

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> & C:\Users\2303a\miniconda3\python.exe C:/Users/2303a/OneDrive/Desktop/AI/lab_9_1.py

- 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.

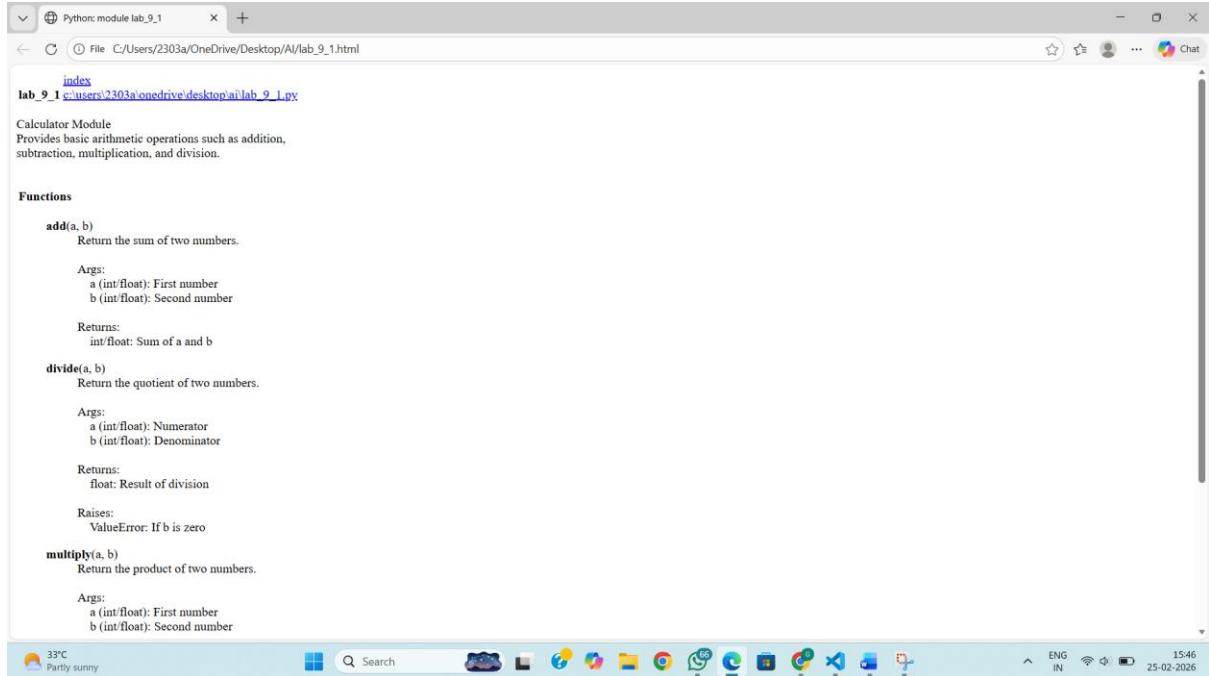
3. Generate and export the module documentation in HTML format using the pydoc utility, and open the generated HTML file in a web browser to verify the output.

```
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
```

```

lab_9_1.html > html > head > title
1   <!DOCTYPE html>
2   <html lang="en">
3   <head>
4   <meta charset="utf-8">
5   <title>Python: module lab_9_1</title>
6   </head><body>
7
8   <table class="heading">
9   <tr class="heading-text decor">
0   <td class="title">&nbs; <br> <strong class="title">lab_9_1</strong></td>
1   <td class="extra"><a href=".">index</a><br><a href="file:c%3A%5Cusers%5C2303a%5Conedrive%5Cdesktop%5Cai%5Clab_9_1.py">c:\users\2303a\onedrive\deskto
2   | <p><span class="code">calculator&nbs;Module<br>
3   Provides&nbs;basic&nbs;arithmetic&nbs;operations&nbs;such&nbs;as&nbs;addition,<br>
4   subtraction,&nbs;multiplication,&nbs;and&nbs;division.</span></p>
5   <p>
6   <table class="section">
7   <tr class="decor functions-decor heading-text">
8   <td class="section-title" colspan=3>&nbs; <br> <strong class="bigsection">Functions</strong></td></tr>
9
0   <tr><td class="decor functions-decor"><span class="code">&nbs;&nbs;&nbs;&nbs;&nbs;</span></td><td>&nbs;</td>
1   <td class="singlecolumn"><dl><dt><a name="-add"><strong>add</strong></a>(a, b)</dt><dd><span class="code">Return&nbs;the&nbs;sum&nbs;of&nbs;two&
2   &nbs;<br>
3   Args:<br>
4   &nbs;&nbs;&nbs;&nbs;(int/float):&nbs;First&nbs;number<br>
5   &nbs;&nbs;&nbs;&nbs;(int/float):&nbs;Second&nbs;number<br>
6   &nbs;<br>
7   Returns:<br>
8   &nbs;&nbs;&nbs;&nbs;int/float:&nbs;Sum&nbs;of&nbs;a&nbs;and&nbs;b</span></dd></dl>
9   | <dl><dt><a name="-divide"><strong>divide</strong></a>(a, b)</dt><dd><span class="code">Return&nbs;the&nbs;quotient&nbs;of&nbs;two&nbs;numbers.
0   &nbs;<br>
1   Args:<br>
2   &nbs;&nbs;&nbs;&nbs;(int/float):&nbs;Numerator<br>

