|  |  |
|---|---|
| **Course Title** | AI Assisted Coding |

Name: G. Snehith Roll Number: 2303A52324 Batch: 45

**Assignment Number: 9.5**(Present assignment number)/**24**(Total number of assignments)

**Lab Experiment: Documentation Generation -Automatic documentation and code comments**

## Lab Objectives

1. To understand automatic documentation generation.
2. To generate code comments and docstrings using AI tools.
3. To learn the importance of documentation in software development.

---

## Lab Outcomes

1. Students will be able to generate documentation automatically for code.
2. Students will be able to add clear comments and docstrings to programs.
3. Students will be able to improve code readability and maintainability using documentation.

## Problem 1: String Utilities Function

Consider the following Python function:

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

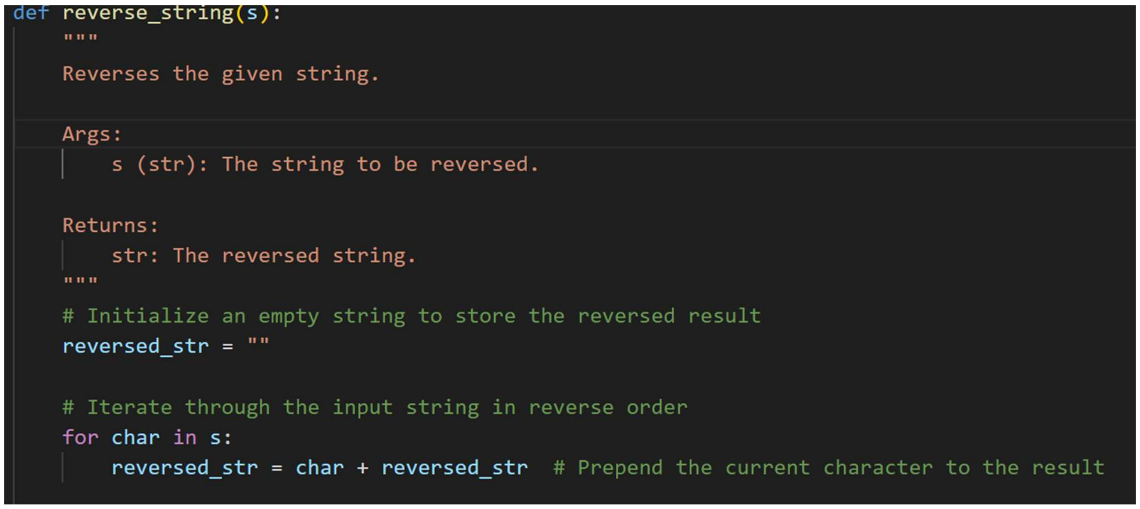
**Task:**

1. Write documentation in:
   - (a) Docstring
   - (b) Inline comments
   - (c) Google-style documentation
2. Compare the three documentation styles.

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

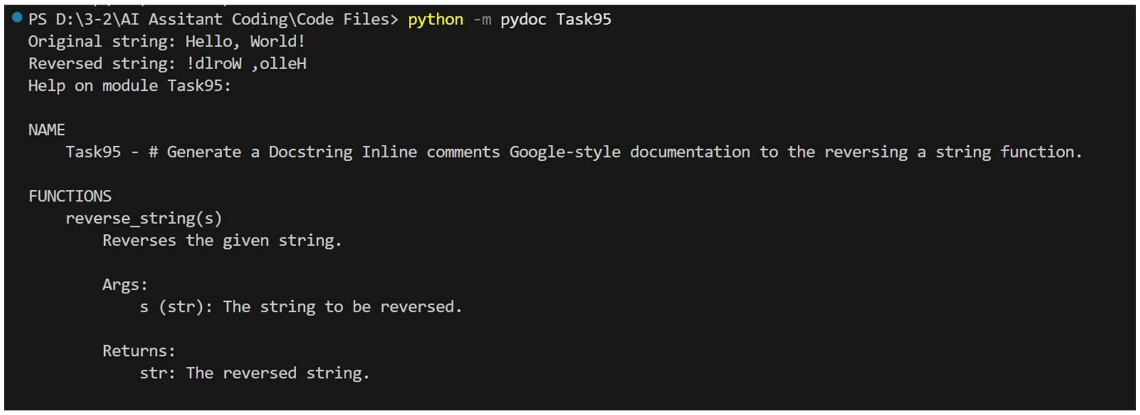
Prompt: # Generate a Docstring Inline comments Google-style documentation to the reversing a string function.

Code:

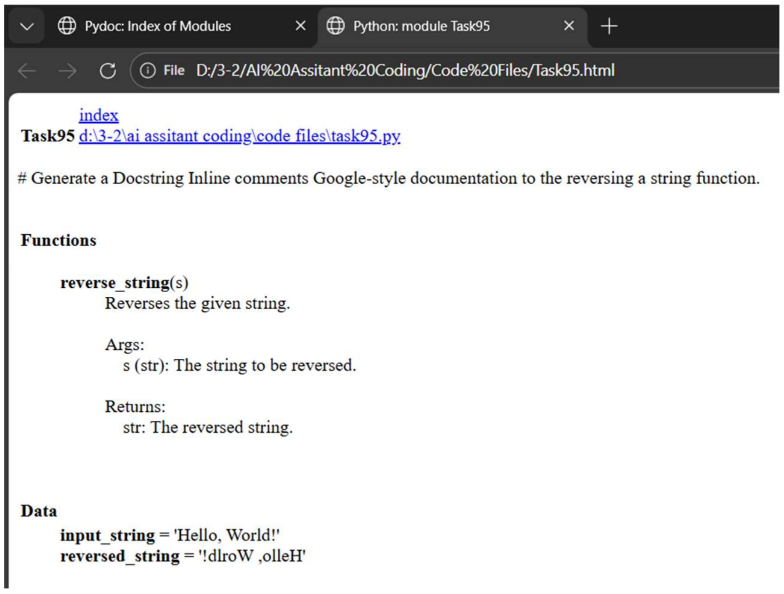


```python
def reverse_string(s):
    """
    Reverses the given string.

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

    Returns:
        str: The reversed string.
    """
    # Initialize an empty string to store the reversed result
    reversed_str = ""

    # Iterate through the input string in reverse order
    for char in s:
        reversed_str = char + reversed_str  # Prepend the current character to the result
```

