# Assignment-9.1

**Name: A. Srivani**

**HT No: 2303A51272**

**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

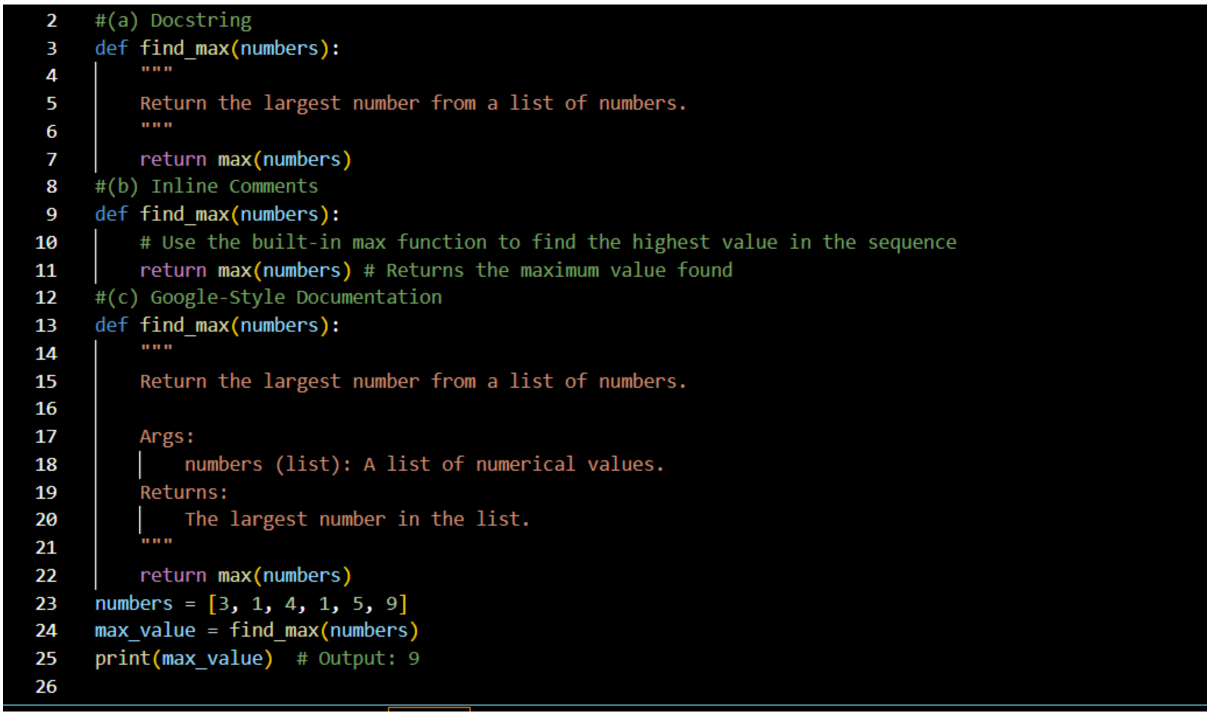
(b) Inline comments

(c) Google-style documentation

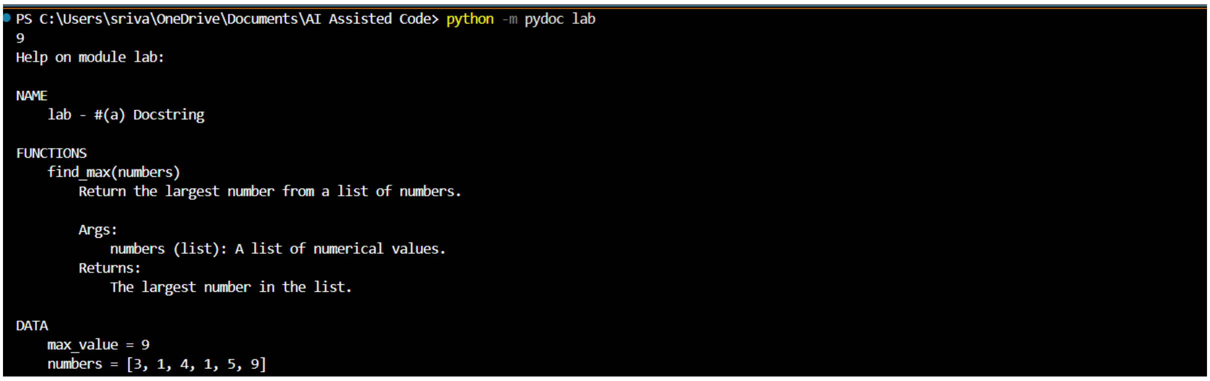
• Critically compare the three approaches. Discuss the advantages, disadvantages, and suitable use cases of each style.

