Assignment - 9.1

Ht. No: 2303A51923

Name: V.Sravani

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:

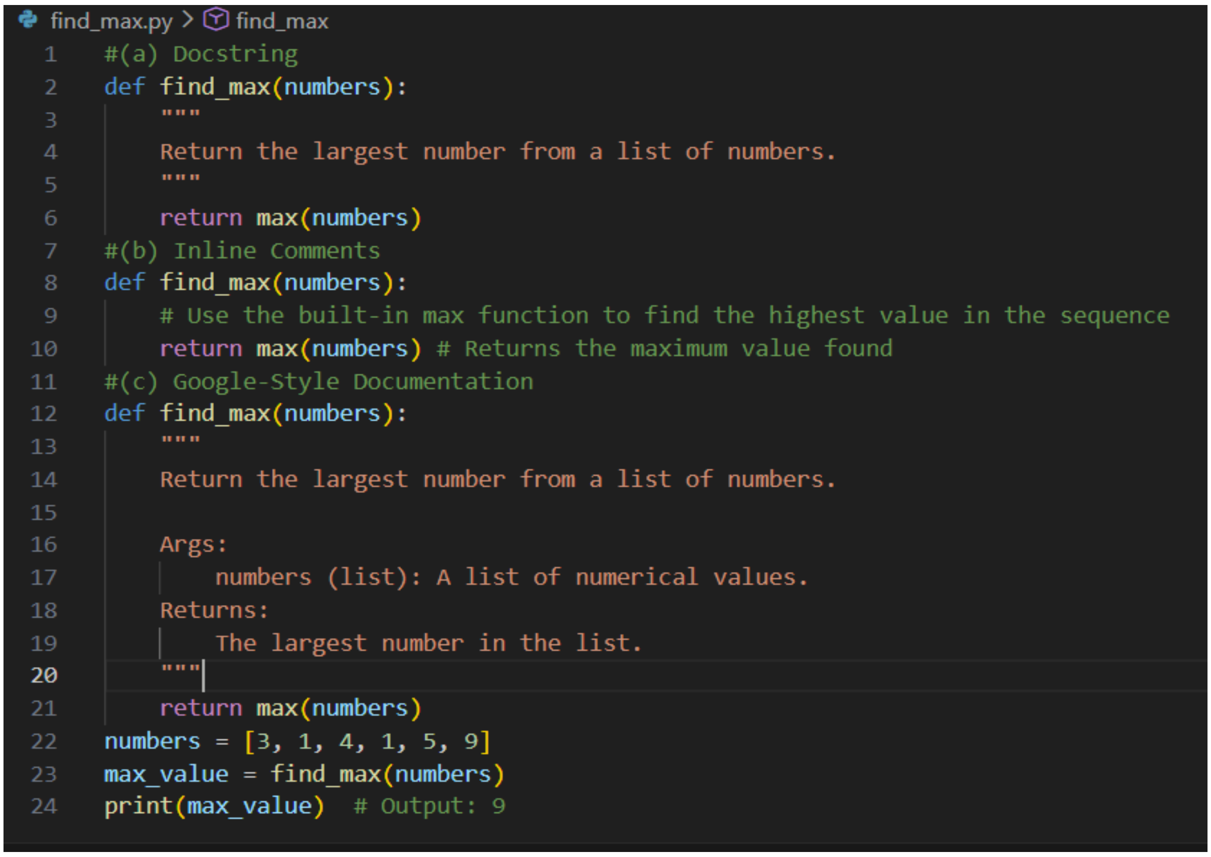
(a) Docstring

(b) Inline comments

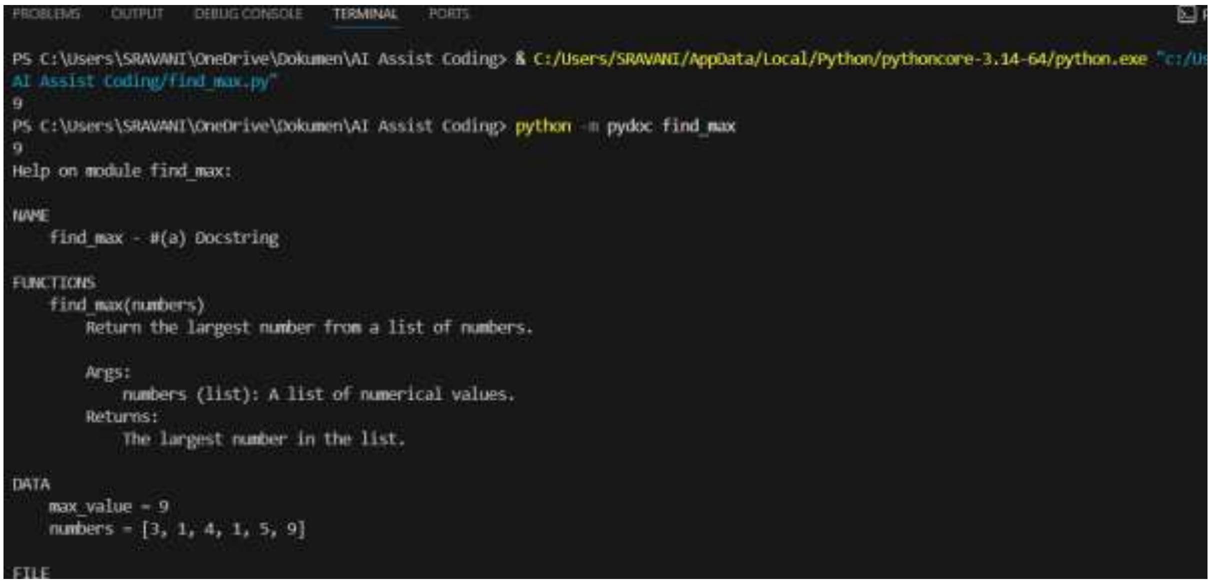
(c) Google-style documentation

• 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.



```
find_max.py > find_max
#(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 C:\Users\SRAVANI\OneDrive\Dokumen\AI Assist Coding> & C:/Users/SRAVANI/AppData/Local/Python/pythoncore-3.14-64/python.exe "c:/Us
