

AI Assisted Coding

Assignment 9.5

Name: Vivek

Hallticket:2303A52009

Task_01 :

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.

Prompt:

""" generate Docstring inline comments and google style documentation for following code """

Outputs:

Inline comment& Docstring documentation

The screenshot shows a code editor window with a dark theme. It displays a Python file named 'prblm1.py' containing the following code:

```
09 > prblm1.py > reverse_string  
""" generate Docstring inline comments and google style documentation for following code """  
def reverse_string(s): # function to reverse a string  
    """Reverse a given string.  
  
    Args:  
        s (str): The string to be reversed.  
  
    Returns:  
        str: The reversed string.  
    """  
    return s[::-1] # return the reversed string using slicing
```

Below the code editor, there is a navigation bar with tabs: PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, PORTS, and POSTMAN CONSOLE.

PyDoc Documentation

Import favorites | Dashboard | Assignment - GDB... | Full Stack Develop... | Practice | Geeksfor... | Practice | Geeksfor...

Python 3.13.7 [tags/v3.13.7:bce1c3, MSC v.1944 64 bit (AMD64)]
Windows-11

quest1

generate Docstring inline comments and google style documentation for following code

Functions

```
reverse_string(s)
    Reverse a given string.

    Args:
        s (str): The string to be reversed.

    Returns:
        str: The reversed string.
```

Task_02

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.

Output:

Inline and Docstring Documentation:

```
test2.py > ...  
""" generate Docstring inline comments and google style documentation for following code """  
def check_strength(password):# function to check the strength of a password  
    """Check if the given password is strong.  
  
    A strong password is defined as having at least 8 characters.  
  
    Args:  
        password (str): The password to be checked."""  
  
    return len(password) >= 8 # return True if the password is strong, otherwise return False  
# Example usage:  
password = "my_secure_password"  
if check_strength(password):  
    print("The password is strong.")  
else:  
    print("The password is weak.")
```

pyDoc Documentation:

```
PS C:\Users\Vivek\OneDrive\Desktop\lab_09> python -m pydoc quest2
The password is strong.
Help on module quest2:

NAME
    quest2 - generate Docstring inline comments and google style documentation for following code

FUNCTIONS
    check_strength(password)
        Check if the given password is strong.

        A strong password is defined as having at least 8 characters.
```

```
△ 0 ⓘ 2 ~~~~~~ PS C:\Users\Vivek\OneDrive\Desktop\lab_09> python -m pydoc -w quest2
KeyboardInterrupt
● PS C:\Users\Vivek\OneDrive\Desktop\lab_09> python -m pydoc -w quest2
The password is strong.
wrote quest2.html
❖ PS C:\Users\Vivek\OneDrive\Desktop\lab_09> python -m pydoc -p 3000
Server ready at http://localhost:3000/
Server commands: [b]rowser, [q]uit
server> □
```

The screenshot shows a browser window with multiple tabs open, all related to Pydoc documentation. The active tab is titled 'quest2'. The page content includes a header with 'Module Index : Topics : Keywords' and a path 'c:\users\livelivek\onedrive\desktop\lab_09\quest2'. The main content area is titled 'quest2' and contains a note about generating Docstring inline comments and Google style documentation for the following code. Below this, there's a section for 'Functions' containing a code snippet for 'check_strength(password)'. The 'Data' section at the bottom contains a variable definition.

```
check_strength(password)
    Check if the given password is strong.

    A strong password is defined as having at least 8 characters.

    Args:
        password (str): The password to be checked.

password = 'my_secure_password'
```

Task-03:

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.

Code:

(Inline and docstring documentation)

(square and cube)

```
quest1.py  quest3.py ^

# quest3.py > ...
1 """ generate Docstring inline comments and google style documentation for Create
2     o square(n)
3     o cube(n)
4     o factorial(n) """
5 def square(n):
6     """Calculate the square of a number.
7
8     Args:
9         n (int or float): The number to be squared.
10
11    Returns:
12        int or float: The square of the input number.
13    """
14    return n ** 2 # return the square of the number
15 def cube(n):
16     """Calculate the cube of a number.
17
18     Args:
19         n (int or float): The number to be cubed.
20
21    Returns:
22        int or float: The cube of the input number.
23    """
24    return n ** 3 # return the cube of the number
25
26 def factorial(n):
```

```
return n ** 3 # return the cube of the number
def factorial(n):
    """Calculate the factorial of a number.

    Args:
        n (int): The number to calculate the factorial for. Must be a non-negative integer.

    Returns:
        int: The factorial of the input number.

    Raises:
        ValueError: If n is a negative integer.
    """
    if n < 0:
        raise ValueError("Factorial is not defined for negative integers.")
    elif n == 0 or n == 1:
        return 1 # return 1 for factorial of 0 and 1
    else:
        result = 1
        for i in range(2, n + 1):
            result *= i # multiply result by each integer from 2 to n
        return result # return the factorial of the number
```

PyDoc

```
q ...
q ...
DESCRIPTION
generate Docstring inline comments and google style documentation for Create a module math_utils.py with functions:
o square(n)
o cube(n)
o factorial(n)

FUNCTIONS
cube(n)
    Calculate the cube of a number.

    Args:
        n (int or float): The number to be cubed.
-- More --
```

The screenshot shows a browser window with multiple tabs open. The active tab is titled 'localhost:3000/quest3.html'. The page content is as follows:

generate Docstring inline comments and google style documentation for Create a module `math_utils.py` with functions:

- o `square(n)`
- o `cube(n)`
- o `factorial(n)`

Functions

`cube(n)`
Calculate the cube of a number.

