ASSIGNMENT – 6.4

2303A51525

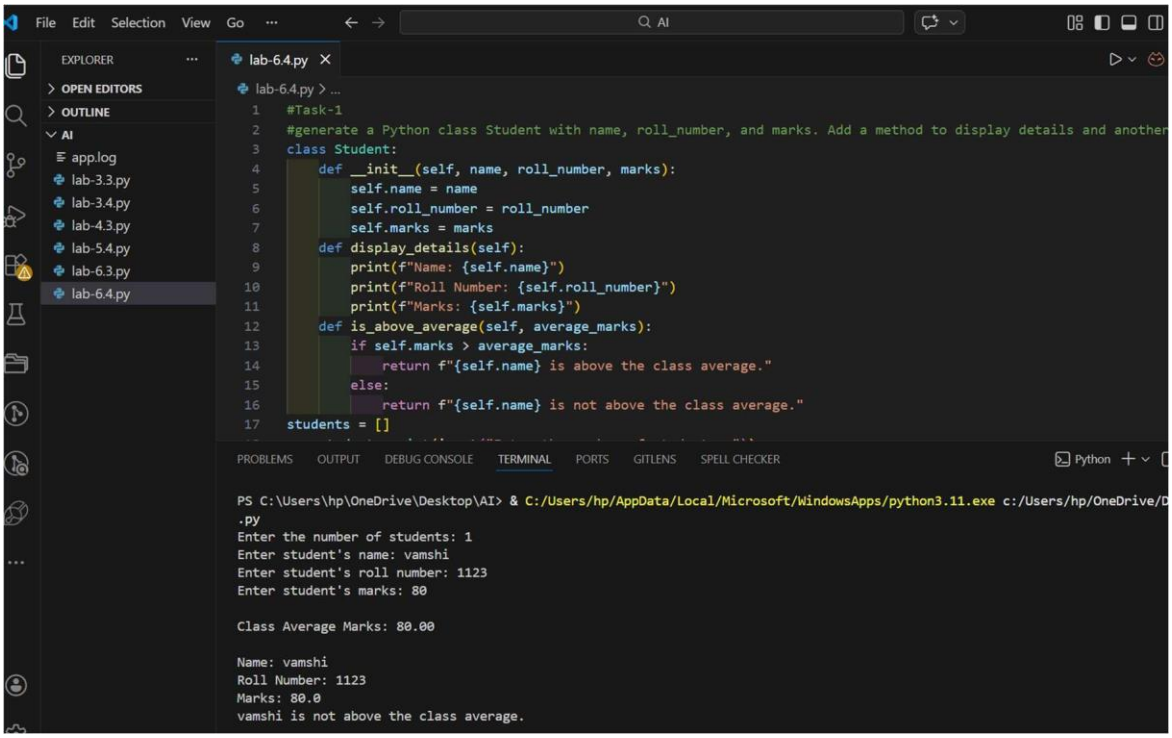
Batch-10

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 methods to display student details and compare marks with class average.
- Takes user input to create multiple student objects and stores them in a list.
- Calculates class average using total marks and number of students.
- Uses if-else to check whether a student is above average

