

Assignment

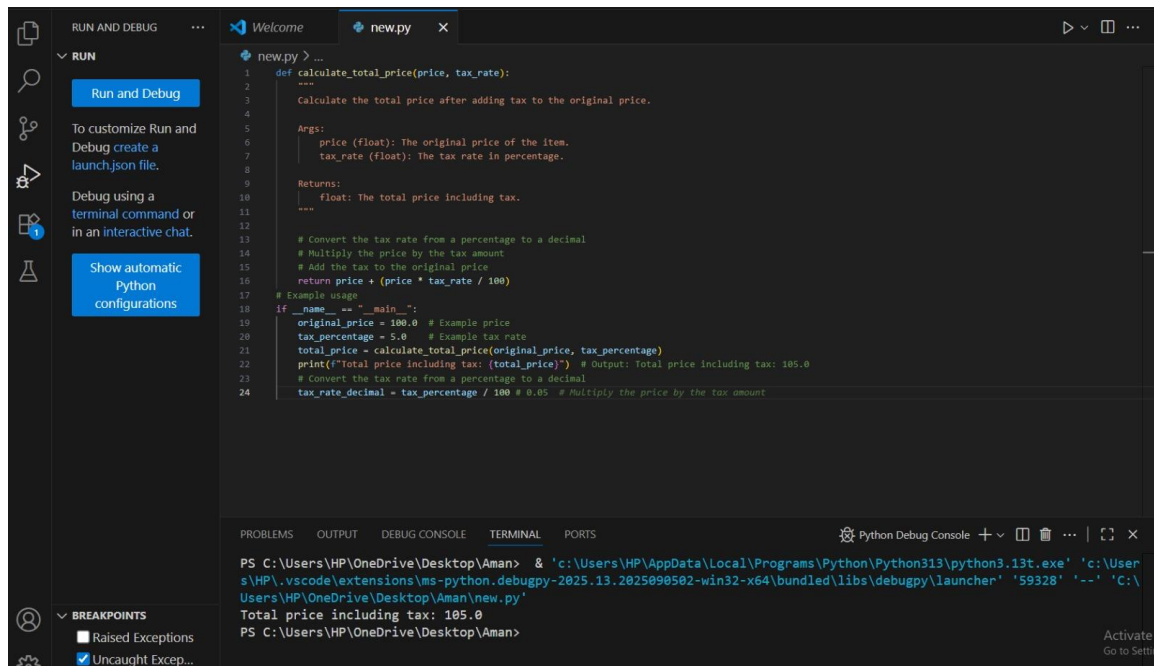
PIN: 2503A52L21

Assignment Title: Lab 9 – Documentation Generation: Automatic Documentation and Code Comments

Task 1: Automatic Code Commenting

Moto: The main goal of Task 1 is to learn how to add inline comments and docstrings to explain code clearly, making it more understandable and maintainable. It also compares auto-generated vs manually written comments.

Code/Output:



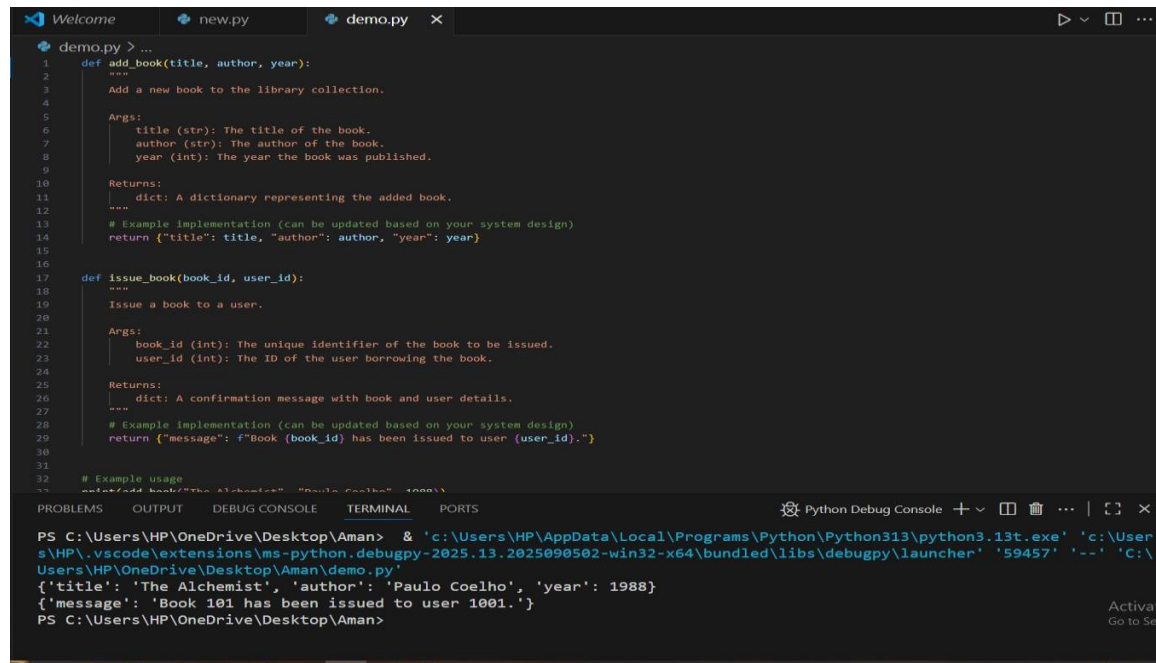
```
1 def calculate_total_price(price, tax_rate):
2     """
3     Calculate the total price after adding tax to the original price.
4
5     Args:
6         price (float): The original price of the item.
7         tax_rate (float): The tax rate in percentage.
8
9     Returns:
10        float: The total price including tax.
11    """
12
13    # Convert the tax rate from a percentage to a decimal
14    # Multiply the price by the tax amount
15    # Add the tax to the original price
16    return price + (price * tax_rate / 100)
17
18 # Example usage
19 if __name__ == "__main__":
20     original_price = 100.0 # Example price
21     tax_percentage = 5.0 # Example tax rate
22     total_price = calculate_total_price(original_price, tax_percentage)
23     print(f"Total price including tax: {total_price}") # Output: Total price including tax: 105.0
24
25 # Convert the tax rate from a percentage to a decimal
26 tax_rate_decimal = tax_percentage / 100 # 0.05 # Multiply the price by the tax amount
```

```
PS C:\Users\HP\OneDrive\Desktop\Aman> & 'c:\Users\HP\AppData\Local\Programs\Python\Python313\python3.13t.exe' 'c:\Users\HP\.vscode\extensions\ms-python.debugpy-2025.13.2025090502-win32-x64\bundled\libs\debugpy\launcher' '59328' '--' 'C:\Users\HP\OneDrive\Desktop\Aman\new.py'
Total price including tax: 105.0
PS C:\Users\HP\OneDrive\Desktop\Aman>
```

Task 2: API Documentation Generator

Moto: The main goal of Task 2 is to practice writing docstrings for functions and using auto-documentation tools like Sphinx or MkDocs to generate structured API documentation.

Code/Output:

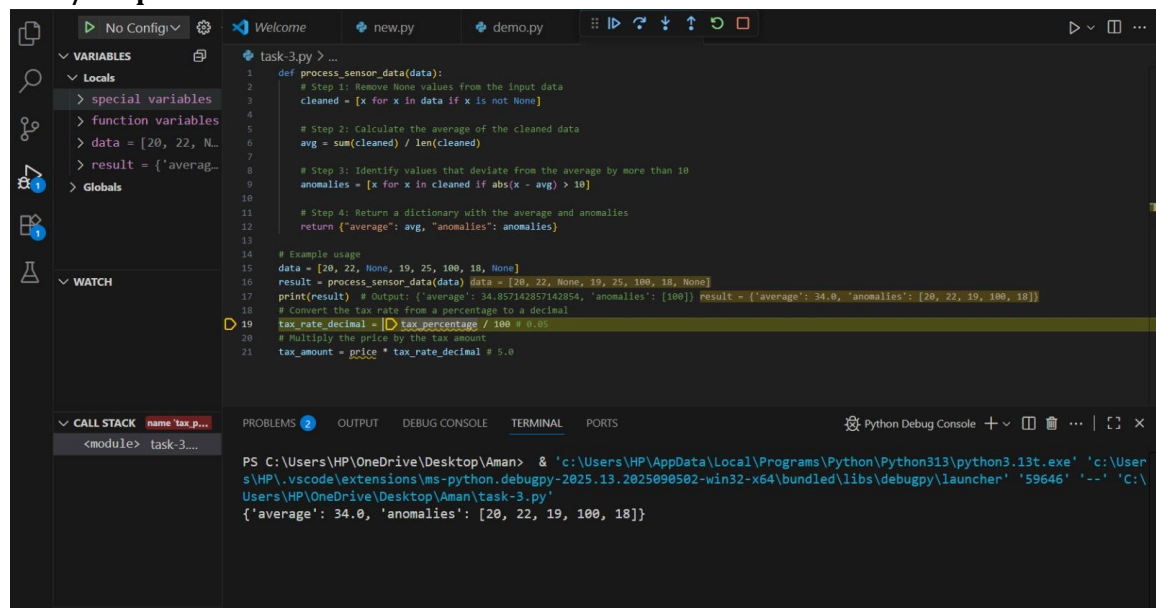


```
demo.py > ...
1 def add_book(title, author, year):
2     """
3     Add a new book to the library collection.
4
5     Args:
6         title (str): The title of the book.
7         author (str): The author of the book.
8         year (int): The year the book was published.
9
10    Returns:
11        dict: A dictionary representing the added book.
12    """
13    # Example Implementation (can be updated based on your system design)
14    return {"title": title, "author": author, "year": year}
15
16
17 def issue_book(book_id, user_id):
18     """
19     Issue a book to a user.
20
21     Args:
22         book_id (int): The unique identifier of the book to be issued.
23         user_id (int): The ID of the user borrowing the book.
24
25     Returns:
26        dict: A confirmation message with book and user details.
27    """
28    # Example Implementation (can be updated based on your system design)
29    return {"message": f"Book {book_id} has been issued to user {user_id}."}
30
31
32 # Example usage
33 add_book("The Alchemist", "Paulo Coelho", 1988)
34
35
36 PS C:\Users\HP\OneDrive\Desktop\Aman> & 'c:\Users\HP\AppData\Local\Programs\Python\Python313\python3.13t.exe' 'c:\Users\HP\.vscode\extensions\ms-python.debugpy-2025.13.2025090502-win32-x64\bundled\libs\debugpy\launcher' '59457' '--' 'C:\Users\HP\OneDrive\Desktop\Aman\demo.py'
{'title': 'The Alchemist', 'author': 'Paulo Coelho', 'year': 1988}
{'message': 'Book 101 has been issued to user 1001.'}
PS C:\Users\HP\OneDrive\Desktop\Aman>
```

Task 3: AI-Assisted Code Summarization

📌 **Moto:** The goal of Task 3 is to practice summarizing functions using concise comments, step-by-step flow explanations, and documenting real-world use cases.

Code/Output:



```
task-3.py > ...
1 def process_sensor_data(data):
2     # Step 1: Remove None values from the input data
3     cleaned = [x for x in data if x is not None]
4
5     # Step 2: Calculate the average of the cleaned data
6     avg = sum(cleaned) / len(cleaned)
7
8     # Step 3: Identify values that deviate from the average by more than 10
9     anomalies = [x for x in cleaned if abs(x - avg) > 10]
10
11    # Step 4: Return a dictionary with the average and anomalies
12    return {"average": avg, "anomalies": anomalies}
13
14    # Example usage
15    data = [20, 22, None, 10, 25, 100, 18, None]
16    result = process_sensor_data(data)
17    print(result) # Output: {'average': 34.857142857142854, 'anomalies': [100]}
18    # Convert the tax rate from a percentage to a decimal
19    tax_rate_decimal = tax_percentage / 100 * 0.05
20    # Multiply the price by the tax amount
21    tax_amount = price * tax_rate_decimal * 5.0
```

Variables:

- Locals
- special variables
- function variables
- data = [20, 22, N...
- result = {'averag...
- Globals