• Recommend which documentation style is most effective for a mathematical utilities library and justify your answer.

**Output:**



```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 C:\Users\sriva\OneDrive\Documents\AI Assisted Code> python -m pydoc lab
9
Help on module lab:

NAME
    lab - #(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

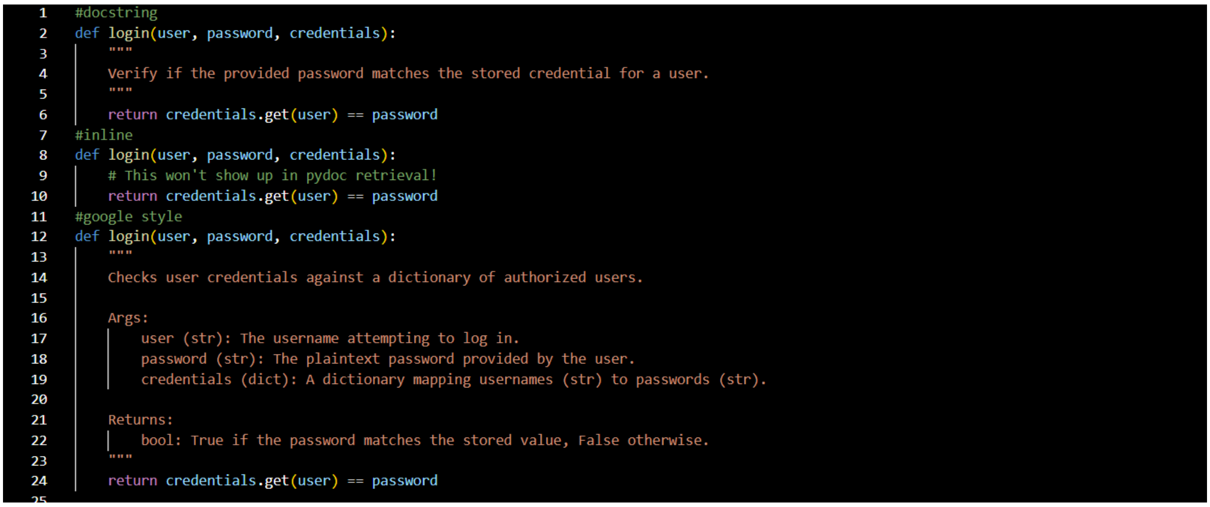
Task:

1. Write documentation in all three formats.

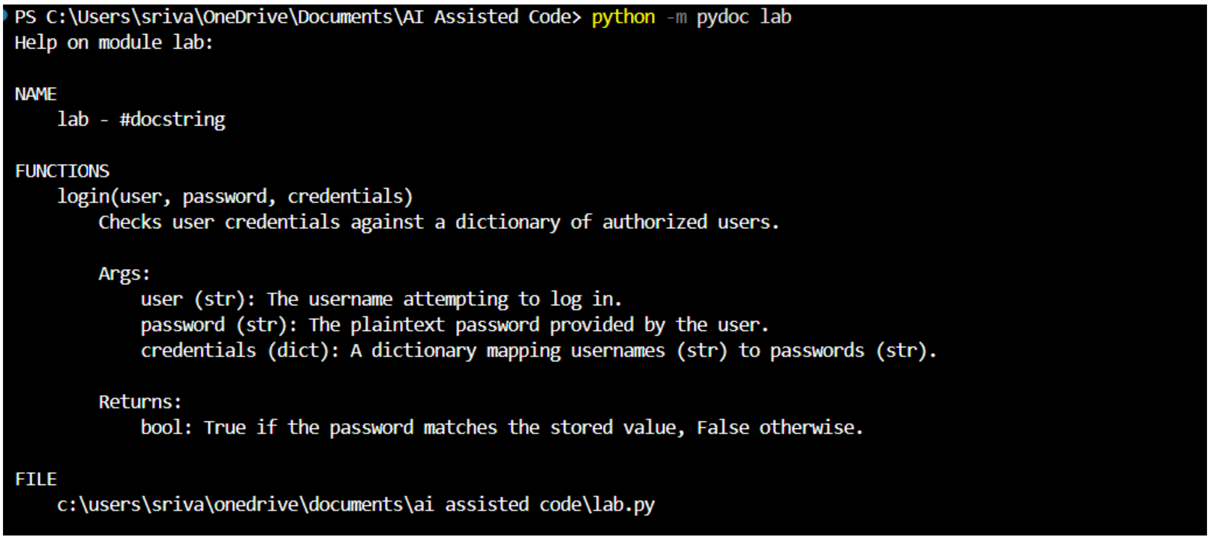
2. Critically compare the approaches.

3. Recommend which style would be most helpful for new

developers onboarding a project, and justify your choice.



