

AI Assisted Coding

Assignment – 9.5

Ht.no : 2303A51679

Batch : 23

Problem 1: String Utilities Function

Consider the following Python function: def

reverse_string(text):

return text[::-1]

Task:

1. Write documentation in:

o (a) Docstring o (b) Inline

comments o (c) Google-style

documentation

2. Compare the three documentation styles.

3. Recommend the most suitable style for a utility-based string

library.

```
Fridaypy > DocExample.py > ...
1  # (a) Docstring
2  def reverse_string(text):
3      """
4      This function takes a string as input and returns the reversed version of that string.
5
6      Parameters:
7      text (str): The string to be reversed.
8
9      Returns:
10     str: The reversed version of the input string.
11     """
12     return text[::-1]
13
14 # (b) Inline comments
15 def reverse_string(text):
16     # This function takes a string as input and returns the reversed version of that string.
17
18     # The input parameter 'text' is expected to be a string.
19
20     # The function uses slicing to reverse the string. The syntax text[::-1] creates a new string that is a reversed version of 'text'.
21
22     return text[::-1]
23
24 # (c) Google-style documentation
25 def reverse_string(text):
26     """
27     Reverses the input string.
28
29     Args:
30         text (str): The string to be reversed.
31
32     Returns:
33         str: The reversed version of the input string.
34     """
35     return text[::-1]
```

```

❖ PS C:\Users\Ganne\OneDrive\Desktop\Ai_Assisted_Coding\Friday.py> python -m pydoc DocExample
Help on module DocExample:

NAME
    DocExample

DESCRIPTION
    # Problem 1: String Utilities Function
    # Consider the following Python function:
    # def reverse_string(text):
    #     return text[::-1]
    # Task:

NAME
    DocExample

DESCRIPTION
    # Problem 1: String Utilities Function
    # Consider the following Python function:
    # def reverse_string(text):
    #     return text[::-1]
    # Task:
    # Consider the following Python function:
    # def reverse_string(text):
    #     return text[::-1]
    # Task:
    # Task:
    # 1. Write documentation in:
    -- More --

```

Problem 2: Password Strength Checker

Consider the function: def

check_strength(password):

return len(password) >= 8

Task:

1. Document the function using docstring, inline comments, and Google style.
2. Compare documentation styles for security-related code.
3. Recommend the most appropriate style.

```
Friday.py > DocExample.py -
41 # (a) Docstring
42 def check_strength(password):
43     """
44     This function checks the strength of a password by verifying if it is at least 8 characters long.
45
46     Parameters:
47     password (str): The password to be checked.
48
49     Returns:
50     bool: True if the password is strong (at least 8 characters), False otherwise.
51     """
52     return len(password) >= 8
53 # (b) inline comments
54 def check_strength(password):
55     """ This function checks the strength of a password by verifying if it is at least 8 characters long.
56
57     # The input parameter 'password' is expected to be a string.
58
59     # The function returns True if the length of the password is greater than or equal to 8, indicating that it is strong. Otherwise, it returns False.
60
61     return len(password) >= 8:
62     """
63 # (c) Google-style documentation
64 def check_strength(password):
65     """
66     Checks the strength of a password.
67
68     Args:
69         password (str): The password to be checked.
70
71     Returns:
72         bool: True if the password is strong (at least 8 characters), False otherwise.
73     """
74     return len(password) >= 8
```

PROBLEMS DEBUG CONSOLE OUTPUT TERMINAL PORTS

PS C:\Users\Ganne\OneDrive\Desktop\Ai_Assisted_Coding\Friday.py> python -m pydoc DocExample

Help on module DocExample:

NAME

DocExample

DESCRIPTION

(a) Docstring

def reverse_string(text):

-- More --

Problem 3: Math Utilities Module

Task:

1. Create a module `math_utils.py` with functions:
 - o `square(n)` o `cube(n)`
 - o `factorial(n)`
2. Generate docstrings automatically using AI tools.
3. Export documentation as an HTML file.

```

Friday.py > math_util.py >...
1 def square(n) :
2     """Returns the square of a number.
3     demonstrates how to use docstrings in Python.
4     Parameters:
5     n (int): The number to be squared.
6     Returns:int: The square of n.
7     """
8     return n * n
9 def cube(n) :
10    """Returns the cube of a number.
11    demonstrates how to use docstrings in Python.
12    Parameters:
13    n (int): The number to be cubed.
14    Returns:int: The cube of n.
15    """
16    return n * n * n
17 def factorial(n) :
18    """Returns the factorial of a number.
19    demonstrates how to use docstrings in Python.
20    Parameters:
21    n (int): The number to calculate the factorial of.
22    Returns:int: The factorial of n.
23    """
24    if n == 0: # check if n is 0 and return 1 if it is because factorial of 0 is 1
25        return 1 # Factorial of 0 is defined to be 1
26    else:
27        return n * factorial(n - 1) # Recursive call to calculate factorial of n
28 print(square.__doc__)
29 print(cube.__doc__)
30 print(factorial.__doc__)
31
32

