School of Computer Science and Artificial Intelligence

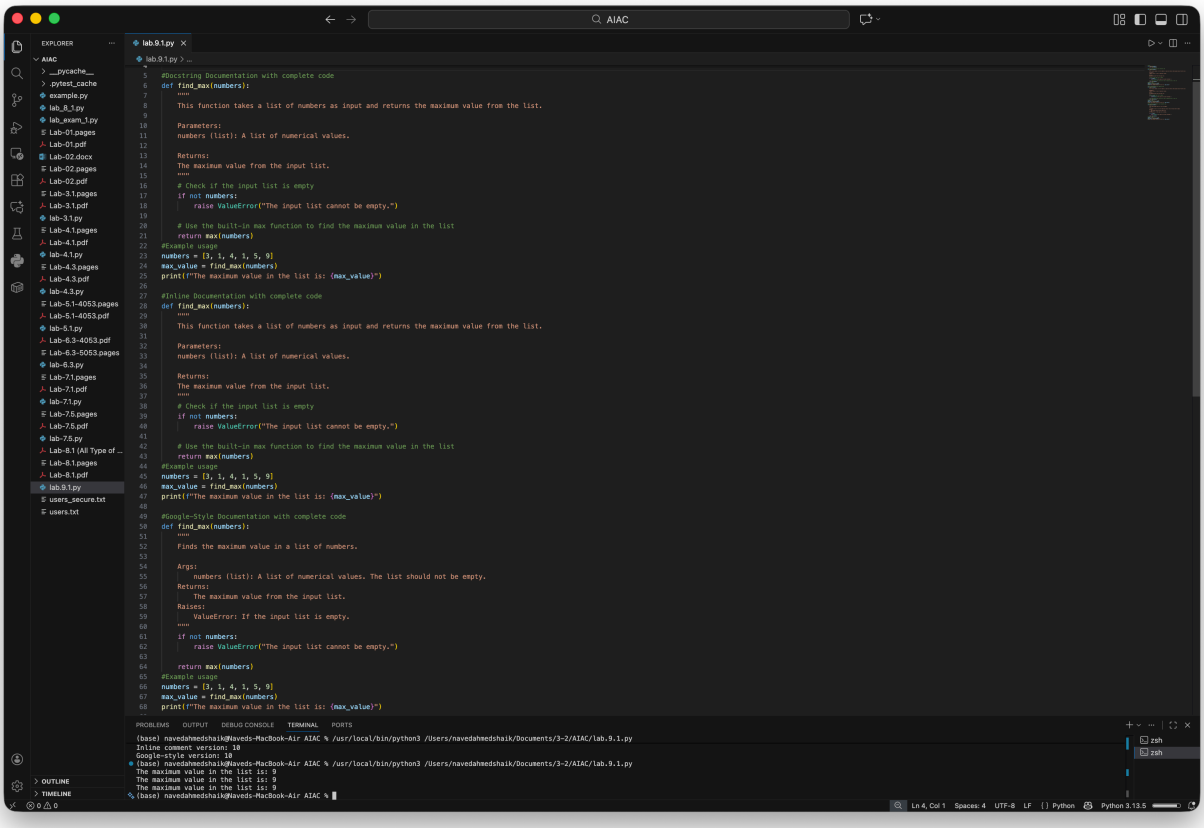
Lab Assignment-9.1

Course Name : AI Assistant Coding
Name of the Student : Shaik Naved Ahmed
Registration No : 2303A54053
Batch No : 47-B

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) Docstring
(b) Inline comments
(c) Google-style documentation
• Critically compare the three approaches. Discuss the advantages, disadvantages, and suitable use cases of each style.
• Recommend which documentation style is most effective for a mathematical utilities library and justify your answer.