```python
#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\sriva\OneDrive\Documents\AI Assisted Code> python -m pydoc lab
Help on module lab:

NAME
    lab - #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.

FILE
    c:\users\sriva\onedrive\documents\ai assisted code\lab.py
```

**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

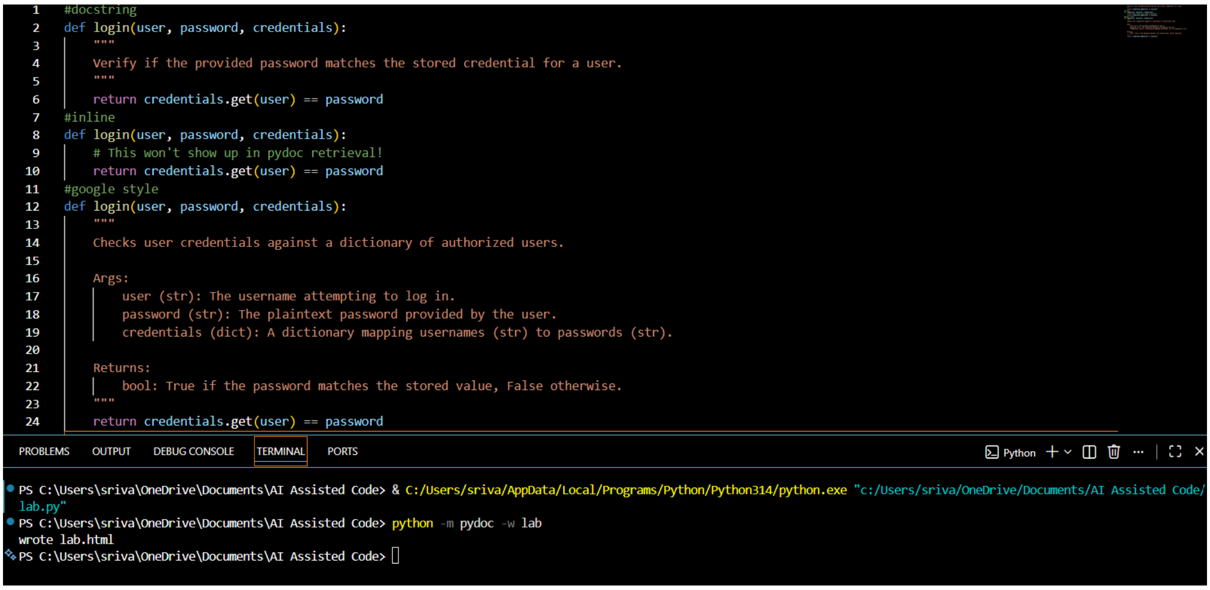
o multiply(a, b) – returns the product of two numbers

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

2. Display the module documentation in the terminal using

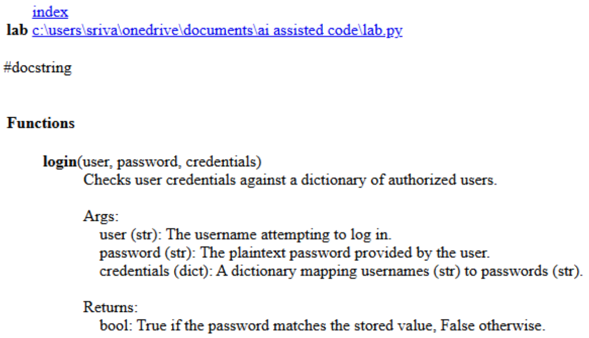
Python’s documentation tools.

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
#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\sriva\OneDrive\Documents\AI Assisted Code> & C:/Users/sriva/AppData/Local/Programs/Python/Python314/python.exe "c:/Users/sriva/OneDrive/Documents/AI Assisted Code/
lab.py"
PS C:\Users\sriva\OneDrive\Documents\AI Assisted Code> python -m pydoc -w lab
wrote lab.html
PS C:\Users\sriva\OneDrive\Documents\AI Assisted Code> 
```



