

ASSIGNMENT-9.1

NAME: RISHIK.P

HALL TICKET NO : 2303A51778

BATCH : 12

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.

PROMPT: #create a python function, use pydoc to generate documentation

CODE :

```
ass9_1.py > find_max
1  #create a python function, use pydoc to generate documentation
2  import numbers
3
4
5  def find_max(numbers):
6      """
7          Find the maximum number in a list of numbers.
8
9      Args:
10         numbers (list): A list of numbers.
11      Returns:
12         int: The maximum number in the list.
13         """
14     if not numbers:
15         raise ValueError("The list of numbers cannot be empty.")
16     max_num = numbers[0]
17     for num in numbers:
18         if num > max_num:
19             max_num = num
20     return max_num
```

OUTPUT:

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

PS C:\Users\Rishik\OneDrive\Desktop\AI> python -m pydoc ass9_1.
Help on module ass9_1 in ass9_1:

NAME
    ass9_1 - #create a python function, use pydoc to generate documentation
    ass9_1 - #create a python function, use pydoc to generate documentation

FUNCTIONS
    find_max(numbers)
        Find the maximum number in a list of numbers.

        Args:
            numbers (list): A list of numbers.
        Returns:
            int: The maximum number in the list.

FILE
    c:\users\rishik\onedrive\desktop\ai\ass9_1.py
```

(b) Inline comments

CODE:

```
ass9_1.py > ...
1  #create a python function, use pydoc to generate documentation
2  import numbers
3
4
5  def find_max(numbers):
6      """
7          Find the maximum number in a list of numbers.
8
9      Args:
10         numbers (list): A list of numbers.
11      Returns:
12         int: The maximum number in the list.
13     """
14     if not numbers:
15         raise ValueError("The list of numbers cannot be empty.")
16     max_num = numbers[0]
17     for num in numbers:
18         if num > max_num:
19             max_num = num
20     return max_num
21
```

OUTPUT :



[index](#)
ass9_1 c:\users\rishik\onedrive\desktop\ai\ass9_1.py

#create a python function, use pydoc to generate documentation

Modules

[numbers](#)

Functions

find_max(numbers)
Find the maximum number in a list of numbers.

Args:
 numbers (list): A list of numbers.
Returns:
 int: The maximum number in the list.

(c) Google-style documentation

```

ass9_1.py > ...
1 #create a python function, use pydoc to generate documentation
2 import numbers
3
4
5 def find_max(numbers):
6     """
7         Find the maximum number in a list of numbers.
8
9     Args:
10        numbers (list): A list of numbers.
11    Returns:
12        int: The maximum number in the list.
13    """
14    if not numbers:
15        raise ValueError("The list of numbers cannot be empty.")
16    max_num = numbers[0]
17    for num in numbers:
18        if num > max_num:
19            max_num = num
20    return max_num
21

```

OUTPUT:

The screenshot shows a web-based Python module index. The main header includes 'Summarize' and navigation icons. Below the header, the URL is 'localhost:8080' and the page title is 'Index of Modules'. The content area is divided into sections for 'Built-in Modules' and user-defined paths.

- 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, interchannels, interqueues, interpreters, json, locale, lprof, md5, multibytecodec, opcode, operator, pickle, random, sha1, sha2, sha3, signal, sre, stat, statistics, string, struct, suggestions, svntable, sysconfig, thread, tokenize, tracemalloc, types, typing, warnings, weakref, winapi, array, atexit, binascii, builtins,cmath,errno,faulthandler,gc,marshal,math,mmap,mixer,nt,sys,time,wineg,xsubtype,zlib.
- C:\Users\Rishik\OneDrive\Desktop\AI:** ass, ass9_1, assignment9_1, labtest1, task
- C:\Users\Rishik\AppData\Local\Python\pythoncore-3.14-64\python314.zip**
- C:\Users\Rishik\AppData\Local\Python\pythoncore-3.14-64\DLLs:** asyncio, bz2, ctypes, lzma, multiprocessing, overlapped, sqlite3, ssl, tkinter, zstd.

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.

PROMPT: #create a python function, use pydoc to generate documentation

CODE :