AI Assist Coding/find_max.py"
9
PS C:\Users\SRAVANI\OneDrive\Dokumen\AI Assist Coding> python -m pydoc find_max
9
Help on module find_max:

NAME
    find_max - #(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]

FILE
```

Problem 2: Consider the following Python function:

def login(user, password, credentials):

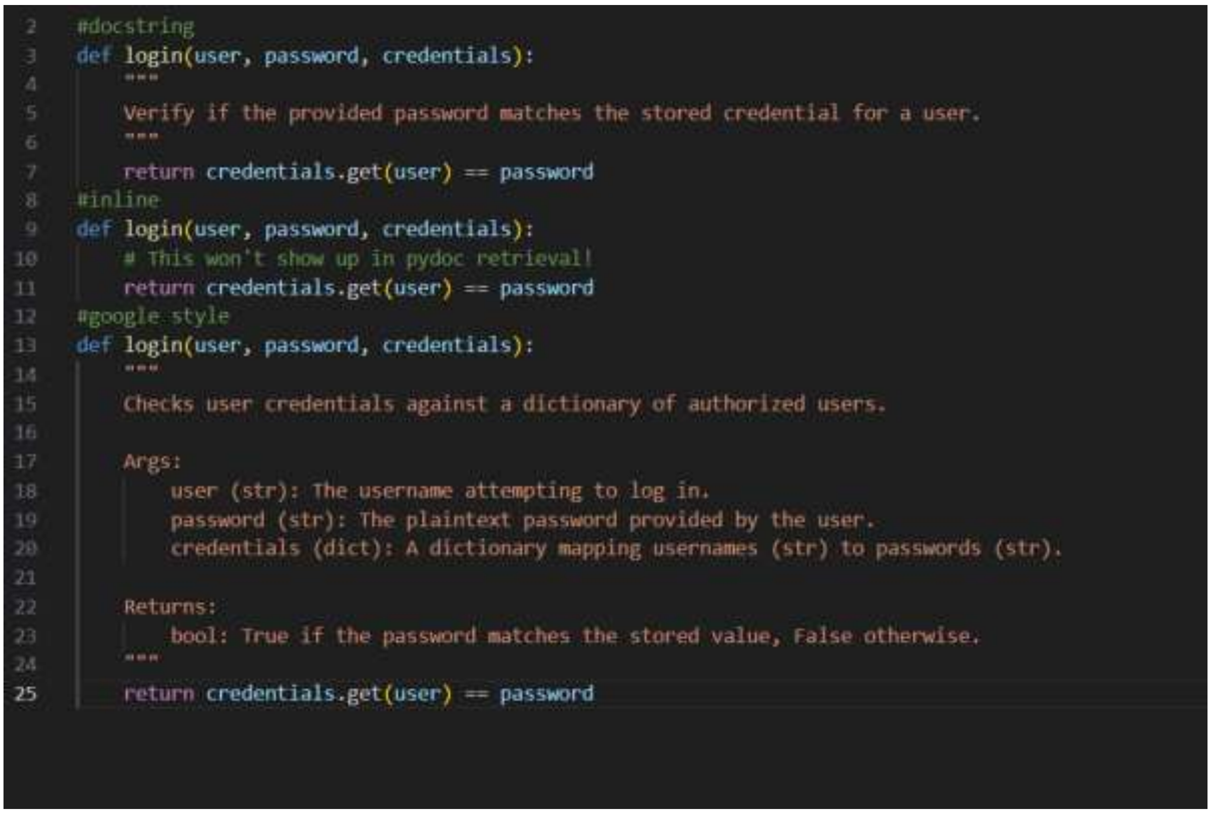
return credentials.get(user) == password

Task:

