Assignment-9.5

2303A51600

ARSHA VARDHINI

PROBLEM-1

PROMPT:

def reverse_string(text):

return text[::-1]

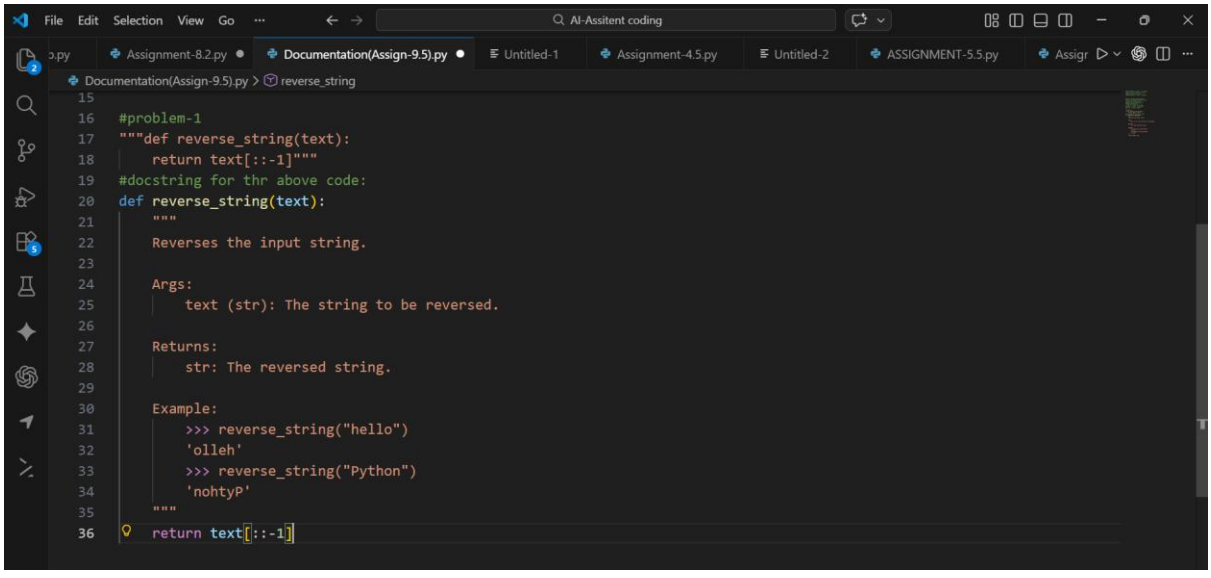
Write documentation in:

(a) Docstring

(b) Inline comments

(c) Google-style documentation

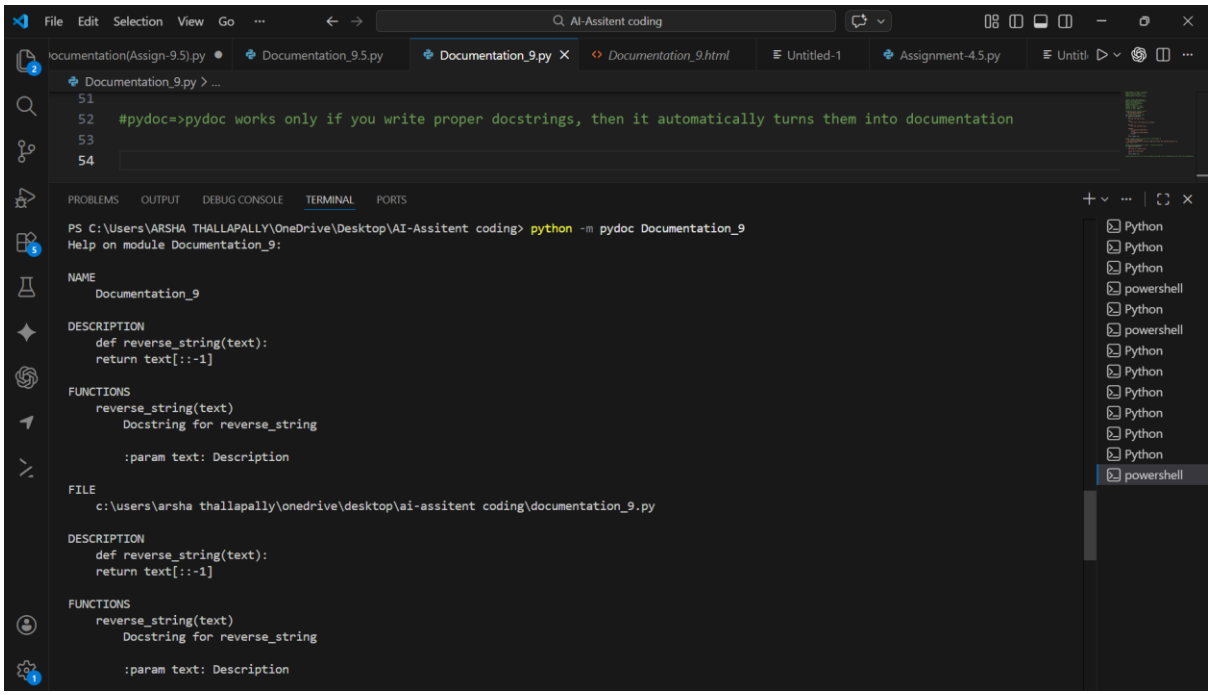
CODE:




```
#Inline comment
"""def reverse_string(text):
    # This function takes a string as input and returns the reversed version of it.
return text[::-1]"""
```

```python
#Google-style documentation
def reverse_string(text):
    """
    Docstring for reverse_string

    :param text: Description
    """
    return text[::-1]
```