```
❸ ass9_1.py > login
1  #create a python function, use pydoc to generate documentation
2  def login(user, password, credentials):
3      """
4          Logs in a user by checking if the provided password matches the stored credentials.
5
6      Args:
7          user (str): The username to log in.
8          password (str): The password to check.
9          credentials (dict): A dictionary mapping usernames to their passwords.
10
11     Returns:
12         bool: True if the login is successful, False otherwise.
13     """
14     return credentials.get(user) == password
```

OUTPUT:

```
● PS C:\Users\Rishik\OneDrive\Desktop\AI> python -m pydoc ass9_1.
Help on module ass9_1 in ass9_1:

NAME
    ass9_1 - #create a python function, use pydoc to generate documentation

FUNCTIONS
    find_max(numbers)
        Find the maximum number in a list of numbers.

        Args:
            numbers (list): A list of numbers.
        Returns:
            int: The maximum number in the list.

FILE
    c:\users\rishik\onedrive\desktop\ai\ass9_1.py
```

(b) Inline comments

CODE:

```
ass9_1.py > login
1  #create a python function, use pydoc to generate documentation
2  def login(user, password, credentials):
3      """
4          Logs in a user by checking if the provided password matches the stored credentials.
5
6      Args:
7          user (str): The username to log in.
8          password (str): The password to check.
9          credentials (dict): A dictionary mapping usernames to their passwords.
10
11     Returns:
12         bool: True if the login is successful, False otherwise.
13         """
14     return credentials.get(user) == password
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

- PS C:\Users\Rishik\OneDrive\Desktop\AI> & C:/Users/Rishik/AppData/Local/Python/pythoncore-3.14-64/python.exe
- PS C:\Users\Rishik\OneDrive\Desktop\AI> python -m pydoc -w ass9_1.
wrote ass9_1..html
- PS C:\Users\Rishik\OneDrive\Desktop\AI> []

OUTPUT:

[index](#)
ass9_1 c:/users/rishik/onedrive/desktop/ai/ass9_1.py

#create a python function, use pydoc to generate documentation

Functions

login(user, password, credentials)

Logs in a user by checking if the provided password matches the stored credentials.

Args:

 user (str): The username to log in.

 password (str): The password to check.

 credentials (dict): A dictionary mapping usernames to their passwords.

Returns:

 bool: True if the login is successful, False otherwise.

(c) Google-style documentation

```

❷ ass9_1.py > ⌂ login
1  #create a python function, use pydoc to generate documentation
2  def login(user, password, credentials):
3      """
4          Logs in a user by checking if the provided password matches the stored credentials.
5
6      Args:
7          user (str): The username to log in.
8          password (str): The password to check.
9          credentials (dict): A dictionary mapping usernames to their passwords.
10
11     Returns:
12         bool: True if the login is successful, False otherwise.
13     """
14     return credentials.get(user) == password

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

PS C:\Users\Rishik\OneDrive\Desktop\AI> & C:/Users/Rishik/AppData/Local/Python/pythoncore-3.14-64/python.exe
● PS C:\Users\Rishik\OneDrive\Desktop\AI> python -m pydoc -p 8080
Server ready at http://localhost:8080/
Server commands: [b]rowser, [q]uit
server> b
server>

```

OUTPUT:

