

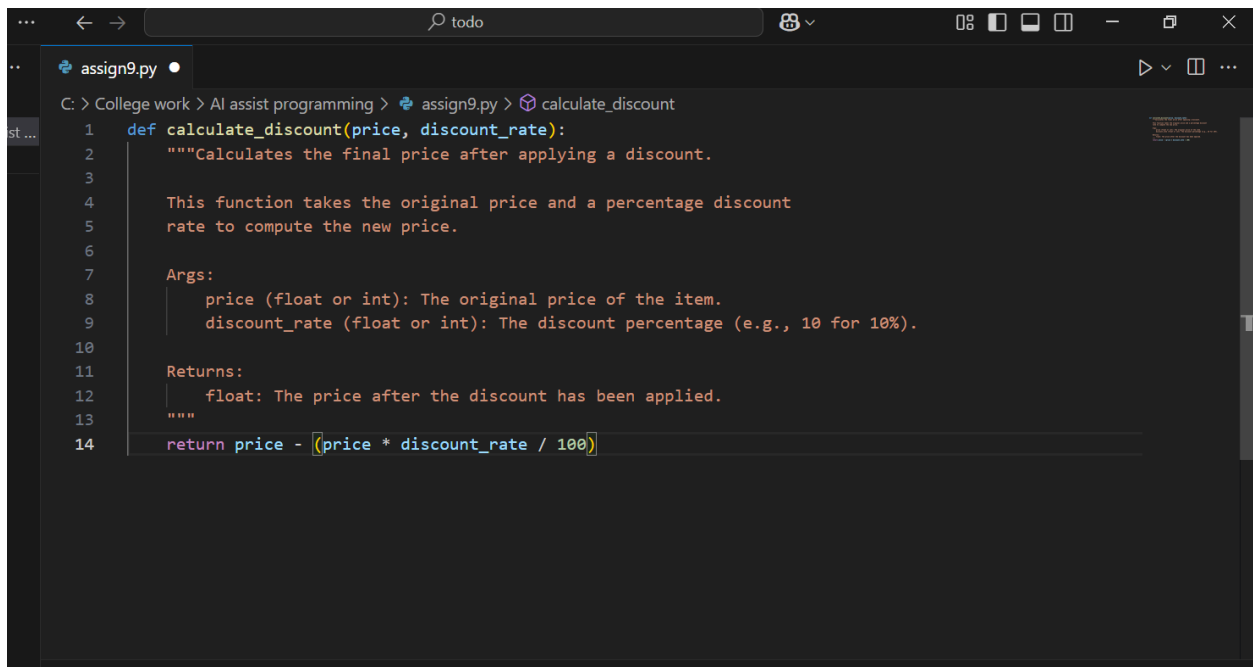
ASSIGNMENT-9.5

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.



The screenshot shows a code editor window with a file named 'assign9.py'. The code defines a function 'calculate_discount' with two parameters: 'price' and 'discount_rate'. The function body consists of a single return statement: 'return price - (price * discount_rate / 100)'. The editor has auto-generated a Google-style docstring for the function, which is displayed in orange text. The docstring includes a description of the function's purpose, its arguments, and its return value.

```
1 def calculate_discount(price, discount_rate):  
2     """Calculates the final price after applying a discount.  
3  
4     This function takes the original price and a percentage discount  
5     rate to compute the new price.  
6  
7     Args:  
8         price (float or int): The original price of the item.  
9         discount_rate (float or int): The discount percentage (e.g., 10 for 10%).  
10  
11     Returns:  
12         float: The price after the discount has been applied.  
13     """  
14     return price - (price * discount_rate / 100)
```

