ASSIGNMENT 9.1

Name: D.Sandeep

Ht.No: 2303A51665

Batch: 23

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,

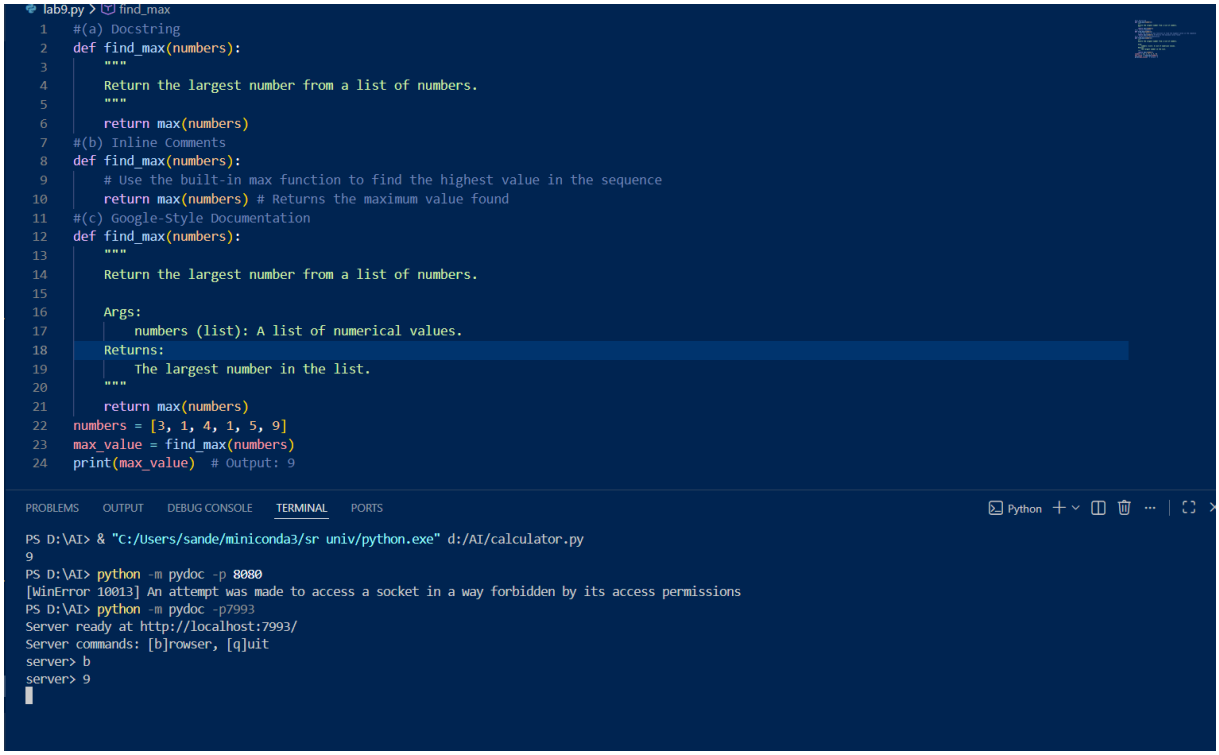
disadvantages, and suitable use cases of each

style.