index
**lab** c:\users\sriva\onedrive\documents\ai assisted code\lab.py

#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 4: Conversion Utilities Module

Task:

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

o decimal_to_binary(n)

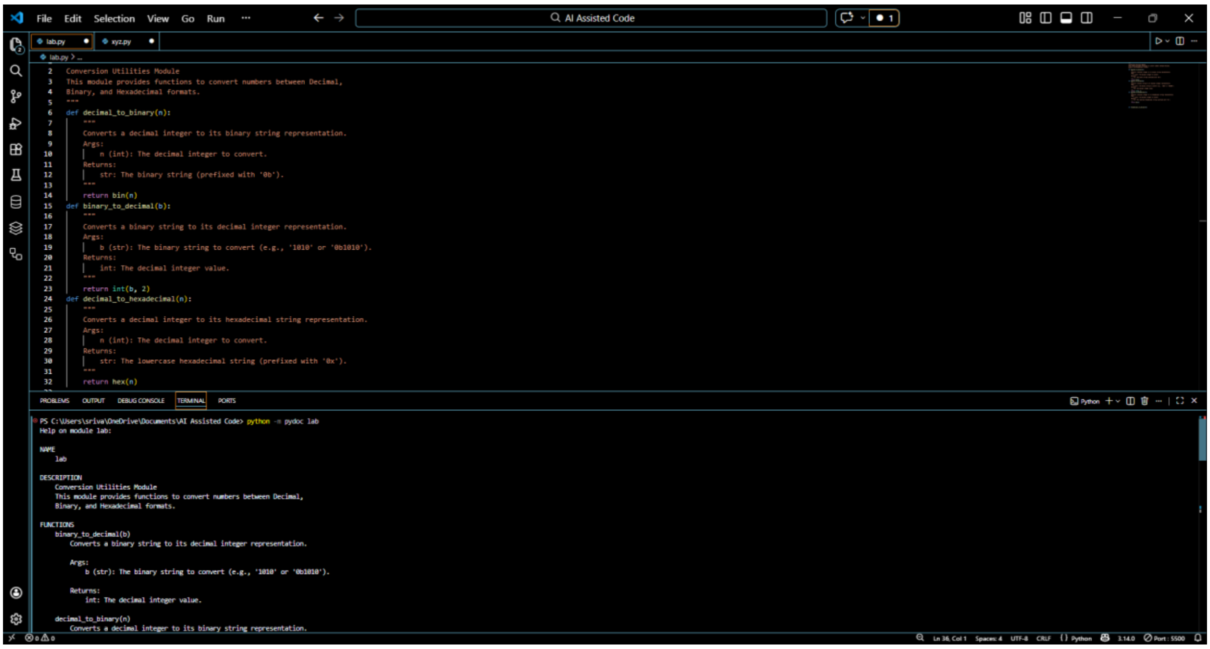
o binary_to_decimal(b)

o decimal_to_hexadecimal(n)

