

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.

```

lab9.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
16      Args:
17         numbers (list): A list of numerical values.
18      Returns:
19         The largest number in the list.
20      """
21     return max(numbers)
22     numbers = [3, 1, 4, 1, 5, 9]
23     max_value = find_max(numbers)
24     print(max_value) # Output: 9

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

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 -p 7993
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](#)

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.

```

3 #!Doctstring
4 def login(user,password,credentials):
5     """This function checks if the provided username and password match the credentials stored in a dictionary.
6     If the credentials are valid, it returns a success message. Otherwise, it raises a ValueError with an error message.
7
8     """
9     if user in credentials and credentials[user] == password:
10         return "Login successful"
11     else:
12         raise ValueError("Invalid username or password")
13 #!inline suggestions
14 def login(user,password,credentials): #to find the user in credentials and check if the password matches
15     if user in credentials and credentials[user] == password:#if the user is found and the password matches, return a success message
16         return "Login successful"
17     else:
18         #if the user is not found or the password does not match, raise a ValueError with an error message
19         raise ValueError("Invalid username or password")
20 #!google style docstring
21 def login(user,password,credentials):
22     """_summary_
23
24     Args:
25         user (_type_): _description_
26         password (_type_): _description_
27         credentials (_type_): _description_
28     returns:
29         _type_: _description_
30     raises:
31         ValueError: _description_
32     """
33     if user in credentials and credentials[user] == password:
34         return "Login successful"
35     else:
36         raise ValueError("Invalid username or password")
37 user="admin"
38 password="admin123"
39 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](#)

calculator [index](#)
[d:\ai\calculator.py](#)

#!Doctstring

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

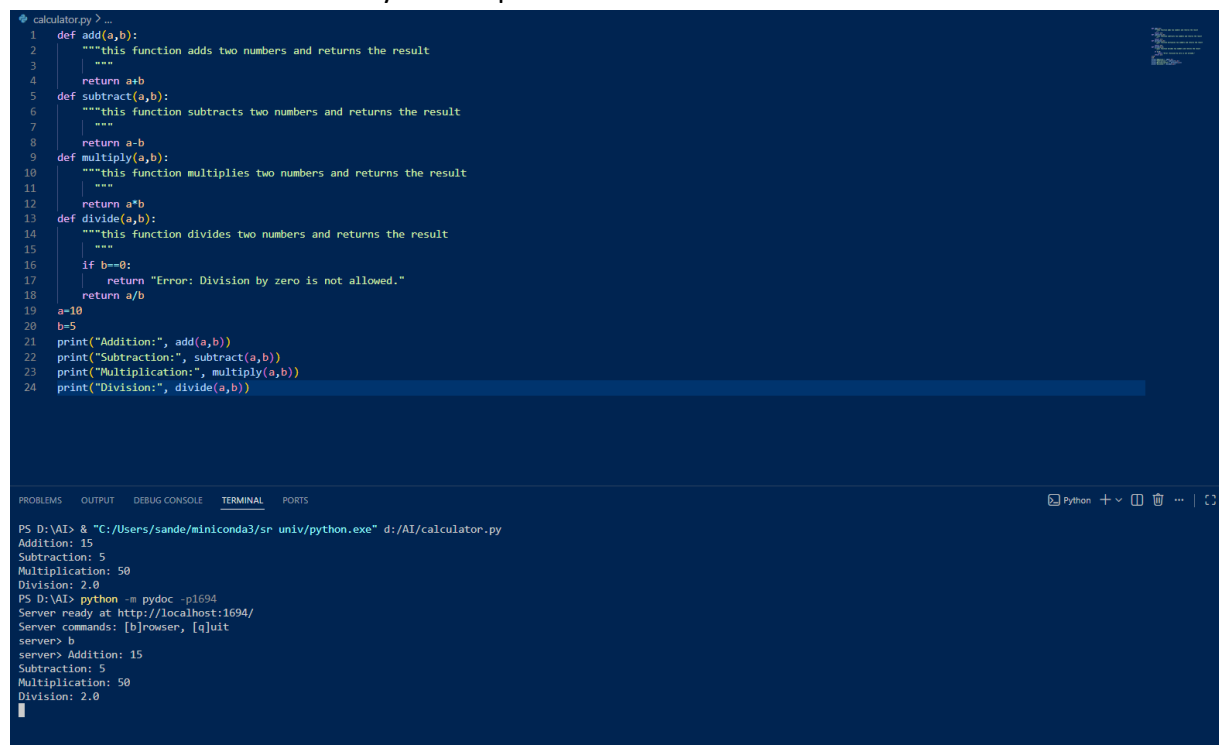
o divide(a, b) – returns the quotient of two