Output:



```
PS D:\3-2\AI Assitant Coding\Code Files> python -m pydoc Task95
Original string: Hello, World!
Reversed string: !dlroW ,olleH
Help on module Task95:

NAME
    Task95 - # Generate a Docstring Inline comments Google-style documentation to the reversing a string function.

FUNCTIONS
    reverse_string(s)
        Reverses the given string.

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

        Returns:
            str: The reversed string.
```



index
**Task95** d:\3-2\ai assitant coding\code files\task95.py

# Generate a Docstring Inline comments Google-style documentation to the reversing a string function.

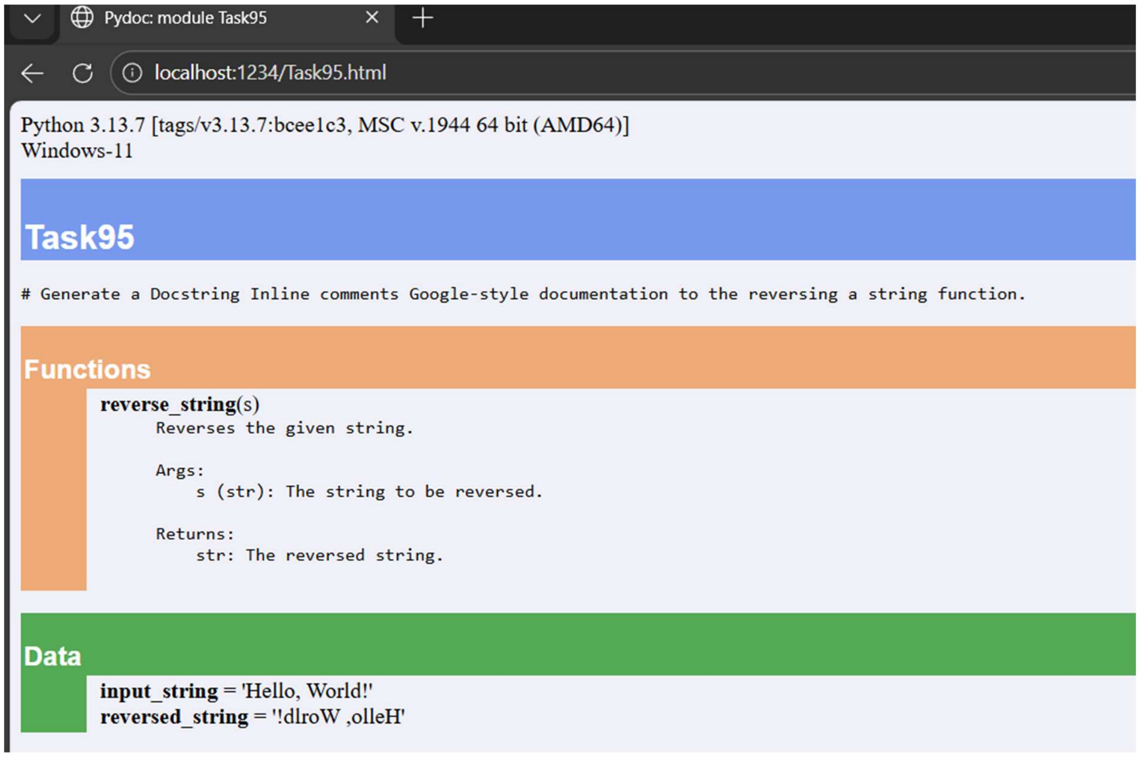
**Functions**

**reverse_string**(s)
Reverses the given string.

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

Returns:
str: The reversed string.

**Data**
**input_string** = 'Hello, World!'
**reversed_string** = '!dlroW ,olleH'



```
Python 3.13.7 [tags/v3.13.7:bcee1c3, MSC v.1944 64 bit (AMD64)]
Windows-11

Task95

# Generate a Docstring Inline comments Google-style documentation to the reversing a string function.

Functions
    reverse_string(s)
        Reverses the given string.

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

        Returns:
            str: The reversed string.

Data
    input_string = 'Hello, World!'
    reversed_string = '!dlroW ,olleH'
```

**Problem 2: Password Strength Checker**

Consider the function:

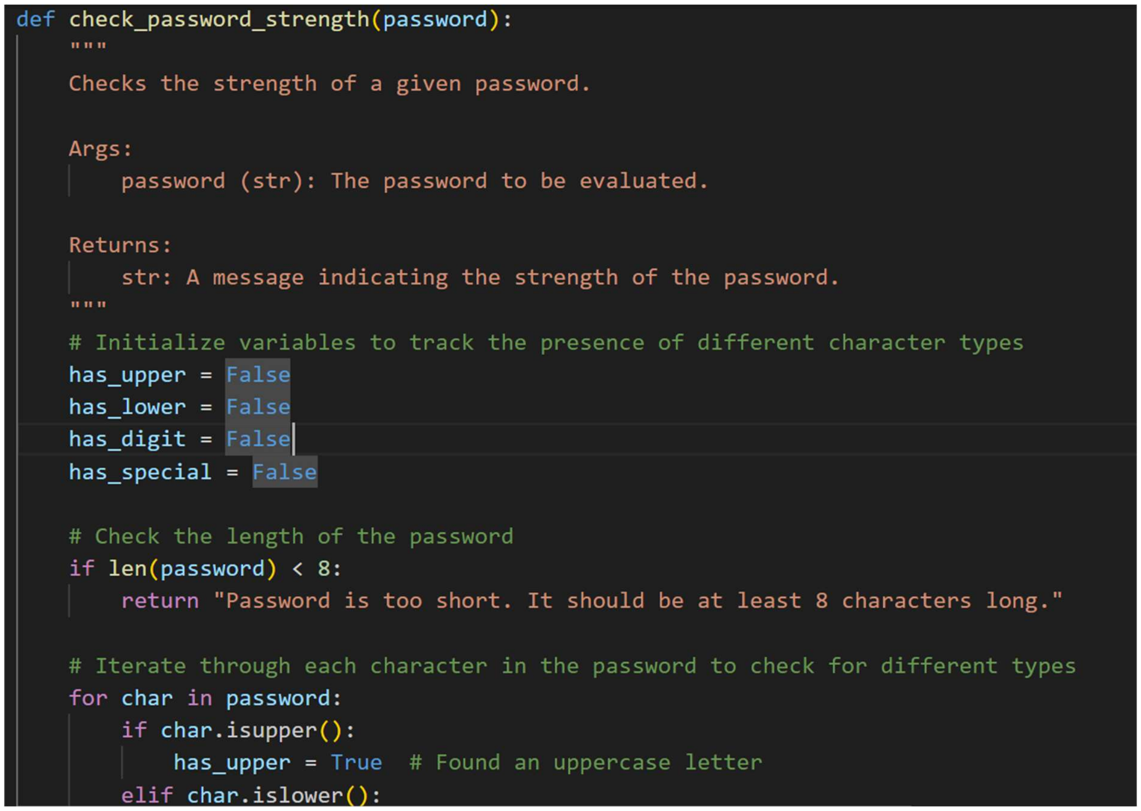
```
def check_strength(password):
    return len(password) >= 8
```

