

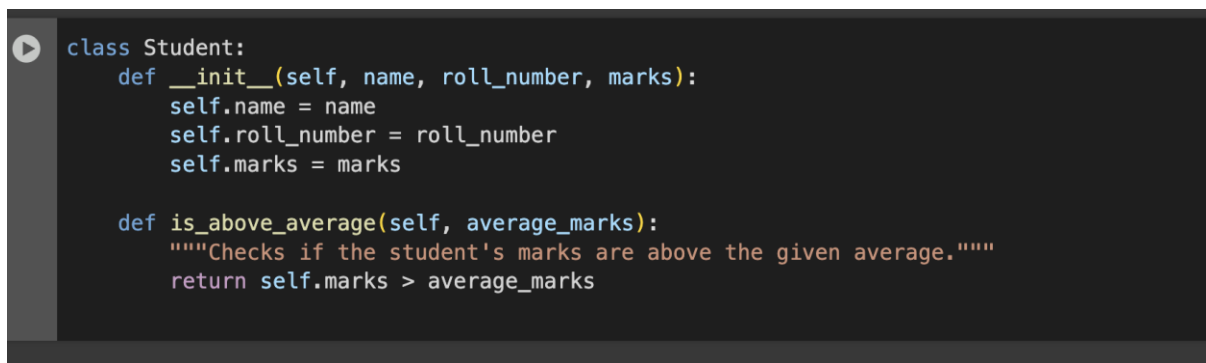
ASSIGNMENT 6.4

Name: A Sai Saketh

Roll no:2403a52179

Task Description #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: Generate a python code class named Student with attributes name, roll_number, and marks.and also check if marks are above average.

A screenshot of a code editor with a dark background. On the left, there is a play button icon. The code is written in a light-colored font. It defines a class named 'Student' with two methods: '__init__' and 'is_above_average'. The '__init__' method takes 'name', 'roll_number', and 'marks' as arguments and assigns them to 'self'. The 'is_above_average' method takes 'average_marks' as an argument and returns a boolean value based on whether 'self.marks' is greater than 'average_marks'. There is a docstring for this method: 'Checks if the student's marks are above the given average.'

```
class Student:
    def __init__(self, name, roll_number, marks):
        self.name = name
        self.roll_number = roll_number
        self.marks = marks

    def is_above_average(self, average_marks):
        """Checks if the student's marks are above the given average."""
        return self.marks > average_marks
```

Task Description #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: generate a python code to he first two lines of a for loop to iterate through a list of numbers. also calculate and print the square of even numbers only.

```
[3] numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

for number in numbers:
    if number % 2 == 0:
        square = number ** 2
        print(f"The square of {number} is {square}")
```

➤ The square of 2 is 4
The square of 4 is 16
The square of 6 is 36
The square of 8 is 64
The square of 10 is 100