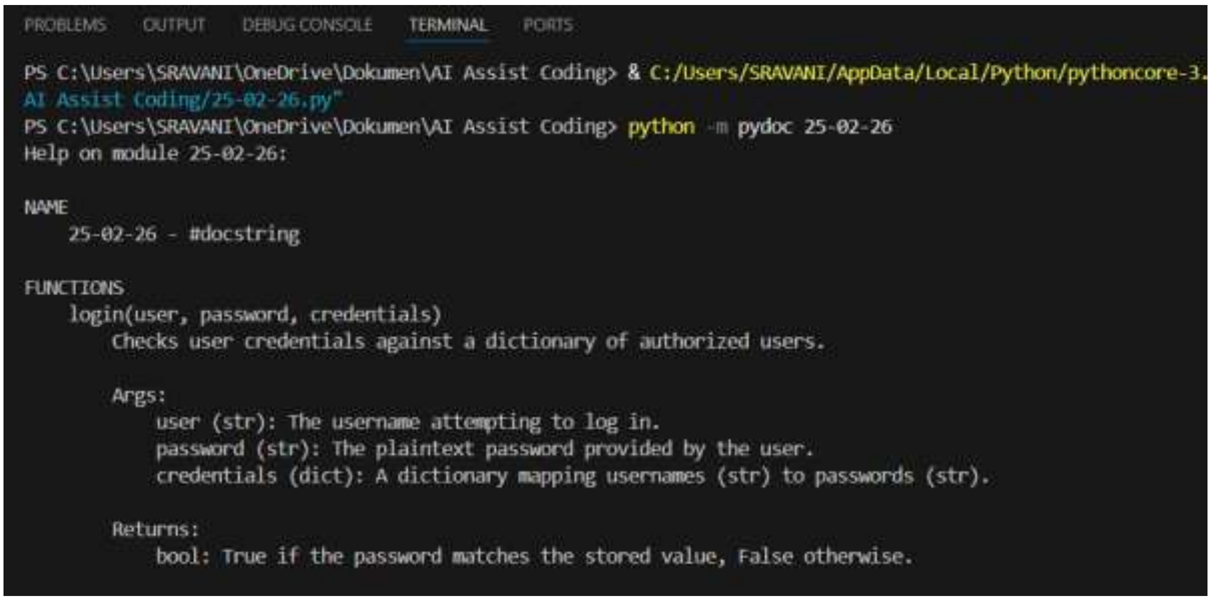
1. Write documentation in all three formats.

2. Critically compare the approaches.

3. Recommend which style would be most helpful for new developers onboarding a project, and justify your choice.



```
#docstring
def login(user, password, credentials):
    """
    Verify if the provided password matches the stored credential for a user.
    """
    return credentials.get(user) == password
#inline
def login(user, password, credentials):
    # This won't show up in pydoc retrieval!
    return credentials.get(user) == password
#google style
def login(user, password, credentials):
    """
    Checks user credentials against a dictionary of authorized users.

    Args:
        user (str): The username attempting to log in.
        password (str): The plaintext password provided by the user.
        credentials (dict): A dictionary mapping usernames (str) to passwords (str).

    Returns:
        bool: True if the password matches the stored value, False otherwise.
    """
    return credentials.get(user) == password
```



```
PS C:\Users\SRAVANI\OneDrive\Dokumen\AI Assist Coding> & C:/Users/SRAVANI/AppData/Local/Python/pythoncore-3.
AI Assist Coding/25-02-26.py"
PS C:\Users\SRAVANI\OneDrive\Dokumen\AI Assist Coding> python -m pydoc 25-02-26
Help on module 25-02-26:

NAME
    25-02-26 - #docstring

FUNCTIONS
    login(user, password, credentials)
        Checks user credentials against a dictionary of authorized users.

        Args:
            user (str): The username attempting to log in.
            password (str): The plaintext password provided by the user.
            credentials (dict): A dictionary mapping usernames (str) to passwords (str).

        Returns:
            bool: True if the password matches the stored value, False otherwise.
```

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

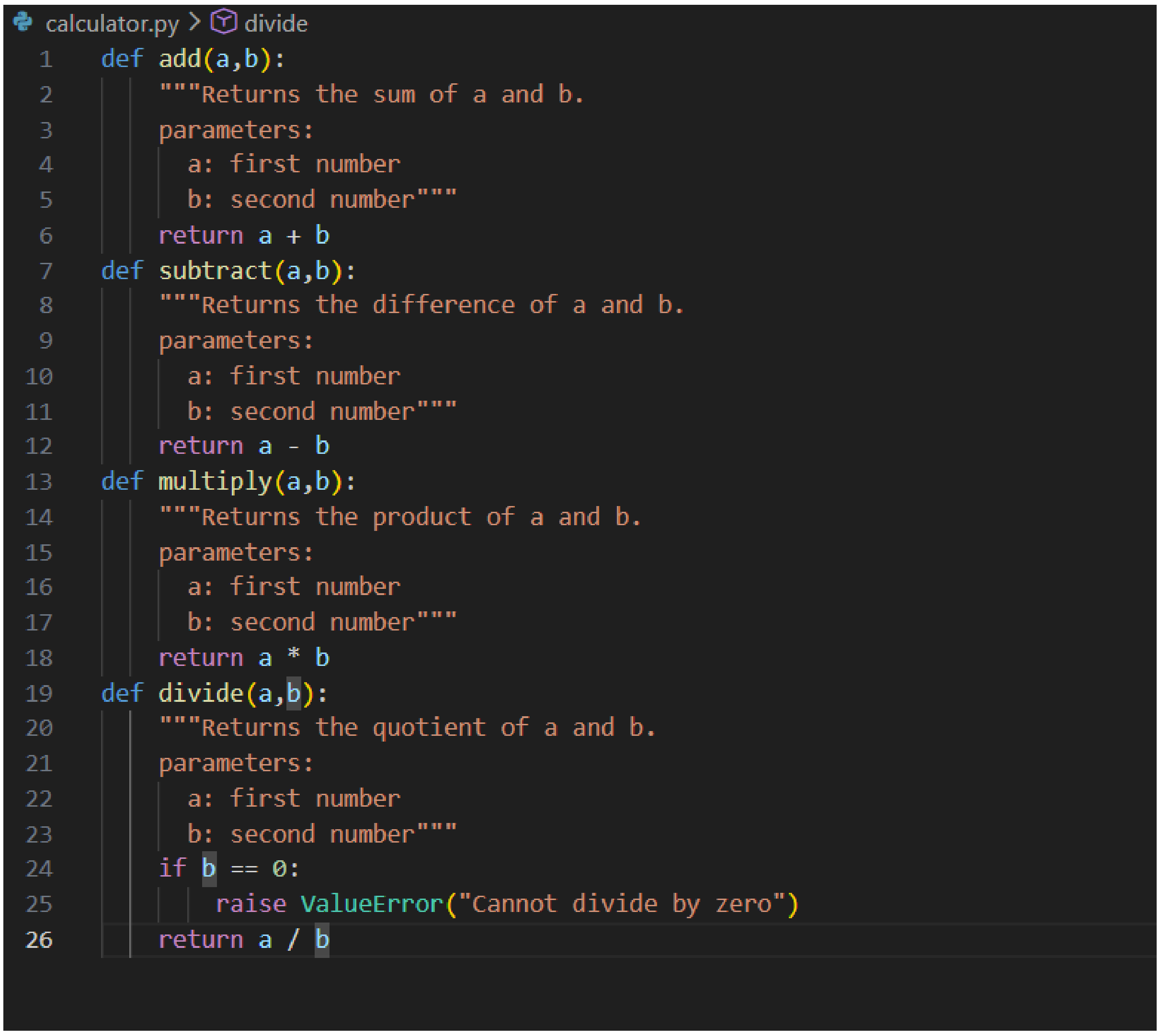
o subtract(a, b) – returns the difference of two numbers

