

## Assignment-9.5

2303A51600

ARSHA VARDHINI

### PROBLEM-1

#### PROMPT:

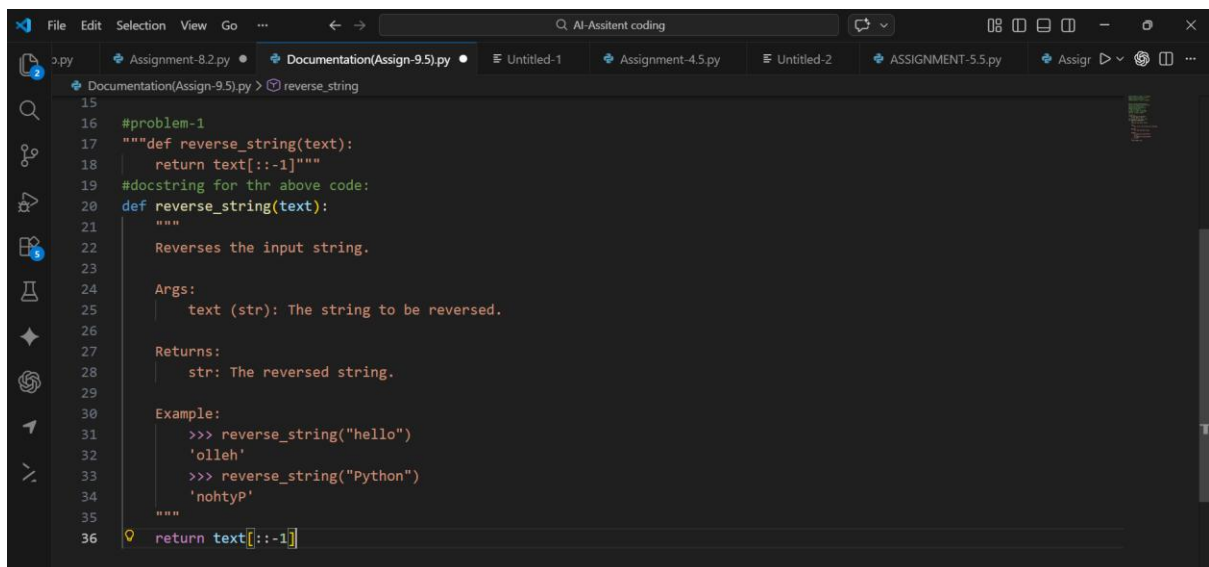
```
def reverse_string(text):
```

```
    return text[::-1]
```

Write documentation in:

- (a) Docstring
- (b) Inline comments
- (c) Google-style documentation

#### CODE:

A screenshot of a code editor window with a dark theme. The editor shows a Python file named 'Documentation(Assign-9.5).py'. The code defines a function 'reverse\_string' with a docstring in Google style. The docstring includes a description, arguments, returns, and an example. The function body is partially visible, showing the return statement.

