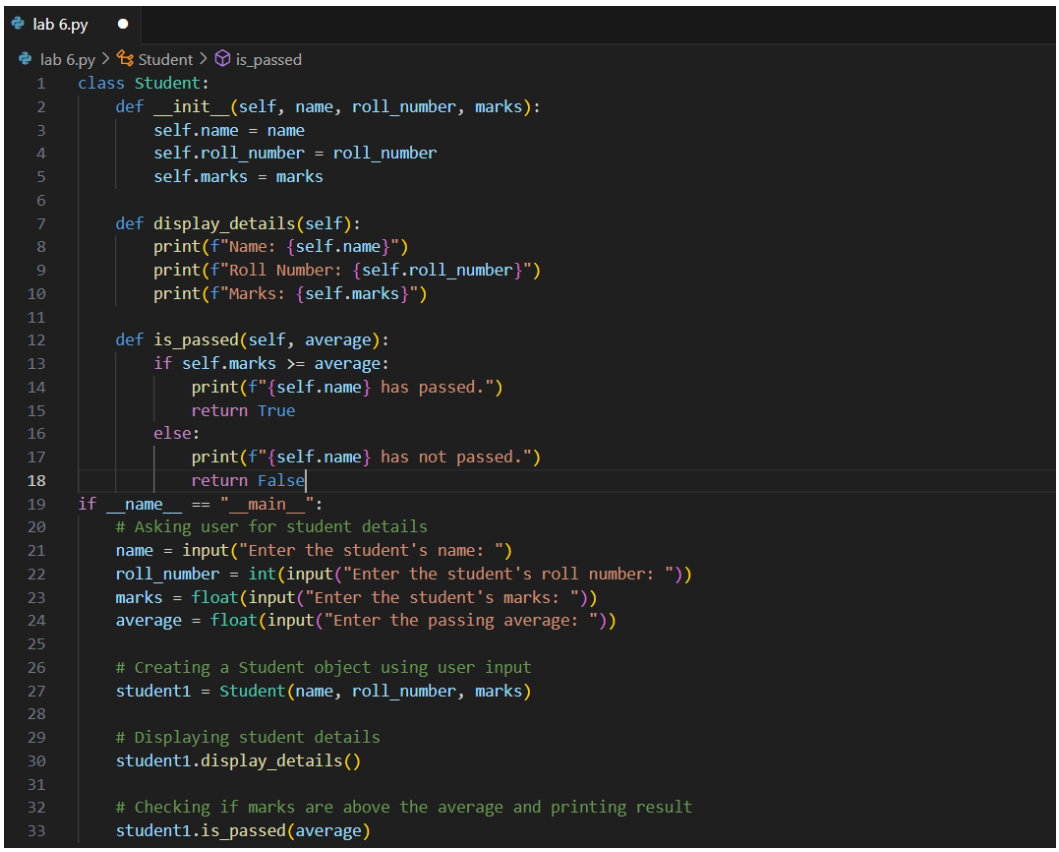
LAB ASSIGNMENT-6

TASK-1:

Prompt:

Generate a python code to ask input to user using a Class named Student with attributes name, roll_number, and marks to complete methods for displaying details and checking if marks are above average. Use classes like display_details() and is_passed(), demonstrating use of if-else conditions.

Code and Output:



lab 6.py

lab 6.py > Student > is_passed

```python
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_passed(self, average):
        if self.marks >= average:
            print(f"{self.name} has passed.")
            return True
        else:
            print(f"{self.name} has not passed.")
            return False
if __name__ == "__main__":
    # Asking user for student details
    name = input("Enter the student's name: ")
    roll_number = int(input("Enter the student's roll number: "))
    marks = float(input("Enter the student's marks: "))
    average = float(input("Enter the passing average: "))

    # Creating a Student object using user input
    student1 = Student(name, roll_number, marks)

    # Displaying student details
    student1.display_details()

    # Checking if marks are above the average and printing result
    student1.is_passed(average)
```

```
PS C:\Users\mahit\OneDrive\Desktop\AIAC> & C:/Users/mahit/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/mahit/OneDrive/Desktop/AIAC/lab 6.py"
Enter the student's name: Vivek
Enter the student's roll number: 97
Enter the student's marks: 99
Enter the passing average: 40
Name: Vivek
Roll Number: 97
Marks: 99.0
Vivek has passed.
```

Code Explanation:

The code defines a `Student` class with the following parts:

- **Constructor (`__init__`)**
  Initializes a student object with attributes: `name` , `roll_number` , and `marks` .
- `display_details` **Method**
  Prints the student's details (name, roll number, and marks) to the console.
- `is_passed` **Method**
  Takes an `average` as a parameter and uses an if-else statement to check if the student's marks are equal to or above the average. It prints whether the student has passed or failed and returns a Boolean result.
- **Main Execution Block**
  When the script is run (rather than imported as a module), the following happens:
  - The program prompts the user to input the student's name, roll number, marks, and the passing average.
  - It creates an instance of the `Student` class using the provided input.
  - It calls the `display_details` method to show the student's information.
  - Finally, it invokes the `is_passed` method to determine if the student's marks meet the passing threshold.

Overall, the code demonstrates basic object-oriented programming concepts and user input handling in Python along with simple conditional logic.

TASK-2:

Prompt:

for number in numbers:

    if number % 2 == 0:

#check square of even numbers only

Code and Output:

lab 6.py

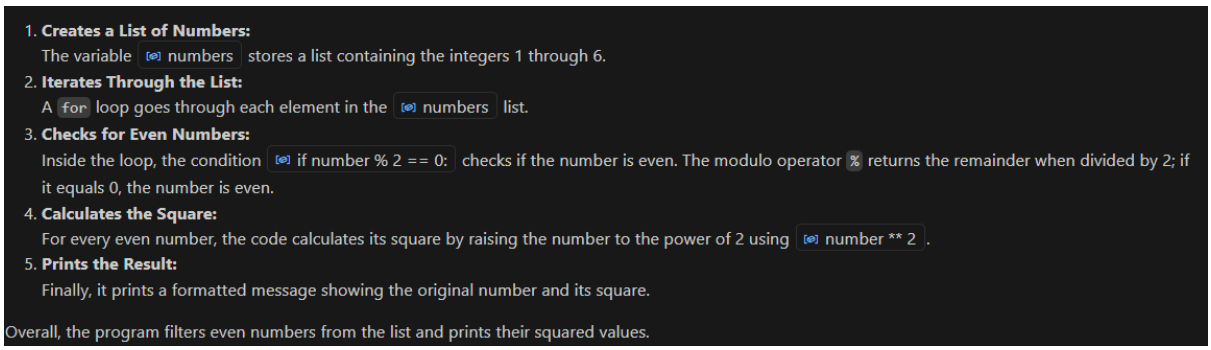
```
numbers = [1, 2, 3, 4, 5, 6]

for number in numbers:
    if number % 2 == 0:
        # Check square of even numbers only
        square = number ** 2
        print(f"The square of {number} is {square}")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\mahit\OneDrive\Desktop\AIAC> & C:/Users/mahit/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/mahit/OneDrive/Desktop/AIAC/lab 6.p
y"
The square of 2 is 4
The square of 4 is 16
The square of 6 is 36
```

Code Explanation:



1. **Creates a List of Numbers:**
   The variable `numbers` stores a list containing the integers 1 through 6.
2. **Iterates Through the List:**
   A `for` loop goes through each element in the `numbers` list.
3. **Checks for Even Numbers:**
   Inside the loop, the condition `if number % 2 == 0:` checks if the number is even. The modulo operator `%` returns the remainder when divided by 2; if it equals 0, the number is even.
4. **Calculates the Square:**
   For every even number, the code calculates its square by raising the number to the power of 2 using `number ** 2`.
5. **Prints the Result:**
   Finally, it prints a formatted message showing the original number and its square.

Overall, the program filters even numbers from the list and prints their squared values.

TASK-3:

Prompt:

Create a class called BankAccount with attributes account_holder and balance. Give a python code to ask input to user also complete methods for deposit(), withdraw(), and check for insufficient balance.

Code and Output:

lab 6.py

```python
class BankAccount:
    def __init__(self, account_holder, balance=0.0):
        self.account_holder = account_holder
        self.balance = balance

    def deposit(self, amount):
        if amount <= 0:
            print("Deposit amount must be positive.")
        else:
            self.balance += amount
            print(f"Deposit successful! New balance: {self.balance}")

    def withdraw(self, amount):
        if amount <= 0:
            print("Withdrawal amount must be positive.")
        elif amount > self.balance:
            print("Insufficient balance!")
        else:
            self.balance -= amount
            print(f"Withdrawal successful! New balance: {self.balance}")

    def display_balance(self):
        print(f"Account Holder: {self.account_holder}, Balance: {self.balance}")

def get_float_input(prompt_text):
    while True:
        try:
            return float(input(prompt_text))
        except ValueError:
            print("Please enter a valid number.")

def main():
    account_holder = input("Enter the account holder's name: ")
    initial_balance = get_float_input("Enter the initial balance: ")
    account = BankAccount(account_holder, initial_balance)

    while True:
        print("\n--- Bank Account Menu ---")
        print("1. Deposit")
        print("2. Withdraw")
        print("3. Check Balance")
        print("4. Exit")

        choice = input("Select an option: ")
```

```
        choice = input("Select an option: ")

        if choice == '1':
            amount = get_float_input("Enter deposit amount: ")
            account.deposit(amount)
        elif choice == '2':
            amount = get_float_input("Enter withdrawal amount: ")
            account.withdraw(amount)
        elif choice == '3':
            account.display_balance()
        elif choice == '4':
            print("Goodbye!")
            break
        else:
            print("Invalid option selected. Please try again.")

if __name__ == "__main__":
    main()
```

```
PS C:\Users\mahit\OneDrive\Desktop\AIAC> & C:/Users/mahit/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/mahit/OneDrive/Desktop/AIAC/lab 6.py"
Enter the account holder's name: Vivek
Enter the initial balance: 90000

--- Bank Account Menu ---
1. Deposit
2. Withdraw
3. Check Balance
4. Exit
Select an option: 1
Enter deposit amount: 10000
Deposit successful! New balance: 100000.0

--- Bank Account Menu ---
1. Deposit
--- Bank Account Menu ---
1. Deposit
2. Withdraw
3. Check Balance
4. Exit
1. Deposit
2. Withdraw
3. Check Balance
4. Exit
2. Withdraw
3. Check Balance
4. Exit
4. Exit
Select an option: 2
Enter withdrawal amount: 30000
Withdrawal successful! New balance: 70000.0
Select an option: 2
Enter withdrawal amount: 30000
Withdrawal successful! New balance: 70000.0
```

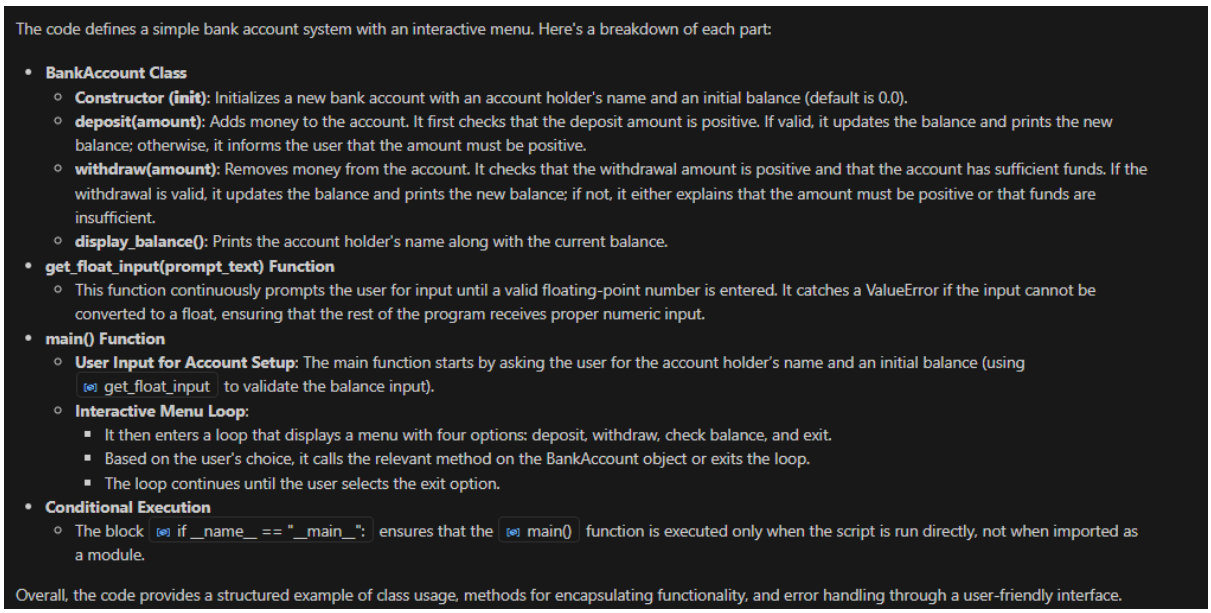
```
--- Bank Account Menu ---
1. Deposit
2. Withdraw
3. Check Balance
4. Exit
2. Withdraw
3. Check Balance
4. Exit
Select an option: 3
4. Exit
Select an option: 3
Account Holder: Vivek, Balance: 70000.0

Select an option: 3
Account Holder: Vivek, Balance: 70000.0

--- Bank Account Menu ---
1. Deposit
Account Holder: Vivek, Balance: 70000.0

--- Bank Account Menu ---
1. Deposit
--- Bank Account Menu ---
1. Deposit
2. Withdraw
3. Check Balance
4. Exit
2. Withdraw
3. Check Balance
4. Exit
4. Exit
Select an option: 4
Select an option: 4
Goodbye!
```

