

ASSIGNMENT – 6.4

2303A51197

Batch-10

Task-1

Prompt: Create a Python class named Student that includes attributes for name, roll number, and marks. Add one method to show student details and another method to determine, using if-else, whether the student's marks are higher than the class average. Accept input from the user.

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 :
```

```
1 # Create a Python class named Student that includes attributes for name, roll number, and marks. Add one method to show student details.
2 class Student:
3     def __init__(self, name, roll_number, marks):
4         self.name = name
5         self.roll_number = roll_number
6         self.marks = marks
7
8     def show_details(self):
9         print(f"Name: {self.name}")
10        print(f"Roll Number: {self.roll_number}")

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\PRAKASH\OneDrive\Desktop\Ai assisted codig> & C:/Users/PRAKASH/AppData/Local/Microsoft/WindowsApps/python3.11.exe "c:/
Enter username: prakash
Enter password: 12345
Login successful: False
PS C:\Users\PRAKASH\OneDrive\Desktop\Ai assisted codig> & C:/Users/PRAKASH/AppData/Local/Microsoft/WindowsApps/python3.11.exe "c:/
Enter the student's name: prakash
Enter the student's roll number: 2303A51197
Enter the student's marks: 99
Enter the class average marks: 85
Name: prakash
Roll Number: 2303A51197
Marks: 99.0
PS C:\Users\PRAKASH\OneDrive\Desktop\Ai assisted codig> █

```

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 Python program that takes sensor readings from the user, loops through them using a for loop, checks for even values with the modulus operator, computes their squares, and displays the output clearly.

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 :

```
#Write a Python program that takes sensor readings from the user, loops through them using a for loop, checks for even values with
# Accept sensor readings from the user
sensor_readings = input("Enter sensor readings separated by commas: ")
# Split the input into a list of readings
readings_list = sensor_readings.split(",")
# Loop through the readings and check for even values
for reading in readings_list:
    try:
        value = float(reading.strip())
        if value % 2 == 0:
            square = value ** 2
            print(f"Even value: {value}, Square: {square}")
    except ValueError:
        print(f"Invalid input: {reading.strip()} is not a number.")
#example input: 10, 15, 20, 25, 30
```

LEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
lid input: is not a number.
lid input: is not a number.
:\Users\PRAKASH\OneDrive\Desktop\Ai assisted codig> & C:/Users/PRAKASH/AppData/Local/Microsoft/WindowsApps/python3.11.exe "c:/Users/PR
r the student's name: Traceback (most recent call last):
le "c:\Users\PRAKASH\OneDrive\Desktop\Ai assisted codig\lab_6.4.py", line 19, in <module>
# name = input("Enter the student's name: ")
boardInterrupt
:\Users\PRAKASH\OneDrive\Desktop\Ai assisted codig> & C:/Users/PRAKASH/AppData/Local/Microsoft/WindowsApps/python3.11.exe "c:/Users/PR
r sensor readings separated by commas: 10,20,30,40,50
value: 10.0, Square: 100.0
value: 20.0, Square: 400.0
value: 30.0, Square: 900.0
value: 40.0, Square: 1600.0
value: 50.0, Square: 2500.0
:\Users\PRAKASH\OneDrive\Desktop\Ai assisted codig> []
```

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: **Develop a Python class called BankAccount with account holder name and balance as attributes. Include methods for depositing and withdrawing money, ensuring withdrawals are restricted when funds are insufficient using if-else conditions, and take 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 :

```

3 class BankAccount:
4     def __init__(self, account_holder_name, balance=0):
5         self.account_holder_name = account_holder_name
6         self.balance = balance
7
8     def deposit(self, amount):
9         if amount > 0:
10             self.balance += amount
11             print(f"Deposited: {amount}. New Balance: {self.balance}")
12         else:
13             print("Deposit amount must be positive.")
14
15     def withdraw(self, amount):
16         if amount > self.balance:
17             print("Insufficient funds. Withdrawal denied.")
18         elif amount <= 0:
19             print("Withdrawal amount must be positive.")
20         else:
21             self.balance -= amount
22             print(f"Withdrew: {amount}. New Balance: {self.balance}")

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

C:\Users\PRAKASH\OneDrive\Desktop\Ai assisted codig>
Invalid input: is not a number.
Invalid input: is not a number. _
Enter value: 50.0, Square: 2500.0
C:\Users\PRAKASH\OneDrive\Desktop\Ai assisted codig> ^C
C:\Users\PRAKASH\OneDrive\Desktop\Ai assisted codig> & C:/Users/PRAKASH/AppData/Local/Microsoft/WindowsApps/python3.11.exe "c:/Users/PRAKASH/OneD
Enter the account holder's name: prakash
Enter the initial balance: 100000
Enter the amount to deposit: 9
Enter the amount to withdraw: 8
Deposited: 9.0. New Balance: 100009.0
Withdrew: 8.0. New Balance: 100001.0
C:\Users\PRAKASH\OneDrive\Desktop\Ai assisted codig>

```

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: Using student details (name and marks), create a Python class

