AI_ASSISTED_CODING

2403A52094

B.Sai ganesh

B-04

➢ **Task 1:**

Start a Python class named Student with attributes name, roll_number, and marks. Prompt GitHub Copilot to complete methods for displaying details and checking if marks are above Average.

Prompt:

Write a Python class named Student with the following:
- Attributes: name, roll_number, and marks
- A method display_details() to print student details
- A method is_above_average(average) that returns True if the student's marks are greater than the given average, otherwise False

Also, show an example usage by creating at least two Student objects and demonstrating both methods.

Code:

```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_above_average(self, average=50):
        return self.marks > average

# Example usage
if __name__ == "__main__":
    student = Student("Alice", "101", 78)
    student.display_details()
    print("Above average?", student.is_above_average())
```

Output:

```
Name: Alice
Roll Number: 101
Marks: 78
Above average? True
```

Observation:

This code defines a Student class with attributes (name, roll number, marks), a method to display details, and another method to check if marks are above average. Then it demonstrates usage with an example student "Alice".

Task 2: Write the first two lines of a for loop to iterate through a list of numbers. Use a comment prompt to let Copilot suggest how to calculate and print the square of even numbers only.

Prompt: Write a Python program that iterates through a list of numbers from 1 to 6. For each even number, print its square in the format: 'Square of X: Y'

Code:

```python
# Iterate through a list of numbers
numbers = [1, 2, 3, 4, 5, 6]
for num in numbers:
    if num % 2 == 0:
        print(f"Square of {num}: {num ** 2}")
```

Output:

```
Square of 2: 4
Square of 4: 16
Square of 6: 36
```

Observation:

o This program iterates through numbers 1–6, selects only even numbers, and prints their squares.

Name: B.sai ganesh
Roll No: 2403A52094
Batch: 24BTCAIAIB04

Task 3: Create a class called BankAccount with attributes account_holder and balance. Use Copilot to complete methods for deposit(), withdraw(), and check for insufficient balance.

Prompt:

Create a Python class named BankAccount with attributes account_holder and balance (default 0). Add methods deposit(amount) to add money if the amount is positive, and withdraw(amount) to subtract money only if the amount is positive and does not exceed the balance. Print appropriate messages for deposits, withdrawals, invalid amounts, and insufficient balance. Also provide example usage by creating accounts, depositing money, and withdrawing money.

Code:

```python
# BankAccount class with Copilot prompts for methods
class BankAccount:
    def __init__(self, account_holder, balance=0):
        self.account_holder = account_holder
        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 <= 0:
            print("Withdrawal amount must be positive.")
        elif amount > self.balance:
            print("Insufficient balance.")
        else:
            self.balance -= amount
            print(f"Withdrew {amount}. New balance: {self.balance}")
# Example usage of BankAccount class

# Create an account for John with an initial balance of 500
account1 = BankAccount("John Doe", 500)

# Deposit money
account1.deposit(200)    # Should add 200, balance becomes 700
account1.deposit(-50)    # Invalid deposit

# Withdraw money
account1.withdraw(100)   # Should subtract 100, balance becomes 600
account1.withdraw(1000)  # Insufficient balance
account1.withdraw(-20)   # Invalid withdrawal

# Another account
account2 = BankAccount("Alice", 1000)
account2.deposit(500)
account2.withdraw(300)
```

Output:

```
Deposited 200. New balance: 700
Deposit amount must be positive.
Withdrew 100. New balance: 600
Insufficient balance.
Withdrawal amount must be positive.
Deposited 500. New balance: 1500
Withdrew 300. New balance: 1200
```

Observation:

This program defines a BankAccount class with methods to deposit and withdraw money. It includes safety checks (like invalid deposits, insufficient balance, and invalid withdrawals). Two accounts (John Doe and Alice) are created, and operations are performed on them.

Name: B.sai ganesh
Roll No: 2403A52094
Batch: 24BTCAIAIB04

Task 4: Define a list of student dictionaries with keys name and score. write a while loop to print the names of students who scored more than 75**.**

Prompt:

Define a list of student dictionaries where each dictionary contains name and score. Write a Python program using a while loop to print the names of students whose score is greater than 75.

Code:

```python
"""
Define a list of student dictionaries with keys name and score. write a while
loop to print the names of students who scored more than 75.
"""
# List of student dictionaries
students = [
    {"name": "Alice", "score": 80},
    {"name": "Bob", "score": 70},
    {"name": "Charlie", "score": 90},
    {"name": "David", "score": 60}
]

# While loop to print names of students who scored more than 75
i = 0
while i < len(students):
    if students[i]["score"] > 75:
        print(students[i]["name"])
    i += 1
```

Output:

```
Alice
Charlie
```

Observation:

This program stores student details in a list of dictionaries and uses a while loop to check each student's score. If the score is greater than 75, it prints the student's name.

Name: B.sai ganesh
Roll No: 2403A52094
Batch: 24BTCAIAIB04

Task 5: Begin writing a class ShoppingCart with an empty items list. Prompt Copilot to generate methods to add_item, remove_item, and use a loop to calculate the total bill using conditional discounts.

Prompt:

Create a Python class named ShoppingCart that allows adding items, removing items, and calculating the total bill. Each item should be stored as a dictionary with keys name and price. The add_item method should append an item to the cart, remove_item should remove it if present, and calculate_total should compute the sum of prices. If the total exceeds 100, apply a 10% discount and display both the discount and final bill. Also, provide example usage by creating a cart, adding items, removing an item, and calculating the total.

Code:

```python
# ShoppingCart class
class ShoppingCart:
    def __init__(self):
        self.items = []

    def add_item(self, item):
        self.items.append(item)
        print(f"Added {item} to cart.")

    def remove_item(self, item):
        if item in self.items:
            self.items.remove(item)
            print(f"Removed {item} from cart.")
        else:
            print(f"Item {item} not found in cart.")

    def calculate_total(self):
        total = 0
        # Assume each item is a dict with 'name' and 'price' keys
        for item in self.items:
            total += item.get('price', 0)
        # Apply a 10% discount if total exceeds 100
        if total > 100:
            discount = total * 0.10
            total -= discount
            print(f"Discount applied: {discount}")
        print(f"Total bill: {total}")
        return total
#Example Usage:
# Create a shopping cart
cart = ShoppingCart()

# Add items
cart.add_item({'name': 'Laptop', 'price': 750})
cart.add_item({'name': 'Headphones', 'price': 50})
cart.add_item({'name': 'Mouse', 'price': 20})

# Remove an item
cart.remove_item({'name': 'Mouse', 'price': 20})

# Calculate total
cart.calculate_total()
```

Output:

```
Added {'name': 'Laptop', 'price': 750} to cart.
Added {'name': 'Headphones', 'price': 50} to cart.
Added {'name': 'Mouse', 'price': 20} to cart.
Removed {'name': 'Mouse', 'price': 20} from cart.
Discount applied: 80.0
Total bill: 720.0
```

Observation:

The code defines a ShoppingCart class with methods to add, remove, and calculate the total price of items. It applies a 10% discount if the total exceeds $100. The example usage demonstrates creating a cart, adding items, removing one, and then calculating the final bill. The final output will be a bill of $720, which is the sum of the remaining items ($800) minus the $80 discount.

Name: B.sai ganesh
Roll No: 2403A52094
Batch: 24BTCAIAIB04