Code Explanation:



The code defines a simple bank account system with an interactive menu. Here's a breakdown of each part:

- **BankAccount Class**
  - **Constructor (init)**: Initializes a new bank account with an account holder's name and an initial balance (default is 0.0).
  - **deposit(amount)**: Adds money to the account. It first checks that the deposit amount is positive. If valid, it updates the balance and prints the new balance; otherwise, it informs the user that the amount must be positive.
  - **withdraw(amount)**: Removes money from the account. It checks that the withdrawal amount is positive and that the account has sufficient funds. If the withdrawal is valid, it updates the balance and prints the new balance; if not, it either explains that the amount must be positive or that funds are insufficient.
  - **display_balance()**: Prints the account holder's name along with the current balance.
- **get_float_input(prompt_text) Function**
  - This function continuously prompts the user for input until a valid floating-point number is entered. It catches a ValueError if the input cannot be converted to a float, ensuring that the rest of the program receives proper numeric input.
- **main() Function**
  - **User Input for Account Setup**: The main function starts by asking the user for the account holder's name and an initial balance (using `get_float_input` to validate the balance input).
  - **Interactive Menu Loop**:
    - It then enters a loop that displays a menu with four options: deposit, withdraw, check balance, and exit.
    - Based on the user's choice, it calls the relevant method on the BankAccount object or exits the loop.
    - The loop continues until the user selects the exit option.
- **Conditional Execution**
  - The block `if __name__ == "__main__":` ensures that the `main()` function is executed only when the script is run directly, not when imported as a module.

Overall, the code provides a structured example of class usage, methods for encapsulating functionality, and error handling through a user-friendly interface.

TASK-4:

Prompt:

```
students = [
   {"name": "Tony Stark", "score": 97},
   {"name": "Steve Rogers", "score": 88},
   {"name": "Thor", "score": 82},
   {"name": "Bruse Banner", "score": 95},
   {"name": "Clint", "score": 74}
]
```

Here, is a list of student dictionaries. Now, give a python code using while loop to print the names of students who scored more than 75.

Code and Output:

lab 6.py

```python
students = [
    {"name": "Tony Stark", "score": 97},
    {"name": "Steve Rogers", "score": 88},
    {"name": "Thor", "score": 82},
    {"name": "Bruse Banner", "score": 95},
    {"name": "Clint", "score": 74}
]

i = 0
while i < len(students):
    if students[i]["score"] > 75:
        print(students[i]["name"])
    i += 1
```

```
PS C:\Users\mahit\OneDrive\Desktop\AIAC> & C:/Users/mahit/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/mahit/OneDrive/Desktop/AIAC/lab 6.py"
Tony Stark
Steve Rogers
Thor
Bruse Banner
```

Code Explanation:

- **Code Purpose**
  - Prints the names of students who scored **more than 75**.
- **Data Structure**
  - students is a **list of dictionaries**.Each dictionary contains:
    - "name" – student's name and "score" – student's numeric score.
- **Loop Logic**
  - i = 0: initializes the counter.
  - while i < len(students): loops through all students by index.
    - Checks if the student's "score" is **greater than 75**. If true, prints the student's "name".
  - i += 1: moves to the next student in the list.
- **Condition**
  - Only students with score > 75 are printed.

TASK-5:

Prompt:

Give a python code to ask input to user also generate methods to add_item, remove_item, and use a loop to calculate the total bill using conditional discounts. Begin writing a class ShoppingCart with an empty items list.

