

Assignment 6.4

AI Assisted Coding

Name: Sunny Kolluri

HTNO: 2303A52444

Batch 43

Task 1:

Prompt:

Student Performance Evaluation System

class Student:

```
#def __init__(self, name, roll_number, marks):
```

```
    self.name = name
```

```
    self.roll_number = roll_number
```

```
    self.marks = marks
```

```
# Create a method to display student details it should print student name, roll  
number and marks clearly. Create a method named check_performance it  
should take class_average as parameter.
```

```
#use if-else to check if student marks are above class average return a message  
about performance
```

```
#Create a sample student object and test the methods
```

```
236 # self.name = name
237 # self.roll_number = roll_number
238 # self.marks = marks
239 # create a method to display student details it should print student name, roll number and marks clearly
240 # create a method named check_performance it should take class_average as parameter
241 # use if-else to check if student marks are above class average return a message about performance
242 # create a sample student object and test the methods
243
244 class Student:
245     def __init__(self, name, roll_number, marks):
246         self.name = name
247         self.roll_number = roll_number
248         self.marks = marks
249
250     def display_details(self):
251         print(f"Name: {self.name}, Roll Number: {self.roll_number}, Marks: {self.marks}")
252
253     def check_performance(self, class_average):
254         if self.marks > class_average:
255             return f"{self.name} performed above average."
256         else:
257             return f"{self.name} performed at or below average."
258
259 # Create a sample student object and test the methods
260 student1 = Student("Alice", "12345", 85)
261 student1.display_details()
262 print(student1.check_performance(75))
```

PS C:\Users\navya\OneDrive\Desktop\AI Assit> & "C:/Users/navya/OneDrive/Desktop/AI Assit/.venv/scripts/python.exe" "C:/Users/navya/OneDrive/Desktop/AI Assit/Assignment 2"

Name: Alice, Roll Number: 12345, Marks: 85
Alice performed above average.
PS C:\Users\navya\OneDrive\Desktop\AI Assit>

Observation – Student Performance Evaluation System

In this task, GitHub Copilot was used to complete the implementation of a Python class based on comment-based prompts. After providing the class skeleton and descriptive comments, Copilot successfully generated the required methods.

The AI assistant created:

- A `display_details()` method that prints student information using class attributes (`self.name`, `self.roll_number`, `self.marks`).
- A `check_performance()` method that uses an **if-else conditional statement** to compare student marks with the class average and return an appropriate message.

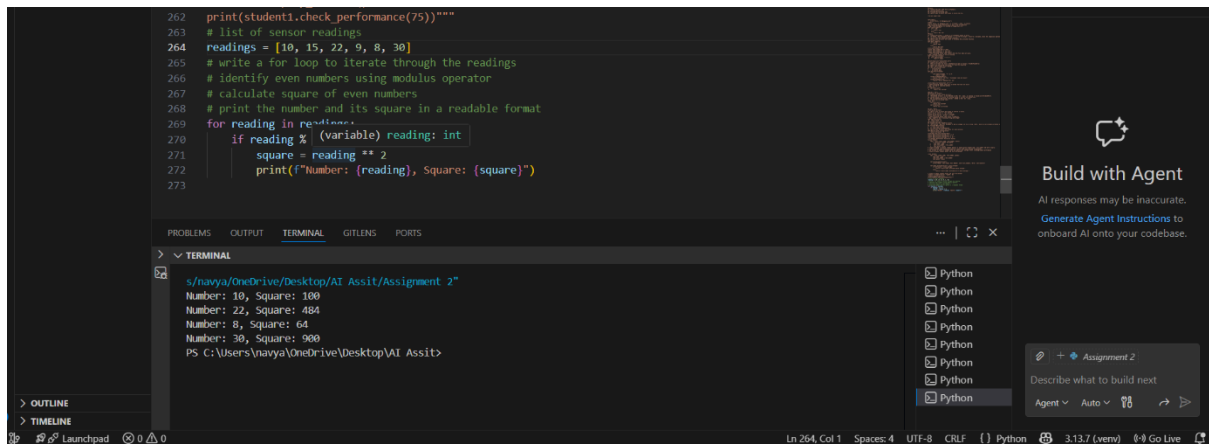
Task 2:

Prompt:

list of sensor readings

readings = [10, 15, 22, 9, 8, 30]

Write a for loop to iterate through the readings identify even numbers using modulus operator Calculate square of even numbers. Print the number and its square in a readable format



Observation

Using the provided comment prompts, GitHub Copilot generated the remaining loop logic successfully. The AI created a proper **for loop**, used the **modulus operator (%)** to identify even numbers, and calculated the square of each valid reading. The output was printed in a clear and readable format, and the program executed without errors. This task shows how AI can assist in writing loops and conditional statements efficiently while still requiring user verification.

