AI ASSISTED CODING

LAB-6.4

NAME:V.KARTHIK REDDDY

ENROLL.NO:2403A52092

BATCH NO:04

TASK-01:

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

Prompt:

Generate a python class named Student with attributes name ,enrollnumber, marks and complete the methods to display details and check the marks are above average or not.

Code:

```
lab6.4.1.py > Student > display_details
 1    class Student:
 2      def __init__(self, name, enrollno, marks):
 3        self.name = name
 4        self.enrollno = enrollno
 5        self.marks = marks
 6
 7      def display_details(self):
 8        print(f"Name: {self.name}")
 9        print(f"Enrollment No: {self.enrollno}")
10        print(f"Marks: {self.marks}")
11
12      def is_above_average(self, average=50):
13        if self.marks > average:
14          print(f"{self.name} has marks above average.")
15        else:
16          print(f"{self.name} does not have marks above average.")
17
18    if __name__ == "__main__":
19      name = input("Enter student name: ")
20      enrollno = input("Enter enrollment number: ")
21      marks = float(input("Enter marks: "))
22      student = Student(name, enrollno, marks)
23      student.display_details()
24      student.is_above_average()
```

Output:

```
PS C:\Users\ramch\OneDrive\Desktop\ai> & C:/Users/ramch/AppData/Local/Programs/Python/Python313/python.exe c:/Users/ra
ch/OneDrive/Desktop/ai/lab6.4.1.py
Enter student name: sandeep
Enter enrollment number: 2724
Enter marks: 87
Name: sandeep
Enrollment No: 2724
Marks: 87.0
sandeep has marks above average.
PS C:\Users\ramch\OneDrive\Desktop\ai>
```

Observation:

The program defines a Student class with attributes name, enrollment number, and marks.
It displays student details and checks if marks are above the average (default 50) using object-oriented concepts.

Task-02:

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:

Generate a python code which finds the even numbers and calculates the squares of it.

Code:

```
lab6.4.2.py > ...
1    numbers = [2, 45, 67, 87, 32,44,66,78]
2    squares_of_even = [x**2 for x in numbers if x % 2 == 0]
3    print(squares_of_even)
```

Output:

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

PS C:\Users\ramch\OneDrive\Desktop\ai> & C:/Users/ramch/AppData/Local/Programs/Python/Python313/python.exe c:/Users/ram
ch/OneDrive/Desktop/ai/lab6.4.2.py
[4, 1024, 1936, 4356, 6084]
PS C:\Users\ramch\OneDrive\Desktop\ai>
```

Observation:

The code processes and finds the even numbers from the list and squares the numbers and store them in the list. The Github copilot gave the appropriate logic for the given prompt.

Task-03:

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:

Generate a python code with the class Bank Account of attributes account holder and balance also generate methods for deposit , withdraw and check for insufficient balance.

Code:

```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:
      self.balance += amount
      return True
    return False

  def withdraw(self, amount):
    if amount > 0 and self.balance >= amount:
      self.balance -= amount
      return True
    return False

  def has_insufficient_balance(self, amount):
    return self.balance < amount

if __name__ == "__main__":
  name = input("Enter account holder name: ")
  balance = float(input("Enter initial balance: "))
  account = BankAccount(name, balance)

  while True:
    print("\n1. Deposit\n2. Withdraw\n3. Check Balance\n4. Exit")
    choice = input("Choose an option: ")
    if choice == "1":
      amount = float(input("Enter amount to deposit: "))
      if account.deposit(amount):
        print(f"Deposited {amount}. New balance: {account.balance}")
      else:
        print("Invalid deposit amount.")
    elif choice == "2":
      amount = float(input("Enter amount to withdraw: "))
      if account.withdraw(amount):
        print(f"Withdrew {amount}. New balance: {account.balance}")
      else:
        print("Insufficient balance or invalid amount.")
    elif choice == "3":
      print(f"Current balance: {account.balance}")
    elif choice == "4":
      print("Exiting.")
      break
    else:
      print("Invalid choice. Try again.")
```

Output:

```
ch/OneDrive/Desktop/ai/lab6.4.3.py
Enter account holder name: Shreshta
Enter initial balance: 10000

1. Deposit
2. Withdraw
3. Check Balance
4. Exit
Choose an option: 1
Enter amount to deposit: 2000
Deposited 2000.0. New balance: 12000.0