2. Use Copilot for auto-generating docstrings.

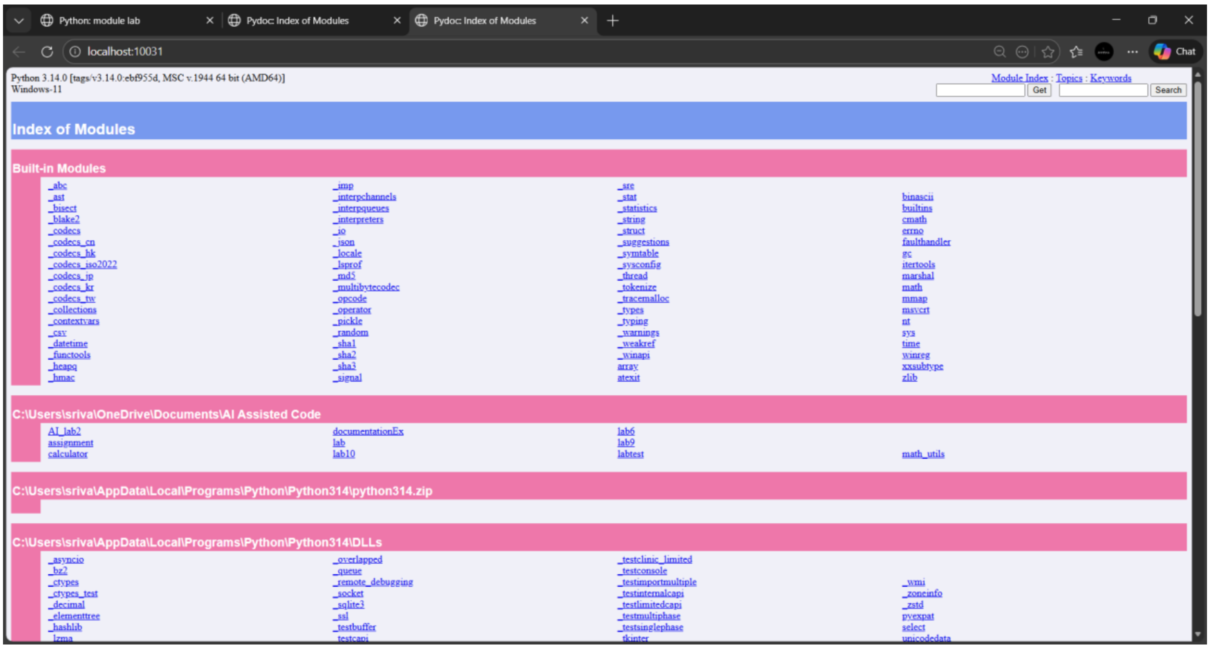
3. Generate documentation in the terminal.