ScholarshipEligibility that checks through an if-else condition whether a student qualifies for a merit-based scholarship (marks above 75) and display the result based on user input.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 :

```

5 class ScholarshipEligibility:
6     def __init__(self, name, marks):
7         self.name = name
8         self.marks = marks
9
10    def check_eligibility(self):
11        if self.marks > 75:
12            print(f"{self.name} is eligible for the merit-based scholarship.")
13        else:
14            print(f"{self.name} is not eligible for the merit-based scholarship.")
15
16    # Accept user input for student details
17    name = input("Enter the student's name: ")
18    marks = float(input("Enter the student's marks: "))
19    # Create a ScholarshipEligibility object
20    student = ScholarshipEligibility(name, marks)
21    # Check scholarship eligibility
22    student.check_eligibility()

```

en value: 50.0, Square: 2500.0
 C:\Users\PRAKASH\OneDrive\Desktop\Ai assisted codig> ^C ...
 C:\Users\PRAKASH\OneDrive\Desktop\Ai assisted codig> & C:/Users/PRAKASH/AppData/Local/Microsoft/WindowsApps/python3.11.exe "
 C:\Users\PRAKASH\OneDrive\Desktop\Ai assisted codig> & C:/Users/PRAKASH/AppData/Local/Microsoft/WindowsApps/python3.11.exe "
 Enter the student's name: nbjb
 Enter the student's marks: 99
 nbjb is eligible for the merit-based scholarship.
 C:\Users\PRAKASH\OneDrive\Desktop\Ai assisted codig>

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({"name": 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:

```



```

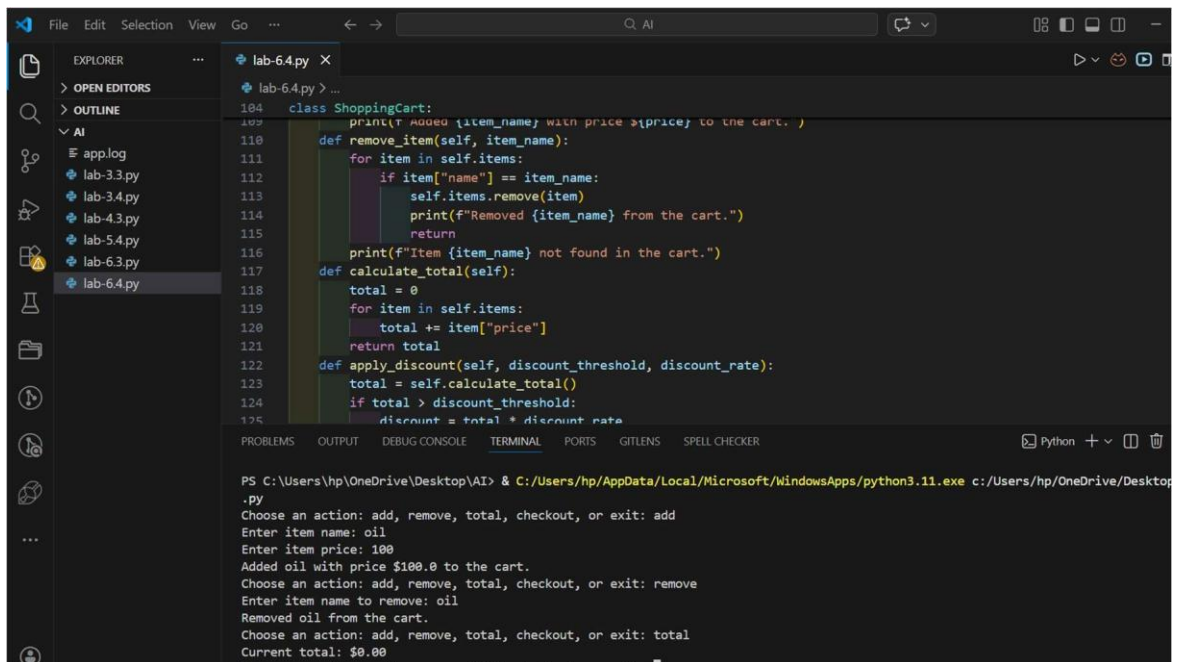
        discount = total *
discount_rate
        total -= discount
print(f"Discount      of
${discount:.2f} applied.")
        return total

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_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
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 :



The image shows a Visual Studio Code editor window with a Python file named `lab-6.4.py` open. The code defines a `ShoppingCart` class with the following methods:

- `add_item(self, item_name, price)`: Adds an item to the cart. It prints a confirmation message and appends the item to `self.items`.
- `remove_item(self, item_name)`: Removes an item from the cart. It iterates through `self.items` and removes the first item matching the name. It prints a confirmation message and returns the removed item name.
- `calculate_total(self)`: Calculates the total price of items in the cart by summing the 'price' attribute of each item.
- `apply_discount(self, discount_threshold, discount_rate)`: Applies a discount to the total if it exceeds a specified threshold. The discount is calculated as `total * discount_rate`.

The terminal at the bottom shows the execution of the script, which prompts the user for actions. The sequence of interactions is:

```
PS C:\Users\hp\OneDrive\Desktop\AI> & C:/Users/hp/AppData/Local/Microsoft/WindowsApps/python3.11.exe c:/Users/hp/OneDrive/Desktop/.../lab-6.4.py
.py
Choose an action: add, remove, total, checkout, or exit: add
Enter item name: oil
Enter item price: 100
Added oil with price $100.0 to the cart.
Choose an action: add, remove, total, checkout, or exit: remove
Enter item name to remove: oil
Removed oil from the cart.
Choose an action: add, remove, total, checkout, or exit: total
Current total: $0.00
```

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.