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

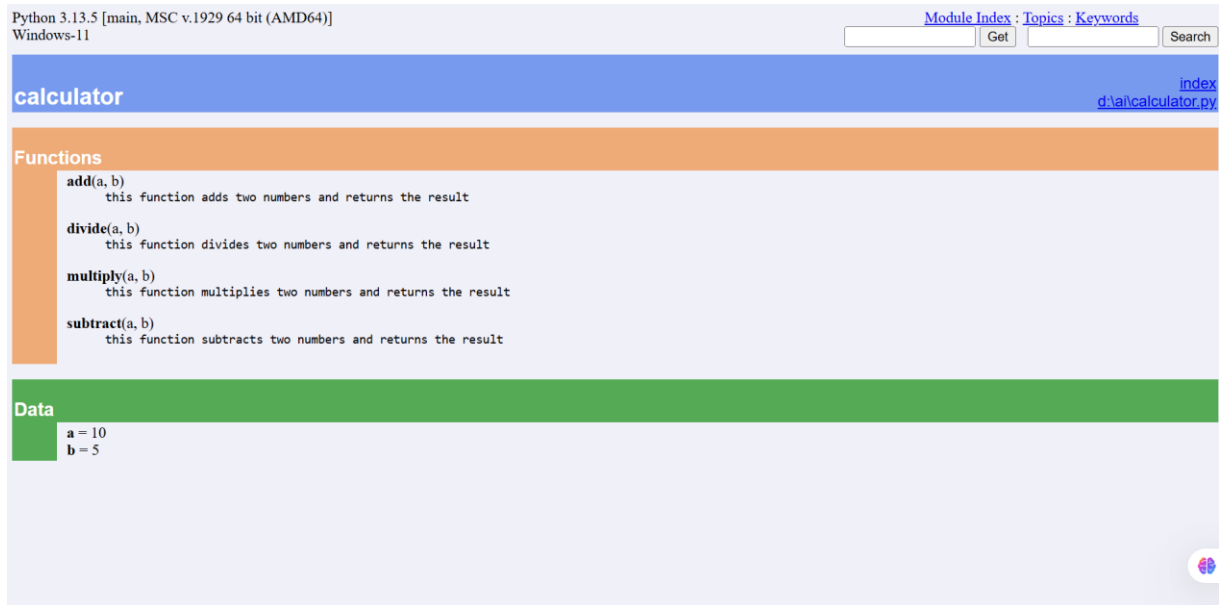
file in a web browser to verify the output.



```
calculator.py > ...
1 def add(a,b):
2     """this function adds two numbers and returns the result
3     """
4     return a+b
5 def subtract(a,b):
6     """this function subtracts two numbers and returns the result
7     """
8     return a-b
9 def multiply(a,b):
10    """this function multiplies two numbers and returns the result
11    """
12    return a*b
13 def divide(a,b):
14    """this function divides two numbers and returns the result
15    """
16    if b==0:
17        return "Error: Division by zero is not allowed."
18    return a/b
19 a=10
20 b=5
21 print("Addition:", add(a,b))
22 print("Subtraction:", subtract(a,b))
23 print("Multiplication:", multiply(a,b))
24 print("Division:", divide(a,b))
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Python + - [] [] [] [] []