SCREENSHOT OF GENERATED CODE:



EXPLANATION:

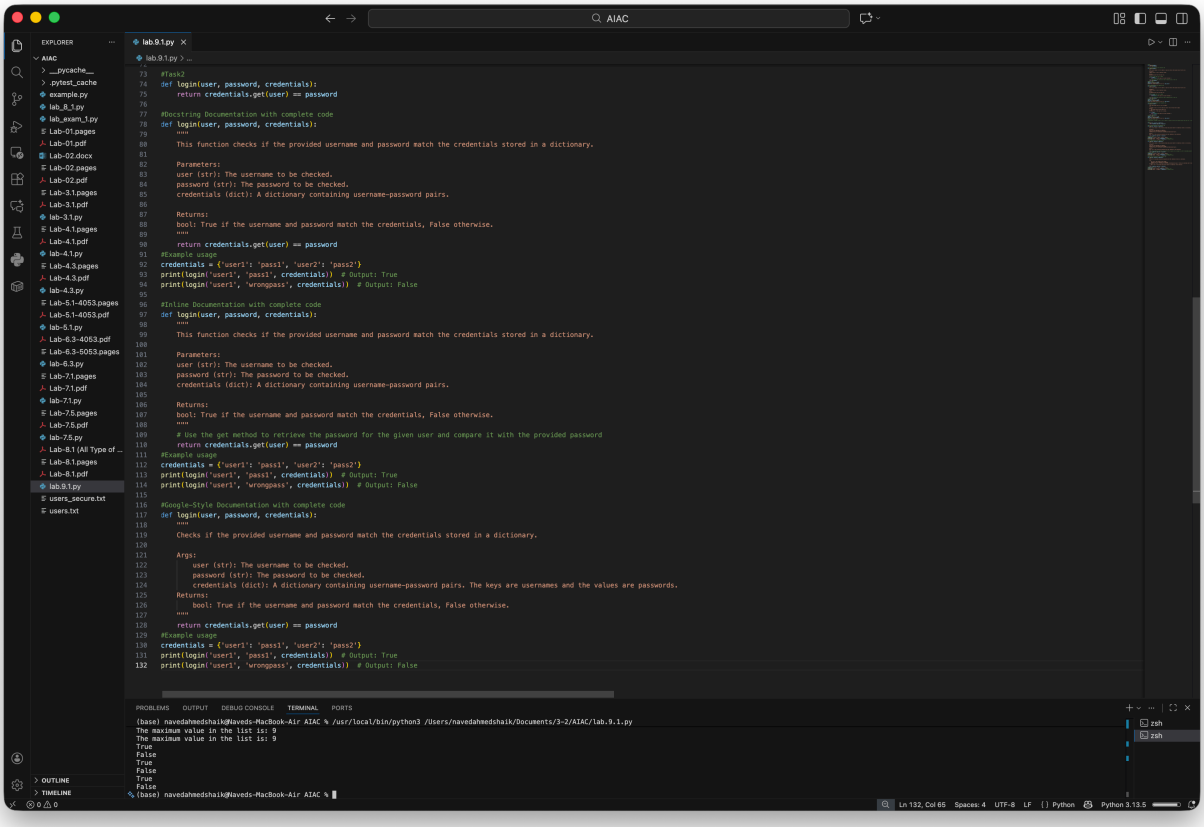
The function `find_max` takes a list of numbers as input and returns the maximum value from that list. It first checks if the input list is empty and raises a ValueError if it is. If the list is not empty, it uses the built-in `max` function to find and return the maximum value. The function is documented using three different styles: docstring documentation, inline documentation, and Google-style documentation, providing clear explanations of the function's purpose, parameters, return value, and potential exceptions.

Style	Advantages	Disadvantages	Suitable Use Cases
Docstring (standard)	Simple, readable, built into Python help tools	Less structured than formal styles	Small scripts, educational code
Inline comments	Explain logic step-by-step	Can clutter code, not accessible via help()	Complex algorithms needing explanation
Google-style	Highly structured, professional, IDE-friendly, good for teams	Slightly longer to write	Libraries, APIs, production code

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.

