

ASSIGNMENT-9.1

NAME:G.CHARANYA

HT.NO:2303A510G6

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.

```

lab_9.1.py > find_max
1  #(a) Docstring
2  def find_max(numbers):
3      """
4      Return the largest number from a list of numbers.
5      """
6      return max(numbers)
7  #(b) Inline Comments
8  def find_max(numbers):
9      # Use the built-in max function to find the highest value in the sequence
10     return max(numbers) # Returns the maximum value found
11 #(c) Google-Style Documentation
12 def find_max(numbers):
13     """
14     Return the largest number from a list of numbers.
15     Args:
16     |     numbers (list): A list of numerical values.
17     Returns:
18     |     The largest number in the list.
19     """
20     return max(numbers)
21 numbers = [3, 1, 4, 1, 5, 9]
22 max_value = find_max(numbers)
23 print(max_value) # Output: 9

```

```

PS C:\Users\2303a\OneDrive\Desktop\AI> & C:\Users\2303a\miniconda3\python.exe c:/Users/2303a/OneDrive/Desktop/AI/lab_9.1.py
PS C:\Users\2303a\OneDrive\Desktop\AI> & C:\Users\2303a\miniconda3\python.exe c:/Users/2303a/OneDrive/Desktop/AI/lab_9.1.py
9
PS C:\Users\2303a\OneDrive\Desktop\AI>

```

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.

```
lab_9_1.py X lab_9_1.html
lab_9_1.py > login
1 #docstring
2 def login(user, password, credentials):
3     """
4     Verify if the provided password matches the stored credential for a user.
5     """
6     return credentials.get(user) == password
7 #inline
8 def login(user, password, credentials):
9     # This won't show up in pydoc retrieval!
10    return credentials.get(user) == password
11 #google style
12 def login(user, password, credentials):
13     """
14     Checks user credentials against a dictionary of authorized users.
15
16     Args:
17         user (str): The username attempting to log in.
18         password (str): The plaintext password provided by the user.
19         credentials (dict): A dictionary mapping usernames (str) to passwords (str).
20
21     Returns:
22         bool: True if the password matches the stored value, False otherwise.
23     """
24    return credentials.get(user) == password
```

```
PS C:\Users\2303a\OneDrive\Desktop\AI> python -m pydoc lab_9_1
Help on module lab_9_1:

NAME
    lab_9_1 - #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).

NAME
    lab_9_1 - #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\2303a\onedrive\desktop\ai\lab_9_1.py
```

Problem 3: Calculator (Automatic Documentation Generation)

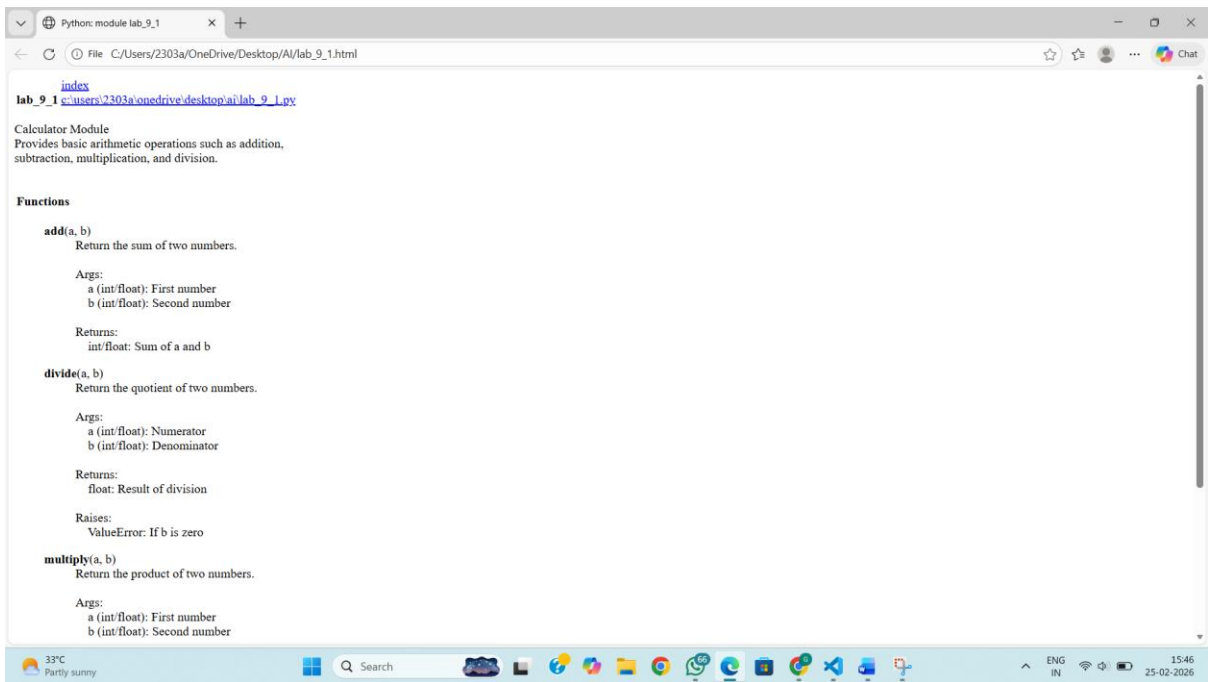
Task: Design a Python module named calculator.py and demonstrate automatic documentation generation.

