

Assignment

HTNO: 2503A52L16

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:

The screenshot shows a Jupyter Notebook interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Toolbar:** Back, Forward, Search, Home, Refresh, Stop, Run Cell, Cell, Kernel, Help.
- Code Area:** A code editor containing three versions of a Python function: `task1_calculate_discount.py`.
 - Version 1: AI-Generated Comments** (Lines 1-17):

```
task1_calculate_discount.py > ...
1   """
2   Lab 9 | Task 1
3   Automatic Code Commenting & Documentation
4   """
5
6   # -----
7   # Version 1: AI-Generated Comments
8   #
9   def calculate_discount(price, discount_rate):
10      # calculates the discount amount
11      discount = price * discount_rate / 100
12      # Subtracts the discount amount from the original price
13      final_price = price - discount
14      return final_price
15
16
17  # -----
18  # Version 2: Manual Comments
19  #
20  def calculate_discount_manual(price, discount_rate):
21      # Step 1: Calculate the discount amount by multiplying price with discount rate
22      discount = price * discount_rate / 100
23      # Step 2: Subtract discount amount from the original price to get final price
24      final_price = price - discount
25      return final_price
26
27
28  # -----
29  # Version 3: Google-Style Docstring
30  #
31  def calculate_discount_google(price, discount_rate):
32      """
33          Calculates the final price after applying a discount.
34
35          Args:
36              price (float): The original price of the item.
37              discount_rate (float): The discount rate as a percentage.
38
39      
```
 - Version 2: Manual Comments** (Lines 20-25)
 - Version 3: Google-Style Docstring** (Lines 31-38)
- Terminal:** PS D:\BTECH\AI Assisted Coding\LABS ASSIGNMENTS\Lab 9 & C:\Users\saikr\AppData\Local\Programs\Python\Python313\python.exe "d:\BTECH\AI Assisted Coding\LABS ASSIGNMENTS\Lab 9\task1_calculate_discount.py"
Original Price: 1000
Discount Rate: 10 %
Discount Amount: 100.0
AI Version Final Price: 900.0
Manual Version Final Price: 900.0
Google Docstring Final Price: 900.0
NumPy Docstring Final Price: 900.0
- PROBLEMS, TERMINAL, and PowerShell tabs:** PROBLEMS, TERMINAL, powershell, etc.
- EXPLORER:** LAB, task1_calculate_discount.py, task2_library_management.py, task3_process_sensor.py, task4_chatbot.py, task4_READMe.md, task4_usage_guide.txt, task4_reflection.txt.
- Bottom Bar:** Ln 5, Col 1, Spaces: 4, UTF-8, CRLF, Python, Python 3.13 (64-bit), Go Live, ba, Background, ENG, IN, 11:52, 12-09-2023.

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:

The screenshot shows a Jupyter Notebook interface with two code cells and a terminal output.

Code Cell 1:

```
task2_library.management.py X
task2_library.management.py @ add_book
...
2 Library Management System
3 Lab 9 Task 2
4 This module provides basic functions to manage books in a library system.
5 ...
6
7 def add_book(title, author, year):
8     """
9         Add a new book to the library.
10
11     Args:
12         title (str): The title of the book.
13         author (str): The author of the book.
14         year (int): The year the book was published.
15
16     Returns:
17         dict: A dictionary containing the details of the added book.
18
19     Example:
20         >>> add_book("1984", "George Orwell", 1949)
21         {'title': '1984', 'author': 'George Orwell', 'year': 1949}
22
23     # For demonstration, we just return a dictionary
24     return {"title": title, "author": author, "year": year}
25
26
27 def issue_book(book_id, user_id):
28     """
29         Issue a book from the library to a user.
30
31     Args:
32         book_id (int): The unique ID of the book to issue.
33         user_id (int): The unique ID of the user borrowing the book.
34
35     Returns:
36         str: A confirmation message with book_id and user_id.
37
```

