

Lab – 3 Assignment 6.4

Roll Number: 2403A52088

Batch: 04

Name: Mohammed Anas Siddiqui

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: Complete methods for displaying and checking if marks are above average

Code:

```
1 class Student:
2     def __init__(self, name, roll_number, marks):
3         self.name = name
4         self.roll_number = roll_number
5         self.marks = marks
6
7     def display_details(self):
8         print(f"Name: {self.name}")
9         print(f"Roll Number: {self.roll_number}")
10        print(f"Marks: {self.marks}")
11        if self.is_passed():
12            print("Status: Passed")
13        else:
14            print("Status: Failed")
15
16    def is_above_average(self, average=50):
17        return self.marks > average
18
19    def is_passed(self, passing_marks=40):
20        if self.marks >= passing_marks:
21            return True
22        else:
23            return False
24
25    #Example usage
26    student1 = Student("John Doe", "12345", 75)
27
28    #display student details
29    student1.display_details()
30
31    # check if marks are above average
32    if student1.is_above_average():
33        print(f"{student1.name} has marks above average.")
34    else:
35        print(f"{student1.name} does not have marks above average.")
36
37    # check if student has passed
38    if student1.is_passed():
39        print(f"{student1.name} has passed.")
40    else:
41        print(f"{student1.name} has failed.")
42
```

Output:

```
PS D:\Anas\2nd Year\AIAC\Lab 6.4> & "C:/Users/Anas/Desktop/Task1.py"
Name: John Doe
Roll Number: 12345
Marks: 75
Status: Passed
John Doe has marks above average.
John Doe has passed.
PS D:\Anas\2nd Year\AIAC\Lab 6.4>
```

Observation:

Github copilot understood the prompt and gave the appropriate code.

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: calculate and print the square of even numbers only

Code:

```
1 numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
2
3 for number in numbers:
4     # calculate and print the square of even numbers only
5     if number % 2 == 0:
6         print(number ** 2)
```

Output:

```
PS D:\Anas\2nd Year\AIAC\Lab 6.4> & "C:/Users/Anas/Desktop/Task2.py"
4
16
36
64
100
PS D:\Anas\2nd Year\AIAC\Lab 6.4>
```

Observation: I have written the first two lines of the code and gave the comment in the third line, the github copilot automatically gave me the rest of the code.

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 class called `BankAccount` with attributes `account_holder` and `balance`. Give code for complete methods for `deposit()`, `withdraw()`, and check for insufficient balance.

Code:

```
1 class BankAccount:
2     def __init__(self, account_holder, balance=0):
3         self.account_holder = account_holder
4         self.balance = balance
5
6     def deposit(self, amount):
7         if amount > 0:
8             self.balance += amount
9             print(f"Deposited: {amount}. New balance: {self.balance}")
10        else:
11            print("Deposit amount must be positive.")
12
13    def withdraw(self, amount):
14        if amount > self.balance:
15            print("Insufficient balance for this withdrawal.")
16        elif amount <= 0:
17            print("Withdrawal amount must be positive.")
18        else:
19            self.balance -= amount
20            print(f"Withdrew: {amount}. New balance: {self.balance}")
21
22    def check_balance(self):
23        print(f"Current balance: {self.balance}")
24
25    # Example usage
26    account = BankAccount("John Doe", 1000)
27    account.check_balance()
28    account.deposit(500)
29    account.withdraw(200)
30    account.withdraw(1500) # Attempt to withdraw more than balance
```

Output:

```
PS D:\Anas\2nd Year\AIAC\Lab 6.4> & "C:/User
/Task3.py"
Current balance: 1000
Deposited: 500. New balance: 1500
Withdrew: 200. New balance: 1300
Insufficient balance for this withdrawal.
PS D:\Anas\2nd Year\AIAC\Lab 6.4>
```

Observation: The github copilot successfully generated the code and the appropriate output.

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

Code:

```
Task4.py > ...
1  ''' Task 4:
2  Define a list of student dictionaries with keys name and score. Ask Copilot to write a while
3  loop to print the names of students who scored more than 75'''
4
5  students = [
6      {"name": "Alice", "score": 85},
7      {"name": "Bob", "score": 70},
8      {"name": "Charlie", "score": 90},
9      {"name": "David", "score": 60},
10     {"name": "Eve", "score": 80}
11 ]
12 index = 0
13 while index < len(students):
14     student = students[index]
15     if student["score"] > 75:
16         print(student["name"])
17     index += 1
18
```

Output:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
PS D:\Anas\2nd Year\AIAC\Lab 6.4> & "C:/U
/Task4.py"
Alice
Charlie
Eve
PS D:\Anas\2nd Year\AIAC\Lab 6.4>
```

Observation:

Github copilot successfully generated the above code and the output is correct.

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: write a code to add, remove items from the list in class shoppingcart, and use loops to calculate total bill using conditional discounts.

Code:

```
1 class ShoppingCart:
2     def __init__(self):
3         self.items = []
4
5     def add_item(self, item, price):
6         self.items.append({"item": item, "price": price})
7         print(f"Added {item} to the cart.")
8
9     def remove_item(self, item):
10        for i in range(len(self.items)):
11            if self.items[i]["item"] == item:
12                del self.items[i]
13                print(f"Removed {item} from the cart.")
14                return
15        print(f"{item} not found in the cart.")
16
17    def calculate_total(self):
18        total = sum(item["price"] for item in self.items)
19        if total > 100:
20            total *= 0.9 # Apply a 10% discount for totals over 100
21        print(f"Total bill: {total}")
22    def display_items(self):
23        if not self.items:
24            print("Your cart is empty.")
25        else:
26            print("Items in your cart:")
27            for item in self.items:
28                print(f"{item['item']}: ${item['price']}")
29
30    # Example usage
31    cart = ShoppingCart()
32    cart.add_item("Apple", 1.5)
33    cart.add_item("Banana", 0.75)
34    cart.display_items()
35    cart.calculate_total()
36    cart.remove_item("Apple")
37    cart.display_items()
38    cart.calculate_total()
39    cart.remove_item("Orange") # Attempt to remove an item not in the cart
```

Output:

```
PS D:\Anas\2nd Year\AIAC\Lab 6.4> & "C:/Users/A.../Task5.py"
Added Apple to the cart.
Added Banana to the cart.
Items in your cart:
Apple: $1.5
Banana: $0.75
Total bill: 2.25
Removed Apple from the cart.
Items in your cart:
Banana: $0.75
Total bill: 0.75
Orange not found in the cart.
PS D:\Anas\2nd Year\AIAC\Lab 6.4>
```

Observation: I gave the empty list in class shoppingcart and asked github copilot to give code for add, remove item and use loops to generate total bill.

