

# LAB ASSIGNMENT-6.4

2303A51337

BATCH-10

## ➤ TASK-1:

### **PROMPT:**

Generate methods to display student details and check if marks are above class average using if-else and self attributes with user input.

### **CODE:**

```
class Student:  
  
    def __init__(self, name, age, marks):  
        self.name = name  
        self.age = age  
        self.marks = marks  
  
    def display_details(self):  
        print(f"Name: {self.name}")  
        print(f"Age: {self.age}")  
        print(f"Marks: {self.marks}")  
  
    def is_above_average(self, average_marks):  
        if self.marks > average_marks:  
            return True  
        else:  
            return False  
  
# Example usage  
if __name__ == "__main__":  
    name = input("Enter student name: ")  
    age = int(input("Enter student age: "))  
    marks = float(input("Enter student marks: "))
```

```

average_marks = float(input("Enter class average marks: "))

student = Student(name, age, marks)

student.display_details()

if student.is_above_average(average_marks):

    print(f"{student.name} has marks above the class average.")

else:

    print(f"{student.name} does not have marks above the class average.")

```

## OUTPUT:

The screenshot shows the Visual Studio Code interface. The code editor displays a Python script named lab64.py. The terminal below it shows the execution of the script and its output. The output pane shows the program's output, which includes user input and the resulting message.

```

File Edit Selection View Go Run Terminal Help ⌘ ⌘ AI Assisted
OPEN EDITORS 1 unsaved
EXPLORER
lab14.py
lab15.py
lab16.py
lab17.py
lab18.py
lab19.py
lab20.py
lab21.py
lab22.py
lab23.py
lab24.py
lab25.py
lab26.py
lab27.py
lab28.py
lab29.py
lab30.py
lab31.py
lab32.py
lab33.py
lab34.py
lab35.py
lab36.py
lab37.py
lab38.py
lab39.py
lab40.py
lab41.py
lab42.py
lab43.py
lab44.py
lab45.py
lab46.py
lab47.py
lab48.py
lab49.py
lab50.py
lab51.py
lab52.py
lab53.py
lab54.py
lab55.py
lab56.py
lab57.py
lab58.py
lab59.py
lab60.py
lab61.py
lab62.py
lab63.py
lab64.py
lab65.py
lab66.py
lab67.py
lab68.py
lab69.py
lab70.py
lab71.py
lab72.py
lab73.py
lab74.py
lab75.py
lab76.py
lab77.py
lab78.py
lab79.py
lab80.py
lab81.py
lab82.py
lab83.py
lab84.py
lab85.py
lab86.py
lab87.py
lab88.py
lab89.py
lab90.py
lab91.py
lab92.py
lab93.py
lab94.py
lab95.py
lab96.py
lab97.py
lab98.py
lab99.py
lab100.py
lab101.py
lab102.py
lab103.py
lab104.py
lab105.py
lab106.py
lab107.py
lab108.py
lab109.py
lab110.py
lab111.py
lab112.py
lab113.py
lab114.py
lab115.py
lab116.py
lab117.py
lab118.py
lab119.py
lab120.py
lab121.py
lab122.py
lab123.py
lab124.py
lab125.py
lab126.py
lab127.py
lab128.py
lab129.py
lab130.py
lab131.py
lab132.py
lab133.py
lab134.py
lab135.py
lab136.py
lab137.py
lab138.py
lab139.py
lab140.py
lab141.py
lab142.py
lab143.py
lab144.py
lab145.py
lab146.py
lab147.py
lab148.py
lab149.py
lab150.py
lab151.py
lab152.py
lab153.py
lab154.py
lab155.py
lab156.py
lab157.py
lab158.py
lab159.py
lab160.py
lab161.py
lab162.py
lab163.py
lab164.py
lab165.py
lab166.py
lab167.py
lab168.py
lab169.py
lab170.py
lab171.py
lab172.py
lab173.py
lab174.py
lab175.py
lab176.py
lab177.py
lab178.py
lab179.py
lab180.py
lab181.py
lab182.py
lab183.py
lab184.py
lab185.py
lab186.py
lab187.py
lab188.py
lab189.py
lab190.py
lab191.py
lab192.py
lab193.py
lab194.py
lab195.py
lab196.py
lab197.py
lab198.py
lab199.py
lab200.py
lab201.py
lab202.py
lab203.py
lab204.py
lab205.py
lab206.py
lab207.py
lab208.py
lab209.py
lab210.py
lab211.py
lab212.py
lab213.py
lab214.py
lab215.py
lab216.py
lab217.py
lab218.py
lab219.py
lab220.py
lab221.py
lab222.py
lab223.py
lab224.py
lab225.py
lab226.py
lab227.py
lab228.py
lab229.py
lab230.py
lab231.py
lab232.py
lab233.py
lab234.py
lab235.py
lab236.py
lab237.py
lab238.py
lab239.py
lab240.py
lab241.py
lab242.py
lab243.py
lab244.py
lab245.py
lab246.py
lab247.py
lab248.py
lab249.py
lab250.py
lab251.py
lab252.py
lab253.py
lab254.py
lab255.py
lab256.py
lab257.py
lab258.py
lab259.py
lab260.py
lab261.py
lab262.py
lab263.py
lab264.py
lab265.py
lab266.py
lab267.py
lab268.py
lab269.py
lab270.py
lab271.py
lab272.py
lab273.py
lab274.py
lab275.py
lab276.py
lab277.py
lab278.py
lab279.py
lab280.py
lab281.py
lab282.py
lab283.py
lab284.py
lab285.py
lab286.py
lab287.py
lab288.py
lab289.py
lab290.py
lab291.py
lab292.py
lab293.py
lab294.py
lab295.py
lab296.py
lab297.py
lab298.py
lab299.py
lab299.py
#task2
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS
cs/AI Assisted/lab64.py"
Using student name: varsha
Enter student age: 20
Enter student marks: 95
Enter class average marks: 90
Name: varsha
Age: 20
Marks: 95.0
varsha has marks above the class average.
PS C:\Users\ariva\Desktop\b3-2 academics\AI Assisted> []