o multiply(a, b) – returns the product of two numbers

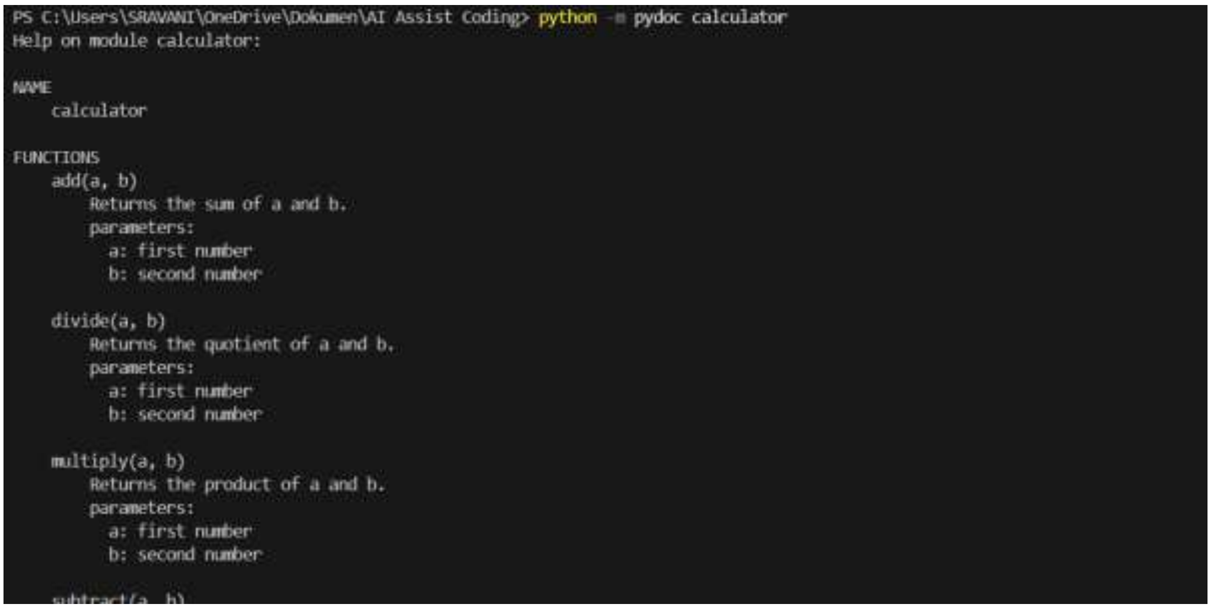
o divide(a, b) – returns the quotient of two numbers