Code and Output:

```
lab 6.py
lab 6.py > main
class ShoppingCart:
    def __init__(self):
        self.items = []

    def add_item(self, name, price, quantity=1):
        for item in self.items:
            if item['name'].lower() == name.lower():
                item['quantity'] += quantity
                return
        self.items.append({'name': name, 'price': price, 'quantity': quantity})

    def remove_item(self, name):
        self.items = [item for item in self.items if item['name'].lower() != name.lower()]

    def calculate_total(self):
        total = 0
        print("\n Cart Summary:")
        for item in self.items:
            name = item['name']
            price = item['price']
            quantity = item['quantity']

            # Bulk discount
            if quantity > 5:
                print(f"Bulk discount applied to {name}")
                price -= 1  # ₹1 off per item

            item_total = price * quantity
            total += item_total
            print(f"{name} - {quantity} x ₹{price:.2f} = ₹{item_total:.2f}")

        # General discount
        if total > 1000:
            print("General discount applied: 10% off on total above ₹1000")
            total *= 0.9

        print(f" Total bill: ₹{total:.2f}")
        return total

def main():
    cart = ShoppingCart()

    while True:
        print("\nChoose an action:")
        print("1. Add item")
```

```
        print("2. Remove item")
        print("3. View total bill")
        print("4. Exit")

        choice = input("Enter your choice (1-4): ").strip()

        if choice == '1':
            name = input("Enter item name: ").strip()
            try:
                price = float(input("Enter price in ₹: ").strip())
                quantity = int(input("Enter quantity: ").strip())
                cart.add_item(name, price, quantity)
                print(f"{quantity} x {name} added to cart.")
            except ValueError:
                print("Invalid input. Please enter valid numbers for price and quantity.")

        elif choice == '2':
            name = input("Enter item name to remove: ").strip()
            cart.remove_item(name)
            print(f"{name} removed from cart (if it existed).")

        elif choice == '3':
            cart.calculate_total()

        elif choice == '4':
            print("Exiting. Thank you for shopping!")
            break

        else:
            print("Invalid choice. Please select a valid option.")

if __name__ == "__main__":
    main()
```

```
PS C:\Users\mahit\OneDrive\Desktop\AIAC> & C:/Users/mahit/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/mahit/OneDrive/Desktop/AIAC/lab 6.py"

Choose an action:
1. Add item
2. Remove item
3. View total bill
4. Exit
Enter your choice (1-4): 1
Enter item name: sugar pack
Enter price in ₹: 99
Enter quantity: 200
200 x sugar pack added to cart.

Choose an action:
1. Add item
2. Remove item
3. View total bill
4. Exit
Enter your choice (1-4): 1
Enter item name: Lays
Enter price in ₹: 10
Enter quantity: 4
4 x Lays added to cart.

Choose an action:
1. Add item
2. Remove item
3. View total bill
4. Exit
Enter your choice (1-4): 3

🧾 Cart Summary:
Bulk discount applied to sugar pack
sugar pack - 200 x ₹98.00 = ₹19600.00
Lays - 4 x ₹10.00 = ₹40.00
General discount applied: 10% off on total above ₹1000
🧮 Total bill: ₹17676.00
```

Code Explanation:

ShoppingCart Class

- **Constructor**

  (__int__)
  Initializes a new shopping cart with an empty list of items (self.items = []).

- **add_item(name, price, quantity=1)**

- **Method**
  Adds an item to the cart.

  - If the item already exists (case-insensitive comparison), it increases its quantity instead of adding a duplicate.

- Otherwise, it appends a new dictionary with name, price, and quantity to the list.

- **remove_item(name)**
- **Method**
  Removes all entries matching the given item name (case-insensitive) from the cart.
  - It uses a list comprehension to filter out items whose name matches the provided name.
- **calculate_total()**
- **Method**
  Calculates and displays the total bill with discounts:
  - Iterates through each item and prints details (name, quantity, price).
  - **Bulk Discount:** If quantity > 5, applies a ₹1 discount per item and displays a message.
  - Calculates item_total for each item and adds it to total.
  - **General Discount:** If the total exceeds ₹1000, applies an additional 10% discount and displays a message.
  - Prints the final bill total and returns the amount.

main() Function

- **Cart**
- **Initialization:**
  Creates a ShoppingCart object to manage items.
- **Interactive Menu**
- **Loop:**
  Continuously displays options until the user exits:
  1. **Add Item:**
     - Prompts for name, price, and quantity.
     - Adds the item using cart.add_item().
     - Handles invalid numeric input using try-except.
  2. **Remove Item:**
     - Prompts for an item name and removes it using cart.remove_item().

3. **View Total Bill:**
    - Calls cart.calculate_total() to display a detailed summary with discounts.
4. **Exit:**
    - Breaks out of the loop and ends the program.

- Handles invalid menu choices gracefully.