SCREENSHOT OF GENERATED CODE:



EXPLANATION:

The function `login` takes three parameters: `user`, `password`, and `credentials`. It checks if the provided username and password match the credentials stored in a dictionary. The function uses the `get` method of the dictionary to retrieve the password associated with the given username and compares it with the provided password. If they match, it returns `True`; otherwise, it returns `False`. The function is documented using three different styles: docstring

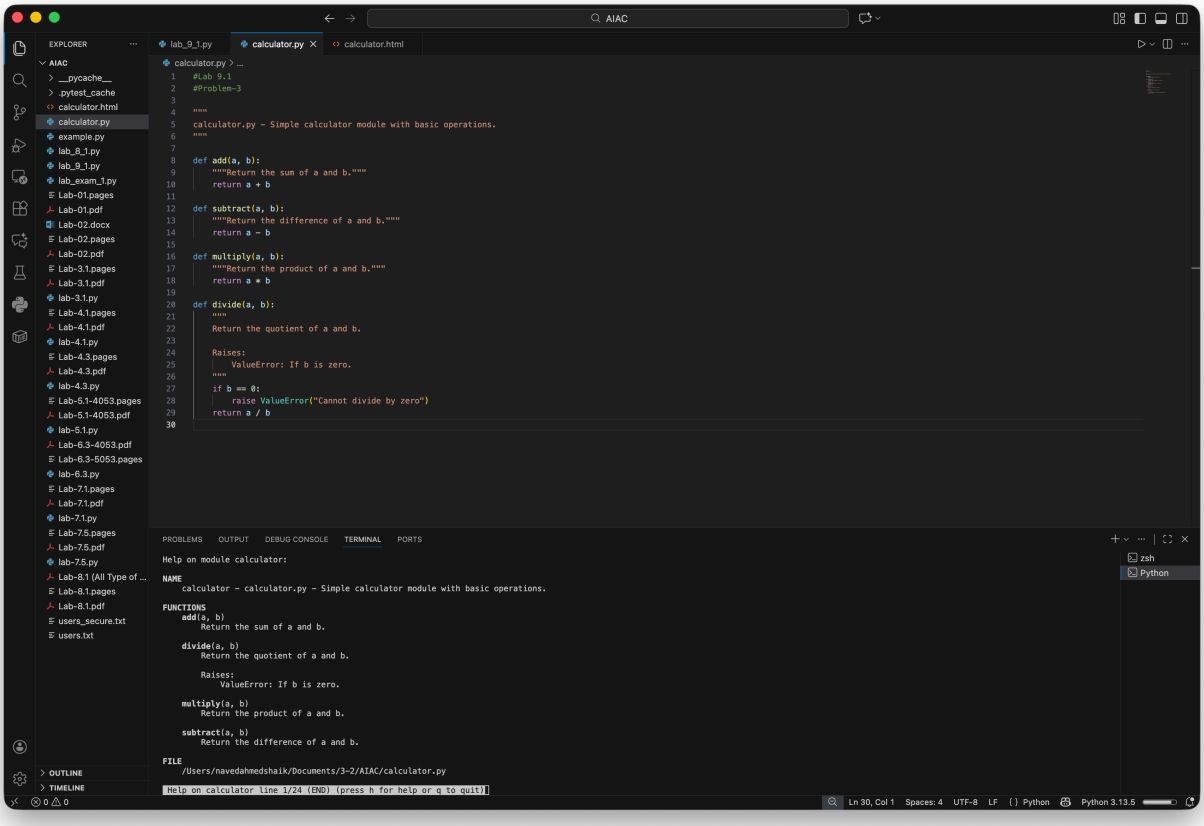
documentation, inline documentation, and Google-style documentation, providing clear explanations of the function's purpose, parameters, return value, and potential exceptions.

Style	Advantages	Disadvantages	Best Use Cases
Docstring (standard)	Simple, readable, built into Python help tools	Less structured for large projects	Small scripts, student projects
Inline comments	Explain logic directly near code	Not visible via help(), can clutter code	Complex internal logic explanation
Google-style	Structured, professional, IDE-friendly, great for teams	Slightly longer to write	Libraries, APIs, collaborative projects

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:
o add(a, b) – returns the sum of two numbers
o subtract(a, b) – returns the difference of two numbers
o multiply(a, b) – returns the product of two numbers
o 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.