The screenshot shows the Python pydoc module index interface. At the top, it displays the Python version (3.14.0) and build details. Below that is a search bar with links to 'Module Index', 'Topics', and 'Keywords'. The main area is divided into sections: 'Index of Modules' (blue header), 'Built-in Modules' (pink header), 'C:\Users\Rishik\OneDrive\Desktop\AI' (pink header), and 'C:\Users\Rishik\AppData\Local\Python\pythoncore-3.14-64\DLLs' (pink header). The 'Built-in Modules' section lists numerous standard library modules like abc, ast, bisect, binascii, builtins, cmath, errno, faulthandler, gc, itertools, marshal, math, mmap, msvcrt, nt, os, time, winreg, and zlib. The 'C:\Users\Rishik\OneDrive\Desktop\AI' section lists user-defined modules: ass and ass9_1. The 'C:\Users\Rishik\AppData\Local\Python\pythoncore-3.14-64\DLLs' section lists various Windows DLL files: async, bz2, crypt, lzma, multiprocessing, overlapped, sqlite3, ssl, tkinter, zstd, and pyexpat.

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.

```
A simple calculator module that performs basic arithmetic operations.
This module demonstrates automatic documentation generation using pydoc.
"""
def add_numbers(a, b):
    """
    Adds two numbers together.

    Args:
        a (float): The first number to add.
        b (float): The second number to add.

    Returns:
        float: The sum of the two numbers.
    """
    return a + b
def subtract_numbers(a, b):
    """
    Subtracts the second number from the first number.

    Args:
        a (float): The number to be subtracted from.
        b (float): The number to subtract.

    Returns:
        float: The difference of the two numbers.
    """
    return a - b
def multiply_numbers(a, b):
    """
    Multiplies two numbers together.

    Args:
        a (float): The first number to multiply.
        b (float): The second number to multiply.

    Returns:
        float: The product of the two numbers.
    """
    return a * b
```

OUTPUT:

```
PS C:\Users\Rishik\OneDrive\Desktop\AI> python -m pydoc calculator
Help on module calculator:

NAME
    calculator - calculator.py

DESCRIPTION
    A simple calculator module that performs basic arithmetic operations.
    This module demonstrates automatic documentation generation using pydoc.

FUNCTIONS
    add_numbers(a, b)
        Adds two numbers together.

        Args:
            a (float): The first number to add.
            b (float): The second number to add.

        Returns:
            float: The sum of the two numbers.

    divide_numbers(a, b)
        Divides the first number by the second number.

        Args:
            a (float): The numerator.
```

```

calculator.py > ...
31 def multiply_numbers(a, b):
32     """
33     Multiplies two numbers together.
34
35     Args:
36         a (float): The first number to multiply.
37         b (float): The second number to multiply.
38
39     Returns:
40         float: The product of the two numbers.
41     """
42     return a * b
43 def divide_numbers(a, b):
44     """
45     Divides the first number by the second number.
46
47     Args:
48         a (float): The numerator.
49         b (float): The denominator.
50
51     Returns:
52         float: The quotient of the two numbers.
53
54     Raises:
55         ValueError: If the denominator is zero.
56     """
57     if b == 0:

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

PS C:\Users\Rishik\OneDrive\Desktop\AI & C:/Users/Rishik/AppData/Local/Python/pythoncore-3.14-64/python.exe calculator.py
PS C:\Users\Rishik\OneDrive\Desktop\AI> python -m pydoc -w calculator
wrote calculator.html
PS C:\Users\Rishik\OneDrive\Desktop\AI> python -m pydoc -p 8080
Server ready at http://localhost:8080/
Server commands: [b]rowser, [q]uit
server> b
server> 

```

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

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

| Get |

Index of Modules

Built-in Modules

abc	_imp	_sre	binascii
_ast	_interpcache	_stat	builtins
_bisect	_interqueues	statistics	cmath
_blake2	_interpreters	string	errno
_codecs	_io	struct	faulthandler
_codecs_cn	_json	suggestions	gc
_codecs_hk	_locale	syntable	itertools
_codecs_iso2022	_lsprof	sysconfig	marshal
_codecs_jp	_md5	thread	math
_codecs_kr	_multibytecodec	tokenize	 mmap
_codecs_tw	_opcode	tracemalloc	msvcrt
collections	_operator	types	nt
contextvars	_pickle	typing	svs
csv	_random	warnings	time
datetime	_sha1	weakref	winreg
functools	_sha2	windapi	xxsubtype
heapq	_sha3	array	zlib
hmac	_signal	_exit	