```

```

PS C:\Users\Ganne\OneDrive\Desktop\AI_Assisted_Coding> cd Friday.py
PS C:\Users\Ganne\OneDrive\Desktop\AI_Assisted_Coding\Friday.py> python -m pydoc Math_util
No Python documentation found for 'Math_util'.
Use help() to get the interactive help utility.
Use help(str) for help on the str class.
PS C:\Users\Ganne\OneDrive\Desktop\AI_Assisted_Coding\Friday.py> python -m pydoc math_util
Help on module math_util:

```

NAME

math_util

DESCRIPTION

```

# def square(n) :
#     """Returns the square of a number.
#     demonstrates how to use docstrings in Python.
#     Parameters:
#     n (int): The number to be squared.
#     Returns:int: The square of n.
#     """
#     return n * n
# def cube(n) :
#     """Returns the cube of a number.
#     demonstrates how to use docstrings in Python.
#     Parameters:
#     n (int): The number to be cubed.
#     Returns:int: The cube of n.
#     """
#     return n * n * n
# def factorial(n) :
#     """Returns the factorial of a number.
#     demonstrates how to use docstrings in Python.
#     Parameters:
#     n (int): The number to calculate the factorial of.
#     Returns:int: The factorial of n.
#     """
#     if n == 0: # check if n is 0 and return 1 if it is because factorial of 0 is 1
#         return 1 # Factorial of 0 is defined to be 1
#     else:
#         return n * factorial(n - 1) # Recursive call to calculate factorial of n
# print(square.__doc__)

```

Problem 4: Attendance Management Module

Task:

1. Create a module attendance.py with functions:
 - o mark_present(student)
 - o mark_absent(student)
 - o get_attendance(student)
2. Add proper docstrings.
3. Generate and view documentation in terminal and browse

```
Friday.py > math_util.py > get_attendance
"""
41 attendance = {}
42 def mark_present(student):
43     """
44     Marks a student as present in the attendance record.
45     Parameters:
46     student (str): The name of the student to be marked as present.
47     """
48     attendance[student] = 'Present'
49 def mark_absent(student):
50     """
51     Marks a student as absent in the attendance record.
52     Parameters:
53     student (str): The name of the student to be marked as absent.
54     """
55     attendance[student] = 'Absent'
56 def get_attendance(student):
57     """
58     Returns the attendance status of a student.
59     Parameters:
60     student (str): The name of the student whose attendance is to be retrieved.
61     Returns:
62     str: The attendance status of the student.
63     """
64     return attendance.get(student, 'Not Found')
```

```
PS C:\Users\Ganne\OneDrive\Desktop\AI_Assisted_Coding\Friday.py> python -m pydoc -w math_util
write math_util.html
PS C:\Users\Ganne\OneDrive\Desktop\AI_Assisted_Coding\Friday.py> & "C:\Program Files\Python312\python.exe" "C:\Users\Ganne\OneDrive\Desktop\AI_Assisted_Coding\Friday.py\math_util.py" > "C:\Users\Ganne\OneDrive\Desktop\AI_Assisted_Coding\Friday.py\math_util.html"
KeyboardInterrupt
PS C:\Users\Ganne\OneDrive\Desktop\AI_Assisted_Coding\Friday.py> python -m pydoc -w math_util
write math_util.html
PS C:\Users\Ganne\OneDrive\Desktop\AI_Assisted_Coding\Friday.py> & "C:\Program Files\Python312\python.exe" "C:\Users\Ganne\OneDrive\Desktop\AI_Assisted_Coding\Friday.py\math_util.py" > "C:\Users\Ganne\OneDrive\Desktop\AI_Assisted_Coding\Friday.py\math_util.html"
PS C:\Users\Ganne\OneDrive\Desktop\AI_Assisted_Coding\Friday.py> & "C:\Program Files\Python312\python.exe" "C:\Users\Ganne\OneDrive\Desktop\AI_Assisted_Coding\Friday.py\math_util.py"
PS C:\Users\Ganne\OneDrive\Desktop\AI_Assisted_Coding\Friday.py> python -m pydoc -p 1234
Server ready at http://localhost:1234/
Server commands: [b]rowser, [q]uit
servers: b
servers: [q]
```