• Recommend which documentation style is most effective

for a mathematical utilities library and justify your

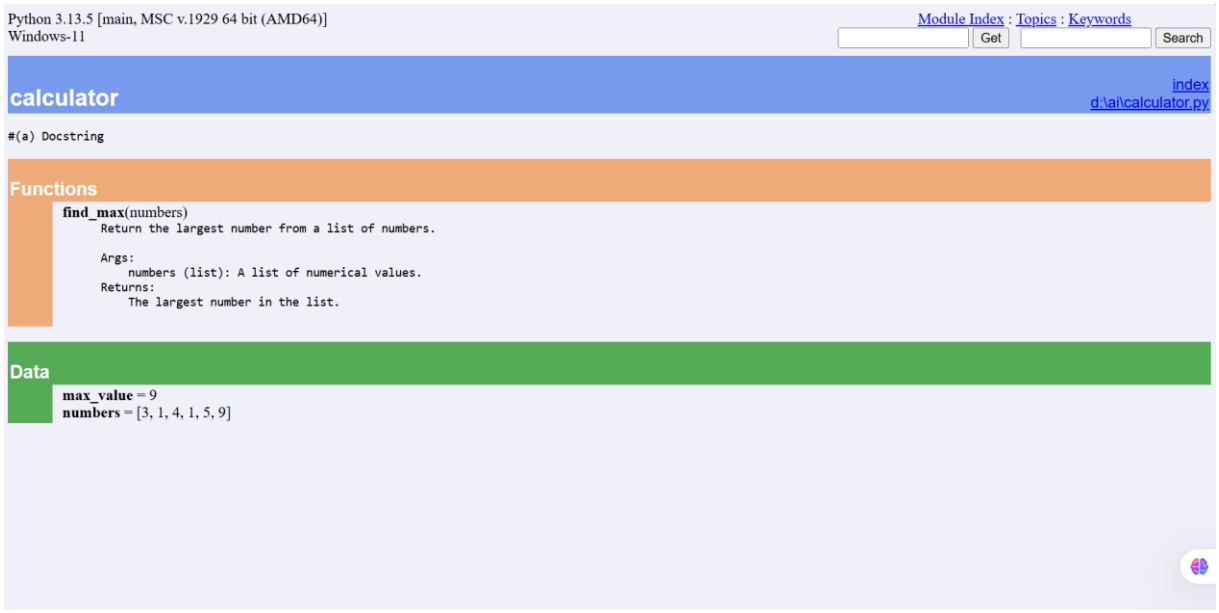
answer.



```python
#(a) Docstring
def find_max(numbers):
    """
    Return the largest number from a list of numbers.
    """
    return max(numbers)
#(b) Inline Comments
def find_max(numbers):
    # Use the built-in max function to find the highest value in the sequence
    return max(numbers) # Returns the maximum value found
#(c) Google-Style Documentation
def find_max(numbers):
    """
    Return the largest number from a list of numbers.

    Args:
        numbers (list): A list of numerical values.
    Returns:
        The largest number in the list.
    """
    return max(numbers)
numbers = [3, 1, 4, 1, 5, 9]
max_value = find_max(numbers)
print(max_value)  # Output: 9
```

```
PS D:\AI> & "C:/Users/sande/miniconda3/sr univ/python.exe" d:/AI/calculator.py
9
PS D:\AI> python -m pydoc -p 8080
[WinError 10013] An attempt was made to access a socket in a way forbidden by its access permissions
PS D:\AI> python -m pydoc -p7993
Server ready at http://localhost:7993/
Server commands: [b]rowser, [q]uit
server> b
server> 9
```



Python 3.13.5 [main, MSC v.1929 64 bit (AMD64)]
Windows-11

Module Index : Topics : Keywords
Get Search

calculator

index
d:\ai\calculator.py

#(a) Docstring

Functions

find_max(numbers)
Return the largest number from a list of numbers.

Args:
numbers (list): A list of numerical values.
Returns:
The largest number in the list.

Data

max_value = 9
numbers = [3, 1, 4, 1, 5, 9]