C:\Users\Rishik\OneDrive\Desktop\AI

ass	assignment9_1	labtest	
ass9_1	calculator	labtest1	task

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

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

asyncio	_lzma	_sqlite3	_zstd
bz2	_multiprocessing	_ssl	_pyexpat
ctypes	_overlapped	_tkinter	

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 > ...
4  A utility module for number system conversions.
5  This module demonstrates automatic documentation generation using pydoc.
6  """
7
8  def decimal_to_binary(n):
9      """
10     Converts a decimal number to binary format.
11
12     Args:
13         n (int): A decimal number
14
15     Returns:
16         str: Binary representation of the decimal number
17     """
18     return bin(n)[2:]
19
20
21 def binary_to_decimal(b):
22     """
23     Converts a binary number to decimal format.
24
25     Args:
26         b (str): A binary number in string format
27
28     Returns:
29         int: Decimal representation of the binary number
30     """
31     return int(b, 2)
32
33
34 def decimal_to_hexadecimal(n):
35     """
36     Converts a decimal number to hexadecimal format.
37
38     Args:
```

OUTPUT:

```
PS C:\Users\Rishik\OneDrive\Desktop\AI> python -m pydoc conversion

NAME
    conversion - conversion.py

DESCRIPTION
    A utility module for number system conversions.
    This module demonstrates automatic documentation generation using pydoc.

FUNCTIONS
    binary_to_decimal(b)
        Converts a binary number to decimal format.

        Args:
            b (str): A binary number in string format

        Returns:
            int: Decimal representation of the binary number

    decimal_to_binary(n)
        Converts a decimal number to binary format.

        Args:
            n (int): A decimal number
-- More --
```

HTML Documentation

[index](#)**conversion** <c:/users/rishik/onedrive/desktop/ai/conversion.py>

conversion.py

A utility module for number system conversions.

This module demonstrates automatic documentation generation using pydoc.

Functions**binary_to_decimal(b)**

Converts a binary number to decimal format.

Args:

b (str): A binary number in string format

Returns:

int: Decimal representation of the binary number

decimal_to_binary(n)

Converts a decimal number to binary format.

Args:

n (int): A decimal number

Returns:

str: Binary representation of the decimal number

decimal_to_hexadecimal(n)

Converts a decimal number to hexadecimal format.

Args:

n (int): A decimal number

Returns:

str: Hexadecimal representation of the decimal number

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

Module Index : Topics : Key
Get

[Index of Modules](#)

Built-in Modules

_abc	_imp	_site	binascii
_ast	_interpcache	_stat	builtins
_bisect	_intersqueues	_statistics	cmath
_blake2	_interpreters	_string	errno
_codecs	_json	_struct	faulthandler
_codecs_cn	_locale	_warnings	gc
_codecs_hk	_lprof	_weakref	itertools
_codecs_iso2022	_multibytecodec	_thread	marshal
_codecs_in	_opcode	_tracemalloc	math
_codecs_kr	_operator	_types	 mmap
_codecs_tw	_pickle	_typing	msvcr
collections	_random	_warnings	nt
contextvars	_sha1	_weakref	sys
csv	_sha2	_winapi	time
datetime	_sha3	array	winreg
functools	_signal	atexit	xxsubtype
heapq			zlib
_hmac			

C:\Users\Rishik\OneDrive\Desktop\AI

ass	assignment9_1	conversion	labtest1
ass9_1	calculator	labtest	task

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

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

_asyncio	_lzma	_sqlite3	_zstd
_bz2	_multiprocessing	_ssl	_zipapp
_ctypes	_overlapped	_xmlreader	