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



```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)
```

```
PS C:\Users\sriva\OneDrive\Documents\AI Assisted Code> python -m pydoc lab
Help on module lab:

NAME
    lab

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.

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



Python 3.14.0 [tags/v3.14.0:ebf955d, MSC v.1944 64 bit (AMD64)]
Windows-11

Module Index : Topics : Keywords

Index of Modules

Built-in Modules

_abc
_ast
_bisect
_blake2
_codecs
_codecs_cn
_codecs_hk
_codecs_iso2022
_codecs_jp
_codecs_kr
_codecs_tw
_collections
_contextvars
_csv
_datetime
_functools
_heapq
_hmac
_imp
_interpchannels
_interpqueues
_interpreters
_io
_json
_locale
_lsprof
_md5
_multibytecodec
_opcode
_operator
_pickle
_random
_sha1
_sha2
_sha3
_signal
_sre
_stat
_statistics
_string
_struct
_suggestions
_symtable
_sysconfig
_thread
_tokenize
_tracemalloc
_types
_typing
_warnings
_weakref
_winapi
array
atexit
binascii
builtins
cmath
errno
faulthandler
gc
itertools
marshal
math
mmap
msvcrt
nt
sys
time
winreg
xxsubtype
zlib

C:\Users\sriva\OneDrive\Documents\AI Assisted Code

AI_lab2
assignment
calculator
documentationEx
lab
lab10
lab6
lab9
labtest
math_utils

C:\Users\sriva\AppData\Local\Programs\Python\Python314\python314.zip

C:\Users\sriva\AppData\Local\Programs\Python\Python314\DLLs

_asyncio
_bz2
_ctypes
_ctypes_test
_decimal
_elementtree
_hashlib
_lzma
_overlapped
_queue
_remote_debugging
_socket
_sqlite3
_ssl
_testbuffer
_testcapi
_testclinic_limited
_testconsole
_testimportmultiple
_testinternalcapi
_testlimitedcapi
_testmultiphase
_testsinglephase
_tkinter
_wmi
_zoneinfo
_zstd
pyexpat
select
unicodedata

## Problem 5 – Course Management Module

Task:

1. Create a module course.py with functions:

o add_course(course_id, name, credits)

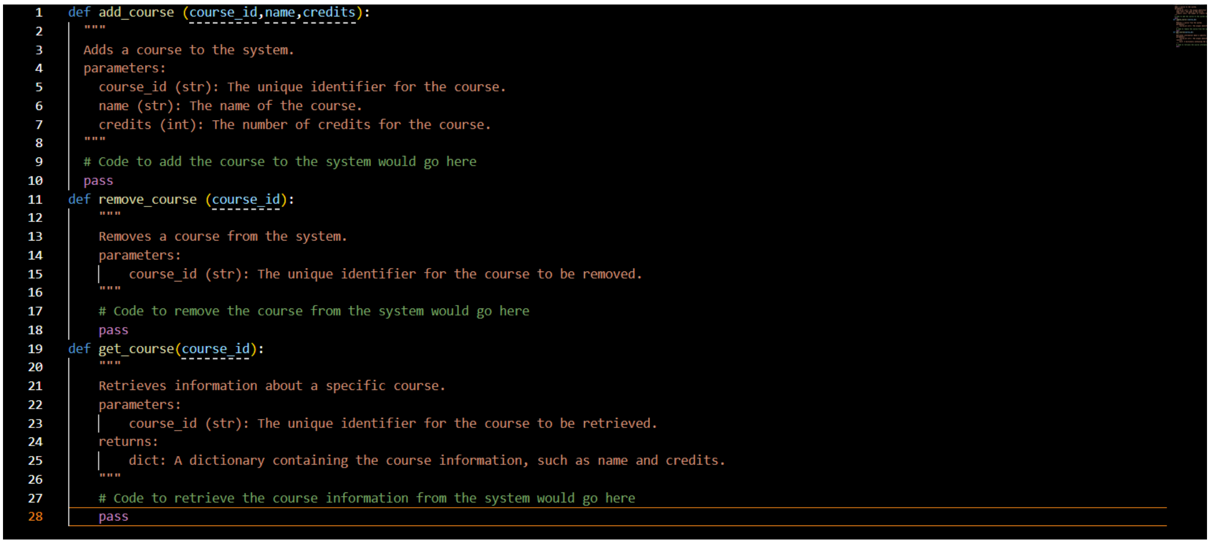
o remove_course(course_id)

o get_course(course_id)

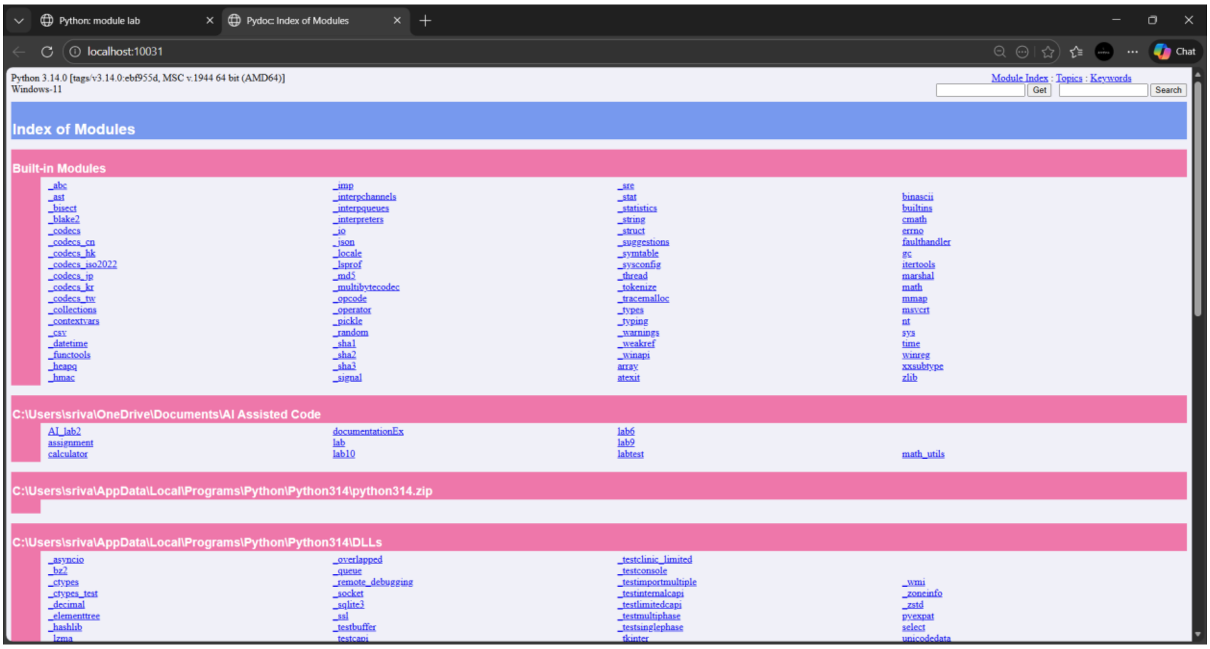
2. Add docstrings with Copilot.

3. Generate documentation in the terminal.

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