SCREENSHOT OF GENERATED CODE:



index
calculator /Users/navedahmedshaik/Documents/3-2/AIAC/calculator.py

calculator.py - Simple calculator module with basic operations.

Functions

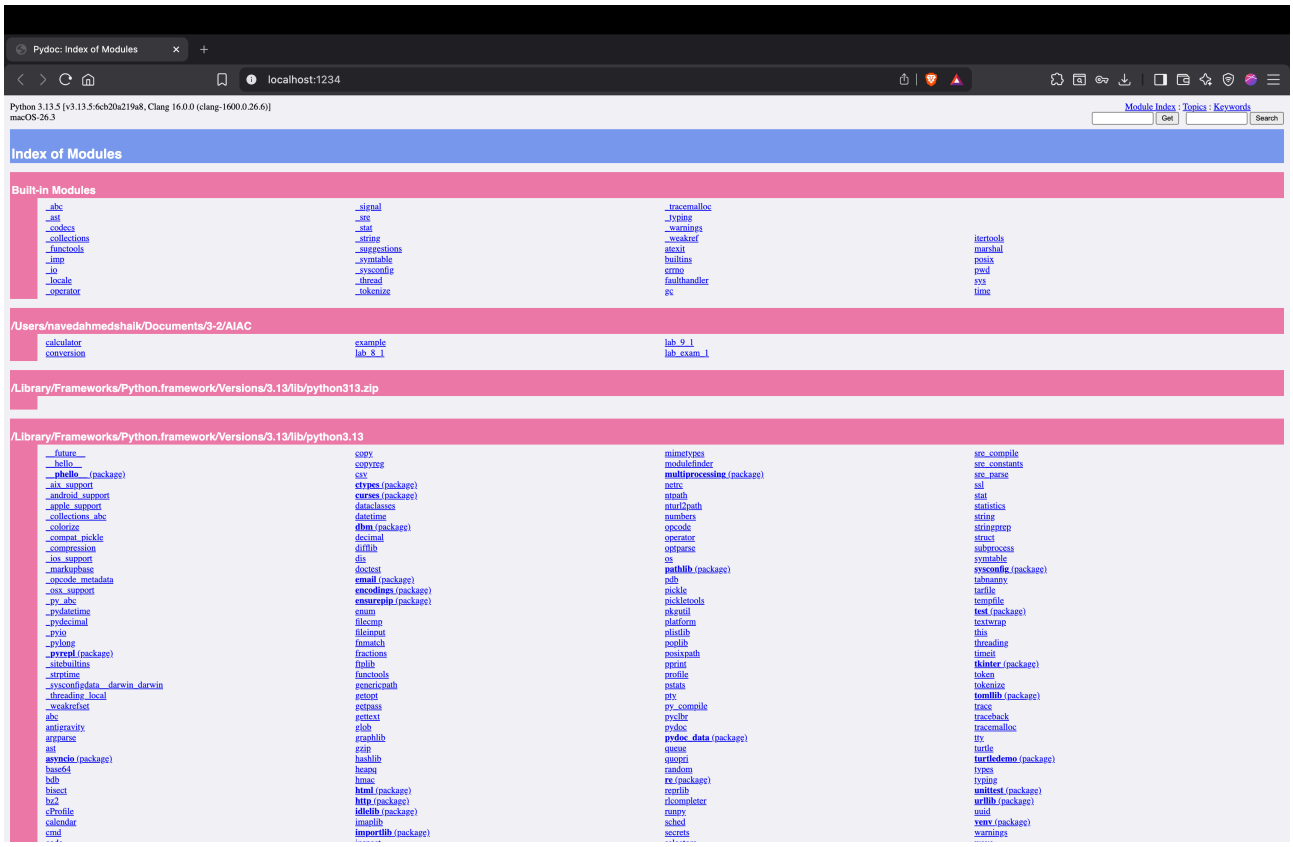
add(a, b)
Return the sum of a and b.

divide(a, b)
Return the quotient of a and b.