```

## EXPLANATION:

- This program creates a Student class with attributes for name, age, and marks.
- The constructor initializes these values when an object is created.
- The display\_details() method prints the student's information using self.
- The is\_above\_average() method compares the student's marks with the class average using an if-else condition and returns a result.
- User input is taken to create a student object, and the program displays the details and prints whether the student's marks are above the class average or not.

## TASK-2

## PROMPT:

For each sensor reading, check if it is even using modulus, square it, and print the output.

## **CODE:**

```
def process_sensor_readings(readings):
```

for reading in readings:

if reading % 2 == 0:

```
squared = reading ** 2
```

prim

else:

```
print(f"Señor")
```

## # Example usage

Name      Class :

`sensor_readings` [15, 15, 22, 33, 15, 33]

100

## **EXPLANATION:**

- This program processes a list of sensor readings using a loop.
- It checks each reading to see if it is even using the modulus operator.
- If the reading is even, it calculates its square and prints the result. If the reading is odd, it prints a message saying no processing is done.
- This shows the use of loops, conditionals, and arithmetic operations in Python.

## **❖ TASK-3:**

### **PROMPT:**

Create a class structure for BankAccount.Create a method to deposit money into the account.The method should take amount as parameter.Add the amount to balance and display updated balance

Create a method to withdraw money.Check if sufficient balance is available using if-else.If balance is enough, deduct amount and print success message Otherwise print 'Insufficient Balance'