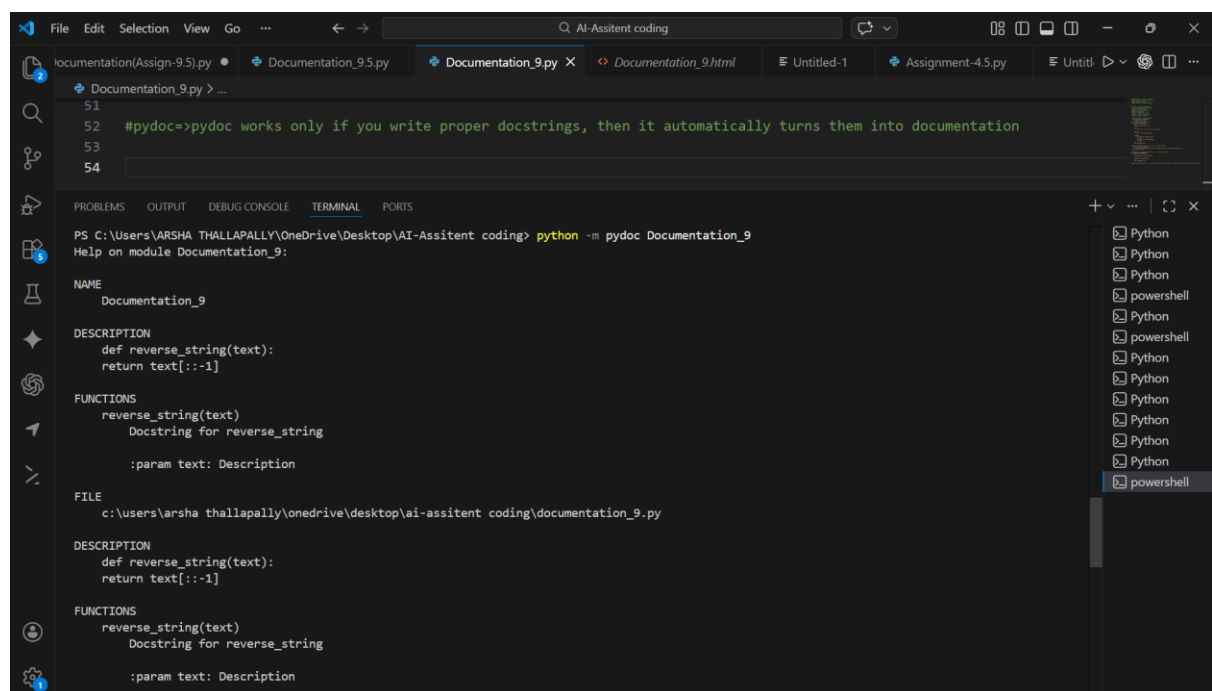
```
15
16 #problem-1
17 """def reverse_string(text):
18     return text[::-1]"""
19 #docstring for thr above code:
20 def reverse_string(text):
21     """
22     Reverses the input string.
23
24     Args:
25         text (str): The string to be reversed.
26
27     Returns:
28         str: The reversed string.
29
30     Example:
31         >>> reverse_string("hello")
32         'olleh'
33         >>> reverse_string("Python")
34         'nohtyP'
35     """
36     return text[::-1]
```

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

```

43 #Google-style documentation
44 def reverse_string(text):
45     """
46     Docstring for reverse_string
47
48     :param text: Description
49     """
50     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.

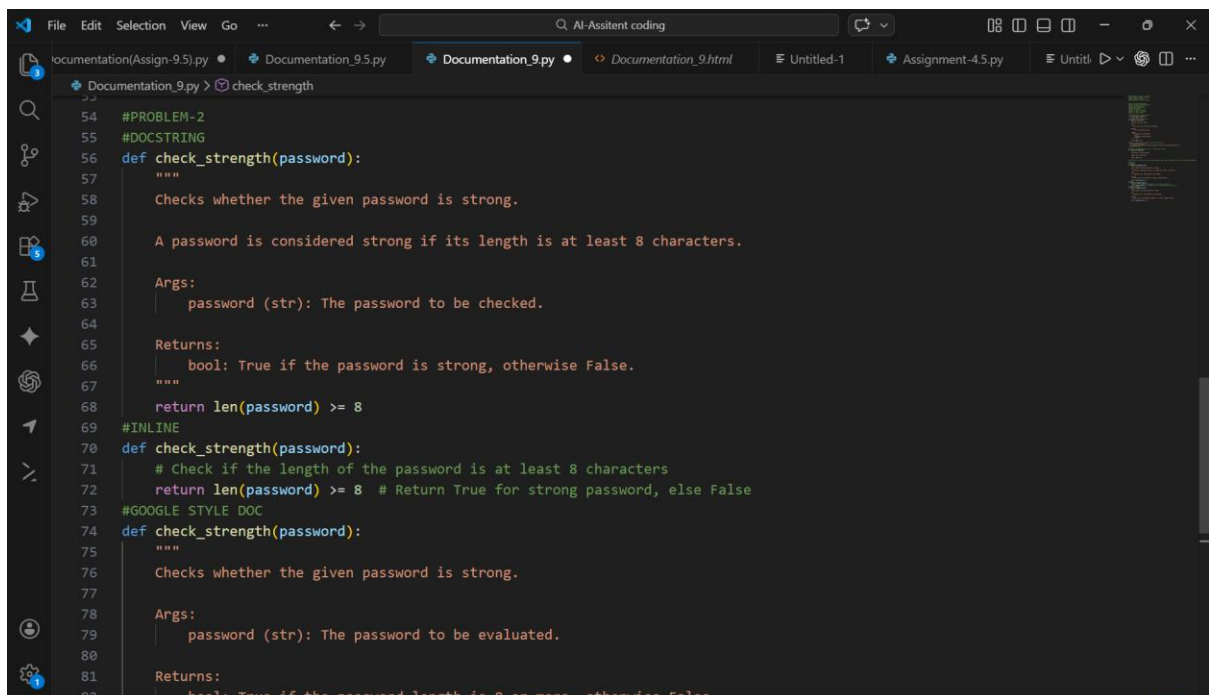
## PROBLEM-2

### PROMPT:

```
def check_strength(password):
```

```
    return len(password) >= 8
```

### CODE:



```
54 #PROBLEM-2
55 #DOCSTRING
56 def check_strength(password):
57     """
58     Checks whether the given password is strong.
59
60     A password is considered strong if its length is at least 8 characters.
61
62     Args:
63         password (str): The password to be checked.
64
65     Returns:
66         bool: True if the password is strong, otherwise False.
67     """
68     return len(password) >= 8
69
70 #INLINE
71 def check_strength(password):
72     # Check if the length of the password is at least 8 characters
73     return len(password) >= 8 # Return True for strong password, else False
74
75 #GOOGLE STYLE DOC
76 def check_strength(password):
77     """
78     Checks whether the given password is strong.
79
80     Args:
81         password (str): The password to be evaluated.
82
83     Returns:
84         bool: True if the password length is 8 or more, otherwise False.
```

### OBSERVATION:

#### 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:

```
85
86 #PROBLEM-3
87 def square(n):
88     """
89     Calculates the square of a given number.
90
91     Args:
92     |   n (int or float): The number to be squared.
93
94     Returns:
95     |   int or float: The square of the input number.
96     """
97     return n * n
98
```

```
85
86 #PROBLEM-3
87 def square(n):
88     """
89     Calculates the square of a given number.
90
91     Args:
92     |   n (int or float): The number to be squared.
93
94     Returns:
95     |   int or float: The square of the input number.
96     """
97     return n * n
98
```

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>
```

Python  
Python  
Python  
powershell  
Python  
powershell  
Python  
Python

Ln 98, Col 1 Spaces: 4 UTF-8 CRLF {} Python 3.14.2

## 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)

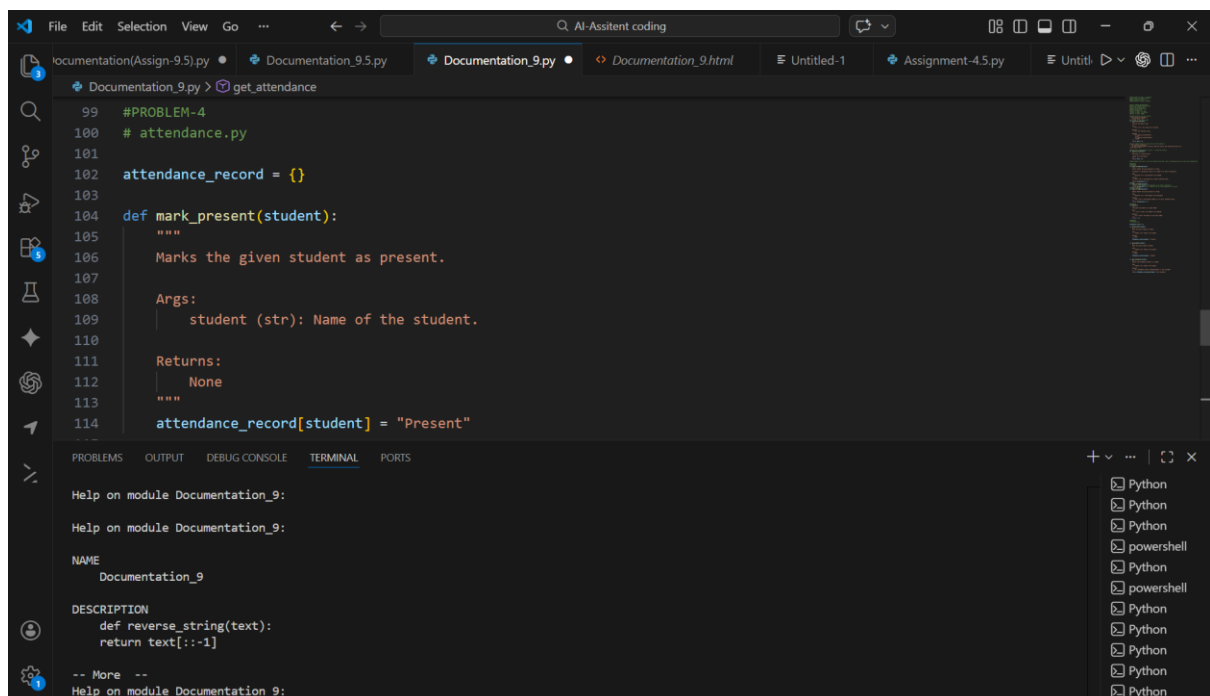
mark\_absent(student)