Task 3:

Prompt:

Banking Transaction Simulation

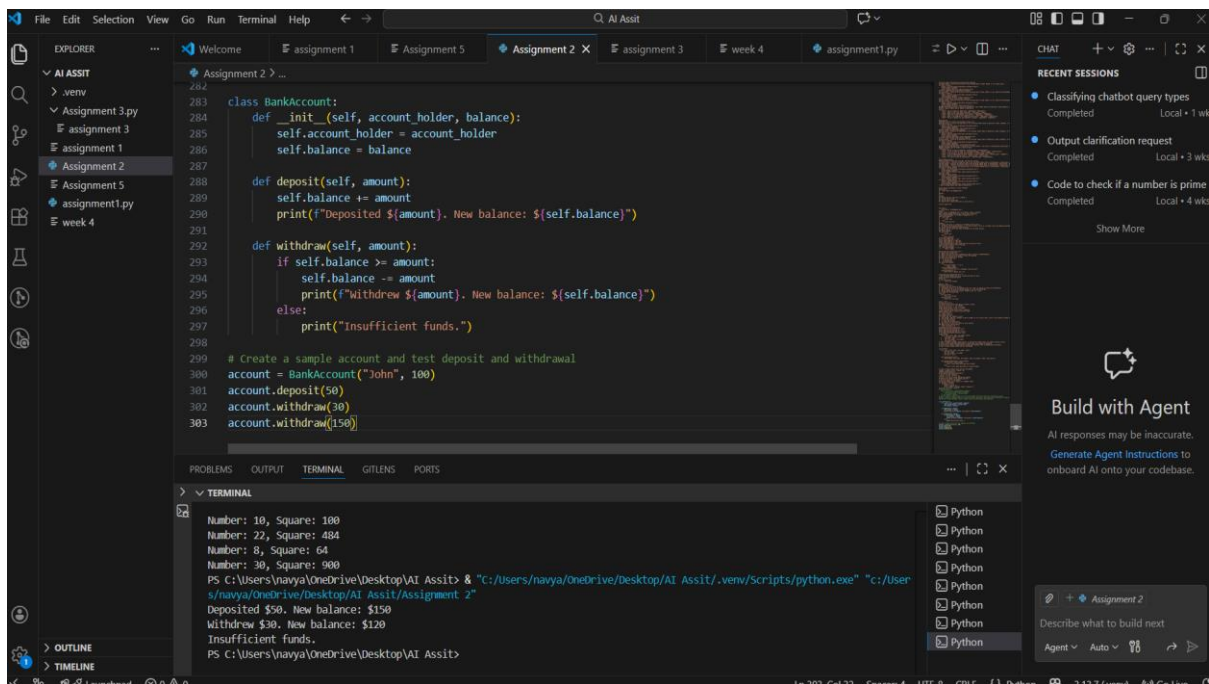
class Bank Account:

```
def __init__(self, account_holder, balance):
```

```
    self.account_holder = account_holder
```

```
    self.balance = balance
```

Create a method to deposit money into the account update the balance and print confirmation message. Create a method to withdraw money. Use if-else to check if balance is sufficient. Prevent withdrawal if balance is insufficient. Print user friendly messages. Create a sample account and test deposit and withdrawal



Observation

GitHub Copilot successfully generated the deposit and withdraw methods based on the comment prompts. The AI used class attributes through self, applied **if-else conditional logic** to prevent withdrawals when the balance was insufficient, and produced clear user-friendly messages. The program executed correctly and demonstrated how AI can help build functional class methods while still requiring human review.

Task 4:

Prompt:

list of students with name and score

students = [{"name": "Asha", "score": 80},

 {"name": "Ravi", "score": 65},

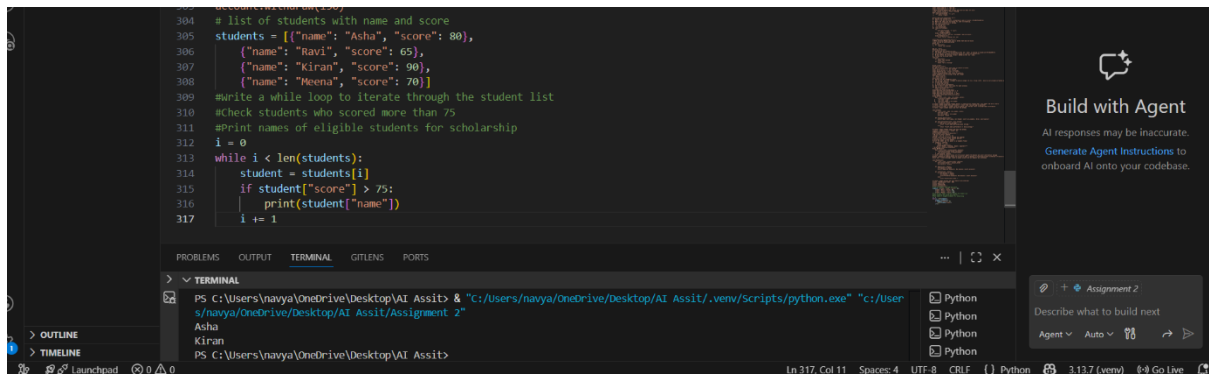
 {"name": "Kiran", "score": 90},

 {"name": "Meena", "score": 70}]

#Write a while loop to iterate through the student list

#Check students who scored more than 75

#Print names of eligible students for scholarship



```
304 # list of students with name and score
305 students = [{"name": "Asha", "score": 80},
306             {"name": "Ravi", "score": 65},
307             {"name": "Kiran", "score": 90},
308             {"name": "Meena", "score": 70}]
309 #write a while loop to iterate through the student list
310 #check students who scored more than 75
311 #Print names of eligible students for scholarship
312 i = 0
313 while i < len(students):
314     student = students[i]
315     if student["score"] > 75:
316         print(student["name"])
317     i += 1
```

PROBLEMS OUTPUT TERMINAL GITLENS PORTS

PS C:\Users\navya\OneDrive\Desktop\AI Assit> & "C:/Users/navya/OneDrive/Desktop/AI Assit/.venv/scripts/python.exe" "C:/User
s/navya/OneDrive/Desktop/AI Assit/Assignment 2"

Asha
Kiran

PS C:\Users\navya\OneDrive\Desktop\AI Assit>

Python
Python
Python
Python

Build with Agent
AI responses may be inaccurate.
Generate Agent Instructions to onboard AI onto your codebase.

Assignment 2
Describe what to build next
Agent Auto

Observation– Scholarship Eligibility (While Loop)

GitHub Copilot generated the while loop based on the given prompts. The loop correctly iterated through the list using an index, applied a conditional check to identify students scoring above 75, and printed the eligible names. The program executed successfully and demonstrated how AI can assist in generating loop and condition-based logic

Task 5:

Prompt:

Online Shopping Cart Module

class Shopping Cart:

def __init__(self):

#Create an empty list to store items each item should contain name, price and quantity

self.items= []

#Create a method to add items to the cart create a method to remove items from the cart

#Create a method to calculate total bill using loop apply discount using if condition when total exceeds a limit

#Create sample cart and test all methods

```
318 # Online Shopping Cart Module
319 #class ShoppingCart:
320
321 # def __init__(self):
322 # Create an empty list to store items each item should contain name, price and quantity
323 # self.items = []
324 # Create a method to add items to the cart create a method to remove items from the cart
325 # Create a method to calculate total bill using loop apply discount using if condition when total exceeds a limit
326 # Create sample cart and test all methods
327 class ShoppingCart:
328     def __init__(self):
329         self.items = []
330
331     def add_item(self, name, price, quantity):
332         self.items.append({"name": name, "price": price, "quantity": quantity})
333
334     def remove_item(self, name):
335         self.items = [item for item in self.items if item["name"] != name]
336
337     def calculate_total(self):
338         total = sum(item["price"] * item["quantity"] for item in self.items)
339         if total > 1000:
340             total *= 0.8 # Apply 20% discount
341         return total
342
343 # Create sample cart and test all methods
344 cart = ShoppingCart()
345 cart.add_item("Laptop", 1000, 1)
346 cart.add_item("Mouse", 25, 2)
347 print(f"Total bill: ${cart.calculate_total()}")
```

Terminal Output:

```
s:\navya\OneDrive\Desktop\AI Assit\Assignment 2>
Total bill: $840.0
PS C:\Users\navya\OneDrive\Desktop\AI Assit>
```

Observation – Online Shopping Cart Module

Using the comment prompts, GitHub Copilot completed the Shopping Cart class with methods to add items, remove items, and calculate the total bill. The AI correctly used loops and conditional statements to compute the total and apply a discount when the amount exceeded a limit. The program ran successfully and showed how AI can help build functional class-based applications efficiently.