### **CODE:**

```
class BankAccount:  
  
    def __init__(self, account_holder, balance):  
        self.account_holder = account_holder  
        self.balance = balance  
  
    # Create a method to deposit money into the account.The method should take amount  
    # as parameter.Add the amount to balance and display updated balance  
  
    def deposit(self, amount):  
        self.balance += amount  
        print(f"Deposit Successful! New Balance: {self.balance}")  
  
    # Create a method to withdraw money.Check if sufficient balance is available using if-  
    # else.If balance is enough, deduct amount and print success message Otherwise print  
    # 'Insufficient Balance'  
  
    def withdraw(self, amount):  
        if amount <= self.balance:  
            self.balance -= amount  
            print(f"Withdrawal Successful! Remaining Balance: {self.balance}")
```

```

else:
    print("Insufficient Balance! Withdrawal Failed.")

# Create a method to display account details

def display_details(self):

    print("Account Holder:", self.account_holder)

    print("Current Balance:", self.balance)

# Sample Usage

if __name__ == "__main__":
    # Creating a BankAccount object

    account = BankAccount("varsha", 5000)

    account.display_details()

    print("\nPerforming Transactions...\n")

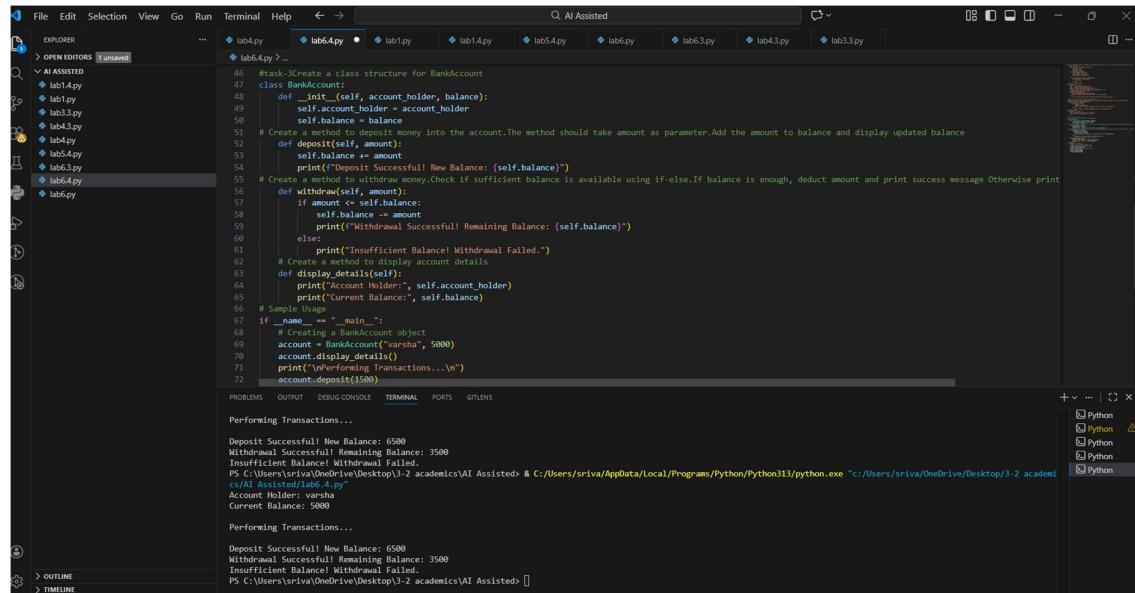
    account.deposit(1500)

    account.withdraw(3000)

    account.withdraw(5000)

```

## OUTPUT:



The screenshot shows a code editor interface with the following details:

- File Structure:** The left sidebar shows an Explorer view with files like lab6.py, lab6.4.py, lab1.py, lab1.4.py, lab5.py, lab6.py, lab6.3.py, lab4.3.py, and lab3.3.py.
- Code Editor:** The main area contains Python code for a BankAccount class and its usage. The code is as follows:

```

46 #task-3>Create a class structure for BankAccount
47 class BankAccount:
48     def __init__(self, account_holder, balance):
49         self.account_holder = account_holder
50         self.balance = balance
51 
52     # Create a method to deposit money into the account. The method should take amount as parameter. Add the amount to balance and display updated balance
53     def deposit(self, amount):
54         self.balance += amount
55         print("Deposit Successful! New Balance: ", self.balance)
56 
57     # Create a method to withdraw money. Check if sufficient balance is available using if-else. If balance is enough, deduct amount and print success message. Otherwise print
58     # insufficient balance
59     def withdraw(self, amount):
60         if amount <= self.balance:
61             self.balance -= amount
62             print("Withdrawal Successful! Remaining Balance: ", self.balance)
63         else:
64             print("Insufficient Balance! Withdrawal Failed.")
65 
66     # Create a method to display account details
67     def display_details(self):
68         print("Account Holder:", self.account_holder)
69         print("Current Balance:", self.balance)
70 
71 # Sample Usage
72 if __name__ == "__main__":
73     # Creating a BankAccount object
74     account = BankAccount("varsha", 5000)
75     account.display_details()
76     print("\nPerforming Transactions...\n")
77     account.deposit(1500)

```

- Terminal Output:** Below the code editor, the terminal window shows the execution of the script and its output:

```

Performing Transactions...
Deposit Successful! New Balance: 6500
Withdrawal Successful! Remaining Balance: 3500
Insufficient Balance! Withdrawal Failed.
PS C:\Users\sriva\OneDrive\Desktop\3-2 academis\AI Assisted> & C:/Users/sriva/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/sriva/OneDrive/Desktop/3-2 academis\AI Assisted\lab6.4.py"
Account Holder: varsha
Current Balance: 5000
Performing Transactions...
Deposit Successful! New Balance: 6500
Withdrawal Successful! Remaining Balance: 3500
Insufficient Balance! Withdrawal Failed.
PS C:\Users\sriva\OneDrive\Desktop\3-2 academis\AI Assisted>

```

## **EXPLANATION:**

This program creates a BankAccount class with account holder and balance attributes. It provides methods to deposit money and withdraw money using if-else to check for sufficient balance. If the balance is not enough, it prints an “Insufficient Balance” message. The program also displays account details and demonstrates transactions using a sample account.

## **❖ TASK-4:**

### **PROMPT:**

Write a while loop to print names of students with score > 75 is eligible and less than 75 is not from the list USING USER INPUT

### **CODE:**

```
students = []

num_students = int(input("Enter the number of students: "))

for i in range(num_students):

    name = input(f"Enter name of student {i + 1}: ")

    score = float(input(f"Enter score of student {i + 1}: "))

    students.append((name, score))

print("\nStudent Eligibility:")

index = 0

while index < len(students):

    name, score = students[index]

    if score > 75:

        print(f"{name} is ELIGIBLE with a score of {score}.")

    else:

        print(f"{name} is NOT ELIGIBLE with a score of {score}.")

    index += 1
```

## OUTPUT:

The screenshot shows the VS Code interface with several files listed in the Explorer sidebar: lab4.py, lab64.py (the active file), lab1.py, lab1.4.py, lab54.py, lab6.py, lab63.py, lab43.py, and lab33.py. The terminal window displays the following code and its execution:

```
#TASK-4B Write a while loop to print names of students with score > 75 IS ELIGIBLE AND LESS THAN 75 NOT ELIGIBLE from the list USING USER INPUT
students = []
for i in range(int(input("Enter the number of students: ")))
    name = input("Enter name of student {i + 1}: ")
    score = float(input("Enter score of student {i + 1}: "))
    students.append((name, score))
print("\nStudent Eligibility:")
index = 0
while index < len(students):
    name, score = students[index]
    if score > 75:
        print(f"{name} is ELIGIBLE with a score of {score}.")
    else:
        print(f"{name} is NOT ELIGIBLE with a score of {score}.")
    index += 1
```