2. Display the module documentation in the terminal using Python's documentation tools.

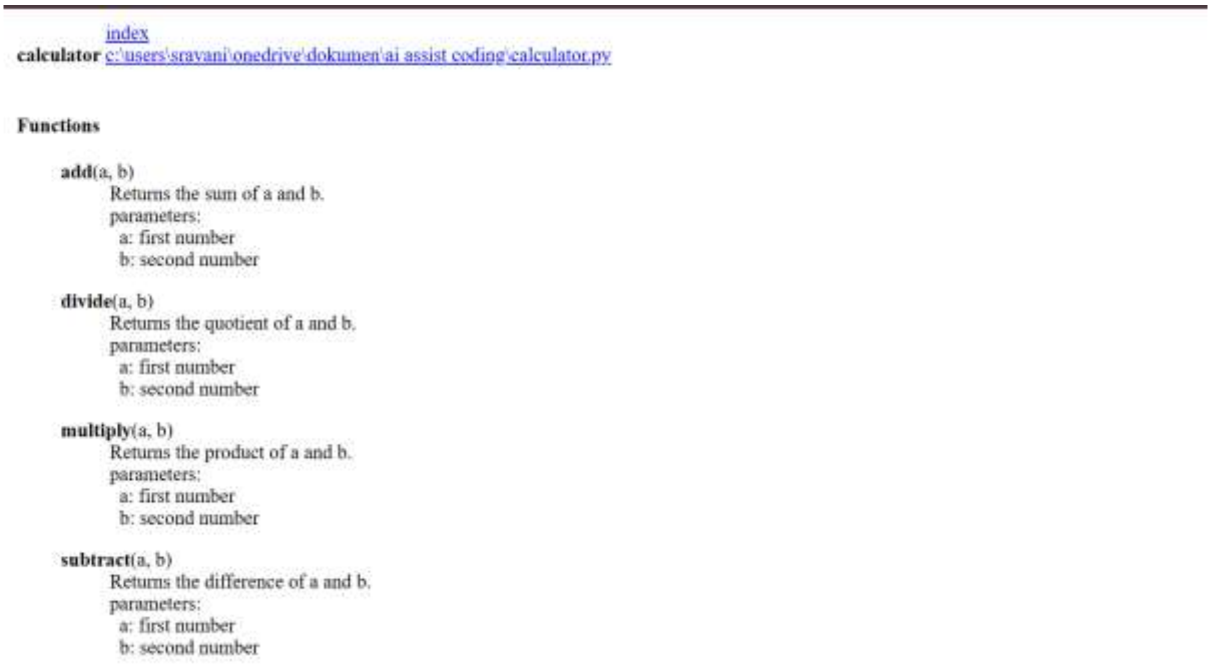
3. Generate and export the module documentation in HTML format using the pydoc utility, and open the generated HTML file in a web browser to verify the output.



```python
def add(a,b):
    """Returns the sum of a and b.
    parameters:
      a: first number
      b: second number"""
    return a + b
def subtract(a,b):
    """Returns the difference of a and b.
    parameters:
      a: first number
      b: second number"""
    return a - b
def multiply(a,b):
    """Returns the product of a and b.
    parameters:
      a: first number
      b: second number"""
    return a * b
def divide(a,b):
    """Returns the quotient of a and b.
    parameters:
      a: first number
      b: second number"""
    if b == 0:
        raise ValueError("Cannot divide by zero")
    return a / b
```



```
PS C:\Users\SRAVANI\OneDrive\Dokumen\AI Assist Coding> python -m pydoc calculator
Help on module calculator:

NAME
    calculator

FUNCTIONS
    add(a, b)
        Returns the sum of a and b.
        parameters:
          a: first number
          b: second number

    divide(a, b)
        Returns the quotient of a and b.
        parameters:
          a: first number
          b: second number

    multiply(a, b)
        Returns the product of a and b.
        parameters:
          a: first number
          b: second number

    subtract(a, b)
```



```
         index
calculator c:\users\sravani\onedrive\dokumen\ai assist coding\calculator.py

Functions

     add(a, b)
          Returns the sum of a and b.
          parameters:
            a: first number
            b: second number

     divide(a, b)
          Returns the quotient of a and b.
          parameters:
            a: first number
            b: second number

     multiply(a, b)
          Returns the product of a and b.
          parameters:
            a: first number
            b: second number

     subtract(a, b)
          Returns the difference of a and b.
          parameters:
            a: first number
            b: second number
```

Problem 4: Conversion Utilities Module

Task:

1. Write a module named conversion.py with functions:

o decimal_to_binary(n)

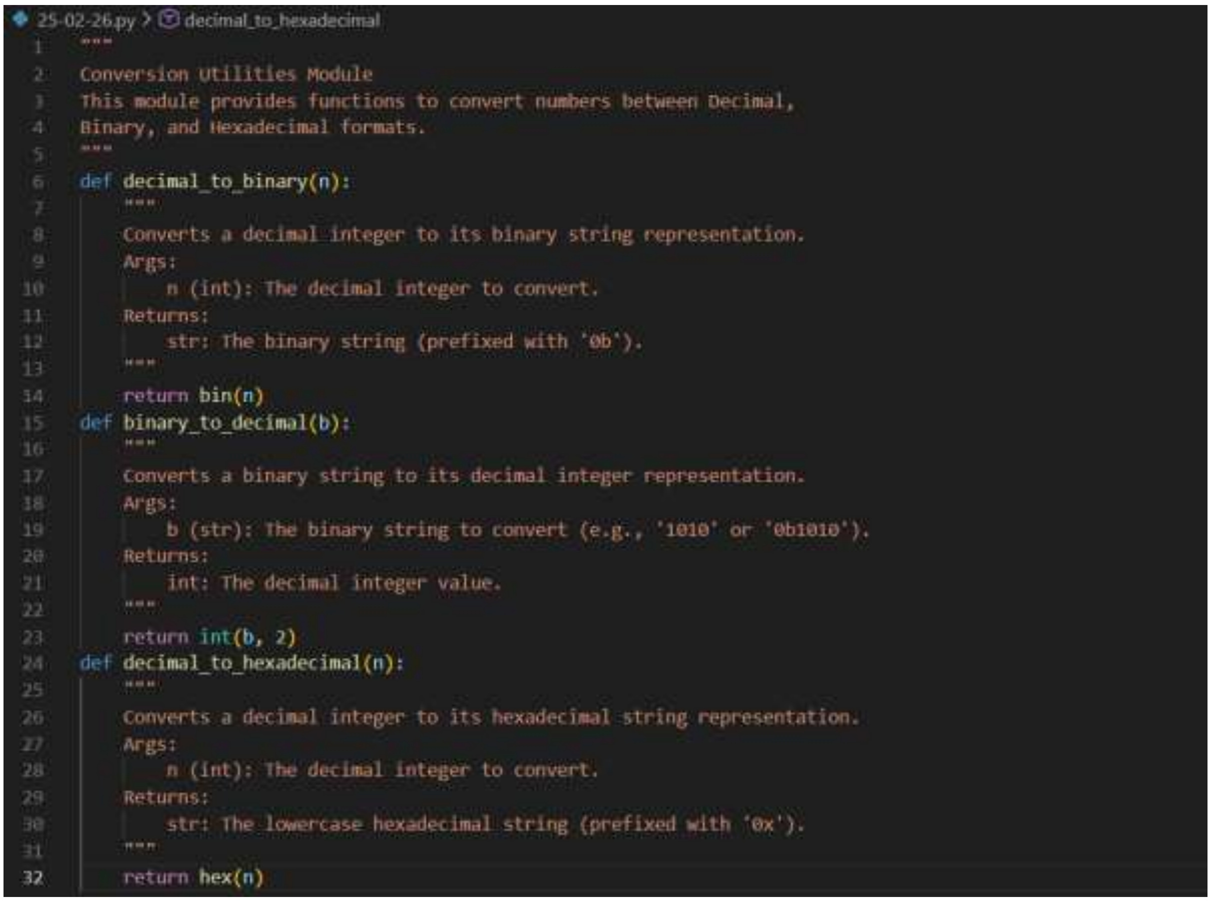
o binary_to_decimal(b)

