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 :

```python
#create a python function, use pydoc to generate documentation
import numbers


def 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.
    """
    if not numbers:
        raise ValueError("The list of numbers cannot be empty.")
    max_num = numbers[0]
    for num in numbers:
        if num > max_num:
            max_num = num
    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:

```python
#create a python function, use pydoc to generate documentation
import numbers


def 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.
    """
    if not numbers:
        raise ValueError("The list of numbers cannot be empty.")
    max_num = numbers[0]
    for num in numbers:
        if num > max_num:
            max_num = num
    return max_num

```

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

```python
#create a python function, use pydoc to generate documentation
import numbers


def 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.
    """
    if not numbers:
        raise ValueError("The list of numbers cannot be empty.")
    max_num = numbers[0]
    for num in numbers:
        if num > max_num:
            max_num = num
    return max_num

```

OUTPUT:

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 :

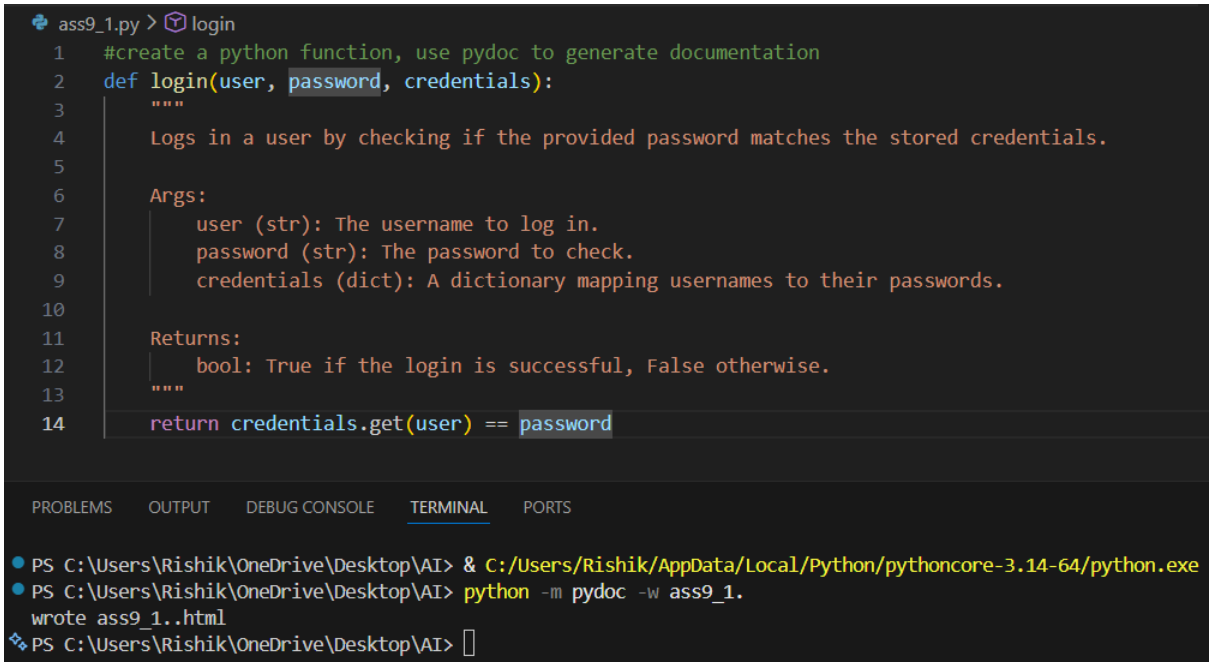
```python
#create a python function, use pydoc to generate documentation
def login(user, password, credentials):
    """
    Logs in a user by checking if the provided password matches the stored credentials.

    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.
    """
    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:



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

```python
#create a python function, use pydoc to generate documentation
def 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.
    """
    return credentials.get(user) == password
```

```
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:

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 > ...

```python
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
def divide_numbers(a, b):
    """
    Divides the first number by the second number.

    Args:
        a (float): The numerator.
        b (float): The denominator.

    Returns:
        float: The quotient of the two numbers.

    Raises:
        ValueError: If the denominator is zero.
    """
    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 c
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 : Keyw
Get

Index of Modules

Built-in Modules

_abc	_imp	_sre	binascii
_ast	_interpchannels	_stat	builtins
_bisect	_interpqueues	_statistics	cmath
_blake2	_interpreters	_string	errno
_codecs	_io	_struct	faulthandler
_codecs_cn	_json	_suggestions	gc
_codecs_hk	_locale	_symtable	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	sys
_csv	_random	_warnings	time
_datetime	_sha1	_weakref	winreg
_functools	_sha2	_winapi	xxsubtype
_heapq	_sha3	array	zlib
_hmac	_signal	atexit	

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	
_bz2	_multiprocessing	_ssl	_zstd
_ctypes	_overlapped	_tkinter	pyexpat

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 > ...

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

def 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
    """
    return bin(n)[2:]


def 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
    """
    return int(b, 2)


def decimal_to_hexadecimal(n):
    """
    Converts a decimal number to hexadecimal format.

    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