Task Description #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: generate a python code to create a class called BankAccount with attributes account_holder and balance for deposit(), withdraw(), and check for insufficient balance.

```
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.")

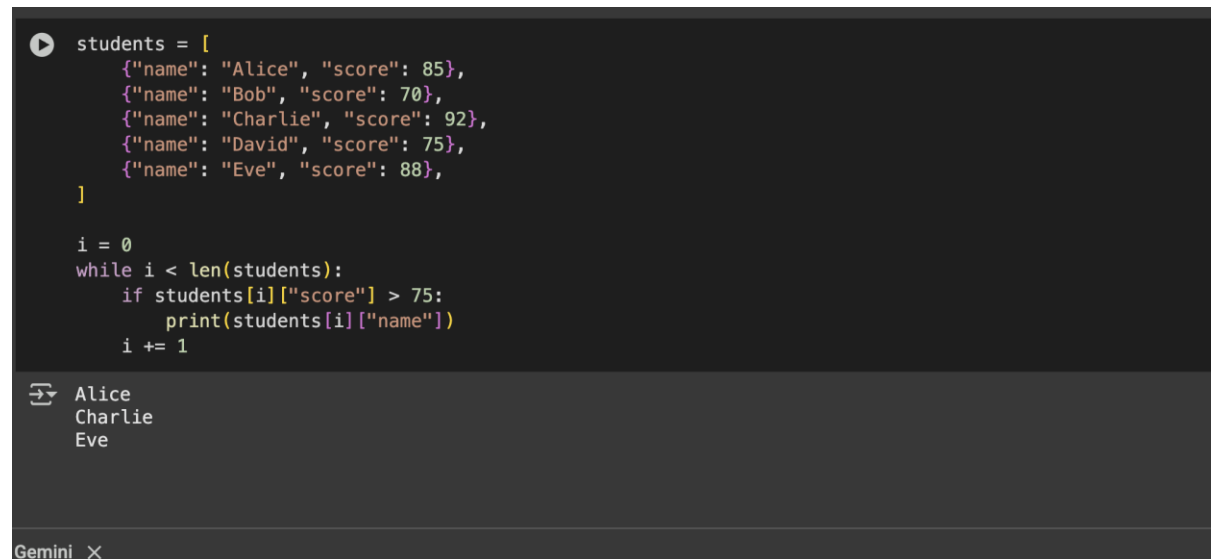
    def withdraw(self, amount):
        if amount > 0:
            if amount <= self.balance:
                self.balance -= amount
                print(f"Withdrew {amount}. New balance: {self.balance}")
            else:
                print("Insufficient funds!")
        else:
            print("Withdrawal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

    def get_balance(self):
        return self.balance
```

Task Description #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: generate a python code to define a list of student dictionaries with keys name and score and write a while loop to print the names of students who scored more than 75.



```
students = [
    {"name": "Alice", "score": 85},
    {"name": "Bob", "score": 70},
    {"name": "Charlie", "score": 92},
    {"name": "David", "score": 75},
    {"name": "Eve", "score": 88},
]

i = 0
while i < len(students):
    if students[i]["score"] > 75:
        print(students[i]["name"])
    i += 1
```

Alice
Charlie
Eve

Gemini X

Task Description #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: generate a python code Begin writing a class ShoppingCart with an empty items list and enerate methods to add_item, remove_item, and use a loop to calculate the total bill using conditional discounts.

```

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

    def add_item(self, item_name, price, quantity=1):
        """Adds an item to the shopping cart."""
        # Check if the item already exists in the cart
        for item in self.items:
            if item["name"] == item_name:
                item["quantity"] += quantity
                print(f"Added {quantity} of {item_name}. Total quantity: {item['quantity']}")
                return

        # If the item is not in the cart, add it as a new item
        self.items.append({"name": item_name, "price": price, "quantity": quantity})
        print(f"Added {quantity} of {item_name} to the cart.")

    def remove_item(self, item_name, quantity=1):
        """Removes an item from the shopping cart."""
        for item in self.items:
            if item["name"] == item_name:
                if item["quantity"] > quantity:
                    item["quantity"] -= quantity
                    print(f"Removed {quantity} of {item_name}. Remaining quantity: {item['quantity']}")
                elif item["quantity"] == quantity:
                    self.items.remove(item)
                    print(f"Removed all {quantity} of {item_name} from the cart.")
                else:
                    print(f"Not enough {item_name} in the cart to remove {quantity}.")
        return

    def __str__(self):
        return

```

```

        print(f"{item_name} not found in the cart.")

    def calculate_total(self, discount_threshold=100, discount_rate=0.10):
        """Calculates the total bill with conditional discounts."""
        total_bill = 0
        for item in self.items:
            total_bill += item["price"] * item["quantity"]

        if total_bill > discount_threshold:
            discount_amount = total_bill * discount_rate
            final_bill = total_bill - discount_amount
            print(f"Total before discount: ${total_bill:.2f}")
            print(f"Discount applied ({discount_rate*100}%): ${discount_amount:.2f}")
            print(f"Final total after discount: ${final_bill:.2f}")
            return final_bill
        else:
            print(f"Total bill: ${total_bill:.2f}")
            return total_bill

    def display_cart(self):
        """Displays the items currently in the cart."""
        if not self.items:
            print("Your shopping cart is empty.")
        else:
            print("Shopping Cart:")
            for item in self.items:
                print(f"- {item['name']}: ${item['price']:.2f} x {item['quantity']}")

```