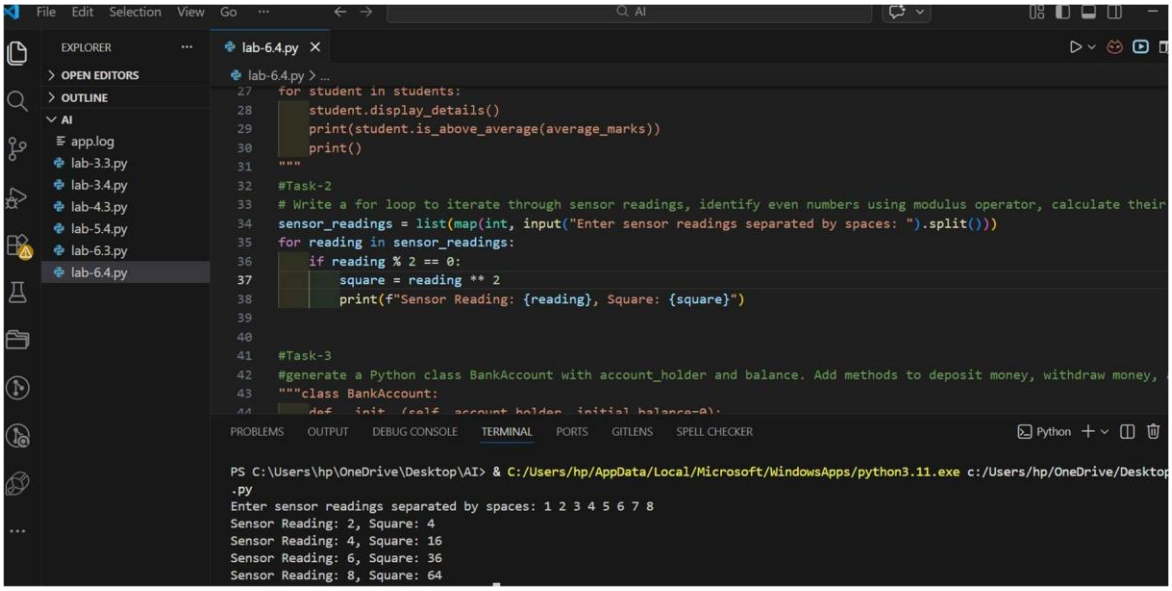
Task-2

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 multiple sensor readings as user input in a list format.
- Uses a for loop to iterate through each sensor reading.
- Applies the modulus operator (%) to identify even numbers.
- Calculates the square of even readings using the exponent operator.
- Prints the reading and its square in a clear format.

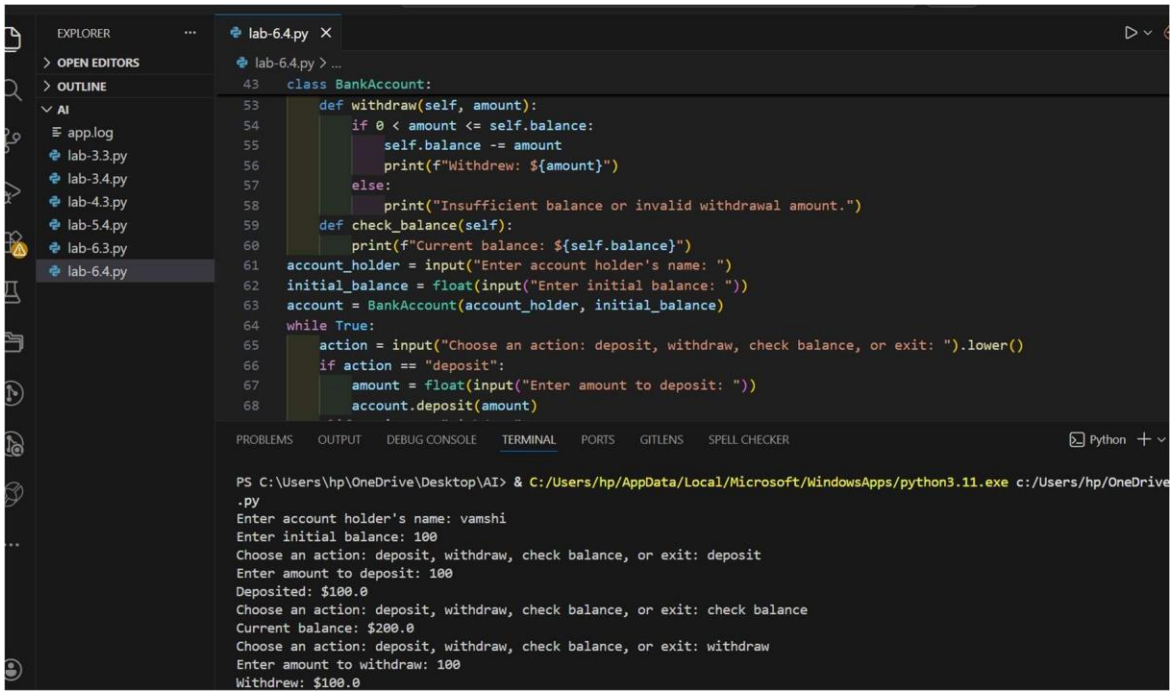
Task-3

Prompt: generate a Python class Bank Account 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 name and balance.
- Includes methods for deposit, withdrawal, and balance checking.
- Uses if-else to prevent withdrawals with insufficient balance.
- Accepts user input through a menu-driven while loop.
- Ensures valid banking operations and safe program exit.

Task-4

Prompt: generate a code Using a list of student dictionaries with name and score, write a class ScholarshipEligibility: def __init__(self, name, marks):

self.name = name

self.marks = marks

def check_eligibility(self):

