SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE		DEPARTMENT OF COMPUTER SCIENCE ENGINEERING	
Program Name: B. Tech		**Assignment Type: Lab**	**Academic Year:**2025-2026
Course Coordinator Name		Dr. Rishabh Mittal	
Hall ticket		**2303A54040**	
CourseCode	23CS002PC304	**Course Title**	AI Assisted Coding
Year/Sem	III/II	**Regulation**	R23
Date and Day of Assignment	**Week5–Monday**	**Time(s)**	23CSBTB01 To 23CSBTB52
Name	P.Mani Sai	**Batch**	47-B
Assignment Number:9.1(Present assignment number)/**24**(Total number of assignments)			

Q.No.	**Question**	***Expected Time to complete***
1	**Lab Experiment: Documentation Generation -Automatic documentation and code comments** **Lab Objectives** 1. To understand automatic documentation generation. 2. To generate code comments and docstrings using AI tools. 3. To learn the importance of documentation in software development. **Lab Outcomes** 1. Students will be able to generate documentation automatically for code. 2. Students will be able to add clear comments and docstrings to programs. 3. Students will be able to improve code readability and	Week5 - Monday

maintainability using documentation.

Problem 1:

Consider the following Python function:

```
def find_max(numbers):

    return max(numbers)
```

Task :

- **Write documentation for the function in all three formats:**

(a) Doc string : In this format, the documentation is written inside triple quotes just below the function definition.

The doc string explains that:

- The function takes a list of numbers as input.
- It finds the maximum value in the list.
- It returns the largest number.

This format keeps the explanation simple and clear. It is placed inside the function, so anyone reading the code can immediately understand what the function does.

(b) Inline comments : In this format, comments are written inside the function using the `#` symbol.

The inline comments explain:

- That the function receives a list of numbers.
- That it uses Python's built-in method to find the maximum value.
- That it returns the result.

Inline comments mainly explain the internal working of the function step by step. They are helpful when the logic is complex.

(c) Google-style documentation : In this format, the documentation is written in a structured way inside the docstring.

It clearly separates:

- A short description of what the function does.
- The arguments it takes (the list of numbers).
- The return value (the maximum number).
- Possible errors (for example, if the list is empty).
- Sometimes an example of usage.

This style is more detailed and organized. It is mostly used in professional projects.

Critically compare the three approaches. Discuss the advantages, disadvantages, and suitable use cases of each style. :

Docstring

- Simple and easy to write.
- Automatically used by documentation tools.
- Good for small programs.
- But it may not clearly mention all details like types and errors.

Inline Comments

- Helpful to explain logic clearly.
- Good for understanding complex code.
- But cannot be used to generate automatic documentation.
- Not suitable as official documentation.

Google-Style Documentation

- Very clear and well-structured.
- Explains arguments and return values properly.
- Suitable for large projects and libraries.
- Takes more time to write.

- **Recommend which documentation style is most effective for a mathematical utilities library and justify your answer :**

For a mathematical utilities library, Google-style documentation is the most effective.

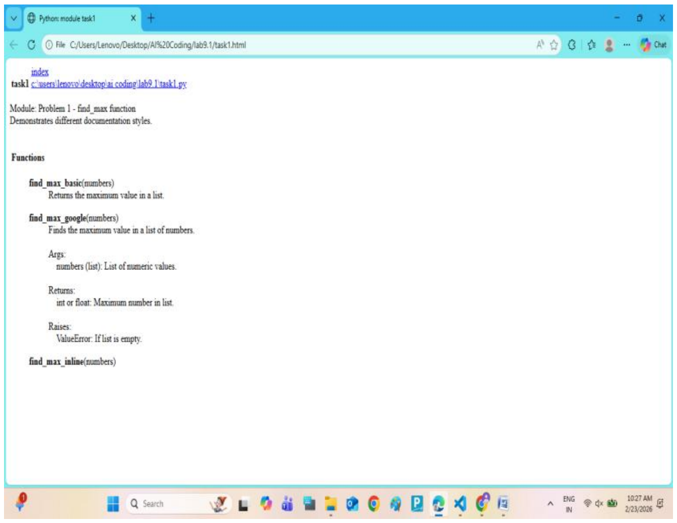
This is because mathematical functions must clearly mention:

- What type of input they expect.
- What output they return.
- What errors may occur.

Since such libraries are reused by many developers, proper and structured documentation is very important.

