AI ASSISTANT CODING - ASSIGNMENT-5.3

Name: Valaboju Harshavardhanu

Roll No: 2303A52354 **Batch:** 41

Task 1: Student Performance Evaluation System

Prompt:

#create a python class named student with the attributes name, roll number and marks and display the details of the student and a method to check weather the student marks above the calculated class average or not

Code:

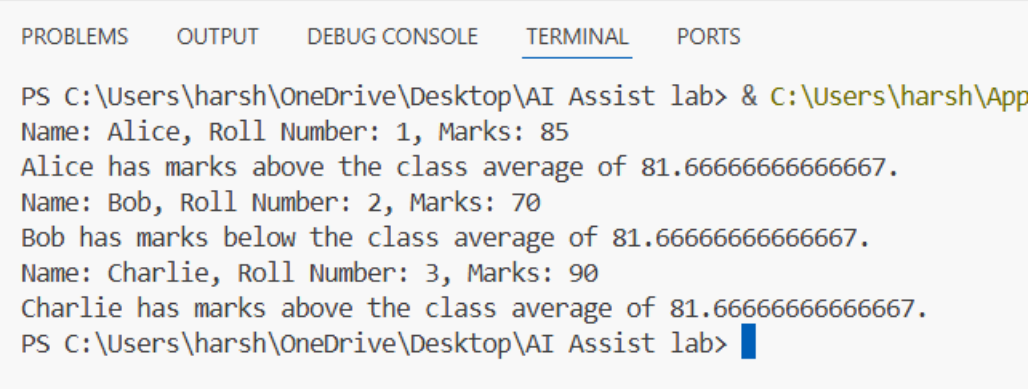
```
#create a python class named student with the attributes name, roll number and marks and disp

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}, Roll Number: {self.roll_number}, Marks: {self.marks}")

    def is_above_average(self, class_average):
        return self.marks > class_average
# Example usage
students = [
    Student("Alice", 1, 85),
    Student("Bob", 2, 70),
    Student("Charlie", 3, 90)
]
total_marks = sum(student.marks for student in students)
class_average = total_marks / len(students)
for student in students:
    student.display_details()
    if student.is_above_average(class_average):
        print(f"{student.name} has marks above the class average of {class_average}.")
    else:
        print(f"{student.name} has marks below the class average of {class_average}.")
```

Output:



Justification:

In the above task we have created a class named student which includes the attributes like name, roll no, marks using the prompt and the output will display the student details along with that it check weather the student's marks are above the class average and the AI will generated the complete code by the given prompt which includes the Expected outcomes

Task 2: Data Processing in a Monitoring System

Prompt:

#write a python code to identify the even numbers and calculate their square and print the result user define input

Code:

```python
#write a python code to identify the even numbers and calculate their square an
def even_numbers_square():
    try:
        user_input = input("Enter a list of numbers separated by spaces: ")
        numbers = list(map(int, user_input.split()))
        even_squares = {num: num**2 for num in numbers if num % 2 == 0}
        print("Even numbers and their squares:")
        for num, square in even_squares.items():
            print(f"{num}: {square}")
    except ValueError:
        print("Invalid input. Please enter numeric values only.")
# Example usage
even_numbers_square()
```