Raises:
ValueError: If b is zero.

multiply(a, b)
Return the product of a and b.

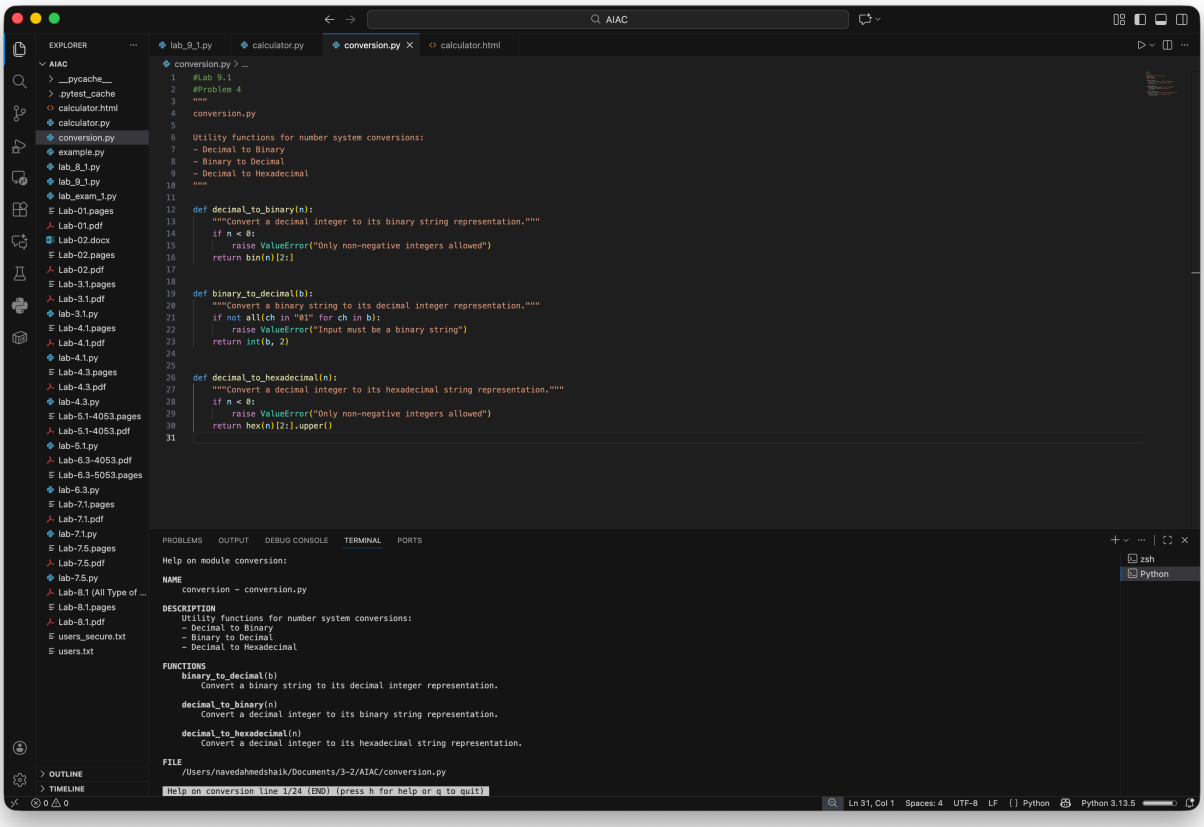
subtract(a, b)
Return the difference of a and b.