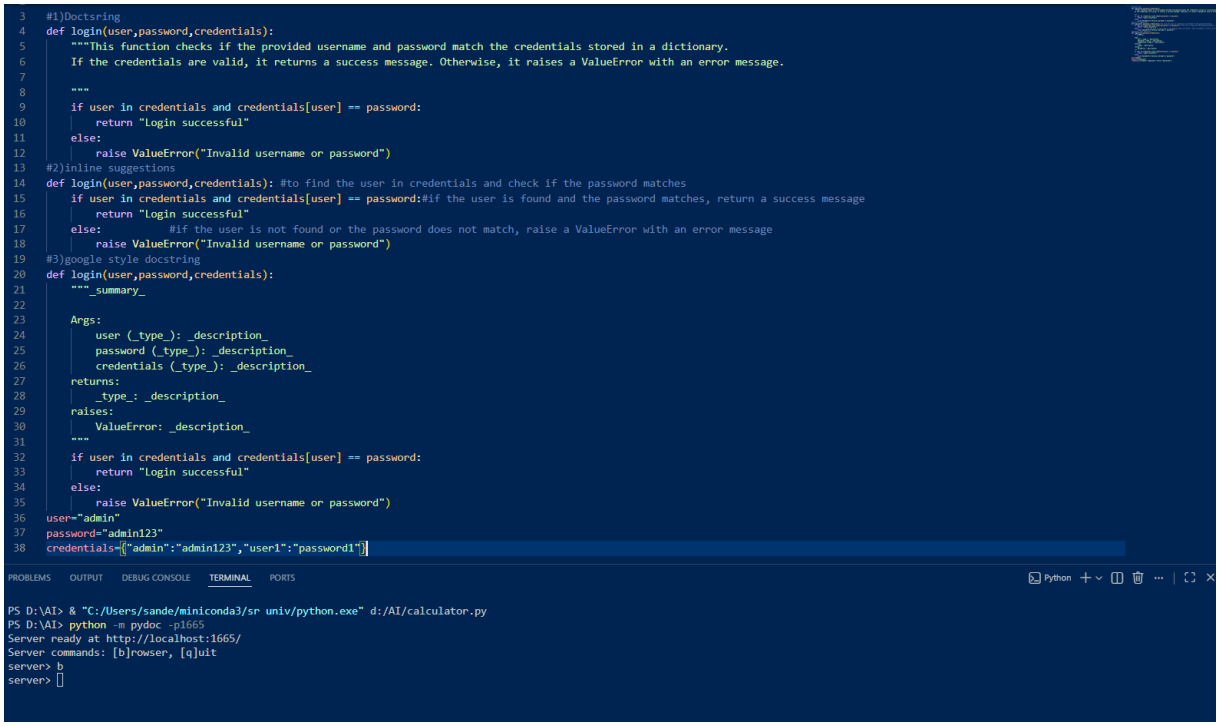
Problem 2: Consider the following Python function:

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.



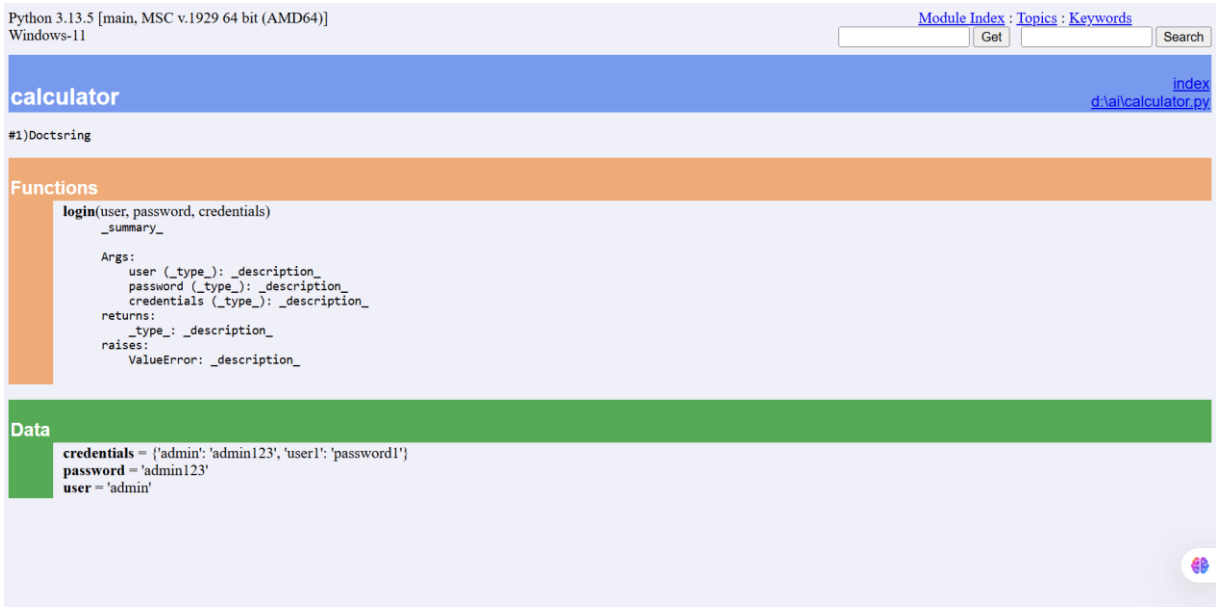
```python
#1)Doctsring
def login(user,password,credentials):
    """This function checks if the provided username and password match the credentials stored in a dictionary.
    If the credentials are valid, it returns a success message. Otherwise, it raises a ValueError with an error message.

    """
    if user in credentials and credentials[user] == password:
        return "Login successful"
    else:
        raise ValueError("Invalid username or password")
#2)inline suggestions
def login(user,password,credentials): #to find the user in credentials and check if the password matches
    if user in credentials and credentials[user] == password:#if the user is found and the password matches, return a success message
        return "Login successful"
    else:          #if the user is not found or the password does not match, raise a ValueError with an error message
        raise ValueError("Invalid username or password")
#3)google style docstring
def login(user,password,credentials):
    """_summary_

    Args:
        user (_type_): _description_
        password (_type_): _description_
        credentials (_type_): _description_
    returns:
        _type_: _description_
    raises:
        ValueError: _description_
    """
    if user in credentials and credentials[user] == password:
        return "Login successful"
    else:
        raise ValueError("Invalid username or password")
user="admin"
password="admin123"
credentials={"admin":"admin123","user1":"password1"}
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS D:\AI> & "C:/Users/sande/miniconda3/sr univ/python.exe" d:/AI/calculator.py
PS D:\AI> python -m pydoc -p1665
Server ready at http://localhost:1665/
Server commands: [b]rowser, [q]uit
server> b
server>
```