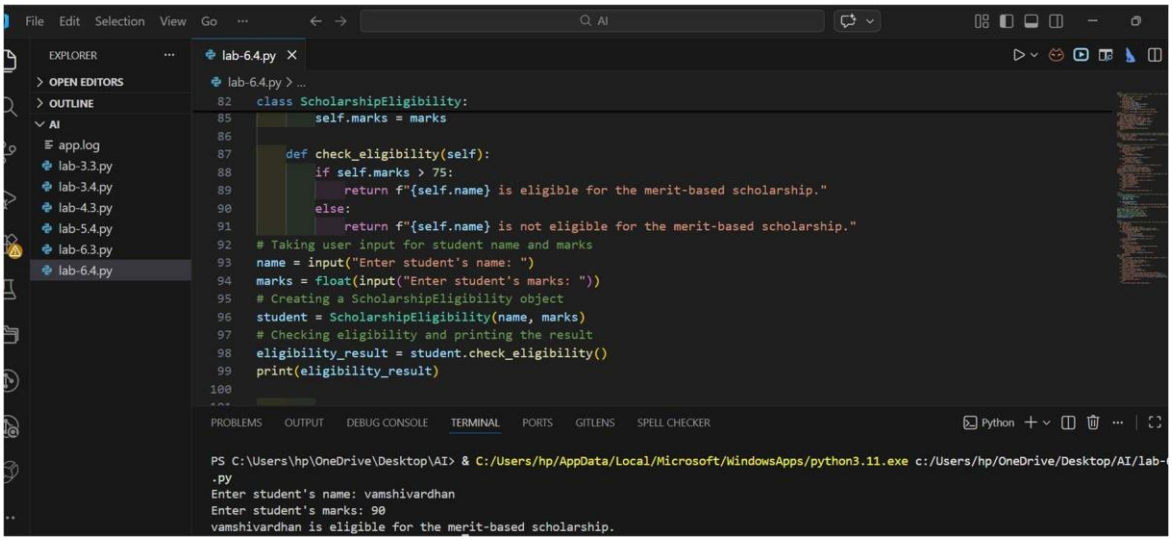
if self.marks > 75:

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 :



Code Analysis :

- Defines a class to store student name and marks.
- Takes user input for student details.
- Uses an if-else condition to check marks greater than 75.
- Determines eligibility for a merit-based scholarship.
- Displays the eligibility result clearly.

Task 5

Prompt: #Create a Python class Shopping Cart 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. class ShoppingCart: def __init__(self): self.items = [] def add_item(self, item_name, price):

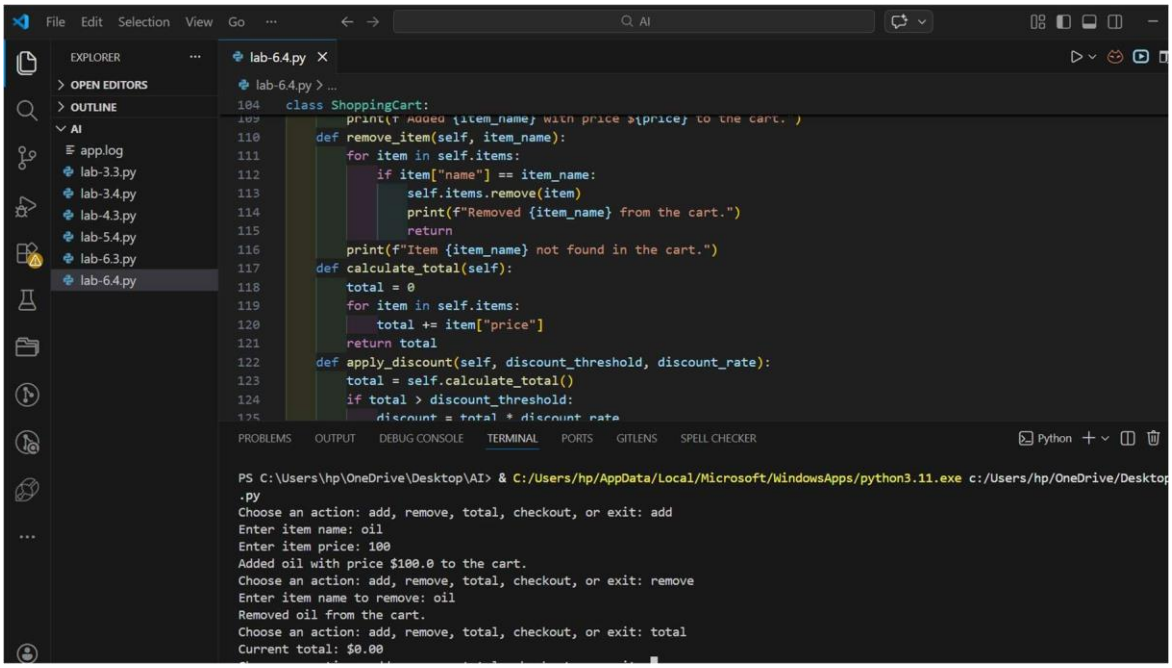
```
        self.items.append({"na
me": item_name, "price":
price})
        print(f"Added
{item_name} with price
${price} to the cart.")
def remove_item(self,
item_name):
        for item in self.items:
            if item["name"] ==
item_name:

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 :

- Implements a ShoppingCart class to store items in a list.
- Provides methods to add and remove items from the cart.
- Uses a loop to calculate the total price of items.
- Applies a discount when total exceeds a user-defined threshold.
- Uses a while loop for continuous user interaction.