o decimal_to_hexadecimal(n)

2. Use Copilot for auto-generating docstrings.

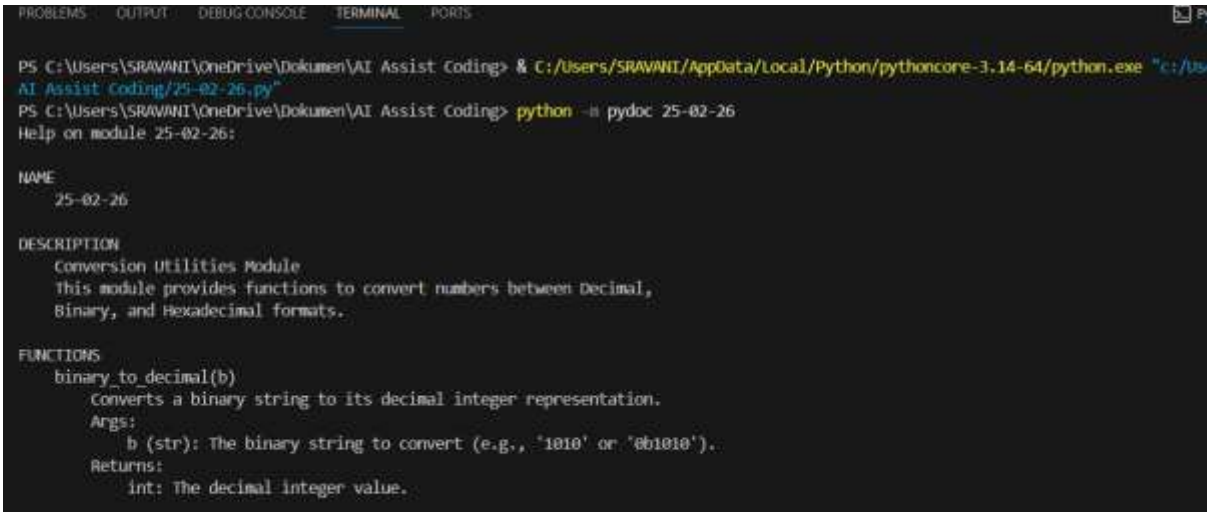
3. Generate documentation in the terminal.

4. Export the documentation in HTML format and open it in a browser.



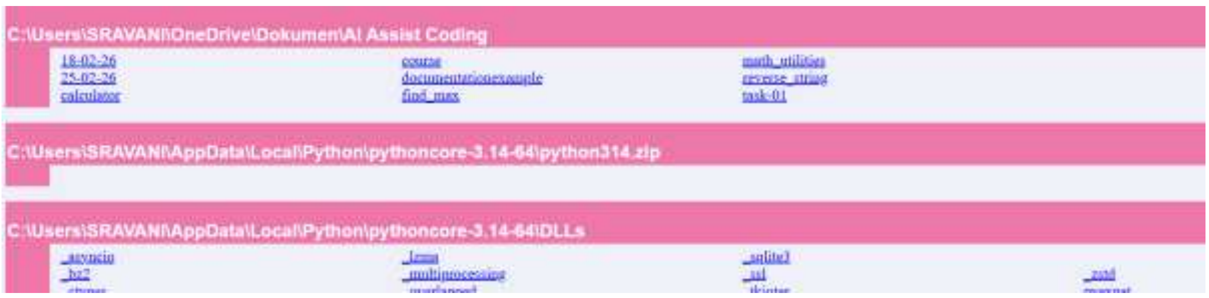
25-02-26.py > decimal_to_hexadecimal

```python
"""
Conversion Utilities Module
This module provides functions to convert numbers between Decimal,
Binary, and Hexadecimal formats.
"""
def decimal_to_binary(n):
    """
    Converts a decimal integer to its binary string representation.
    Args:
        n (int): The decimal integer to convert.
    Returns:
        str: The binary string (prefixed with '0b').
    """
    return bin(n)
def binary_to_decimal(b):
    """
    Converts a binary string to its decimal integer representation.
    Args:
        b (str): The binary string to convert (e.g., '1010' or '0b1010').
    Returns:
        int: The decimal integer value.
    """
    return int(b, 2)
def decimal_to_hexadecimal(n):
    """
    Converts a decimal integer to its hexadecimal string representation.
    Args:
        n (int): The decimal integer to convert.
    Returns:
        str: The lowercase hexadecimal string (prefixed with '0x').
    """
    return hex(n)
```



PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\SRAVANI\OneDrive\Dokumen\AI Assist Coding> & C:/Users/SRAVANI/AppData/Local/Python/pythoncore-3.14-64/python.exe "c:/Us
AI Assist Coding/25-02-26.py"
PS C:\Users\SRAVANI\OneDrive\Dokumen\AI Assist Coding> python -m pydoc 25-02-26
Help on module 25-02-26:

NAME
    25-02-26

DESCRIPTION
    Conversion Utilities Module
    This module provides functions to convert numbers between Decimal,
    Binary, and Hexadecimal formats.

FUNCTIONS
    binary_to_decimal(b)
        Converts a binary string to its decimal integer representation.
        Args:
            b (str): The binary string to convert (e.g., '1010' or '0b1010').
        Returns:
            int: The decimal integer value.
```



```
C:\Users\SRAVANI\OneDrive\Dokumen\AI Assist Coding
    18-02-26                course                      math_utilities
    25-02-26                documentationexample        reverse_string
    calculator              find_max                    task-01

C:\Users\SRAVANI\AppData\Local\Python\pythoncore-3.14-64\python314.zip

C:\Users\SRAVANI\AppData\Local\Python\pythoncore-3.14-64\DLLs
    _asyncio                _lzma                       _sqlite3
    _bz2                    _multiprocessing            _ssl                    _zstd
```

Problem 5 – Course Management Module

Task:

1. Create a module course.py with functions:

o add_course(course_id, name, credits)

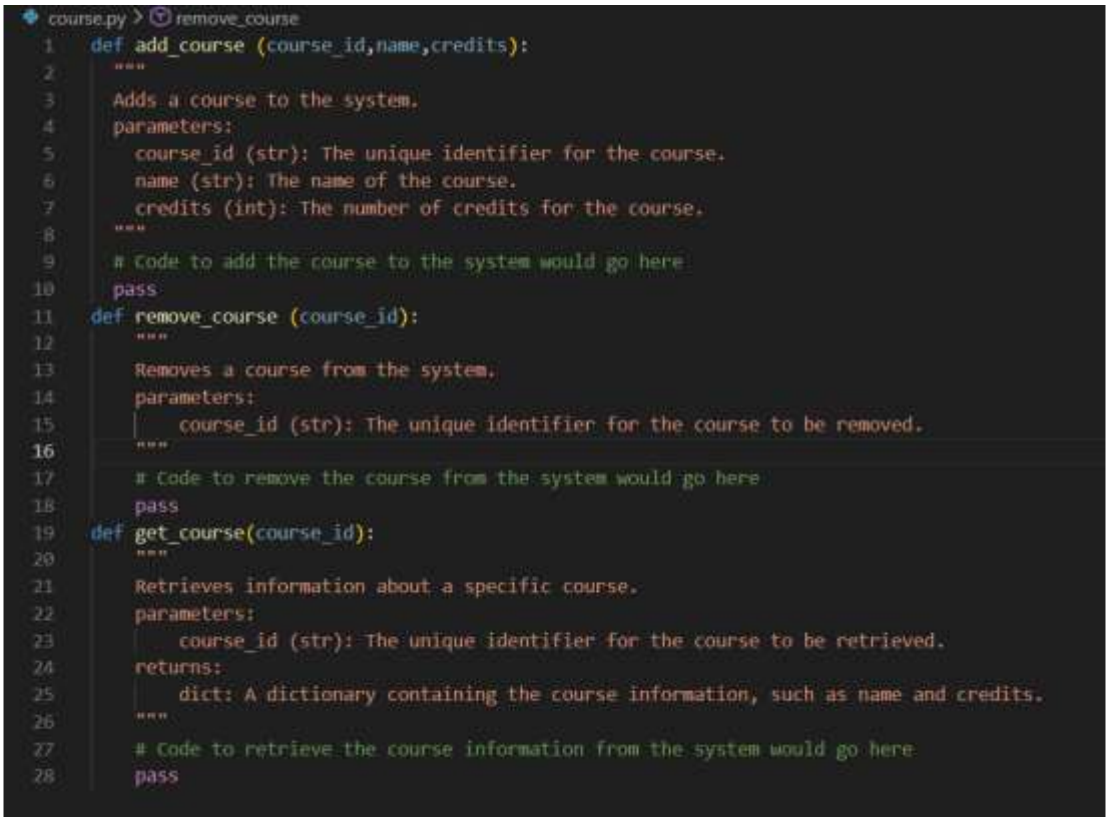
o remove_course(course_id)

o get_course(course_id)

2. Add docstrings with Copilot.

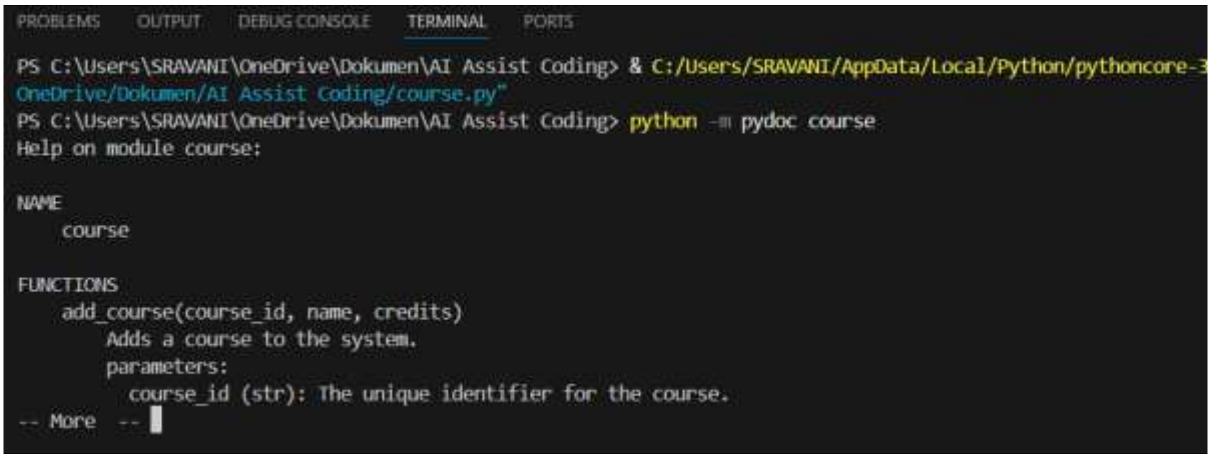
3. Generate documentation in the terminal.

4. Export the documentation in HTML format and open it in a browser.

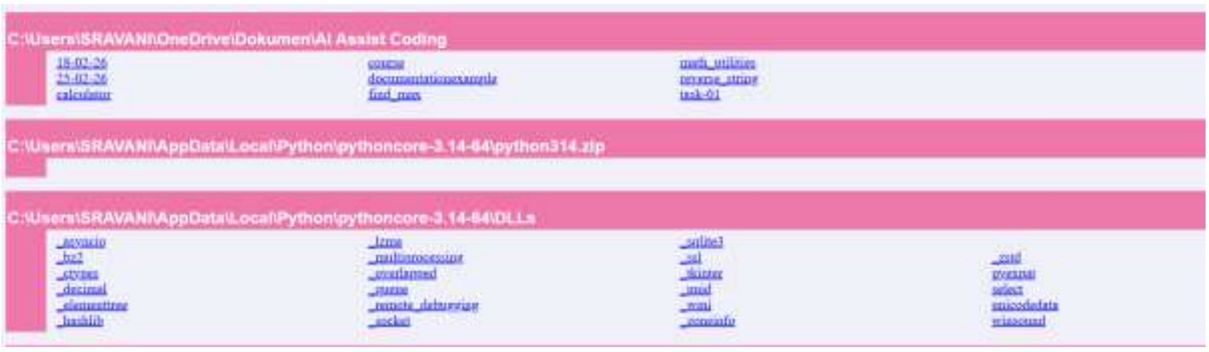


course.py > remove_course