**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.

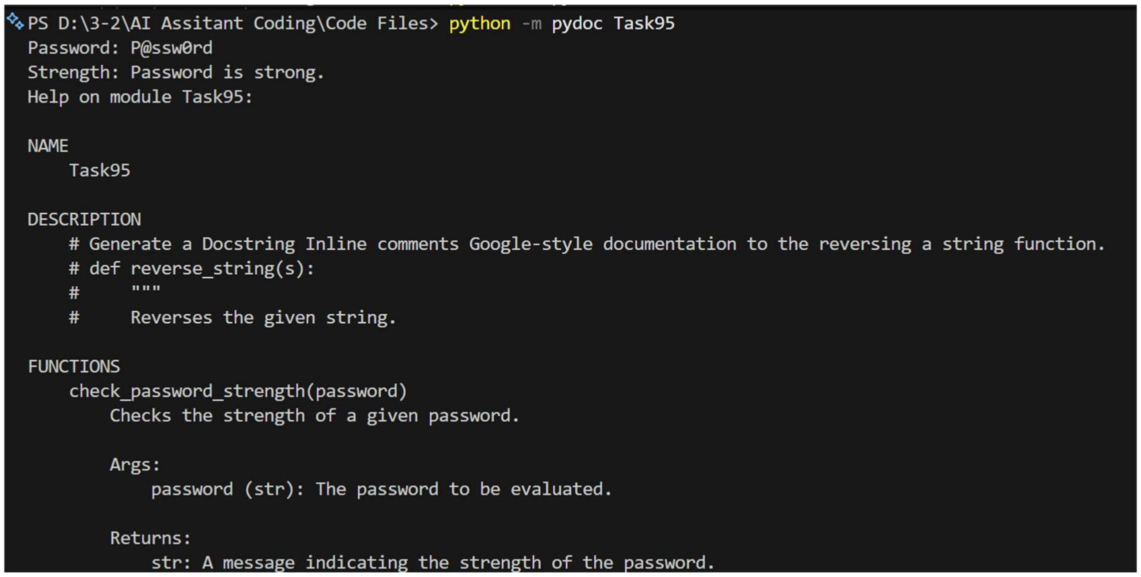
Prompt: generate a 1. Document the function using docstring, inline comments, and Google style 2. Compare documentation styles for security-related code. for Password Strength Checker function

Code:

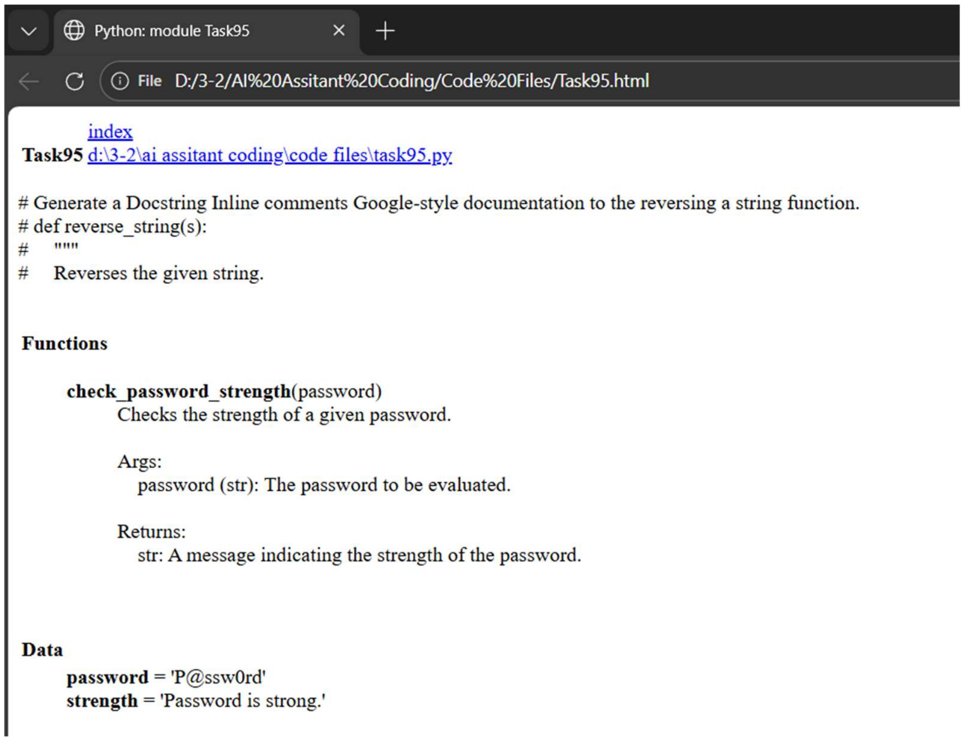


```python
def check_password_strength(password):
    """
    Checks the strength of a given password.

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

    Returns:
        str: A message indicating the strength of the password.
    """
    # Initialize variables to track the presence of different character types
    has_upper = False
    has_lower = False
    has_digit = False
    has_special = False

    # Check the length of the password
    if len(password) < 8:
        return "Password is too short. It should be at least 8 characters long."

    # Iterate through each character in the password to check for different types
    for char in password:
        if char.isupper():
            has_upper = True  # Found an uppercase letter
        elif char.islower():
```

Output:



```
PS D:\3-2\AI Assitant Coding\Code Files> python -m pydoc Task95
Password: P@ssw0rd
Strength: Password is strong.
Help on module Task95:

NAME
    Task95

DESCRIPTION
    # Generate a Docstring Inline comments Google-style documentation to the reversing a string function.
    # def reverse_string(s):
    #     """
    #     Reverses the given string.

FUNCTIONS
    check_password_strength(password)
        Checks the strength of a given password.

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

        Returns:
            str: A message indicating the strength of the password.
```