math_util

```

# def square(n) :
#     """Returns the square of a number.
#     demonstrates how to use docstrings in Python.
#     Parameters:
#     n (int): The number to be squared.
#     Returns:int: The square of n.
#     """
#     return n * n
# def cube(n) :
#     """Returns the cube of a number.
#     demonstrates how to use docstrings in Python.
#     Parameters:
#     n (int): The number to be cubed.
#     Returns:int: The cube of n.
#     """
#     return n * n * n
# def factorial(n) :
#     """Returns the factorial of a number.
#     demonstrates how to use docstrings in Python.
#     Parameters:
#     n (int): The number to calculate the factorial of.
#     Returns:int: The factorial of n.
#     """
#     if n == 0: # check if n is 0 and return 1 if it is because factorial of 0 is 1
#         return 1 # factorial of 0 is defined to be 1
#     else:
#         return n * factorial(n - 1) # Recursive call to calculate factorial of n
# print(square.__doc__)
# print(cube.__doc__)
# print(factorial.__doc__)

```

Functions

```

def get_attendance(student):
    """Returns the attendance status of a student.
    Parameters:
    student (str): The name of the student whose attendance is to be retrieved.
    Returns:
    str: The attendance status of the student.

def mark_absent(student):
    """Marks a student as absent in the attendance record.
    Parameters:
    student (str): The name of the student to be marked as absent.

def mark_present(student):
    """Marks a student as present in the attendance record.
    Parameters:
    student (str): The name of the student to be marked as present.

```

Data

```

attendance = {}

```

Problem 5: File Handling Function

Consider the function: def

read_file(filename):

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

return f.read()

Task:

1. Write documentation using all three formats.
2. Identify which style best explains exception handling.
3. Justify your recommendation.

```

89 #DocString style:
90 def read_file(filename):
91     """
92     Reads the content of a file and returns it as a string.
93
94     Parameters:
95     filename (str): The name of the file to be read.
96
97     Returns:
98     str: The content of the file.
99
100     Raises:
101     FileNotFoundError: If the specified file does not exist.
102     IOError: If an I/O error occurs while reading the file.
103     """
104     try:
105         with open(filename, 'r') as f:
106             return f.read()
107     except FileNotFoundError:
108         print(f"Error: The file '{filename}' was not found.")
109         raise
110     except IOError as e:
111         print(f"An I/O error occurred: {e}")
112         raise
113 # Google style Docstring:
114 def read_file(filename):
115     """
116     Reads the content of a file and returns it as a string.
117
118     Args:
119     | filename (str): The name of the file to be read.
120
121     Returns:
122     | str: The content of the file.
123     Raises:
124     | FileNotFoundError: If the specified file does not exist.
125     | IOError: If an I/O error occurs while reading the file.
126     """
127     try:
128         with open(filename, 'r') as f:
129             return f.read()
130     except FileNotFoundError:
131         print(f"Error: The file '{filename}' was not found.")
132         raise
133     except IOError as e:

```

```
math_utility X
In[100]: math_utility.py
110 ~ except IOError as e:
111 ~     print("An I/O error occurred: {e}")
112 ~     raise
113 # Python style Docstring:
114 ~ def read_file(filename):
115 ~     """
116 ~     Reads the content of a file and returns it as a string.
117 ~
118 ~     (param filename): The name of the file to be read.
119 ~     (type filename): str
120 ~     (return): The content of the file.
121 ~     (type): str
122 ~     (raises FileNotFoundError): If the specified file does not exist..
123 ~     (raises IOError): If an I/O error occurs while reading the file.
124 ~     """
125 ~
126 ~     try:
127 ~         with open(filename, 'r') as f:
128 ~             return f.read()
129 ~     except FileNotFoundError:
130 ~         print(f"Error: The file '{filename}' was not found.")
131 ~         raise
132 ~     except IOError as e:
133 ~         print(f"An I/O error occurred: {e}")
134 ~         raise
135 ~
136 ~ # Recommendation:
137 ~ # The Google style Docstring tool explains exception handling because it clearly separates the description of the function, its parameters, return value, and exceptions in a structured
138 ~ # format. This makes it easier for developers to quickly understand the function's behavior and the potential errors that may arise, enhancing readability and maintainability of the code.
139
140 PYTHON DOCUMENTATION AUTHOR README COPY
141
142 PS C:\Users\Victor\OneDrive\Desktop\AI_Assisted_Coding\Friday.py> python -m pydoc math_util
143 help on module math_util:
144
145 NAME
146     math_util
147
148 DESCRIPTION
149     # Problem 1: Math Utilities Module
150     # Task:
151
152 NAME
153     math_util
154
155 DESCRIPTION
156     # Problem 1: Math Utilities Module
157     # Task:
158     # 1. Create a module math_util.py with functions:
```