OBSERVATION:

Docstring:
Gives a general description of the function. It is readable and can be accessed using help() and pydoc.

Inline comments:
Explain the working of code line by line. They are useful for understanding logic but cannot be used to generate documentation automatically.

Google-style documentation:
More structured and professional. It clearly defines parameters and return values and is supported by documentation tools.

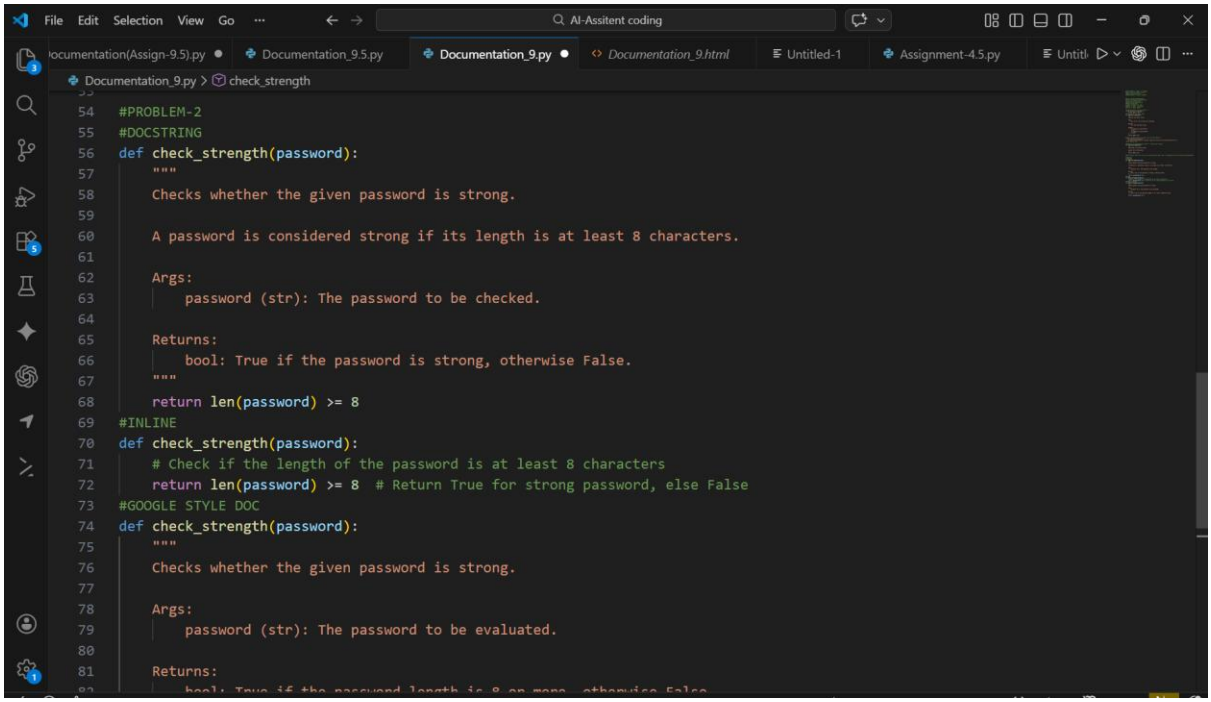
PROBLEM-2

PROMPT:

def check_strength(password):

return len(password) >= 8

CODE:



OBERVATION:

Docstring:
Gives a general description of the function and its purpose. Useful for understanding what the function does.

Inline comments:
Explain the logic step by step. Helpful for developers but not suitable for generating external documentation.

Google-style documentation:
Provides clear, structured, and detailed information about inputs and outputs. Best for maintaining clarity in security-related functions.

PROBLEM-3

PROMPT: reate a module math_utils.py with functions:

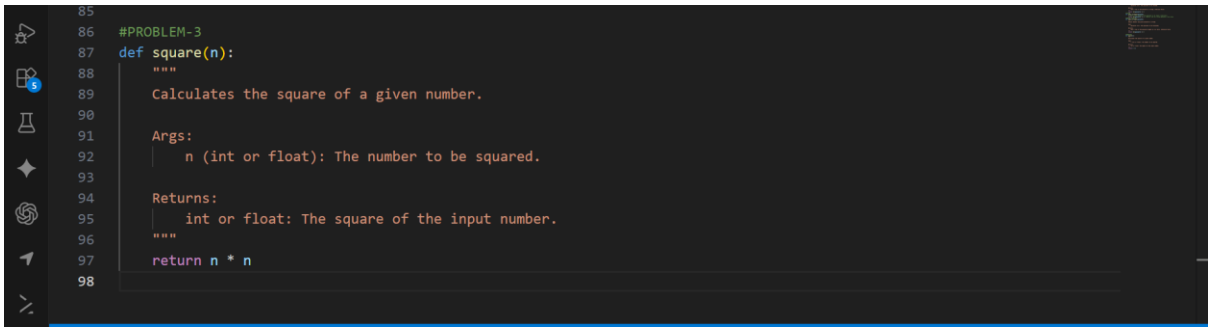
square(n)

cube(n)

factorial(n)

Generate docstrings automatically using AI tools.

CODE:



```python
#PROBLEM-3
def square(n):
    """
    Calculates the square of a given number.

    Args:
        n (int or float): The number to be squared.

    Returns:
        int or float: The square of the input number.
    """
    return n * n
```

```python
#PROBLEM-3
def square(n):
    """
    Calculates the square of a given number.

    Args:
        n (int or float): The number to be squared.

    Returns:
        int or float: The square of the input number.
    """
    return n * n
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

wrote Documentation_9.html
PS C:\Users\ARSHA THALLAPALLY\OneDrive\Desktop\AI-Assitent coding> python -m pydoc -w  Documentation_9
wrote Documentation_9.html
PS C:\Users\ARSHA THALLAPALLY\OneDrive\Desktop\AI-Assitent coding>
```

OBSERVATION:

Automatically generated docstrings provide a quick and consistent way to describe function purpose, parameters, and return values. They reduce manual

effort, improve readability, and ensure uniform documentation across the module.