**Task95** d:\3-2\ai assitant coding\code files\task95.py

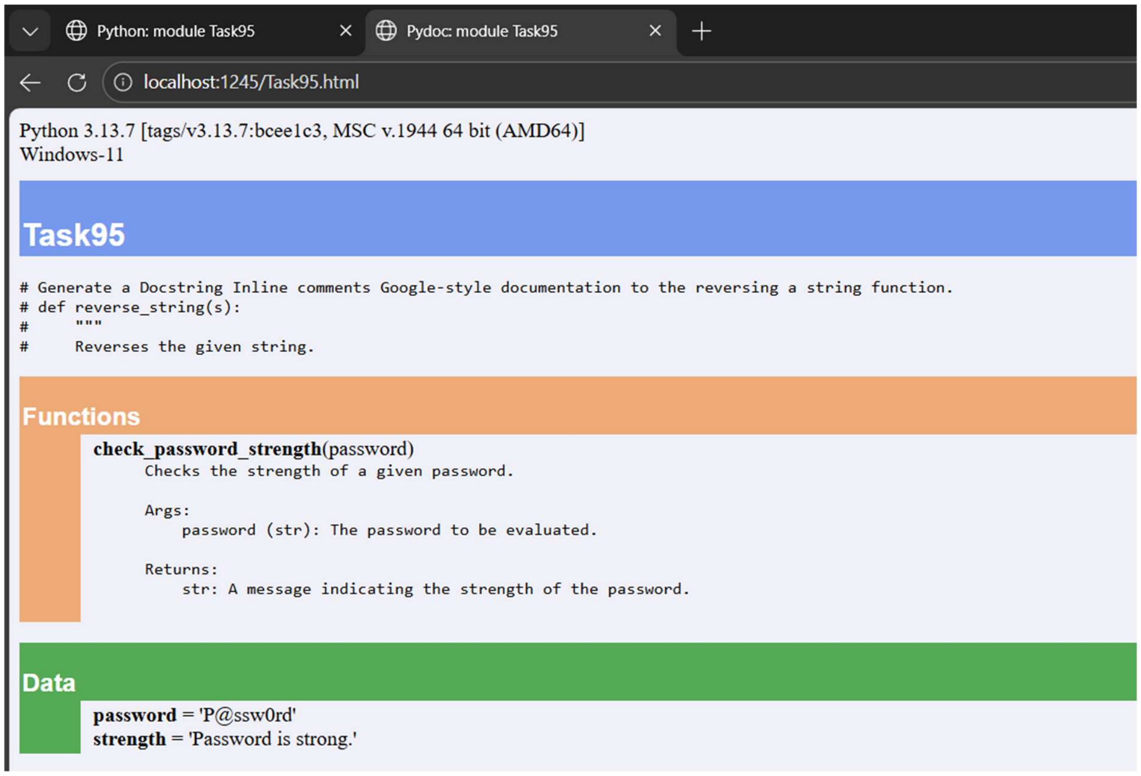
```
# Generate a Docstring Inline comments Google-style documentation to the reversing a string function.
# def reverse_string(s):
#     """
#     Reverses the given string.
```

**Functions**

```
check_password_strength(password)
    Checks the strength of a given password.

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

    Returns:
        str: A message indicating the strength of the password.
```

**Data**

**password** = 'P@ssw0rd'
**strength** = 'Password is strong.'



Python 3.13.7 [tags/v3.13.7:bcee1c3, MSC v.1944 64 bit (AMD64)]
Windows-11

**Task95**

```
# Generate a Docstring Inline comments Google-style documentation to the reversing a string function.
# def reverse_string(s):
#     """
#     Reverses the given string.
```

**Functions**

```
check_password_strength(password)
    Checks the strength of a given password.

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

    Returns:
        str: A message indicating the strength of the password.
```

**Data**

**password** = 'P@ssw0rd'
**strength** = 'Password is strong.'

**Problem 3: Math Utilities Module**

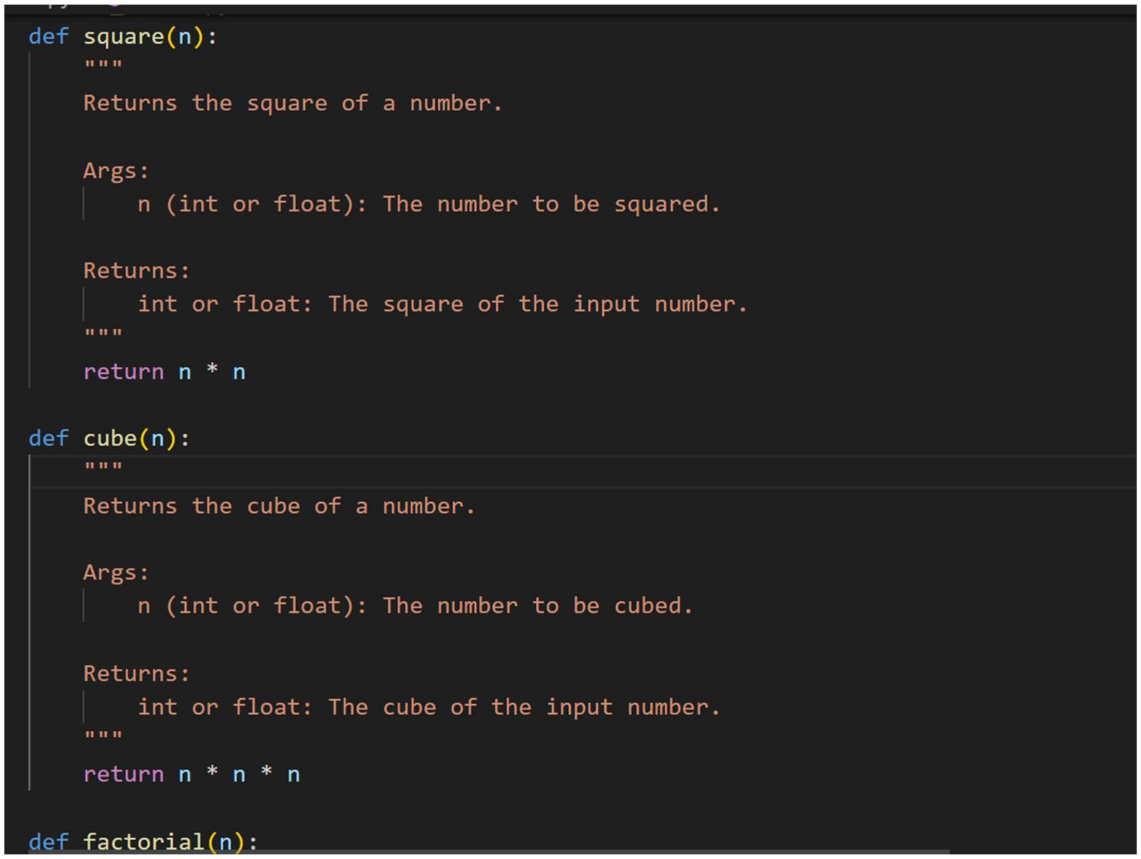
**Task:**

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

Prompt: # Generate a 1. Create a module math_utils.py with functions:,square(n),cube(n),factorial(n),

