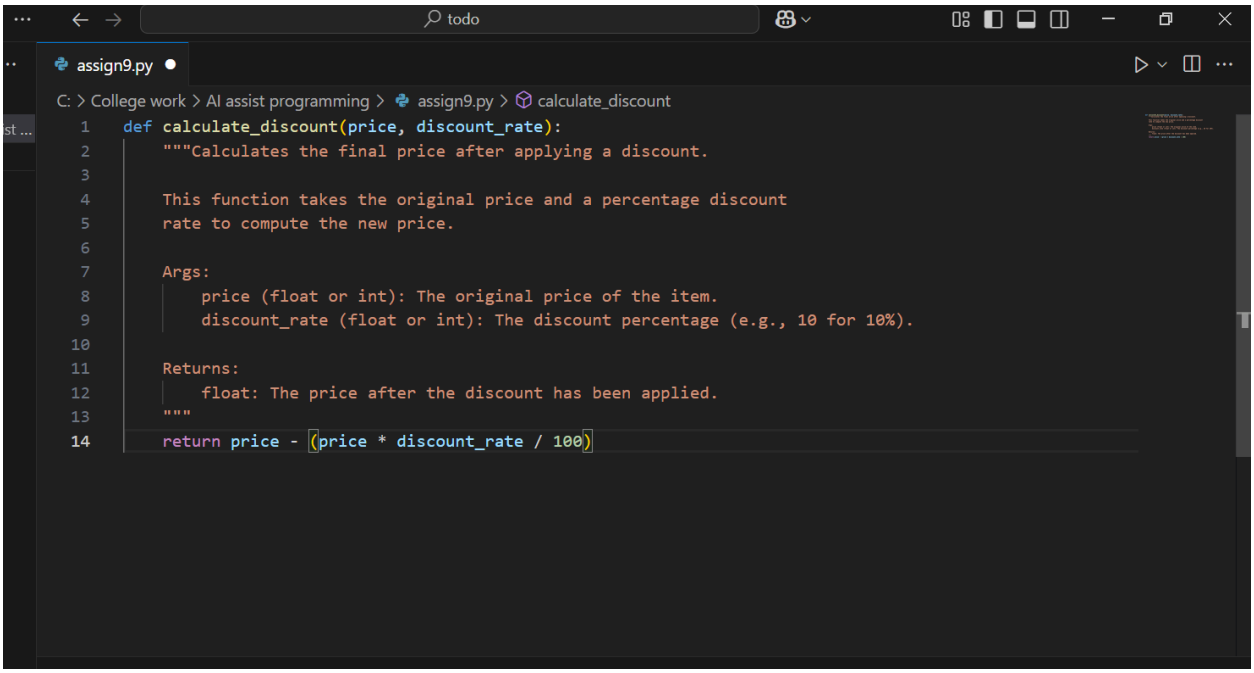
ASSIGNMENT-9.5

Task 1: (Automatic Code Commenting)
Scenario: You have been given a Python function without comments.

def calculate_discount(price, discount_rate):
return price - (price * discount_rate / 100)

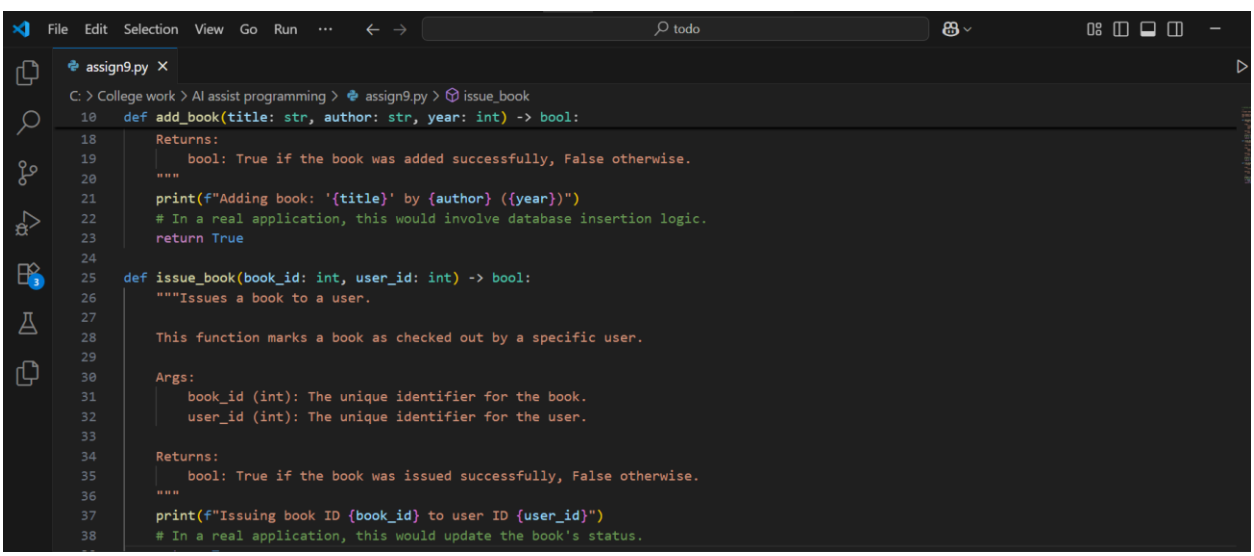
• Use an AI tool (or manually simulate it) to generate line-by-line comments for the function.
• Modify the function so that it includes a docstring in Google-style or NumPy-style format.
• Compare the auto-generated comments with your manually written version.