Prompt used : python -m pydoc task1

python -m pydoc -w task1



```
Python: module task1
File  C:/Users/Lenovo/Desktop/AI%20Coding/lab9.1/task1.html

        index
task1 c:\users\lenovo\desktop\ai coding\lab9.1\task1.py

Module: Problem 1 - find_max function
Demonstrates different documentation styles.


Functions

    find_max_basic(numbers)
        Returns the maximum value in a list.

    find_max_google(numbers)
        Finds the maximum value in a list of numbers.

        Args:
          numbers (list): List of numeric values.

        Returns:
          int or float: Maximum number in list.

        Raises:
          ValueError: If list is empty.

    find_max_inline(numbers)
```

Problem 2: Consider the following Python function:

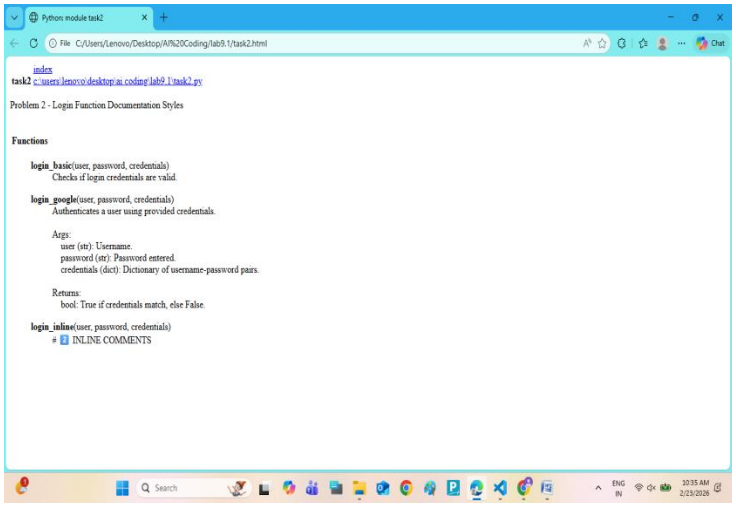
```
def login(user, password, credentials):

return credentials.get(user) == password
```

Task:

1. Write documentation in all three formats.
2. Critically compare the approaches.
3. Recommend which style would be most helpful for new developers onboarding a project, and justify your choice.

Prompt used : python -m pydoc task2

python -m pydoc -w task2



```
Python: module task2
File  C:/Users/Lenovo/Desktop/AI%20Coding/lab9.1/task2.html

     index
task2 c:\users\lenovo\desktop\ai coding\lab9.1\task2.py

Problem 2 - Login Function Documentation Styles


Functions

    login_basic(user, password, credentials)
        Checks if login credentials are valid.

    login_google(user, password, credentials)
        Authenticates a user using provided credentials.

        Args:
          user (str): Username.
          password (str): Password entered.
          credentials (dict): Dictionary of username-password pairs.

        Returns:
          bool: True if credentials match, else False.

    login_inline(user, password, credentials)
        # 2 INLINE COMMENTS
```

Problem 3: Calculator (Automatic Documentation Generation)

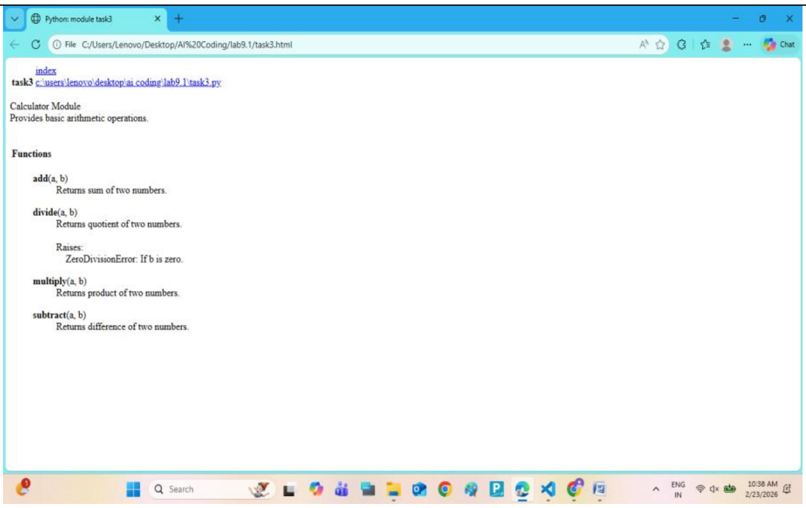
Task:Design a Python module named calculator.py and demonstrate automatic documentation generation.

Instructions:

1. Create a Python module calculator.py that includes the following functions, each written with appropriate docstrings:
   - add(a, b) – returns the sum of two numbers
   - subtract(a, b) – returns the difference of two numbers
   - multiply(a, b) – returns the product of two numbers
   - divide(a, b) – returns the quotient of two numbers
2. Display the module documentation in the terminal using Python's documentation tools.
3. Generate and export the module documentation in HTML format using the pydoc utility, and open the generated HTML file in a web browser to verify the output.