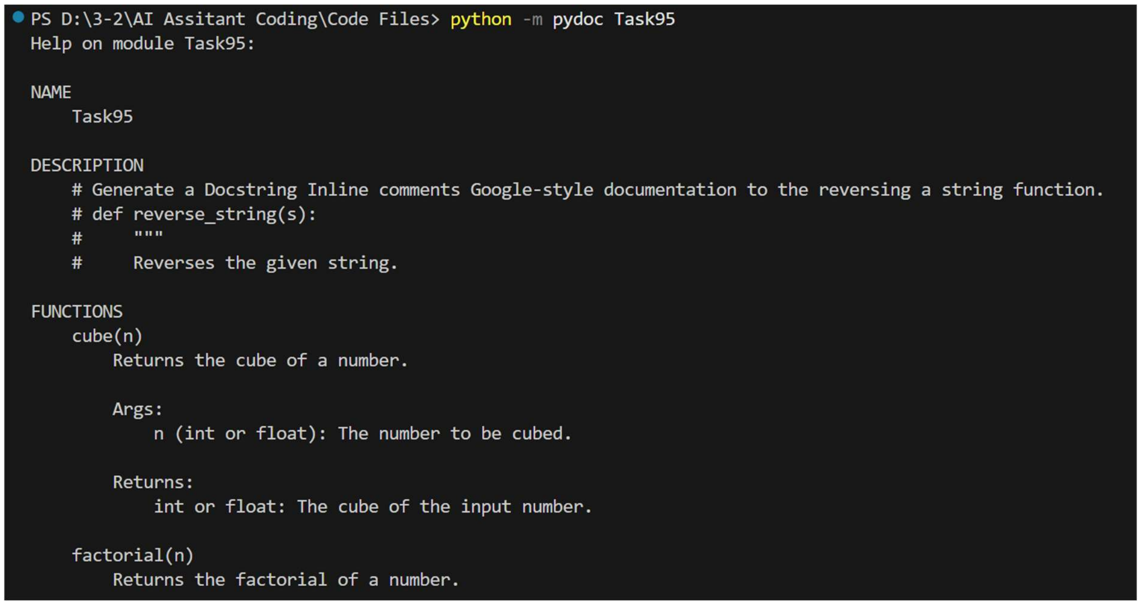
# Generate docstrings automatically

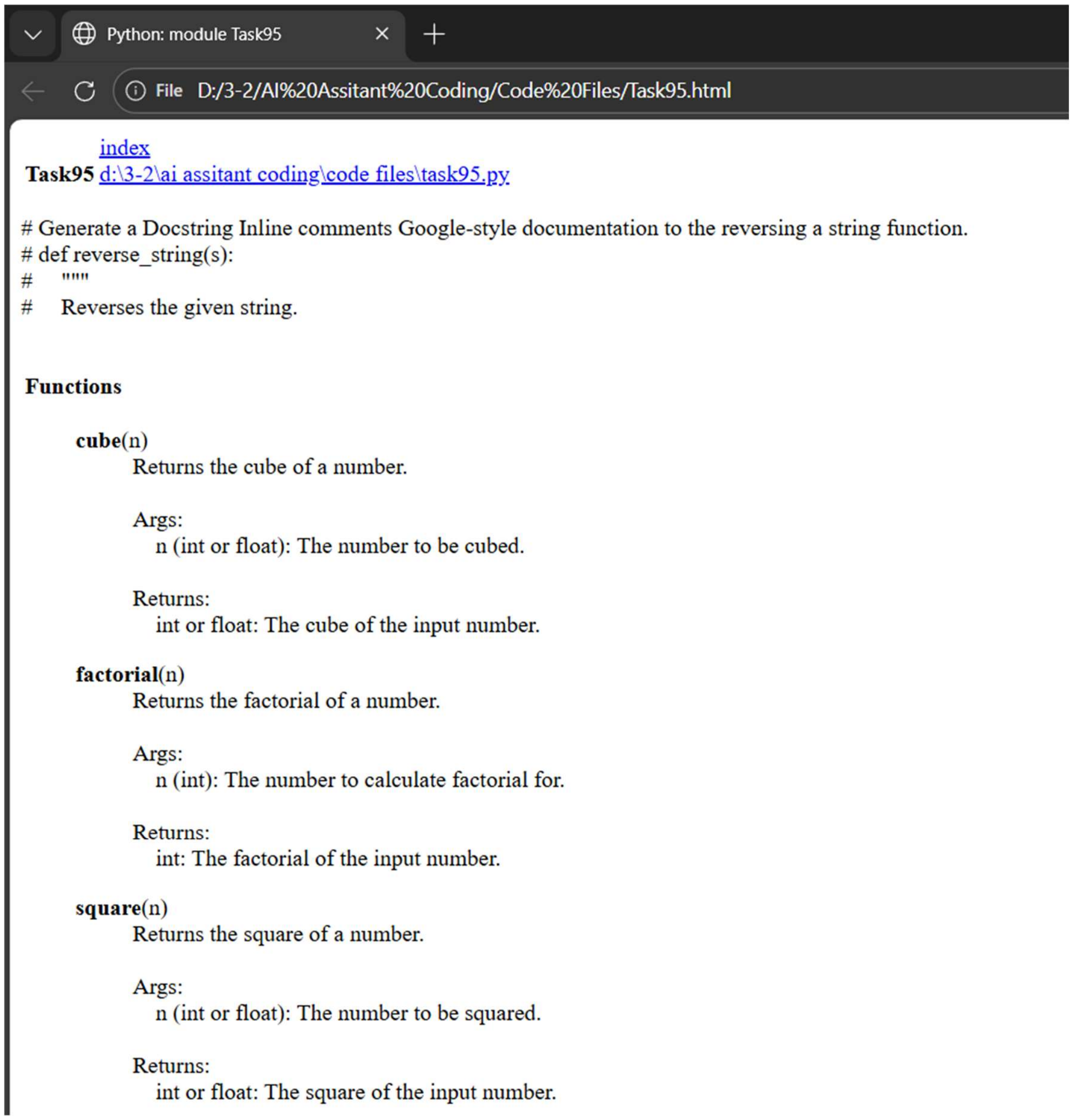
Code:



```
def square(n):
    """
    Returns 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.
    """
    return n * n

def cube(n):
    """
    Returns 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.
    """
    return n * n * n

def factorial(n):
```