```
PS D:\AI> & "C:/Users/sande/miniconda3/sr univ/python.exe" d:/AI/calculator.py
Addition: 15
Subtraction: 5
Multiplication: 50
Division: 2.0
PS D:\AI> python -m pydoc -p1694
Server ready at http://localhost:1694/
Server commands: [b]rowser, [q]uit
server> b
server> Addition: 15
Subtraction: 5
Multiplication: 50
Division: 2.0
```



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.

The image shows a VS Code editor window with a Python script named `calculator.py`. The script defines three functions: `decimal_to_binary(n)`, `binary_to_decimal(b)`, and `decimal_to_hexadecimal(n)`. It also includes test code for each function. The terminal output shows the script being run successfully, with the following commands and results:

```
PS D:\AI> python -m pydoc -p1747
Server ready at http://localhost:1747/
Server commands: [b]rowser, [q]uit
server> b
server> 110
6
E1
[]
```

The documentation page for the `calculator` module is displayed below the editor. It includes a search bar, a list of functions, and a data section.

calculator

[index](#)
[d:\ai\calculator.py](#)

Functions

- binary_to_decimal(b)**
this function converts a binary number (given as a string) to decimal and returns it as an integer
- decimal_to_binary(n)**
this function converts a decimal number to binary and returns it as a string
- decimal_to_hexadecimal(n)**
this function converts a decimal number to hexadecimal and returns it as a string

Data

b = '110'
n = 225

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.py > ...
1 def add_course(course_id,name,credits):
2     """_summary_
3
4     Args:
5         course_id (_type_): _description_
6         name (_type_): _description_
7         credits (_type_): _description_
8
9     Returns:
10         _type_: _description_
11     """
12     course = {
13         "course_id": course_id,
14         "name": name,
15         "credits": credits
16     }
17     return course
18
19 course_id=101
20 name="Introduction to Computer Science"
21 credits=3
22 course_info = add_course(course_id, name, credits)
23 print(course_info)
24
25 def remove_course(course_id):
26     """_summary_
27
28     Args:
29         course_id (_type_): _description_
30
31     Returns:
32         _type_: _description_
33     """
34     # Code to remove the course from the database or list
35     print(f"Course with ID {course_id} has been removed.")
36     course_id=101
37     remove_course(course_id)
38     print(f"Course with ID {course_id} has been removed.")
39
40 def get_course(course_id):
41     """_summary_
42
43     Args:
44         course_id (_type_): _description_
45
46     Returns:
47         _type_: _description_
48     """
49     # Code to retrieve the course information from the database or list
50     course_info = {
51         "course_id": course_id,
52         "name": "Introduction to Computer Science",
53         "credits": 3
54     }
55     return course_info
56 course_id=101
57 course_info = get_course(course_id)
58 print(course_info)
```

PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL** PORTS

```
PS D:\AI> python -m pydoc -p 1118
Server ready at http://localhost:1118/
Server commands: [b]rowser, [q]uit
server> b
Server ready at http://localhost:1118/
Server commands: [b]rowser, [q]uit
server> b
server> b
server> {'course_id': 101, 'name': 'Introduction to Computer Science', 'credits': 3}
Course with ID 101 has been removed.
Course with ID 101 has been removed.
Course with ID 101 has been removed.
Course with ID 101 has been removed.
{'course_id': 101, 'name': 'Introduction to Computer Science', 'credits': 3}
[]
```

calculator

[index](#)
[d:\ai\calculator.py](#)

Functions

```
add_course(course_id, name, credits)
    _summary_

    Args:
        course_id (_type_): _description_
        name (_type_): _description_
        credits (_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, 'credits': 3, 'name': 'Introduction to Computer Science'}
credits = 3
name = 'Introduction to Computer Science'
```