   Prompt used : python -m pydoc task3

   python -m pydoc -w task3



index
task3 c:\users\lenovo\desktop\ai coding\lab9.1\task3.py

Calculator Module
Provides basic arithmetic operations.

Functions

add(a, b)
Returns sum of two numbers.

divide(a, b)
Returns quotient of two numbers.

Raises:
ZeroDivisionError: If b is zero.

multiply(a, b)
Returns product of two numbers.

subtract(a, b)
Returns difference of two numbers.

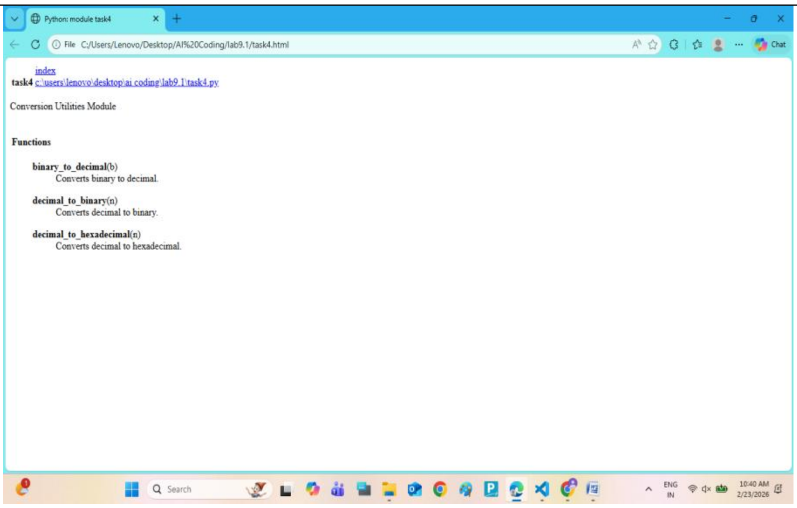
Problem 4:Conversion Utilities Module

Task:

1. Write a module named conversion.py with functions:
   - decimal_to_binary(n)
   - binary_to_decimal(b)
   - decimal_to_hexadecimal(n)
2. Use Copilot for auto-generating docstrings.
3. Generate documentation in the terminal.
4. Export the documentation in HTML format and open it in a browser.

   Prompt used : python -m pydoc task4

   python -m pydoc -w task4



```
index
task4 c:\users\lenovo\desktop\ai coding\lab9.1\task4.py

Conversion Utilities Module

Functions

    binary_to_decimal(b)
        Converts binary to decimal.

    decimal_to_binary(n)
        Converts decimal to binary.

    decimal_to_hexadecimal(n)
        Converts decimal to hexadecimal.
```

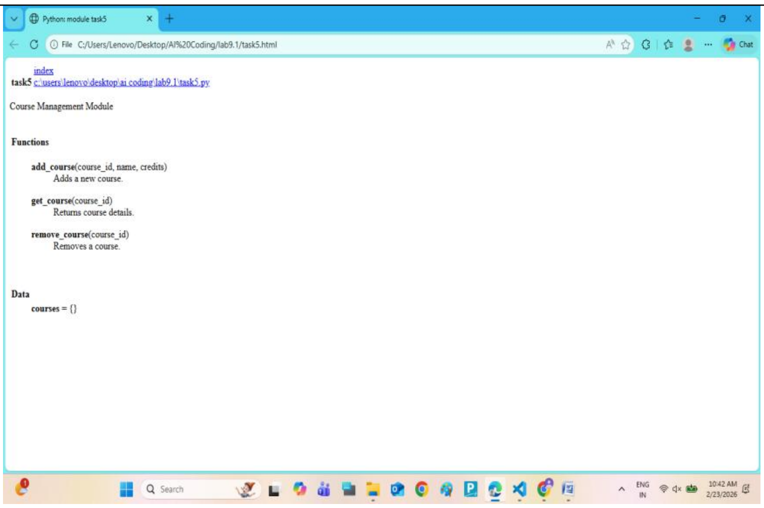
Problem 5 – Course Management Module

Task:

1. Create a module course.py with functions:
   - add_course(course_id, name, credits)
   - remove_course(course_id)
   - get_course(course_id)
2. Add docstrings with Copilot.
3. Generate documentation in the terminal.
4. Export the documentation in HTML format and open it in a browser.

Prompt used : python -m pydoc task5

python -m pydoc -w task5

	 Python: module task5 File C:/Users/Lenovo/Desktop/AI%20Coding/lab9.1/task5.html index **task5** c:\users\lenovo\desktop\ai coding\lab9.1\task5.py Course Management Module **Functions** **add_course**(course_id, name, credits) Adds a new course. **get_course**(course_id) Returns course details. **remove_course**(course_id) Removes a course. **Data** **courses** = {}	