Output:


```
PS D:\3-2\AI Assitant Coding\Code Files> python -m pydoc Task95
Help on module Task95:

NAME
    Task95

DESCRIPTION
    # Generate a Docstring Inline comments Google-style documentation to the reversing a string function.
    # def reverse_string(s):
    #     """
    #     Reverses the given string.

FUNCTIONS
    cube(n)
        Returns 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)
        Returns the factorial of a number.
```


Python: module Task95

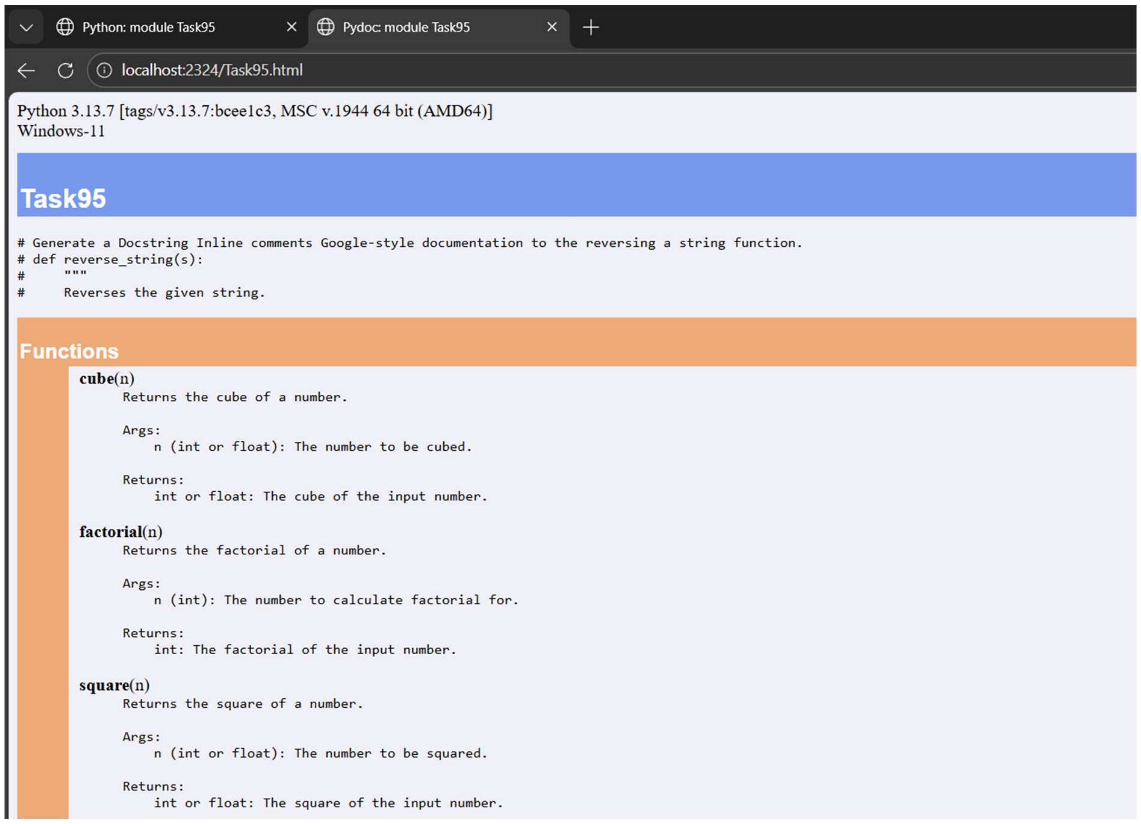
File D:/3-2/AI%20Assitant%20Coding/Code%20Files/Task95.html

index
**Task95** d:\3-2\ai assitant coding\code files\task95.py

```
# Generate a Docstring Inline comments Google-style documentation to the reversing a string function.
# def reverse_string(s):
#    """
#    Reverses the given string.
```

**Functions**

```
cube(n)
    Returns 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)
    Returns the factorial of a number.

    Args:
        n (int): The number to calculate factorial for.

    Returns:
        int: The factorial of the input number.

square(n)
    Returns 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.
```



```
Python 3.13.7 [tags/v3.13.7:bcee1c3, MSC v.1944 64 bit (AMD64)]
Windows-11

Task95

# Generate a Docstring Inline comments Google-style documentation to the reversing a string function.
# def reverse_string(s):
#     """
#     Reverses the given string.

Functions
    cube(n)
        Returns 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)
        Returns the factorial of a number.

        Args:
            n (int): The number to calculate factorial for.

        Returns:
            int: The factorial of the input number.

    square(n)
        Returns 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.
```

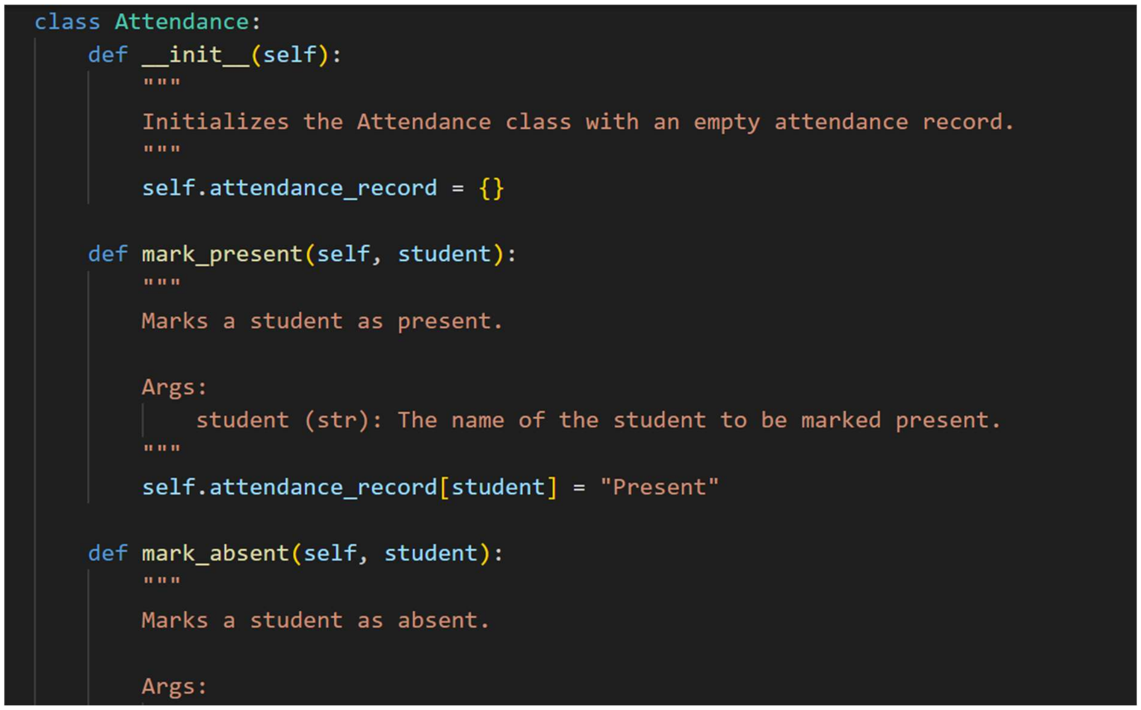
**Problem 4: Attendance Management Module**

