

Assignment-9.5

2303A51600

ARSHA VARDHINI

PROBLEM-1

PROMPT:

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

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

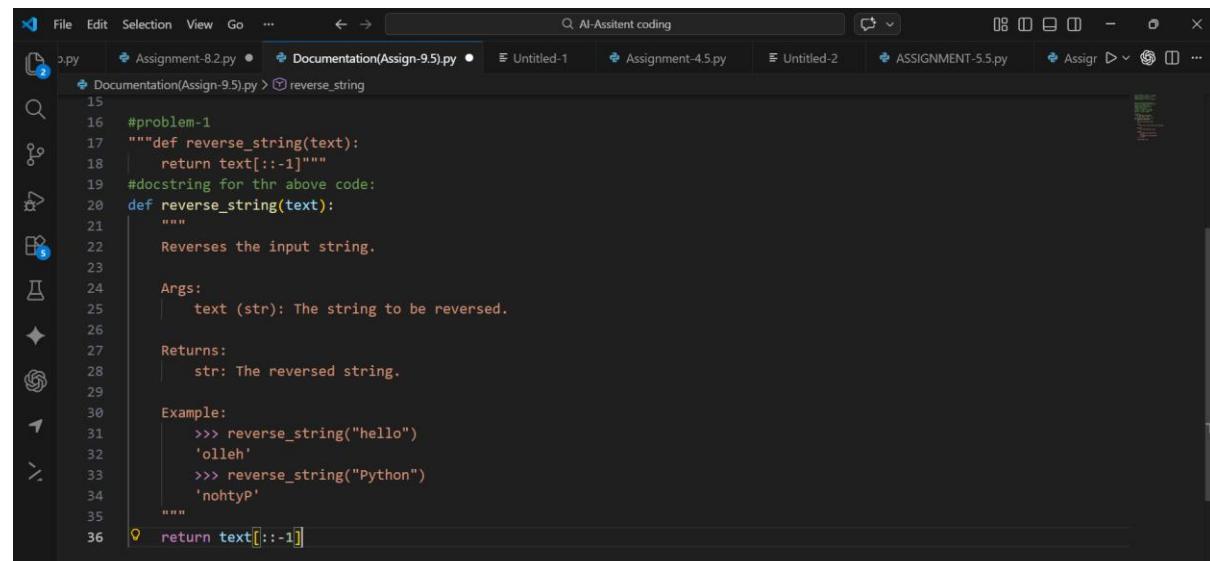
Write documentation in:

(a) Docstring

(b) Inline comments

(c) Google-style documentation

CODE:



```
File Edit Selection View Go ... AI-Assitant coding
Documentation(Assign-9.5).py > reverse_string
15
16     #problem-1
17     """def reverse_string(text):
18         return text[::-1]"""
19     #docstring for the 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]
```

The screenshot shows a Visual Studio Code (VS Code) interface with the following details:

- File Explorer:** On the left, it shows a file tree with several Python files and a powershell folder.
- Editor:** The main editor area displays the content of `Documentation_9.py`. The code defines a function `reverse_string` which returns the text reversed. A docstring is present, and the function is annotated with `:param text: Description`.
- Terminal:** The terminal tab shows the command `python -m pydoc Documentation_9` being run, followed by the generated documentation for the module.
- Output:** The output tab shows the generated documentation for the `Documentation_9` module, including its name, description, and the definition of the `reverse_string` function with its docstring and parameter annotation.
- Status Bar:** The status bar at the bottom indicates the file path as `c:/users/arsha thallapally/onedrive/desktop/ai-assitant coding/documentation_9.py`.

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:

The screenshot shows a code editor with three tabs open: Documentation_9.py, Documentation_9.html, and Untitled-1. The Documentation_9.py tab contains the following code:

```
#PROBLEM-2  
#DOCSTRING  
def check_strength(password):  
    """  
        Checks whether the given password is strong.  
        A password is considered strong if its length is at least 8 characters.  
    Args:  
        password (str): The password to be checked.  
    Returns:  
        bool: True if the password is strong, otherwise False.  
    """  
    return len(password) >= 8  
#INLINE  
def check_strength(password):  
    # Check if the length of the password is at least 8 characters  
    return len(password) >= 8 # Return True for strong password, else False  
#GOOGLE STYLE DOC  
def check_strength(password):  
    """  
        Checks whether the given password is strong.  
    Args:  
        password (str): The password to be evaluated.  
    Returns:  
        bool: True if the password length is 8 or more, otherwise False.
```

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: Create a module `math_utils.py` with functions:

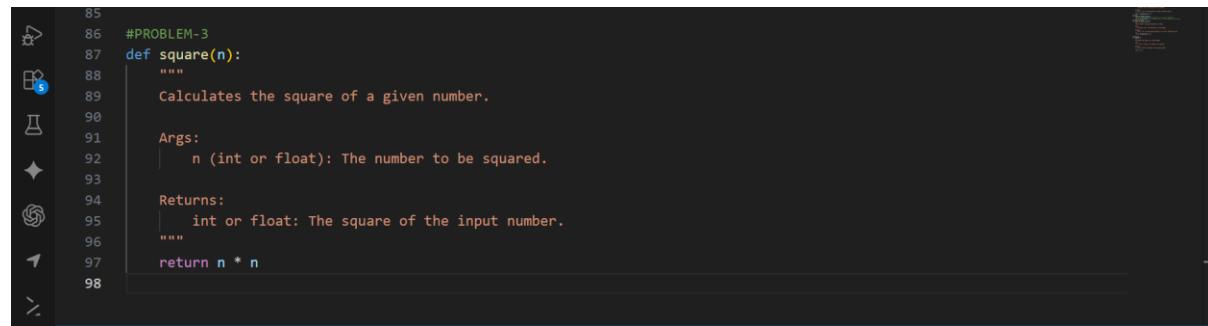
`square(n)`

`cube(n)`

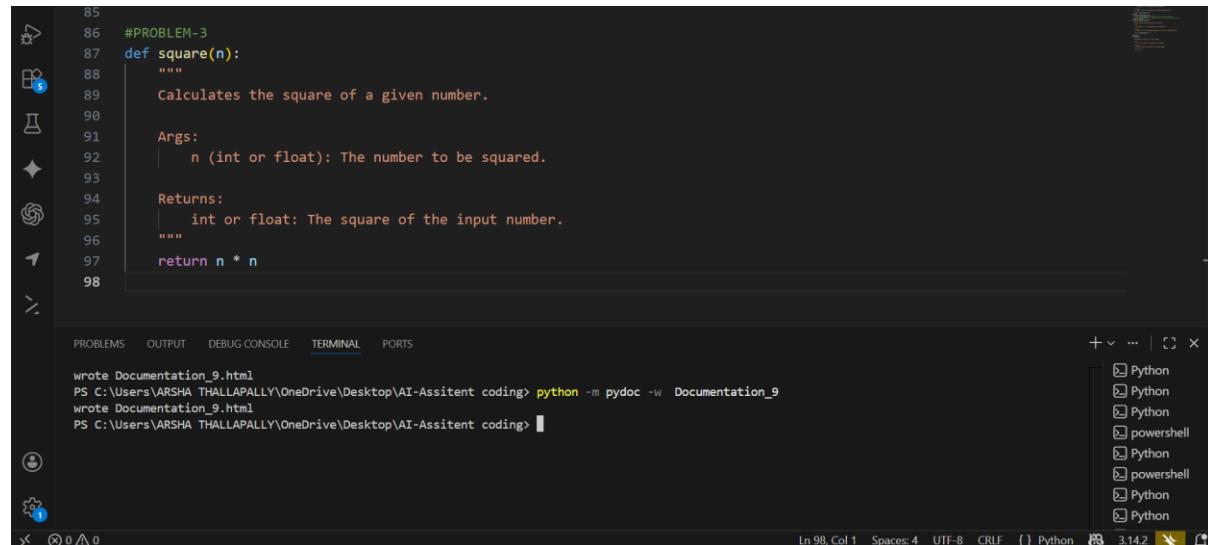
`factorial(n)`

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-Assitant coding> python -m pydoc -w Documentation_9
PS C:\Users\ARSHA THALLAPALLY\OneDrive\Desktop\AI-Assitant coding>
```

Python Python Python powershell Python Python Python Python

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:

The screenshot shows a code editor interface with several tabs at the top: Documentation(Assign-9.5).py, Documentation_9.py, Documentation_9.html, Untitled-1, Assignment-4.5.py, and Untitled. The Documentation_9.py tab is active, displaying the following 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"
```

Below the code editor, there is a terminal window showing the output of running the module:

```
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:
```

To the right of the editor, there is a sidebar showing a list of open files or tabs, all labeled with the word "Python".

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:

```
documentation(Assign-9.5.py) Documentation_9.5.py Documentation_9.py Documentation_9.html Untitled-1 Assignment-4.5.py Untitled
```

```
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()

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
```

Help on module Documentation_9:

Help on module Documentation_9:

NAME Documentation_9

DESCRIPTION

```
def reverse_string(text):
    return text[::-1]
```

Ln 160, Col 24 Spaces: 4 UTF-8 CRLF {} Python 3.14.2

```
161
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
```

Ln 178, Col 24 Spaces: 4 UTF-8 CRLF {} Python 3.14.2

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.