get\_attendance(student)

Add proper docstrings.

Generate and view documentation in terminal and browse

### CODE:



```
99 #PROBLEM-4
100 # attendance.py
101
102 attendance_record = {}
103
104 def mark_present(student):
105     """
106     Marks the given student as present.
107
108     Args:
109         student (str): Name of the student.
110
111     Returns:
112         None
113     """
114     attendance_record[student] = "Present"
```

Help on module Documentation\_9:

Help on module Documentation\_9:

NAME

Documentation\_9

DESCRIPTION

def reverse\_string(text):  
 return text[::-1]

-- More --

Help on module Documentation\_9:

### 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:

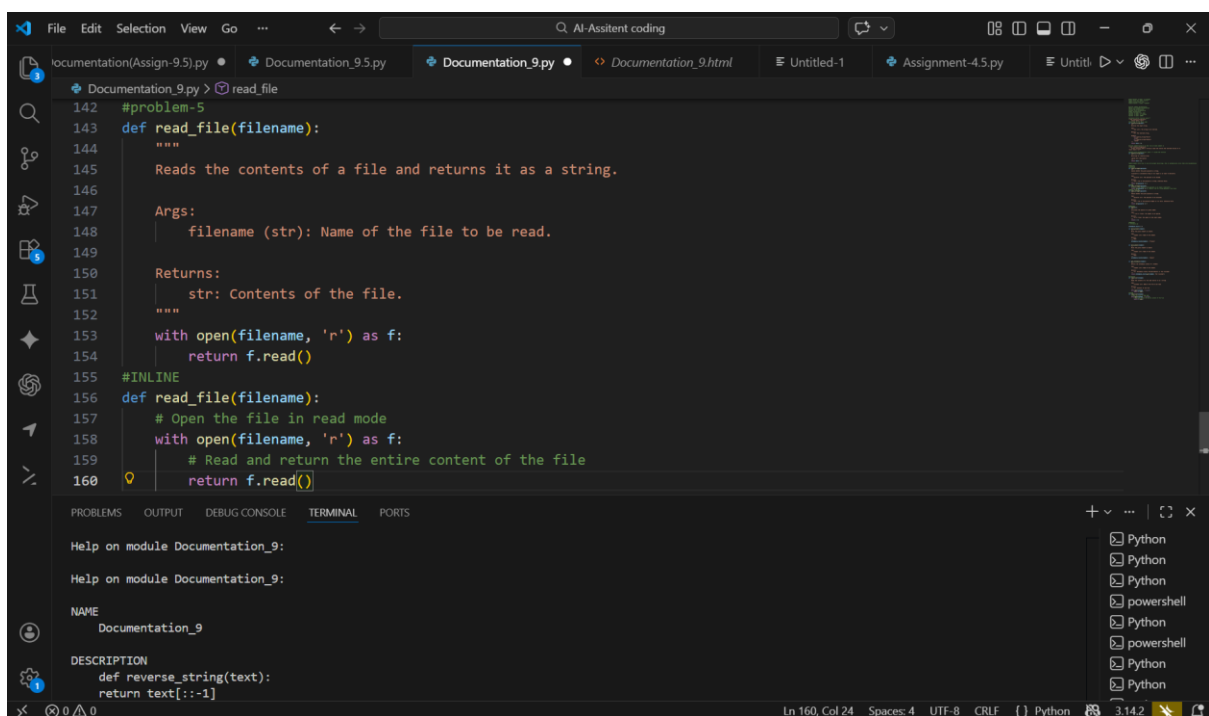
```
def read_file(filename):
```

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

```
    return f.read()
```