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.

calculator.py >  divide

```
1  """
2  Calculator Module
3  This module provides basic arithmetic operations:
4  addition, subtraction, multiplication, and division.
5  """
6  def add(a, b):
7      """
8      Return the sum of two numbers.
9      Args:
10         a (int or float): First number
11         b (int or float): Second number
12
13     Returns:
14         int or float: Sum of a and b
15     """
16     return a + b
17  def subtract(a, b):
18      """
19      Return the difference of two numbers.
20      Args:
21         a (int or float): First number
22         b (int or float): Second number
23
24     Returns:
25         int or float: Difference of a and b
26     """
27     return a - b
28  def multiply(a, b):
29      """
30      Return the product of two numbers.
31      Args:
32         a (int or float): First number
33         b (int or float): Second number
34      Returns:
35         int or float: Product of a and b
36      """
37     return a * b
```

[illegible]

Problem 4: Conversion Utilities Module

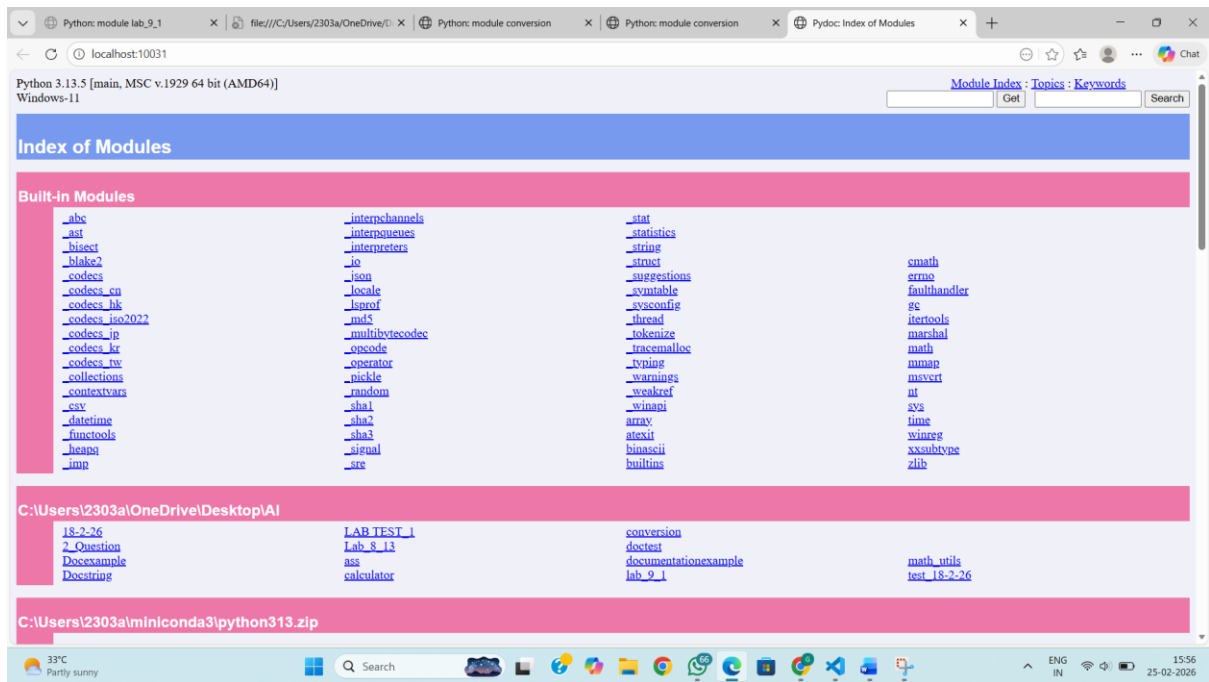
Task:

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.

```
conversion.py / decimal_to_hexadecimal
1  """
2  Conversion Utilities Module
3  This module provides functions to convert numbers
4  between decimal, binary, and hexadecimal formats.
5  """
6  def decimal_to_binary(n):
7      """
8      Convert a decimal integer to its binary representation.
9      Args:
10         n (int): Decimal number
11      Returns:
12         str: Binary representation of the number
13      """
14      return bin(n)[2:]
15  def binary_to_decimal(b):
16      """
17      Convert a binary string to its decimal equivalent.
18      Args:
19         b (str): Binary number as a string
20      Returns:
21         int: Decimal representation of the binary number
22      """
23      return int(b, 2)
24  def decimal_to_hexadecimal(n):
25      """
26      Convert a decimal integer to its hexadecimal representation.
27      Args:
28         n (int): Decimal number
29      Returns:
30         str: Hexadecimal representation of the number
31      """
32      return hex(n)[2:]
```

```
PS C:\Users\2303a\OneDrive\Desktop\AI> python -m pydoc -w conversion
wrote conversion.html
PS C:\Users\2303a\OneDrive\Desktop\AI> python -m pydoc -p 10031
Server ready at http://localhost:10031/
wrote conversion.html
PS C:\Users\2303a\OneDrive\Desktop\AI> python -m pydoc -p 10031
Server ready at http://localhost:10031/
Server ready at http://localhost:10031/
Server commands: [b]rowser, [q]uit
server> 
```



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.

```

1  """
2  Course Management Module
3  This module provides functionality to manage courses,
4  including adding, removing, and retrieving course details.
5  """
6  courses = {}
7  def add_course(course_id, name, credits):
8      """
9      Add a new course to the course dictionary.
10     Args:
11         course_id (str): Unique identifier of the course.
12         name (str): Name of the course.
13         credits (int): Number of credits assigned to the course.
14     Returns:
15         None
16     """
17     courses[course_id] = {
18         "name": name,
19         "credits": credits
20     }
21 def remove_course(course_id):
22     """
23     Remove a course from the system.
24     Args:
25         course_id (str): Unique identifier of the course.
26     Returns:
27         dict or None: Removed course details if course exists,
28         otherwise None.
29     """
30     return courses.pop(course_id, None)
31 def get_course(course_id):
32     """
33

```

```

def get_course(course_id):
    """
    Retrieve course details by course ID.
    Args:
        course_id (str): Unique identifier of the course.
    Returns:
        dict or None: Course details if found, otherwise None.
    """
    return courses.get(course_id)

```

```

PS C:\Users\2303a\OneDrive\Desktop\AI> python -m pydoc -p 10031
Server ready at http://localhost:10031/
PS C:\Users\2303a\OneDrive\Desktop\AI> & C:\Users\2303a\miniconda3\python.exe c:/Users/2303a/OneDrive/Desktop/AI/course.py
PS C:\Users\2303a\OneDrive\Desktop\AI> python -m pydoc -p 10031
Server ready at http://localhost:10031/
PS C:\Users\2303a\OneDrive\Desktop\AI> python -m pydoc -p 10031
Server ready at http://localhost:10031/
Server ready at http://localhost:10031/
Server commands: [b]rowser, [q]uit
server>

```

Python: module lab_9_1 X file:///C:/Users/2303a/OneDrive/ Python: module conversion X Python: module conversion X Pydoc: Index of Modules X Pydoc: Index of Modules X + - Chat

localhost:10031

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

Module Index : [Topics](#) : [Keywords](#)

Index of Modules

Built-in Modules

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

C:\Users\2303a\OneDrive\Desktop\AI

18-2-26	LAB TEST 1	conversion	lab_9_1
2_Question	Lab_8_13	course	math_utils
Docexample	ass	doctest	test_18-2-26
Docstring	calculator	documentationexample	

C:\Users\2303a\miniconda3\python313.zip

33°C Partly sunny Search ENG IN 16:03 25-02-2026