```python
def add_course (course_id,name,credits):
  """
  Adds a course to the system.
  parameters:
    course_id (str): The unique identifier for the course.
    name (str): The name of the course.
    credits (int): The number of credits for the course.
  """
  # Code to add the course to the system would go here
  pass
def remove_course (course_id):
    """
    Removes a course from the system.
    parameters:
        course_id (str): The unique identifier for the course to be removed.
    """
    # Code to remove the course from the system would go here
    pass
def get_course(course_id):
    """
    Retrieves information about a specific course.
    parameters:
        course_id (str): The unique identifier for the course to be retrieved.
    returns:
        dict: A dictionary containing the course information, such as name and credits.
    """
    # Code to retrieve the course information from the system would go here
    pass
```



```
PS C:\Users\SRAVANI\OneDrive\Dokumen\AI Assist Coding> & C:/Users/SRAVANI/AppData/Local/Python/pythoncore-3
OneDrive/Dokumen/AI Assist Coding/course.py"
PS C:\Users\SRAVANI\OneDrive\Dokumen\AI Assist Coding> python -m pydoc course
Help on module course:

NAME
    course

FUNCTIONS
    add_course(course_id, name, credits)
        Adds a course to the system.
        parameters:
          course_id (str): The unique identifier for the course.
-- More  --
```



C:\Users\SRAVANI\OneDrive\Dokumen\AI Assist Coding

18-02-26	course	math_utilities
25-02-26	documentationexample	reverse_string
calculator	find_max	task-01

C:\Users\SRAVANI\AppData\Local\Python\pythoncore-3.14-64\python314.zip

C:\Users\SRAVANI\AppData\Local\Python\pythoncore-3.14-64\DLLs

_asyncio	_lzma	_sqlite3	
_bz2	_multiprocessing	_ssl	_zstd
_ctypes	_overlapped	_tkinter	pyexpat
_decimal	_queue	_uuid	select
_elementtree	_remote_debugging	_wmi	unicodedata
_hashlib	_socket	_zoneinfo	winsound