Python 3.13.5 [main, MSC v.1929 64 bit (AMD64)]
Windows-11

Module Index : Topics : Keywords
Get Search

calculator

index
d:\ai\calculator.py

#1)Doctsring

Functions

login(user, password, credentials)
    summary

    Args:
        user (_type_): _description_
        password (_type_): _description_
        credentials (_type_): _description_
    returns:
        type: _description_
    raises:
        ValueError: _description_

Data

credentials = {'admin': 'admin123', 'user1': 'password1'}
password = 'admin123'
user = 'admin'

Problem 3: Calculator (Automatic Documentation Generation)

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

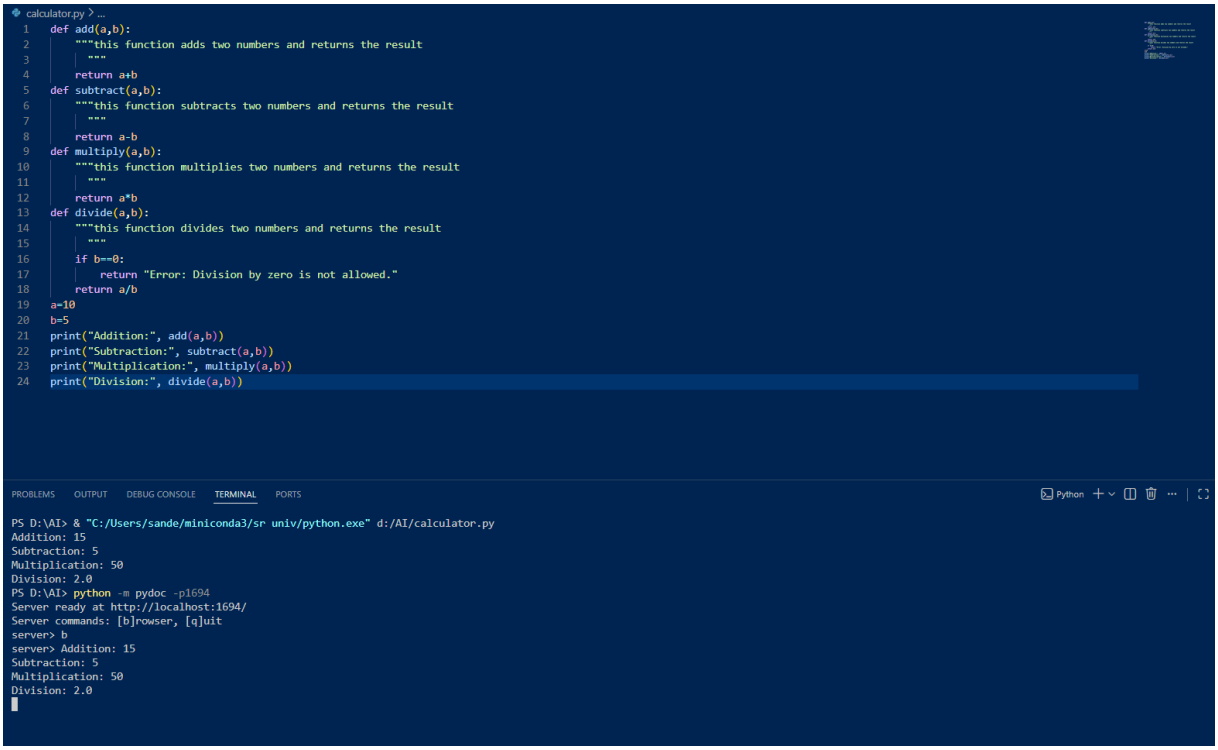
Instructions:

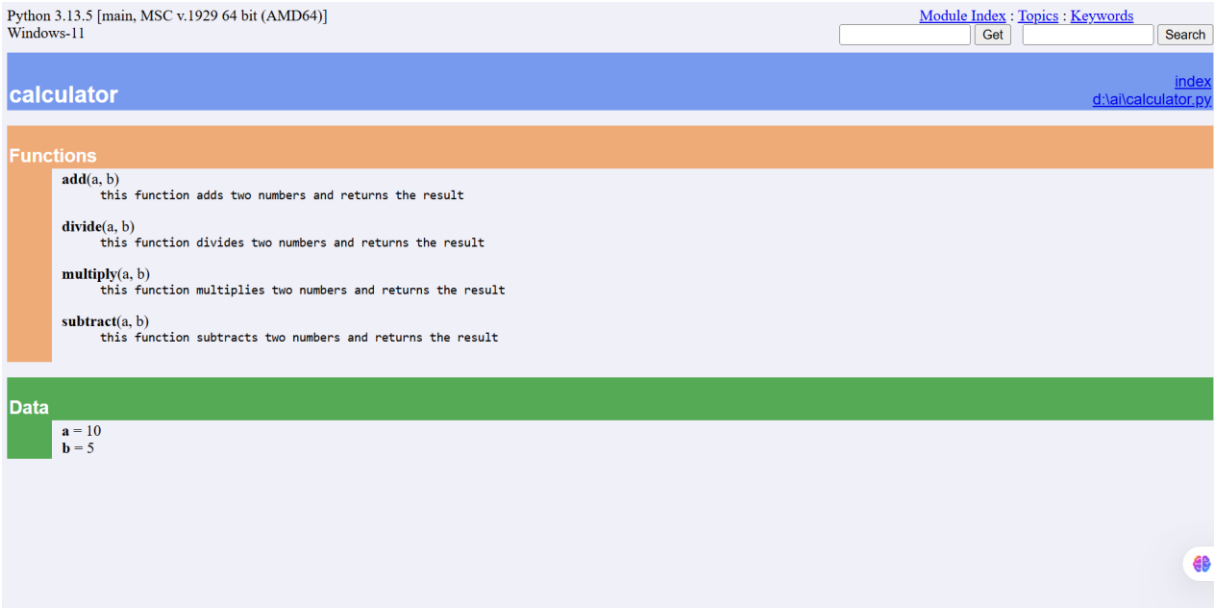
1. Create a Python module calculator.py that includes the following functions, each written with appropriate docstrings:

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.