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
```

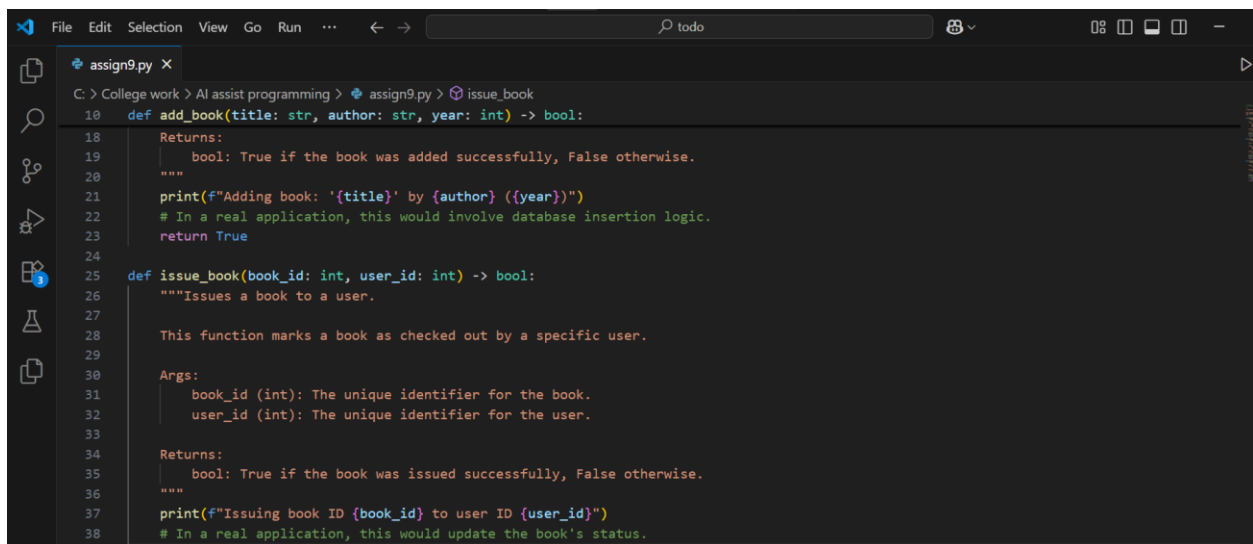
```
pass
```

```
def issue_book(book_id, user_id):
```

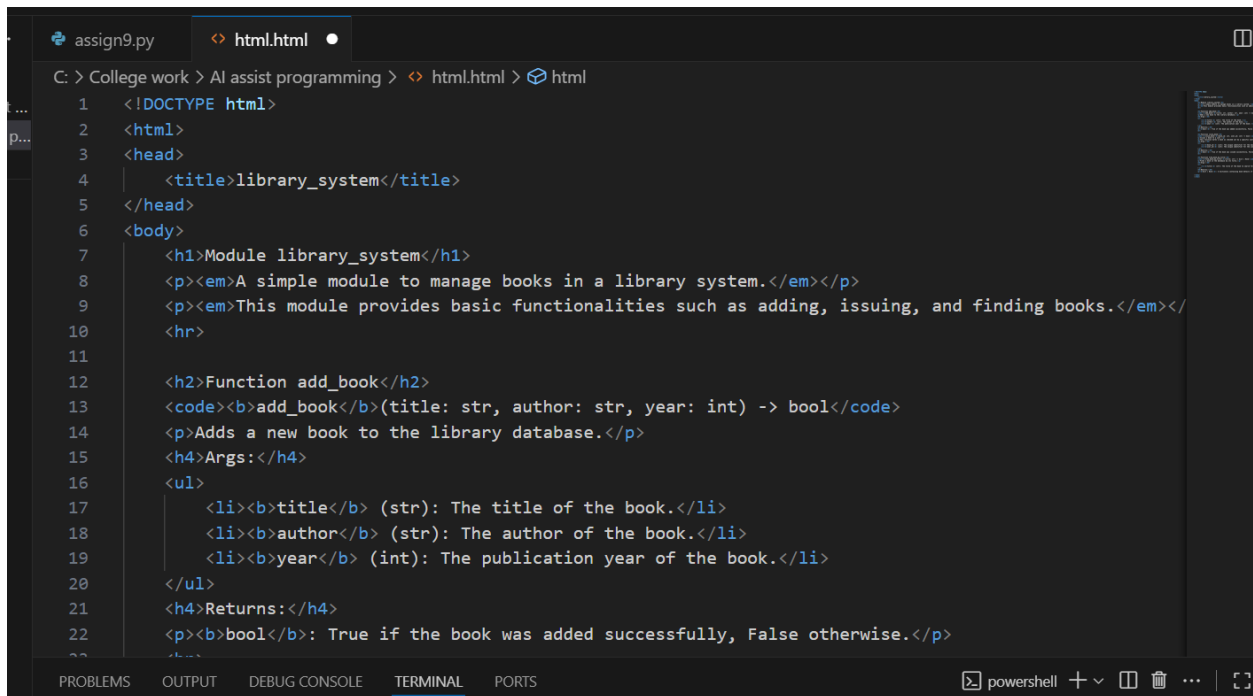
```
# code to issue book
```