PROBLEM-4

PROMPT:

mark_present(student)

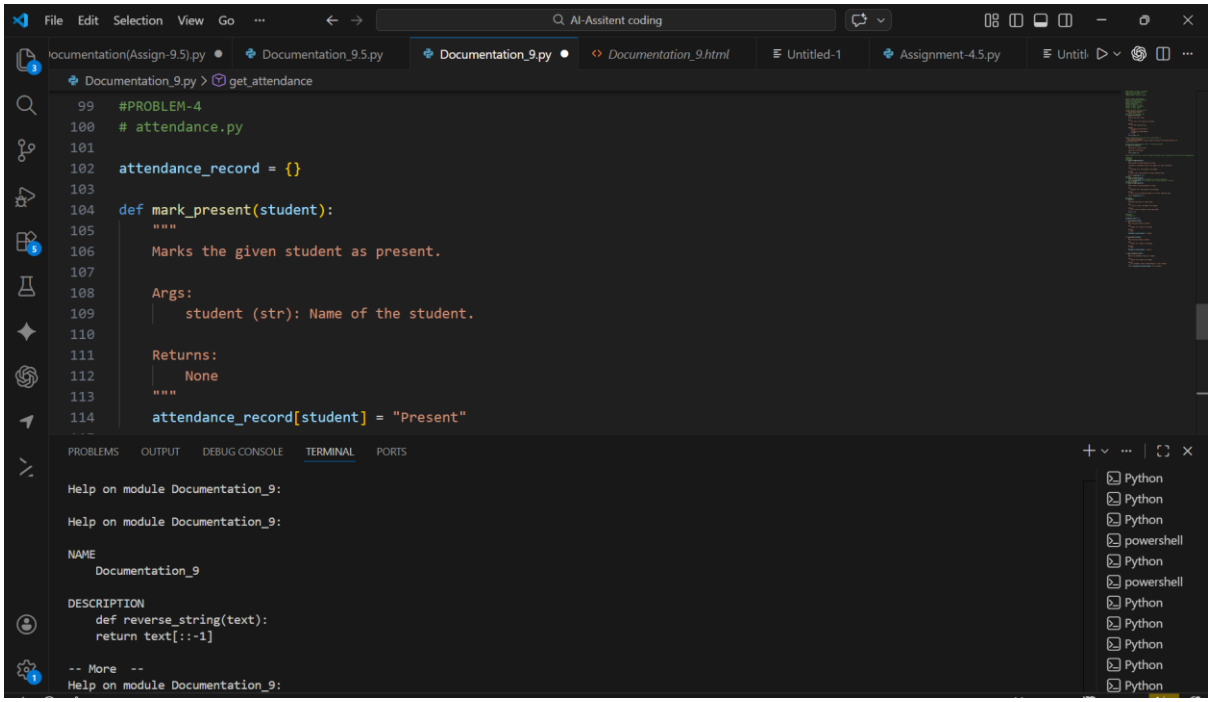
mark_absent(student)

get_attendance(student)

Add proper docstrings.

Generate and view documentation in terminal and browse

CODE:



OBSERVATION:

When documentation is generated using pydoc, it can be viewed directly in the terminal or exported as an HTML file and opened in a browser. This shows that docstrings are successfully converted into readable documentation.

PROBLEM-5

PROMPT:

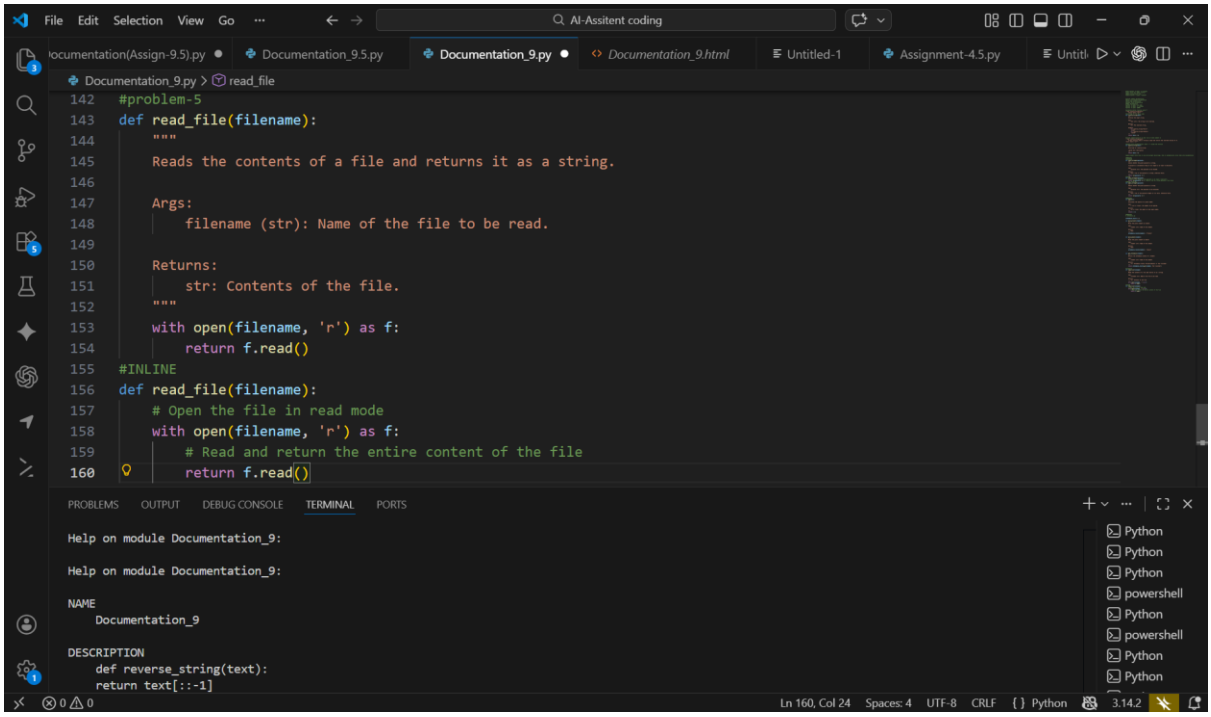
def read_file(filename):

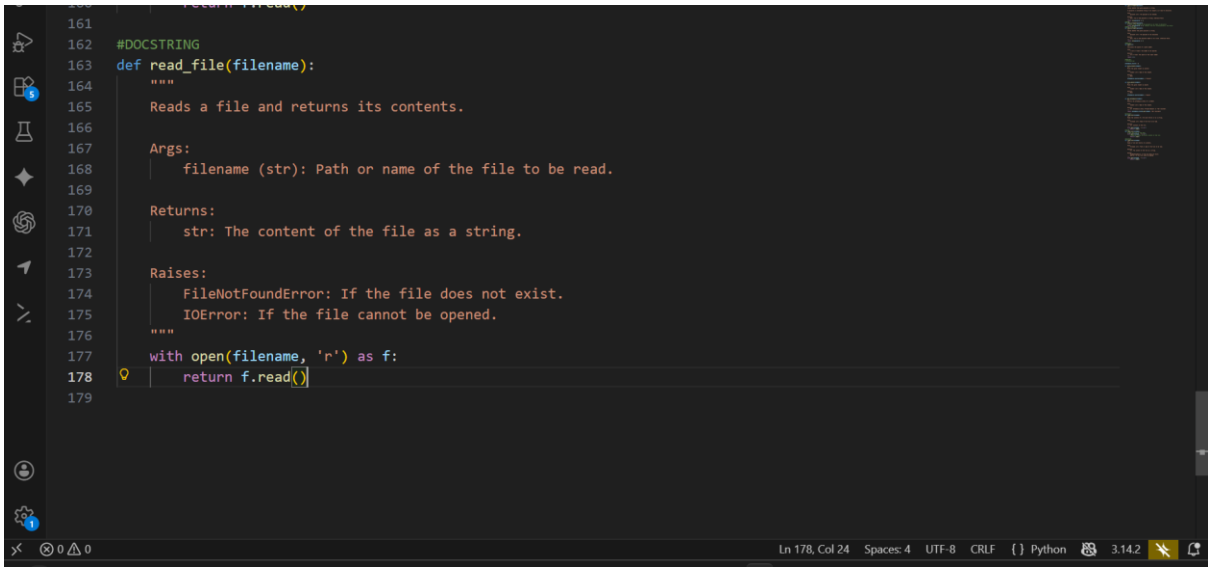
with open(filename, 'r') as f:

return f.read()

Write documentation using all three formats.

CODE:





OBSERVATION:

Docstring is added by writing a description inside triple quotes below the function header.
Inline comments are added using # near important lines of code.
Google-style documentation is added by writing a structured docstring with Args and Returns sections.