```



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)

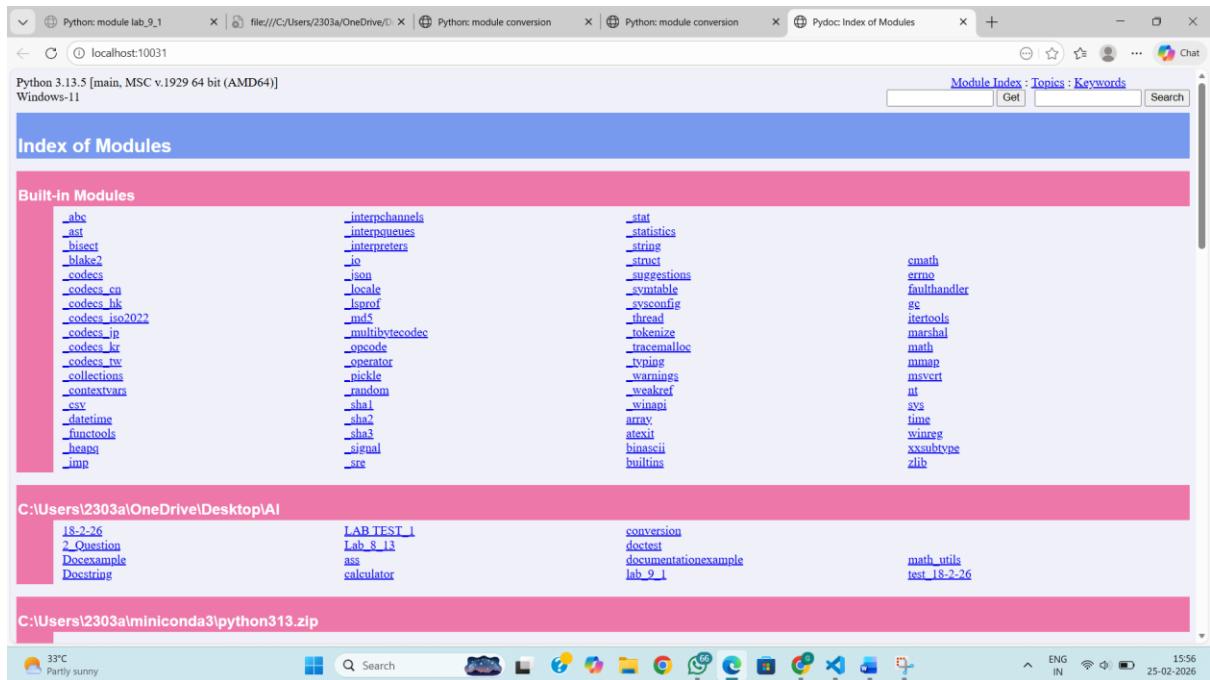
2. Use Copilot for auto-generating docstrings.

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
27    Returns:
28        dict or None: Removed course details if course exists,
29                      otherwise None.
30    """
31     return courses.pop(course_id, None)
32 def get_course(course_id):
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> []

```

The screenshot shows a web browser window with multiple tabs open. The active tab is titled "Pydoc Index of Modules" and displays the "Index of Modules". The page lists numerous built-in Python modules, such as abc, ast, bisect, blake2, codecs, codecs_cn, codecs_hk, codecs_iso2022, codecs_jp, codecs_kr, codecs_tw, collections, contextvars, csv, datetime, functools, heapq, imp, interpchannels, interpoqueues, interpreters, json, locale, lprof, mdu, multibytecodec, opcode, operator, pickle, random, sha1, sha2, sha256, signal, sre, stat, statistics, string, struct, suggestions, sysconfig, tokenize, tracemalloc, typing, warnings, weakref, winapi, array, atexit, binascii, builtins, calendar, cmath, errno, faulthandler, gc, itertools, marshal, math, mmap, msvcrt, nt, os, time, wincrt, xxsubtype, zlib.

Below the module index, there are sections for "C:/Users/2303a/OneDrive/Desktop/AI" and "C:/Users/2303a/miniconda3/python313.zip".

At the bottom of the screen, the Windows taskbar is visible, showing the date (25-02-2026), time (16:03), and system status (ENG IN).