

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 :

The screenshot shows the VS Code interface with the file `lab-6.py` open. The terminal window displays the following output:

```

PS C:\Users\saipr\OneDrive\Desktop\AI assistant> & 'c:\Users\saipr\AppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\saipr\vscode\extensions\ms-python.python.debug-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '58092' '--' 'c:\Users\saipr\OneDrive\Desktop\AI assistant\lab-6.py'
Enter the number of students: 2
Enter student's name: sai
Enter student's roll number: 1355
Enter student's marks: 100
Enter student's name: hari
Enter student's roll number: 10
Enter student's marks: 200
Class Average Marks: 150.00
Name: sai
Roll Number: 1355
Marks: 100.0
sai is not above the class average.

② Name: hari
Roll Number: 10
Marks: 200.0
hari is above the class average.

```

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 :

```
File Edit Selection View Go Run Terminal Help ⏮ ⏯ 🔍 AI assistant ⌂ ⌂ ⌂

lab-6.4.py ✘ lab-2.py
lab-6.4.py > [?] sensor_readings
26     print(f"Class Average: {average_marks:.2f}")
27     print(f"{'='*50}\n")
28     for student in students:
29         student.display_details()
30         student.is_above_average(average_marks)"""
31
32 #Task-2
33 # 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.
34 sensor_readings = list(map(int, input("Enter sensor readings separated by spaces: ").split()))
35 for reading in sensor_readings:
36     if reading % 2 == 0:
37         square = reading ** 2
38         print(f"Sensor Reading: {reading}, Square: {square}")
39
40
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS
Python Dev
```

```
PS C:\Users\saipr\OneDrive\Desktop\AI assistant> & 'c:\Users\saipr\AppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\saipr\.vscode\extensions\ms-python.debug-langs\debugpy\launcher' '58976' '---' 'c:\Users\saipr\OneDrive\Desktop\AI assistant\lab-6.4.py'
Enter sensor readings separated by spaces: 1 2 3 4 5 6 7 8
Sensor Reading: 2, Square: 4
Sensor Reading: 4, Square: 16
Sensor Reading: 6, Square: 36
Sensor Reading: 8, Square: 64
```

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 :

The screenshot shows a VS Code interface with a dark theme. The left sidebar has 'File', 'Edit', 'Selection', 'View', 'Go', 'Run', 'Terminal', 'Help' menus. The top bar has a search field 'Q AI assistant' and a close button. The main area shows a Python file 'lab-64.py' with the following code:

```
42 #generate a Python class BankAccount with account_holder and balance. Add methods to deposit money, withdraw money, and prevent withdrawals when balance is insufficient
43 class BankAccount:
44     def __init__(self, account_holder, initial_balance=0):
45         self.account_holder = account_holder
46         self.balance = initial_balance
47     def deposit(self, amount):
48         if amount > 0:
49             self.balance += amount
50             print("Deposited: ${amount}")
51         else:
52             print("Deposit amount must be positive.")
53     def withdraw(self, amount):
54         if 0 < amount <= self.balance:
55             self.balance -= amount
56             print("Withdraw: ${amount}")
57         else:
58             print("Insufficient balance or invalid withdrawal amount.")
59     def check_balance(self):
60         print(f"Current balance: ${self.balance}")
61 account_holder = input("Enter account holder's name: ")
62 initial_balance = float(input("Enter initial balance: "))
63 account = BankAccount(account_holder, initial_balance)
64 while True:
65     action = input("Choose an action: deposit, withdraw, check balance, or exit: ").lower()
```

The terminal below shows the execution of the script:

```
PS C:\Users\saipr\OneDrive\Desktop\AI assistant> & "c:\Users\saipr\AppData\Local\Programs\Python\Python313\python.exe" "c:\Users\saipr\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher" "52e32" ... "c:\Users\saipr\OneDrive\Desktop\AI assistant\lab-6.4.py"
Enter account holder's name: sai
Enter initial balance: 10
Choose an action: deposit, withdraw, check balance, or exit: deposit
Enter amount to deposit: 100
Deposited: $100.0
Choose an action: deposit, withdraw, check balance, or exit: check balance
Current balance: $110.0
```

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 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:
            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 :

The screenshot shows a VS Code interface with the following details:

- File Explorer:** Shows a tree view with files like "lab-6.4.py" and "ScholarshipEligibility".
- Code Editor:** Displays Python code for a "ScholarshipEligibility" class. The code includes a constructor, a check_eligibility method, and a main block that takes user input for name and marks, creates a student object, and prints the result of the check.
- Terminal:** Shows the command "PS C:\Users\saipr\OneDrive\Desktop\AI assistant> & "c:\Users\saipr\AppData\Local\Programs\Python\Python313\python.exe" "c:\Users\saipr\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\lib\idlelib\debugpy\launcher" 49975" -- "c:\Users\saipr\OneDrive\Desktop\AI assistant\lab-6.4.py"".
- Output:** Shows the terminal output:

```
Enter student's name: sai
Enter student's marks: 100
sai is eligible for the merit-based scholarship.
```

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({"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_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(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 :

```
File Edit Selection View Go Run Terminal Help ⏎ ↻ AI assistant 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 #Task-5 #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 up to 10% off. class ShoppingCart:     def __init__(self):         self.items = []     def add_item(self, item_name, price):         self.items.append({"name": item_name, "price": price})         print("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     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 PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS PS C:\Users\saipr\OneDrive\Desktop\AI assistant> c; cd 'c:\Users\saipr\OneDrive\Desktop\AI assistant'; & 'c:\Users\saipr\AppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\saipr\vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '62046' '--' 'c:\Users\saipr\OneDrive\Desktop\AI assistant\lab-6.4.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: checkout Enter discount threshold: 10 Enter discount rate (as a decimal): 0.1 Discount of $10.00 applied. Final total after discount (if applicable): $90.00 choose an action: add, remove, total, checkout, or exit: total Current total: $100.00
```

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.