WATCH

CALL STACK

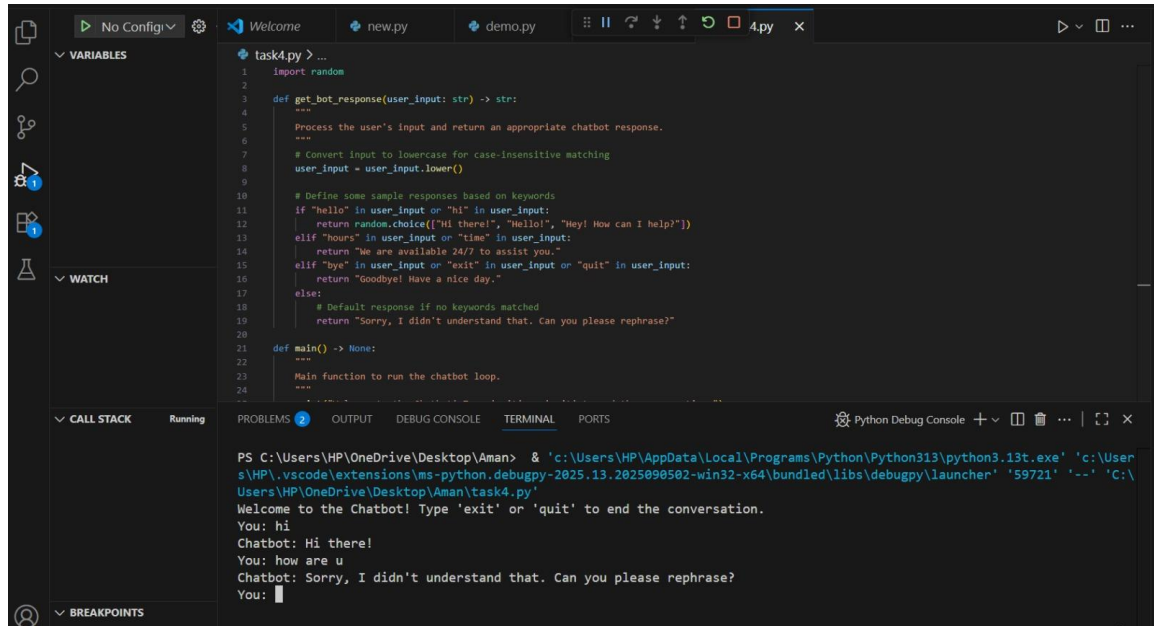
name 'tax.p...'
<module> task-3....

```
PS C:\Users\HP\OneDrive\Desktop\Aman> & 'c:\Users\HP\AppData\Local\Programs\Python\Python313\python3.13t.exe' 'c:\Users\HP\.vscode\extensions\ms-python.debugpy-2025.13.2025090502-win32-x64\bundled\libs\debugpy\launcher' '59646' '--' 'C:\Users\HP\OneDrive\Desktop\Aman\task-3.py'
{'average': 34.0, 'anomalies': [20, 22, 19, 100, 18]}
```

Task 4: Real-Time Project Documentation

📌 **Moto:** The main goal of Task 4 is to practice creating real-time project documentation with README, inline comments, AI-assisted guides, and reflection on automated documentation.

Code/Output:



The screenshot displays the Visual Studio Code interface. The editor window shows a Python file named `task4.py` with the following code:

```
1 import random
2
3 def get_bot_response(user_input: str) -> str:
4     """
5     Process the user's input and return an appropriate chatbot response.
6     """
7     # Convert input to lowercase for case-insensitive matching
8     user_input = user_input.lower()
9
10    # Define some sample responses based on keywords
11    if "hello" in user_input or "hi" in user_input:
12        return random.choice(["Hi there!", "Hello!", "Hey! How can I help?"])
13    elif "hours" in user_input or "time" in user_input:
14        return "We are available 24/7 to assist you."
15    elif "bye" in user_input or "exit" in user_input or "quit" in user_input:
16        return "Goodbye! Have a nice day."
17    else:
18        # Default response if no keywords matched
19        return "Sorry, I didn't understand that. Can you please rephrase?"
20
21 def main() -> None:
22     """
23     Main function to run the chatbot loop.
24     """
25     while True:
26         user_input = input("You: ")
27         if user_input.lower() in ["exit", "quit", "bye"]:
28             break
29         bot_response = get_bot_response(user_input)
30         print(f"Chatbot: {bot_response}")
```

The bottom panel shows the **TERMINAL** output:

```
PS C:\Users\HP\OneDrive\Desktop\Aman> & 'c:\Users\HP\AppData\Local\Programs\Python\Python313\python3.13t.exe' 'c:\Users\HP\.vscode\extensions\ms-python.debugpy-2025.13.2025090502-win32-x64\bundle\libs\debugpy\launcher' '59721' '--' 'C:\Users\HP\OneDrive\Desktop\Aman\task4.py'
Welcome to the Chatbot! Type 'exit' or 'quit' to end the conversation.
You: hi
Chatbot: Hi there!
You: how are u
Chatbot: Sorry, I didn't understand that. Can you please rephrase?
You: 
```

Observation

In this lab, I learned the importance of inline comments, docstrings, and documentation tools. Automated documentation keeps code easy to maintain, while manual notes provide context. Combining both approaches helps in building reliable and maintainable projects.