Output:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\harsh\OneDrive\Desktop\AI Assist lab> & C:\Users\harsh\AppData\Local\P
Enter a list of numbers separated by spaces: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
Even numbers and their squares:
2: 4
4: 16
6: 36
8: 64
10: 100
12: 144
14: 196
PS C:\Users\harsh\OneDrive\Desktop\AI Assist lab>
```

Justification:

In the above task we have initialized a for loop for iteration, after this by using the prompts the code performs the task that identifying the even numbers and calculating their squares and printing. Which includes the modulus operator, conditional statements as the Expected outcomes

Task 3: Banking Transaction Simulation

Prompt:

#generate a python class named BankAccount with attributes account_number, account_holder_name and balance. Include methods to deposit, withdraw and display account details and check for sufficient balance before withdrawal input is user defined

Code:

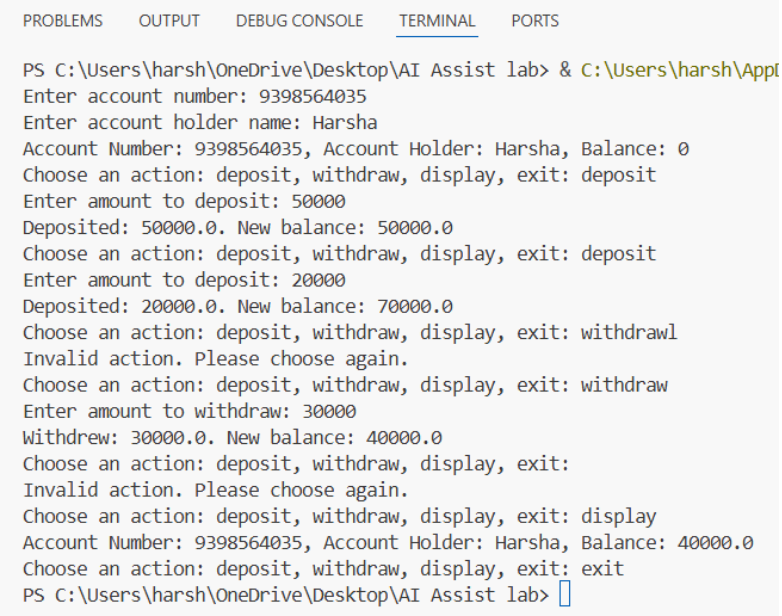
```
#generate a python class named BankAccount with attributes account_number, account_holder_name and balance. Include methods to
class BankAccount:
    def __init__(self, account_number, account_holder_name, balance=0):
        self.account_number = account_number
        self.account_holder_name = account_holder_name
        self.balance = balance

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

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

    def display_account_details(self):
        print(f"Account Number: {self.account_number}, Account Holder: {self.account_holder_name}, Balance: {self.balance}")
# Example usage
account_number = input("Enter account number: ")
account_holder_name = input("Enter account holder name: ")
account = BankAccount(account_number, account_holder_name)
account.display_account_details()
while True:
    action = input("Choose an action: deposit, withdraw, display, 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 == "display":
        account.display_account_details()
    elif action == "exit":
        break
    else:
        print("Invalid action. Please choose again.")
```

Output:



Justification:

In the above task we have developed a Basic banking system which includes the a Class BankAccount with the Attributes account_holder, balance and the generated code performs the tasks like Depositing the money withdrawl of money and the prevents insufficient money withdrawls which gives the best and simple system. AI will been generated the complete code by the given prompt which includes the Expected outcomes

Task 4: Student Scholarship Eligibility Check

Prompt:

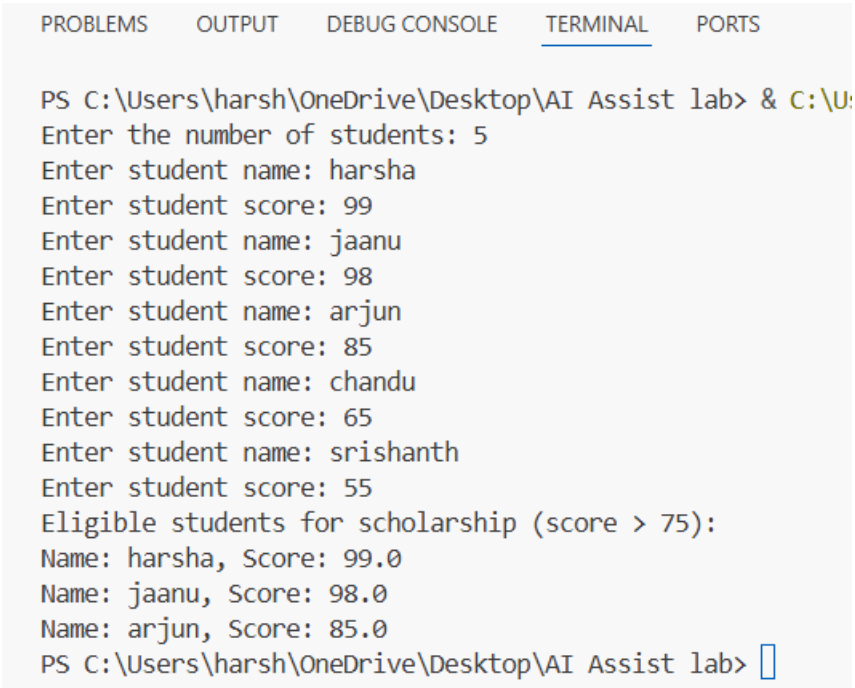
#generate a python code for student scholarship eligibility check using a list of dictionary with name and score and print the eligible students whose score is more than 75 using while loop with the user defined input

Code:

```python
#generate a python code for student scholarship eligibility check using a list of dictior
def scholarship_eligibility():
    students = []
    try:
        n = int(input("Enter the number of students: "))
        for _ in range(n):
            name = input("Enter student name: ")
            score = float(input("Enter student score: "))
            students.append({"name": name, "score": score})

        print("Eligible students for scholarship (score > 75):")
        i = 0
        while i < len(students):
            if students[i]["score"] > 75:
                print(f"Name: {students[i]['name']}, Score: {students[i]['score']}")
            i += 1
    except ValueError:
        print("Invalid input. Please enter numeric values for scores.")
# Example usage
scholarship_eligibility()
```

Output:



Justification:

In the above code we are able to check the Student scholarship Eligibility using a while loop at the beginning and the prompts which includes at the middle and we define a list of dictionaries. We can clearly see that through the iterations and index handling and

condition checks we have got the clear output this shows the fairness of the code generated by A.I. which is most important.

Task 5: Online Shopping Cart Module

Prompt:

#create a class named ShoppingCart with an list to store the items each item include name, price and quantity include methods to add item, remove item, view cart and calculate total price and apply discount based on total price user defined input

Code:

```
#create a class named ShoppingCart with an list to store the items each item include r
class ShoppingCart:
    def __init__(self):
        self.items = []

    def add_item(self, name, price, quantity):
        self.items.append({"name": name, "price": price, "quantity": quantity})
        print(f"Added {quantity} of {name} at ${price} each to the cart.")

    def remove_item(self, name):
        for item in self.items:
            if item["name"] == name:
                self.items.remove(item)
                print(f"Removed {name} from the cart.")
                return
        print(f"Item {name} not found in the cart.")

    def view_cart(self):
        if not self.items:
            print("Your cart is empty.")
            return
        print("Items in your cart:")
        for item in self.items:
            print(f"{item['quantity']} x {item['name']} at ${item['price']} each")

    def calculate_total(self):
        total = sum(item["price"] * item["quantity"] for item in self.items)
        discount = 0
        if total > 100:
            discount = total * 0.1  # 10% discount for orders over $100
        total_after_discount = total - discount
        print(f"Total price: ${total:.2f}")
        if discount > 0:
            print(f"Discount applied: ${discount:.2f}")
            print(f"Total after discount: ${total_after_discount:.2f}")
        return total_after_discount
```

```python
# Example usage
cart = ShoppingCart()
while True:
    action = input("Choose an action: add, remove, view, total, exit: ").lower()
    if action == "add":
        name = input("Enter item name: ")
        price = float(input("Enter item price: "))
        quantity = int(input("Enter item quantity: "))
        cart.add_item(name, price, quantity)
    elif action == "remove":
        name = input("Enter item name to remove: ")
        cart.remove_item(name)
    elif action == "view":
        cart.view_cart()
    elif action == "total":
        cart.calculate_total()
    elif action == "exit":
        break
    else:
        print("Invalid action. Please choose again.")
```

Output:

```
PS C:\Users\harsh\OneDrive\Desktop\AI Assist lab> & C:\Users
Choose an action: add, remove, view, total, exit: add
Enter item name: laptop
Enter item price: 50000
Enter item quantity: 1
Added 1 of laptop at $50000.0 each to the cart.
Choose an action: add, remove, view, total, exit: add
Enter item name: bag
Enter item price: 600
Enter item quantity: 2
Added 2 of bag at $600.0 each to the cart.
Choose an action: add, remove, view, total, exit: bat
Invalid action. Please choose again.
Choose an action: add, remove, view, total, exit: add
Enter item name: bat
Enter item price: 1200
Enter item quantity: 1
Added 1 of bat at $1200.0 each to the cart.
Choose an action: add, remove, view, total, exit: remove
Enter item name to remove: bat
Removed bat from the cart.
Choose an action: add, remove, view, total, exit: view
Items in your cart:
1 x laptop at $50000.0 each
2 x bag at $600.0 each
Choose an action: add, remove, view, total, exit: total
Total price: $51200.00
Discount applied: $5120.00
Total after discount: $46080.00
Choose an action: add, remove, view, total, exit: exit
PS C:\Users\harsh\OneDrive\Desktop\AI Assist lab>
```

Justification:

In the above code we are able to create a shopping cart system where we have initialized the with the Class and the Attributes and using the prompt we have performed the tasks like adding items and remove and calculating the total bill and applying the conditional discounts and the output generated by the code from the prompt gives the demonstrating cart functionality.