**Task:**

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

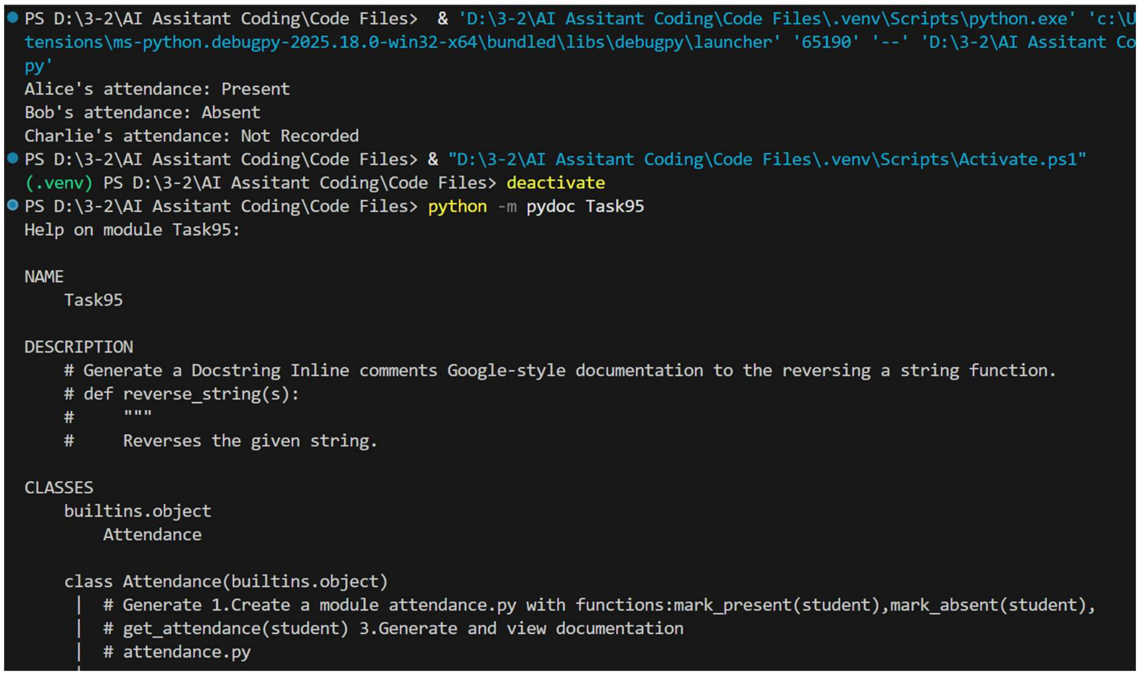
Prompt: Generate 1.Create a module attendance.py with functions:mark_present(student),mark_absent(student),
# get_attendance(student) 3.Generate and view documentation

Code:

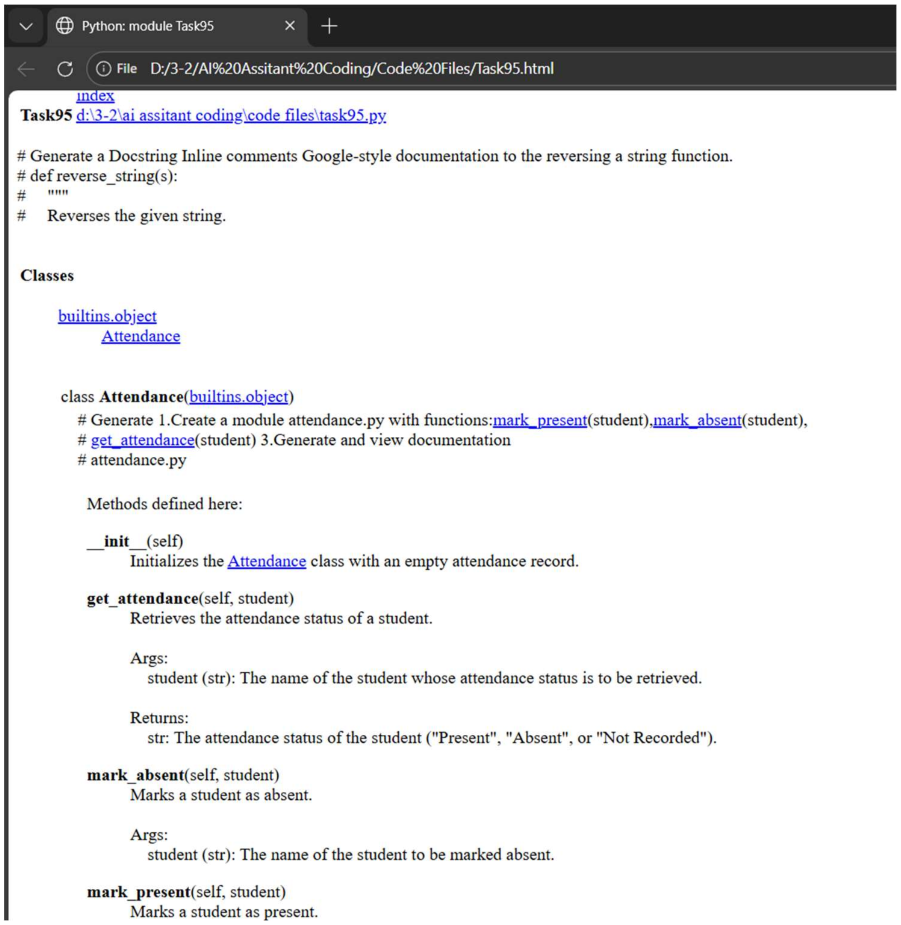


```
class Attendance:
    def __init__(self):
        """
        Initializes the Attendance class with an empty attendance record.
        """
        self.attendance_record = {}

    def mark_present(self, student):
        """
        Marks a student as present.

        Args:
            student (str): The name of the student to be marked present.
        """
        self.attendance_record[student] = "Present"

    def mark_absent(self, student):
        """
        Marks a student as absent.

        Args:
```