Task 2: (API Documentation Generator)
Scenario: A team is building a Library Management System with multiple functions.

def add_book(title, author, year):
 # code to add book
pass
def issue_book(book_id, user_id):
 # code to issue book
Pass

• Write a Python script that uses docstrings for each function (with input, output, and description).
• Use a documentation generator tool (like pdoc, Sphinx, or MkDocs) to automatically create HTML documentation.
• Submit both the code and the generated documentation as output.



assign9.py html.html

C: > College work > AI assist programming > html.html > html

```html
<!DOCTYPE html>
<html>
<head>
    <title>library_system</title>
</head>
<body>
    <h1>Module library_system</h1>
    <p><em>A simple module to manage books in a library system.</em></p>
    <p><em>This module provides basic functionalities such as adding, issuing, and finding books.</em></
    <hr>

    <h2>Function add_book</h2>
    <code><b>add_book</b>(title: str, author: str, year: int) -> bool</code>
    <p>Adds a new book to the library database.</p>
    <h4>Args:</h4>
    <ul>
        <li><b>title</b> (str): The title of the book.</li>
        <li><b>author</b> (str): The author of the book.</li>
        <li><b>year</b> (int): The publication year of the book.</li>
    </ul>
    <h4>Returns:</h4>
    <p><b>bool</b>: True if the book was added successfully, False otherwise.</p>
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS powershell

TASK 3: (AI-Assisted Code Summarization)

Scenario: You are reviewing a colleague's codebase containing long functions

```
def process_sensor_data(data):
cleaned = [x for x in data if x is not None]
avg = sum(cleaned)/len(cleaned)
anomalies = [x for x in cleaned if abs(x - avg) > 10]
return {"average": avg, "anomalies": anomalies}
```

• Generate a summary comment explaining the purpose of the function in 2–3 lines.

• Create a flow-style comment (step-by-step explanation).

• Write a short paragraph of documentation describing possible use cases of this function in real-world scenarios

STEP BY STEP EXPLANATION:

assign9.py ● html.html ●

C: > College work > AI assist programming > assign9.py > process_sensor_data

```python
def process_sensor_data(data):
    # 1. Clean the data: Create a new list 'cleaned' that excludes any 'None' values
    #    from the input 'data' list.
    cleaned = [x for x in data if x is not None]

    # 2. Calculate average: Compute the average of the numbers in the 'cleaned' list.
    #    This will raise an error if 'cleaned' is empty.
    avg = sum(cleaned) / len(cleaned)

    # 3. Find anomalies: Create a list of all values from 'cleaned' that are more
    #    than 10 units away (higher or lower) from the average.
    anomalies = [x for x in cleaned if abs(x - avg) > 10]

    # 4. Return results: Output a dictionary containing the calculated average and the
    #    list of identified anomalies.
    return {"average": avg, "anomalies": anomalies}
```

TASK 4 : (Real-Time Project Documentation)

Scenario: You are part of a project team that develops a Chatbot Application. The team needs documentation for maintainability.

• Write a README.md file for the chatbot project (include project description, installation steps, usage, and example).

File Edit Selection View Go Run ··· todo

chatbot.py readme.md ×

C: > College work > AI assist programming > chatbot > readme.md > # Simple Echo Chatbot > ## Example Interaction

```md
# Simple Echo Chatbot

## Project Description

Simple Echo Chatbot is a basic, command-line chatbot application built with Python. It's designed to demonstrate fundamental chatbot
logic, including parsing user input and providing rule-based responses. The primary goal of this project is to provide a clean and
well-documented codebase for maintainability and educational purposes.

## Features

* **Rule-Based Responses**: Responds to specific keywords and phrases.
* **Case-Insensitive**: Understands commands regardless of capitalization.
* **Dynamic Time Function**: Can provide the current time.
* **Simple & Extendable**: Easy to add new commands and responses.

## Installation

To get a local copy up and running, follow these simple steps.

**Prerequisites:**
* Python 3.8 or later must be installed on your system.

**Steps:**
1.  Clone the repository (or simply save the script below):
    ```sh
    git clone [https://github.com/example/chatbot-project.git](https://github.com/example/chatbot-project.git)
    ```
2. Navigate to the project directory:
    ```sh
    cd chatbot-project
    ```
```

Ln 51, Col 4 Spaces: 4 UTF-8 CRLF {} Markdown

• Add inline comments in the chatbot's main Python script (focus on explaining logic, not trivial code).
• Use an AI-assisted tool (or simulate it) to generate a usage guide in plain English from your code comments.
• Reflect: How does automated documentation help in real-time projects compared to manual documentation?

chatbot.py ✕ readme.md

C: > College work > AI assist programming > chatbot > chatbot.py > ...

```python
# chatbot.py
import datetime
import random

def get_response(user_input):
    """
    Generates a response based on predefined rules.
    """
    # Normalize the user's input to lowercase to make matching case-insensitive.
    # This ensures "Hello", "hello", and "HELLO" are all treated the same.
    processed_input = user_input.lower()

    # --- Rule-Based Logic ---
    # Check for greeting keywords to provide a friendly response.
    if "hello" in processed_input or "hi" in processed_input:
        return random.choice(["Hi! How can I help you today?", "Hello there!", "Hey!"])

    # Provide the current time if requested.
    # This demonstrates calling another function to perform a dynamic action.
    if "time" in processed_input:
        now = datetime.datetime.now()
        return f"The current time is {now.strftime('%I:%M %p')}."

    # Offer help if the user asks for it.
    if "help" in processed_input:
        return "You can ask me for the 'time', or greet me. Type 'bye' to exit."

    # Check for farewell keywords to end the conversation gracefully.
    if "bye" in processed_input or "quit" in processed_input:
        return "Goodbye! Have a great day."
```