Code Cell 2:

```
Q Lab 9
PROBLEMS OUTPUT TERMINAL ... _> powershell +> ... ○ PS D:\BTech\AI Assisted Coding\LABS ASSIGNMENTS\Lab 9 & C:\Users\saikr\AI ppData\local\Programs\Python\python313\python.exe "d:/BTech/AI Assisted C oding/LABS ASSIGNMENTS/Lab 9/task2_library.management.py"
{'title': 'The Hobbit', 'author': 'J.R.R. Tolkien', 'year': 1937}
Book 101 has been issued to user 2001.
○ PS D:\BTech\AI Assisted Coding\LABS ASSIGNMENTS\Lab 9 > ■
```

Terminal Output:

```
PS D:\BTech\AI Assisted Coding\LABS ASSIGNMENTS\Lab 9 & C:\Users\saikr\AI ppData\local\Programs\Python\python313\python.exe "d:/BTech/AI Assisted C oding/LABS ASSIGNMENTS/Lab 9/task2_library.management.py"
{'title': 'The Hobbit', 'author': 'J.R.R. Tolkien', 'year': 1937}
Book 101 has been issued to user 2001.
○ PS D:\BTech\AI Assisted Coding\LABS ASSIGNMENTS\Lab 9 > ■
```

Explorer:

- task1_calculate_discount.py
- task2_library.management.py
- task3_process_sensor.py
- task4_chatbot.py
- task4_READMeEnd
- task4_usage_guide.txt
- task4_reflection.txt

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 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 shows a code editor window titled "Q Lab 9" with the file "task4_chatbot.py" open. The code defines a chatbot application with predefined responses for "hello", "what is your name?", and "bye". It includes docstrings for the main function and the response function. The terminal tab shows the execution of the script and a conversation with the chatbot. The Explorer tab lists other files in the project directory.

```
task4_chatbot.py >...  
1  #> 2. - chatbot.py  
2  """  
3  Chatbot Application  
4  Lab 9 [ ] Task 4  
5  """  
6  
7  # Predefined responses stored in a dictionary  
8  responses = {  
9      "hello": "Hi there! How can I help you?",  
10     "what is your name?": "I am your friendly Python chatbot.",  
11     "bye": "Goodbye! Have a nice day."  
12 }  
13  
14 def chatbot_response(user_input):  
15     """  
16         Returns chatbot's response based on user input.  
17  
18     Args:  
19         user_input (str): The message typed by the user.  
20  
21     Returns:  
22         str: chatbot's reply (predefined) or default message if not found.  
23  
24     # Normalize input by converting to lowercase and stripping spaces  
25     cleaned_input = user_input.lower().strip()  
26  
27     # Match user input with predefined responses  
28     if cleaned_input in responses:  
29         return responses[cleaned_input]  
30  
31     # If input is unknown, return a default response  
32     return "Sorry, I don't understand that."  
33  
34  
35 if __name__ == "__main__":  
36     print("Welcome to the Chatbot! Type 'bye' to exit.")  
PS D:\BTECH\AI Assisted Coding\LABS ASSIGNMENTS\Lab 9> & C:\Users\saikr\Anaconda3\envs\py313\python.exe "d:\BTECH\AI Assisted Coding\LABS ASSIGNMENTS\Lab 9\task4_chatbot.py"  
Welcome to the Chatbot! Type 'bye' to exit.  
You: hello  
Chatbot: Hi there! How can I help you?  
You: what is your name?  
Chatbot: I am your friendly Python chatbot.  
You: bye  
Chatbot: Goodbye! Have a nice day.  
PS D:\BTECH\AI Assisted coding\LABS ASSIGNMENTS\Lab 9>  
EXPLORER LAB... 🔍 🗂️ 📁 🗃 🗃 🗃  
task1_calculate_discount.py  
task2_library_management.py  
task3_process_sensor.py  
task4_chatbot.py  
task4_README.md  
task4_usage_guide.txt  
task4reflection.txt  
Ln 12, Col 2 Spaces: 4 UTRF- CRLF {} Python Python 3.13 (64-bit) Go Live ba Background 🔍  
ENG IN 11:56 12-09-2025 📅
```

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.