1. Deposit
2. Withdraw
3. Check Balance
4. Exit
Choose an option: 2
Enter amount to withdraw: 500
Withdrew 500.0. New balance: 11500.0

1. Deposit
2. Withdraw
3. Check Balance
4. Exit
Choose an option: 3
Current balance: 11500.0

1. Deposit
2. Withdraw
```

Observation:

The code creates a class with name BankAccount and the methods to deposit, withdraw and check the insufficient balance with the menu driven application which takes the input from the user. The github copilot gave the appropriate code according to the scenario.

Task-04:

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

Prompt:

Use a while loop to iterate over the list of the dictionaries and print the students names whose marks are above 75.

Code:

```python
lab6.4.4.py > ...
1   students = [
2       {"shreya": 82},
3       {"martin": 68},
4       {"shreshta": 90},
5       {"akhi": 74},
6       {"skruthi": 88}
7   ]
8
9   i = 0
10  while i < len(students):
11      for name, marks in students[i].items():
12          if marks > 75:
13              print(name)
14      i += 1
```

Output:

```
Choose an option: 4
Exiting.
PS C:\Users\ramch\OneDrive\Desktop\ai> & C:/Users/ramch/AppData/Local/Programs/Python/Python313/python.exe c:/Users/ram
ch/OneDrive/Desktop/ai/lab6.4.4.py
shreya
shreshta
skruthi
PS C:\Users\ramch\OneDrive\Desktop\ai> []
```

Observation:

The code iterates over the list of dictionaries and prints the names of the students who got marks above 75. The Github copilot processed well.

Task-05:

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

Prompt:

Generate a python class named shopping cart with empty lists then also generate the methods to additem,removeitem,use loops to calculate bill applying the discount.

Code:

```
lab6.4.5.py > ⛬ ShoppingCart > ⊘ calculate_total
1    class ShoppingCart:
2        def __init__(self):
3            self.items = []  # Each item is a dict: {'name': str, 'price': float, 'quantity': int}
4
5        def add_item(self, name, price, quantity=1):
6            self.items.append({'name': name, 'price': price, 'quantity': quantity})
7
8        def remove_item(self, name):
9            self.items = [item for item in self.items if item['name'] != name]
10
11       def calculate_total(self):
12           total = 0
13           for item in self.items:
14               total += item['price'] * item['quantity']
15           return total
16
17       def apply_discount(self, discount_percent):
18           total = self.calculate_total()
19           discount = total * (discount_percent / 100)
20           return total - discount
21
22   # Example usage:
23   if __name__ == "__main__":
24       cart = ShoppingCart()
25       while True:
26           action = input("Enter action (add, remove, total, discount, quit): ").strip().lower()
27           if action == "add":
28               name = input("Item name: ")
29               price = float(input("Item price: "))
30               quantity = int(input("Quantity: "))
31               cart.add_item(name, price, quantity)
32               print(f"Added {quantity} x {name} at ${price:.2f} each.")
33           elif action == "remove":
34               name = input("Item name to remove: ")
35               cart.remove_item(name)
36               print(f"Removed {name} from cart.")
37           elif action == "total":
38               print(f"Cart total: ${cart.calculate_total():.2f}")
39           elif action == "discount":
40               percent = float(input("Discount percent: "))
41               discounted_total = cart.apply_discount(percent)
42               print(f"Total after {percent}% discount: ${discounted_total:.2f}")
43           elif action == "quit":
44               print("Exiting.")
45               break
46           else:
47               print("Unknown action. Try again.")
⚠ 0
```

Output:

```
PS C:\Users\ramch\OneDrive\Desktop\ai> & C:/Users/ramch/AppData/Local/Programs/Python/Python313/python.exe c:/Users/ram
ch/OneDrive/Desktop/ai/lab6.4.5.py
Enter action (add, remove, total, discount, quit): add
Item name: chair
Item price: 400
Quantity: 2
Added 2 x chair at $400.00 each.
Enter action (add, remove, total, discount, quit): discount
Discount percent: 5
Total after 5.0% discount: $760.00
Enter action (add, remove, total, discount, quit): quit
Exiting.
PS C:\Users\ramch\OneDrive\Desktop\ai>
```

Observation:

The code creates a class named shopping cart with the empty lists and the methods to add , remove and calculating the bills applying discounts. The github copilot understood the scenario in a better way and processed the output well.