The terminal output shows the program's execution and the resulting student eligibility status:

```
PS C:\Users\sriva\OneDrive\Desktop\3-2 academics> & C:/Users/sriva/AppData/Local/Programs/Python/Python311/python.exe "c:/Users/sriva/OneDrive/Desktop/3-2 academics/lab64.py"
Enter the number of students: 3
Enter name of student 1: varsha
Enter score of student 1: 95
Enter name of student 2: ammu
Enter score of student 2: 73
Enter name of student 3: srivarsha
Enter score of student 3: 75

Student Eligibility:
varsha is ELIGIBLE with a score of 95.0.
ammu is NOT ELIGIBLE with a score of 73.0.
srivarsha is NOT ELIGIBLE with a score of 75.0.
```

## EXPLANATION:

This program collects student details name and score from the user and stores them in a list. It then uses a while loop to go through each student in the list one by one. For every student, it checks whether the score is greater than 75. If the score is above 75, the student is marked as eligible for the scholarship; otherwise, the student is marked as not eligible.

## ❖ TASK-5:

### PROMPT:

Generate methods to add/remove items, calculate total with loop, and apply discount using if-else

### CODE:

```
class ShoppingCart:

    def __init__(self):
        self.items = {}

    def add_item(self, item_name, price, quantity):
        if item_name in self.items:
            self.items[item_name]['quantity'] += quantity
```

```
else:
    self.items[item_name] = {'price': price, 'quantity': quantity}
    print(f"Added {quantity} of {item_name} to cart.")

def remove_item(self, item_name, quantity):
    if item_name in self.items:
        if quantity >= self.items[item_name]['quantity']:
            del self.items[item_name]
            print(f"Removed all of {item_name} from cart.")
        else:
            self.items[item_name]['quantity'] -= quantity
            print(f"Removed {quantity} of {item_name} from cart.")
    else:
        print(f"{item_name} not found in cart.")

def calculate_total(self):
    total = 0
    for item in self.items.values():
        total += item['price'] * item['quantity']
    return total

def apply_discount(self, discount_rate):
    total = self.calculate_total()
    if discount_rate > 0 and discount_rate < 1:
        discount_amount = total * discount_rate
        total_after_discount = total - discount_amount
        print(f"Discount Applied: ${discount_amount:.2f}")
    return total_after_discount

else:
    print("Invalid discount rate. No discount applied.")
return total
```

```

# Example usage

if __name__ == "__main__":
    cart = ShoppingCart()
    cart.add_item("Apple", 0.5, 10)
    cart.add_item("Banana", 0.3, 5)
    cart.add_item("Orange", 0.8, 8)
    print(f"Total before discount: ${cart.calculate_total():.2f}")
    total_after_discount = cart.apply_discount(0.1) # 10% discount
    print(f"Total after discount: ${total_after_discount:.2f}")
    cart.remove_item("Banana", 2)
    print(f"Total after removing items: ${cart.calculate_total():.2f}")

```

## OUTPUT:

The screenshot shows a code editor interface with the following details:

- File Explorer:** Shows multiple Python files: lab1.py, lab1A.py, lab1B.py, lab2.py, lab3.py, lab3A.py, lab3B.py, lab4.py, lab4A.py, lab4B.py, lab5.py, lab5A.py, lab5B.py, lab6.py, lab6A.py, lab6B.py.
- Terminal:** Displays the execution of the code. The output is as follows:
 

```

Added 10 of Apple to cart.
Added 5 of Banana to cart.
Added 8 of Orange to cart.
Total before discount: $12.90
Discount applied: $1.29
Total after discount: $11.61
Removed 2 of Banana from cart.
Total after removing items: $12.30
PS C:\Users\sriva\OneDrive\Desktop\3-2 academics\AI Assisted>
      
```
- Bottom Status Bar:** Shows file path (PS C:\Users\sriva\OneDrive\Desktop\3-2 academics\AI Assisted>), line number (Ln 144, Col 1), and other development status.

## EXPLANATION:

This program creates a ShoppingCart class that stores items with their price and quantity. It allows adding and removing items from the cart, calculates the total cost using a loop, and applies a discount using a condition. The example shows item addition, discount application, and total updates after removal.