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

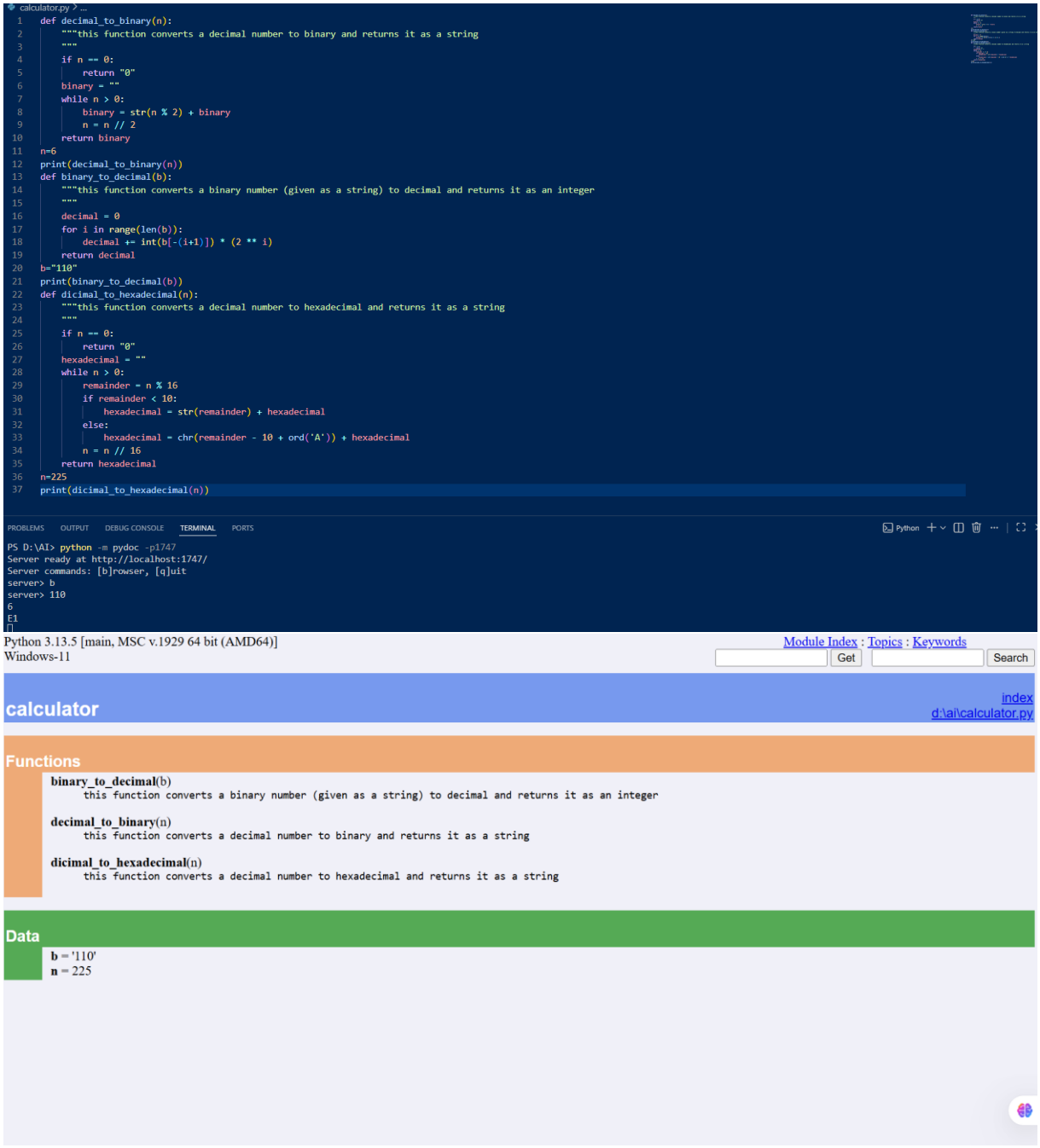
decimal_to_hexadecimal(n)

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.



Problem 5 – Course Management Module

Task:

1. Create a module course.py with functions:

o add_course(course_id, name, credits)