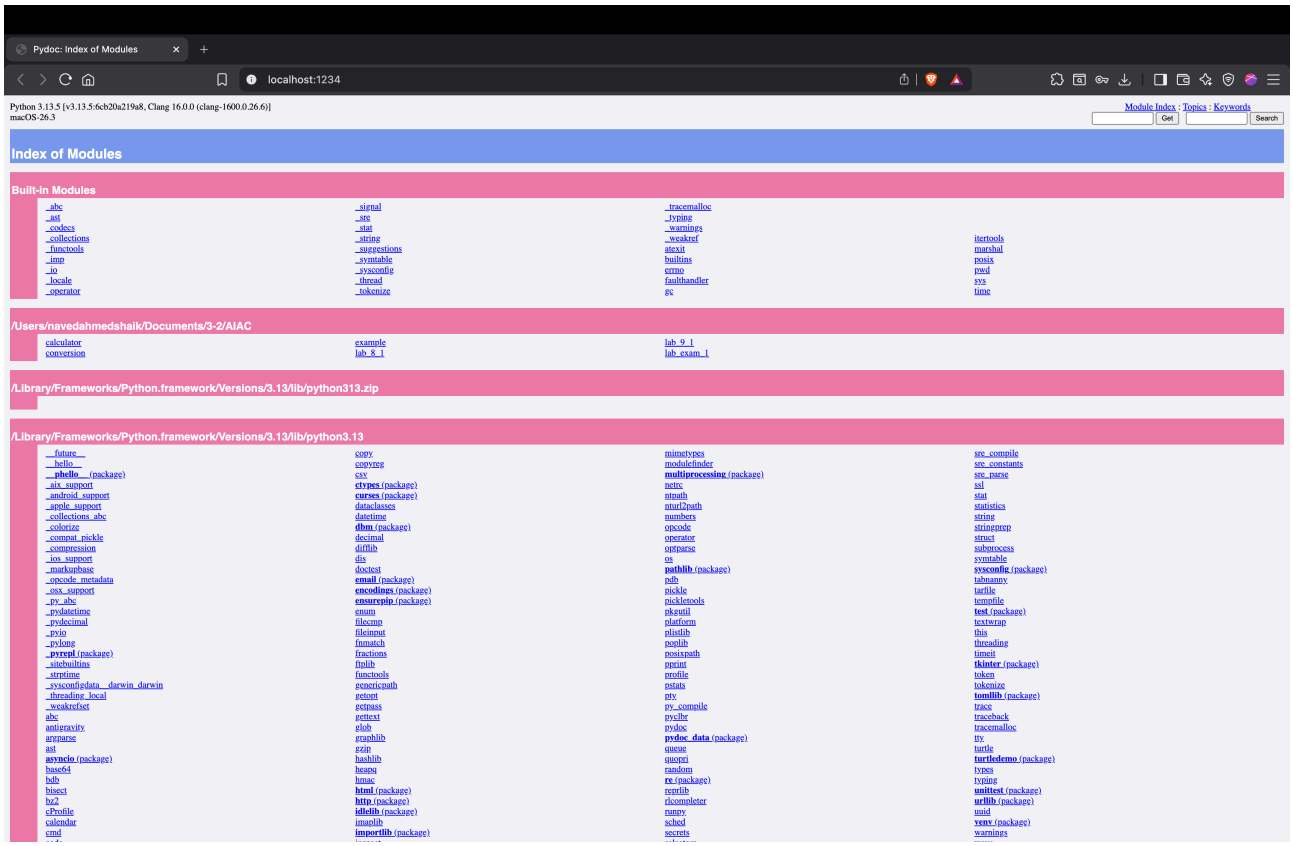


Problem 4: Conversion Utilities Module

Task:
1. Write a module named conversion.py with functions:
o decimal_to_binary(n)
o binary_to_decimal(b)
o 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.

SCREENSHOT OF GENERATED CODE:





index

conversion /Users/navedahmedshaik/Documents/3-2/AIAC/conversion.py

conversion.py

Utility functions for number system conversions:
- Decimal to Binary
- Binary to Decimal
- Decimal to Hexadecimal

Functions

binary_to_decimal(b)
Convert a binary string to its decimal integer representation.

