holder and balance.
- Includes methods to deposit and withdraw money.
- Uses if-else to prevent withdrawal when balance is insufficient.

- Accepts user input to perform banking operations repeatedly.
- Displays current balance and transaction messages.

Task-4

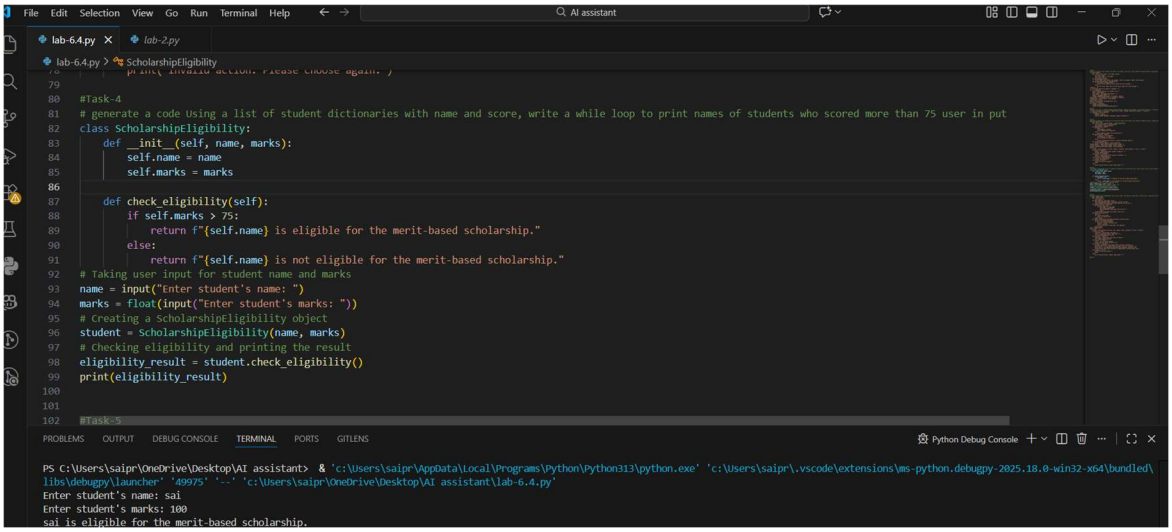
Prompt: Generate student scholoarship eligibility for a merit based scholarship system where students with marks above 75 are eligible for scholarship,and method to check eligibility.

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."
name = input("Enter student's name: ")
marks = float(input("Enter student's marks: "))
student = ScholarshipEligibility(name, marks)
eligibility_result = student.check_eligibility()
print(eligibility_result)
```

Output :



Code Analysis :

- The code defines a ScholarshipEligibility class with attributes name and marks.
- User input is taken to dynamically assign student details at runtime.
- The check_eligibility() method uses an if-else condition to evaluate eligibility.
- Students scoring more than 75 marks are considered eligible for the scholarship.
- The result is returned as a message and printed in a user-friendly format.

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:

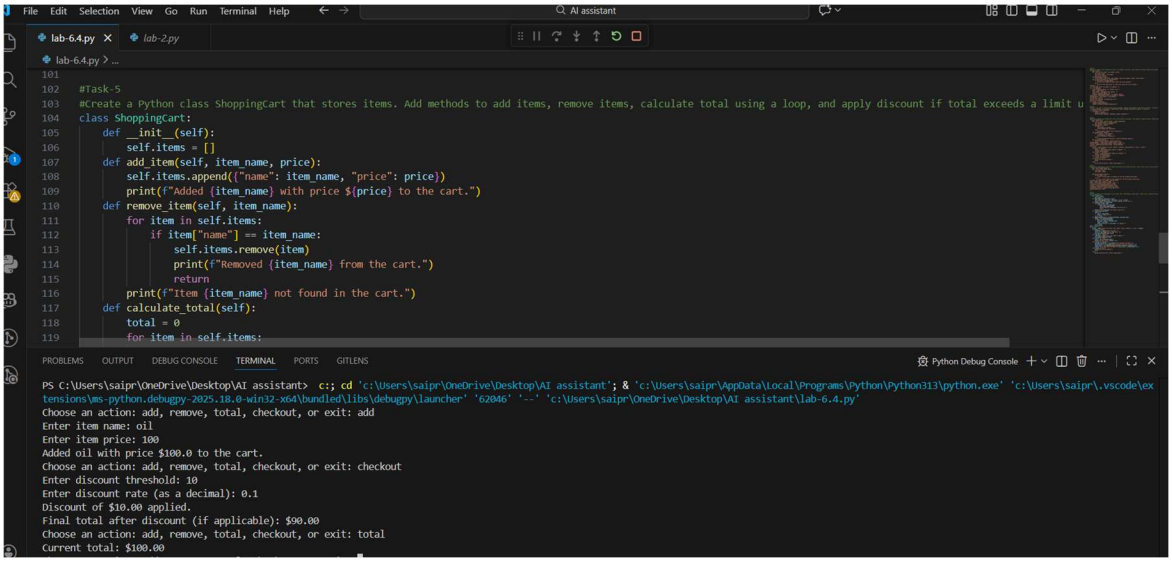
```
    def __init__(self):
        self.items = []
    def add_item(self,
item_name, price):
```

```
        self.items.append({"na
me": item_name, "price":
price})
        print(f"Added
{item_name} with price
${price} to the cart.")
    def remove_item(self,
item_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):
        total = 0
        for item in self.items:
            total +=
item["price"]
        return total
    def apply_discount(self,
discount_threshold,
discount_rate):
```

```
        total =
self.calculate_total()
        if total >
discount_threshold:
            discount = total *
discount_rate
            total -= discount
            print(f"Discount of
${discount:.2f} applied.")
        return total
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_na
me, 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(discou
nt_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 :



Code Analysis :

- Defines a ShoppingCart class to store items and prices.
- Allows adding and removing items using class methods.
- Calculates the total price using a loop.
- Applies a discount when total exceeds a given threshold using if.
- Provides a menu-driven interface with user input.