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

Returns:
int or float: The cube of the input number.

`factorial(n)`
Calculate the factorial of a number.

Args:
n (int): The number to calculate the factorial for. Must be a non-negative integer.

Returns:
int: The factorial of the input number.

Raises:
ValueError: If n is a negative integer.

`square(n)`
Calculate the square of a number.

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

Returns:
int or float: The square of the input number.

Task_4:

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

Code:

Inline and DocString Documentation:

```

attendance = {} # dictionary to store attendance records
def mark_present(student):
    """Mark a student as present.

    Args:
        student (str): The name of the student to mark as present.
    """
    attendance[student] = 'Present' # mark the student as present in the attendance dictionary

def mark_absent(student):
    """Mark a student as absent.

    Args:
        student (str): The name of the student to mark as absent.
    """
    attendance[student] = 'Absent' # mark the student as absent in the attendance dictionary

def get_attendance(student):
    """Get the attendance status of a student.

    Args:
        student (str): The name of the student to check attendance for.

    Returns:
        str: The attendance status of the student ('Present', 'Absent', or 'Not Recorded').
    """
    return attendance.get(student, 'Not Recorded') # return the attendance status of the student, or

```

PyDoc Documentation:

```

② PS C:\Users\Vivek\OneDrive\Desktop\lab_09> python -m pydoc quest4
Present
Absent
Not Recorded
Help on module quest4:

NAME
    quest4

DESCRIPTION
    generate Docstring inline comments and google style documentation for Create a modu
tions:
    Task:
        1. Create a module attendance.py with functions:
            mark_present(student)
            mark_absent(student)
            get_attendance(student)
            Add proper docstrings.

FUNCTIONS
    get_attendance(student)
^CTraceback (most recent call last):
    ...

```

```

PS C:\Users\Vivek\OneDrive\Desktop\lab_09> python -m pydoc -w quest4
Present
Absent
Not Recorded
wrote quest4.html
PS C:\Users\Vivek\OneDrive\Desktop\lab_09> python -m pydoc -p 3000
Server ready at http://localhost:3000/
Server commands: [b]rowser, [q]uit

```

quest4

generate Docstring inline comments and google style documentation for Create a module `math_utils.py` with functions:

Task:

1. Create a module `attendance.py` with functions:

```
mark_present(student)
mark_absent(student)
get_attendance(student)
```

Add proper docstrings.

Functions

get_attendance(student)

Get the attendance status of a student.

Args:

student (str): The name of the student to check attendance for.

Returns:

str: The attendance status of the student ('Present', 'Absent', or 'Not Recorded').

mark_absent(student)

Mark a student as absent.

Args:

student (str): The name of the student to mark as absent.

mark_present(student)

Mark a student as present.

Args:

student (str): The name of the student to mark as present.

Data

```
attendance = {'Alice': 'Present', 'Bob': 'Absent'}
```

Task_05:

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

Inline and DocString Documentation

```
""" Consider the function:
def read_file(filename):
    with open(filename, 'r') as f:
        return f.read()
Task:
1. Write documentation using Docstring inline comments and google style documentation for the above function.
"""

def read_file(filename):
    """Read the contents of a file.

    Args:
        filename (str): The name of the file to be read.

    Returns:
        str: The contents of the file as a string.

    Raises:
        FileNotFoundError: If the specified file does not exist.
        IOError: If there is an error reading the file.
    """
    with open(filename, 'r') as f: # open the file in read mode
        return f.read() # return the contents of the file as a string

# Example usage:
try:
    content = read_file('example.txt')
    print(content)
except FileNotFoundError:
    print("The specified file was not found.")
except IOError:
    print("An error occurred while reading the file.")
```

PyDoc:

```
PS C:\Users\Vivek\OneDrive\Desktop\lab_09> python -m pydoc quest5
Vivek Lakum
```

Help on module quest5:

NAME
quest5

DESCRIPTION
Consider the function:
-- More -- █

KeyboardInterrupt

```
PS C:\Users\Vivek\OneDrive\Desktop\lab_09> python -m pydoc -w quest5
Vivek Lakum
```

wrote quest5.html

```
PS C:\Users\Vivek\OneDrive\Desktop\lab_09> python -m pydoc -p 3000
Server ready at http://localhost:3000/
Server commands: [b]rowser, [q]uit
server> █
```

Python 3.13.7 [tags/v3.13.7:bceelc3, MSC v.1944 64 bit (AMD64)]
Windows-11

quest5

Consider the function:

```
def read_file(filename):
    with open(filename, 'r') as f:
        return f.read()
```

Task:
1. Write documentation using Docstring inline comments and google style documentation for the above function.

Functions

```
read_file(filename)
    Read the contents of a file.

    Args:
        filename (str): The name of the file to be read.

    Returns:
        str: The contents of the file as a string.

    Raises:
        FileNotFoundError: If the specified file does not exist.
        IOError: If there is an error reading the file.
```

Data

```
content = 'Vivek Lakum \n'
```