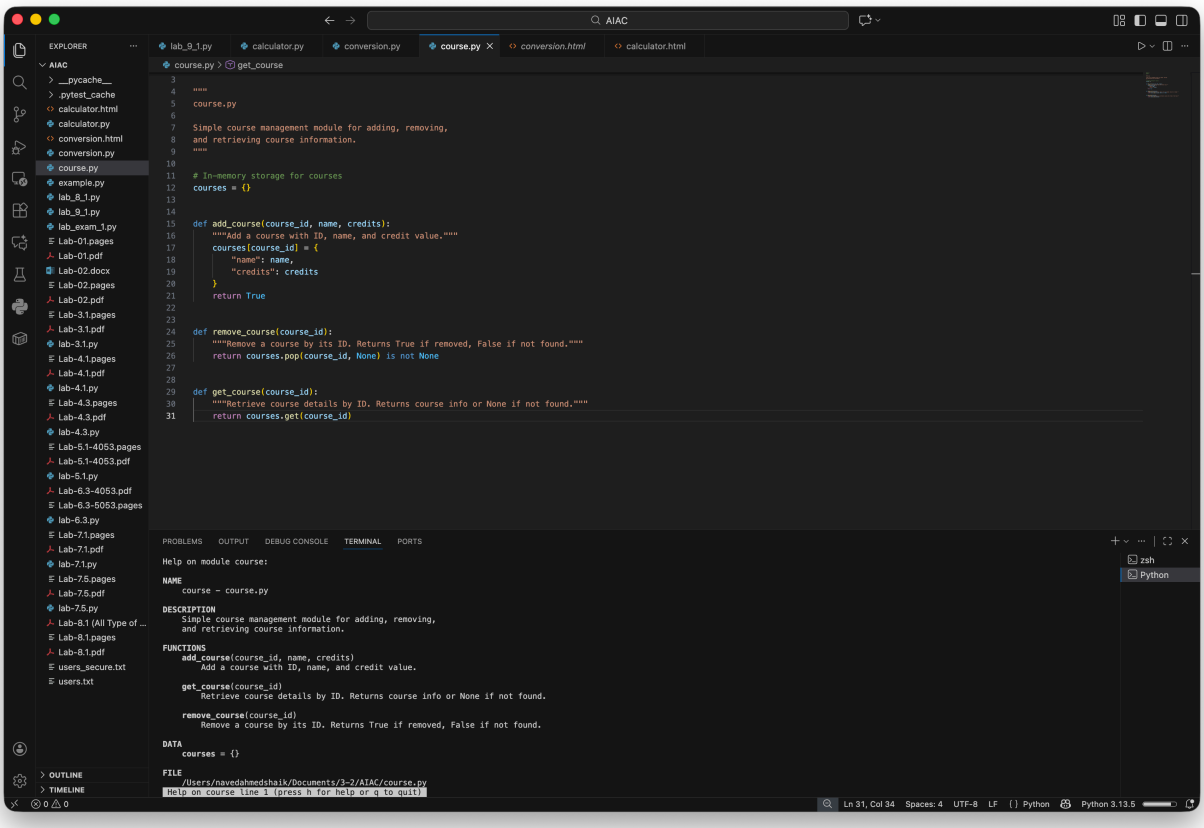
decimal_to_binary(n)
Convert a decimal integer to its binary string representation.

decimal_to_hexadecimal(n)
Convert a decimal integer to its hexadecimal string representation.

Problem 5 – Course Management Module

Task:
1. Create a module course.py with functions:
o add_course(course_id, name, credits)
o remove_course(course_id)
o 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.

SCREENSHOT OF GENERATED CODE:



index

course /Users/navedahmedshaik/Documents/3-2/AIAC/course.py

course.py

Simple course management module for adding, removing,
and retrieving course information.

Functions

add_course(course_id, name, credits)
Add a course with ID, name, and credit value.

get_course(course_id)
Retrieve course details by ID. Returns course info or None if not found.

remove_course(course_id)
Remove a course by its ID. Returns True if removed, False if not found.

Data

courses = {}