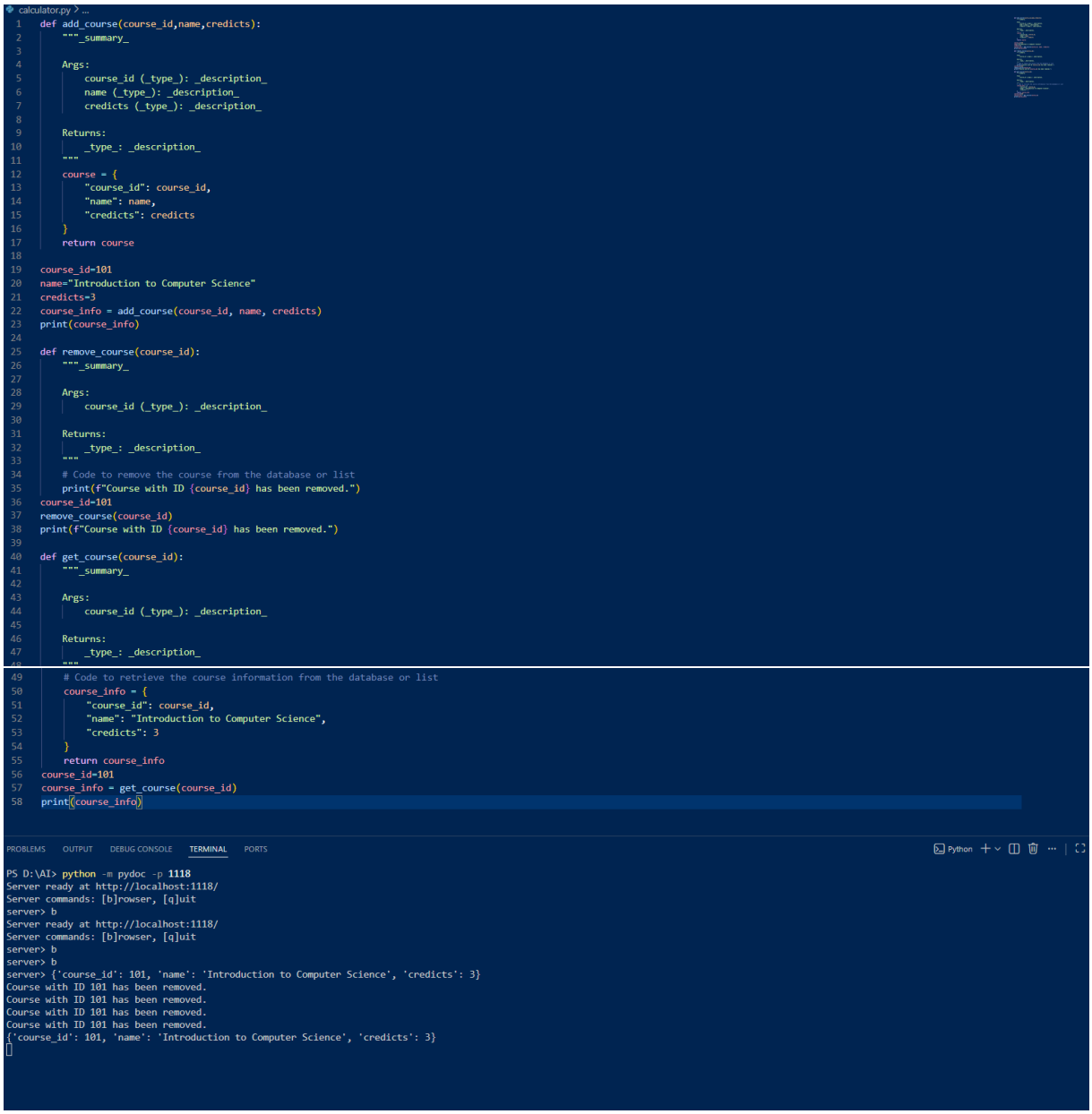
o remove_course(course_id) o

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.



calculator

index
d:\ai\calculator.py

Functions

add_course(course_id, name, credicts)

```
    _summary_

    Args:
        course_id (_type_): _description_
        name (_type_): _description_
        credicts (_type_): _description_

    Returns:
        _type_: _description_
```

get_course(course_id)

```
    _summary_

    Args:
        course_id (_type_): _description_

    Returns:
        _type_: _description_
```

remove_course(course_id)

```
    _summary_

    Args:
        course_id (_type_): _description_

    Returns:
        _type_: _description_
        _type_: _description_
```

Data

course_id = 101
course_info = {'course_id': 101, 'credicts': 3, 'name': 'Introduction to Computer Science'}
credicts = 3
name = 'Introduction to Computer Science'