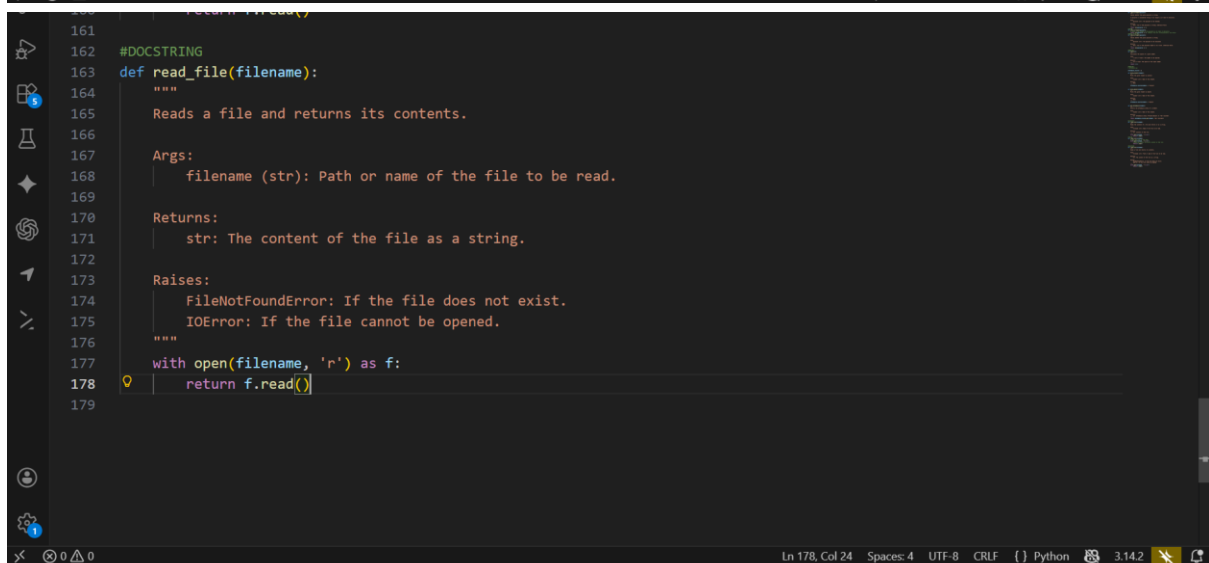
Write documentation using all three formats.

### CODE:



The screenshot shows a VS Code editor window with the file 'Documentation\_9.py' open. The code defines a function `read_file` with inline documentation. The documentation includes a docstring, an 'Args' section, and a 'Returns' section. The function uses `with open(filename, 'r') as f:` to open the file and `return f.read()` to read its contents. The status bar at the bottom indicates 'Ln 160, Col 24', 'Spaces: 4', 'UTF-8', 'CRLF', and 'Python 3.14.2'.

```
142 #problem-5
143 def read_file(filename):
144     """
145     Reads the contents of a file and returns it as a string.
146
147     Args:
148         filename (str): Name of the file to be read.
149
150     Returns:
151         str: Contents of the file.
152     """
153     with open(filename, 'r') as f:
154         return f.read()
155 #INLINE
156 def read_file(filename):
157     # Open the file in read mode
158     with open(filename, 'r') as f:
159         # Read and return the entire content of the file
160         return f.read()
```



The screenshot shows a VS Code editor window with the file 'Documentation\_9.py' open. The code defines a function `read_file` with docstring documentation. The documentation includes a docstring, an 'Args' section, a 'Returns' section, and a 'Raises' section. The function uses `with open(filename, 'r') as f:` to open the file and `return f.read()` to read its contents. The status bar at the bottom indicates 'Ln 178, Col 24', 'Spaces: 4', 'UTF-8', 'CRLF', and 'Python 3.14.2'.

```
161 #problem-5
162 #DOCSTRING
163 def read_file(filename):
164     """
165     Reads a file and returns its contents.
166
167     Args:
168         filename (str): Path or name of the file to be read.
169
170     Returns:
171         str: The content of the file as a string.
172
173     Raises:
174         FileNotFoundError: If the file does not exist.
175         IOError: If the file cannot be opened.
176     """
177     with open(filename, 'r') as f:
178         return f.read()
179
```

## 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.