```
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
```



Python 3.14.0 [tags/v3.14.0:ebf955d, MSC v.1944 64 bit (AMD64)]
Windows-11

Module Index : Topics : Keywords

## Index of Modules

### Built-in Modules

_abc
_ast
_bisect
_blake2
_codecs
_codecs_cn
_codecs_hk
_codecs_iso2022
_codecs_jp
_codecs_kr
_codecs_tw
_collections
_contextvars
_csv
_datetime
_functools
_heapq
_hmac

_imp
_interpchannels
_interpqueues
_interpreters
_io
_json
_locale
_lsprof
_md5
_multibytecodec
_opcode
_operator
_pickle
_random
_sha1
_sha2
_sha3
_signal

_sre
_stat
_statistics
_string
_struct
_suggestions
_symtable
_sysconfig
_thread
_tokenize
_tracemalloc
_types
_typing
_warnings
_weakref
_winapi
array
atexit

binascii
builtins
cmath
errno
faulthandler
gc
itertools
marshal
math
mmap
msvcrt
nt
sys
time
winreg
xxsubtype
zlib

### C:\Users\sriva\OneDrive\Documents\AI Assisted Code

AI_lab2
assignment
calculator

documentationEx
lab
lab10

lab6
lab9
labtest

math_utils

### C:\Users\sriva\AppData\Local\Programs\Python\Python314\python314.zip

### C:\Users\sriva\AppData\Local\Programs\Python\Python314\DLLs

_asyncio
_bz2
_ctypes
_ctypes_test
_decimal
_elementtree
_hashlib
_lzma

_overlapped
_queue
_remote_debugging
_socket
_sqlite3
_ssl
_testbuffer
_testcapi

_testclinic_limited
_testconsole
_testimportmultiple
_testinternalcapi
_testlimitedcapi
_testmultiphase
_testsinglephase
_tkinter

_wmi
_zoneinfo
_zstd
pyexpat
select
unicodedata