```
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.



```
File Edit Selection View Go Run ... < > todo
assign9.py X
C: > College work > AI assist programming > assign9.py > issue_book
10 def add_book(title: str, author: str, year: int) -> bool:
18     Returns:
19         bool: True if the book was added successfully, False otherwise.
20     """
21     print(f"Adding book: '{title}' by {author} ({year})")
22     # In a real application, this would involve database insertion logic.
23     return True
24
25 def issue_book(book_id: int, user_id: int) -> bool:
26     """Issues a book to a user.
27
28     This function marks a book as checked out by a specific user.
29
30     Args:
31         book_id (int): The unique identifier for the book.
32         user_id (int): The unique identifier for the user.
33
34     Returns:
35         bool: True if the book was issued successfully, False otherwise.
36     """
37     print(f"Issuing book ID {book_id} to user ID {user_id}")
38     # In a real application, this would update the book's status.
```



```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>library_system</title>
5 </head>
6 <body>
7   <h1>Module library_system</h1>
8   <p><em>A simple module to manage books in a library system.</em></p>
9   <p><em>This module provides basic functionalities such as adding, issuing, and finding books.</em></p>
10  <hr>
11
12  <h2>Function add_book</h2>
13  <code><b>add_book</b>(title: str, author: str, year: int) -> bool</code>
14  <p><em>Adds a new book to the library database.</em></p>
15  <h4>Args:</h4>
16  <ul>
17    <li><b>title</b> (str): The title of the book.</li>
18    <li><b>author</b> (str): The author of the book.</li>
19    <li><b>year</b> (int): The publication year of the book.</li>
20  </ul>
21  <h4>Returns:</h4>
22  <p><b>bool</b>: True if the book was added successfully, False otherwise.</p>
```

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
1 def process_sensor_data(data):
2     # 1. Clean the data: Create a new list 'cleaned' that excludes any 'None' values
3     #   from the input 'data' list.
4     cleaned = [x for x in data if x is not None]
5
6     # 2. Calculate average: Compute the average of the numbers in the 'cleaned' list.
7     #   This will raise an error if 'cleaned' is empty.
8     avg = sum(cleaned) / len(cleaned)
9
10    # 3. Find anomalies: Create a list of all values from 'cleaned' that are more
11    #   than 10 units away (higher or lower) from the average.
12    anomalies = [x for x in cleaned if abs(x - avg) > 10]
13
14    # 4. Return results: Output a dictionary containing the calculated average and the
15    #   list of identified anomalies.
16    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 X
C: > College work > AI assist programming > chatbot > readme.md > # Simple Echo Chatbot > ## Example Interaction
1 # Simple Echo Chatbot
2
3 ## Project Description
4
5 Simple Echo Chatbot is a basic, command-line chatbot application built with Python. It's designed to demonstrate fundamental chatbot
6 logic, including parsing user input and providing rule-based responses. The primary goal of this project is to provide a clean and
7 well-documented codebase for maintainability and educational purposes.
8
9 ## Features
10
11 * **Rule-Based Responses**: Responds to specific keywords and phrases.
12 * **Case-Insensitive**: Understands commands regardless of capitalization.
13 * **Dynamic Time Function**: Can provide the current time.
14 * **Simple & Extendable**: Easy to add new commands and responses.
15
16 ## Installation
17
18 To get a local copy up and running, follow these simple steps.
19
20 **Prerequisites:**
21 * Python 3.8 or later must be installed on your system.
22
23 **Steps:**
24 1. Clone the repository (or simply save the script below):
25    ```sh
26    git clone [https://github.com/example/chatbot-project.git](https://github.com/example/chatbot-project.git)
27    ```
28 2. Navigate to the project directory:
29    ```sh
30    cd chatbot-project
31    ```
```

- 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 X  readme.md
C: > College work > AI assist programming > chatbot > chatbot.py > ...
1  # chatbot.py
2  import datetime
3  import random
4
5  def get_response(user_input):
6      """
7      Generates a response based on predefined rules.
8      """
9      # Normalize the user's input to lowercase to make matching case-insensitive.
10     # This ensures "Hello", "hello", and "HELLO" are all treated the same.
11     processed_input = user_input.lower()
12
13     # --- Rule-Based Logic ---
14     # Check for greeting keywords to provide a friendly response.
15     if "hello" in processed_input or "hi" in processed_input:
16         return random.choice(["Hi! How can I help you today?", "Hello there!", "Hey!"])
17
18     # Provide the current time if requested.
19     # This demonstrates calling another function to perform a dynamic action.
20     if "time" in processed_input:
21         now = datetime.datetime.now()
22         return f"The current time is {now.strftime('%I:%M %p')}."
23
24     # Offer help if the user asks for it.
25     if "help" in processed_input:
26         return "You can ask me for the 'time', or greet me. Type 'bye' to exit."
27
28     # Check for farewell keywords to end the conversation gracefully.
29     if "bye" in processed_input or "quit" in processed_input:
30         return "Goodbye! Have a great day."
31
```