Output:



```
PS D:\3-2\AI Assitant Coding\Code Files>  & 'D:\3-2\AI Assitant Coding\Code Files\.venv\Scripts\python.exe' 'c:\U
tensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '65190' '--' 'D:\3-2\AI Assitant Co
py'
Alice's attendance: Present
Bob's attendance: Absent
Charlie's attendance: Not Recorded
PS D:\3-2\AI Assitant Coding\Code Files> & "D:\3-2\AI Assitant Coding\Code Files\.venv\Scripts\Activate.ps1"
(.venv) PS D:\3-2\AI Assitant Coding\Code Files> deactivate
PS D:\3-2\AI Assitant Coding\Code Files> python -m pydoc Task95
Help on module Task95:

NAME
    Task95

DESCRIPTION
    # Generate a Docstring Inline comments Google-style documentation to the reversing a string function.
    # def reverse_string(s):
    #     """
    #     Reverses the given string.

CLASSES
    builtins.object
        Attendance

    class Attendance(builtins.object)
     |  # Generate 1.Create a module attendance.py with functions:mark_present(student),mark_absent(student),
     |  # get_attendance(student) 3.Generate and view documentation
     |  # attendance.py
```


Python: module Task95

File D:/3-2/AI%20Assitant%20Coding/Code%20Files/Task95.html

index
**Task95** d:\3-2\ai assitant coding\code files\task95.py

```
# Generate a Docstring Inline comments Google-style documentation to the reversing a string function.
# def reverse_string(s):
#     """
#     Reverses the given string.
```

**Classes**

builtins.object
    Attendance

class **Attendance**(builtins.object)
```
# Generate 1.Create a module attendance.py with functions:mark_present(student),mark_absent(student),
# get_attendance(student) 3.Generate and view documentation
# attendance.py
```

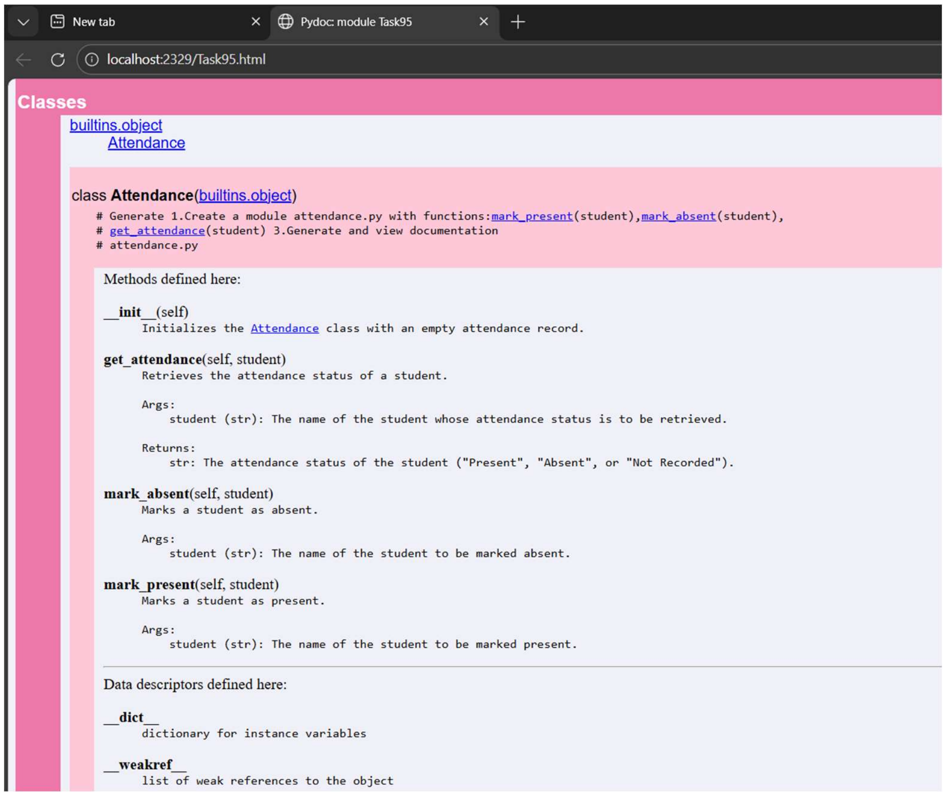
Methods defined here:

**__init__**(self)
    Initializes the Attendance class with an empty attendance record.

**get_attendance**(self, student)
```
Retrieves the attendance status of a student.

Args:
    student (str): The name of the student whose attendance status is to be retrieved.

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

**mark_absent**(self, student)
```
Marks a student as absent.

Args:
    student (str): The name of the student to be marked absent.
```

**mark_present**(self, student)
    Marks a student as present.


New tab | Pydoc: module Task95

localhost:2329/Task95.html

**Classes**

builtins.object
    Attendance

class **Attendance**(builtins.object)
```
# Generate 1.Create a module attendance.py with functions:mark_present(student),mark_absent(student),
# get_attendance(student) 3.Generate and view documentation
# attendance.py
```

Methods defined here:

**__init__**(self)
```
Initializes the Attendance class with an empty attendance record.
```

**get_attendance**(self, student)
```
Retrieves the attendance status of a student.

Args:
    student (str): The name of the student whose attendance status is to be retrieved.

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

**mark_absent**(self, student)
```
Marks a student as absent.

Args:
    student (str): The name of the student to be marked absent.
```

**mark_present**(self, student)
```
Marks a student as present.

Args:
    student (str): The name of the student to be marked present.
```

Data descriptors defined here:

**__dict__**
```
dictionary for instance variables
```

**__weakref__**
```
list of weak references to the object
```

**Problem 5: File Handling Function**

Consider the function:

def read_file(filename):

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

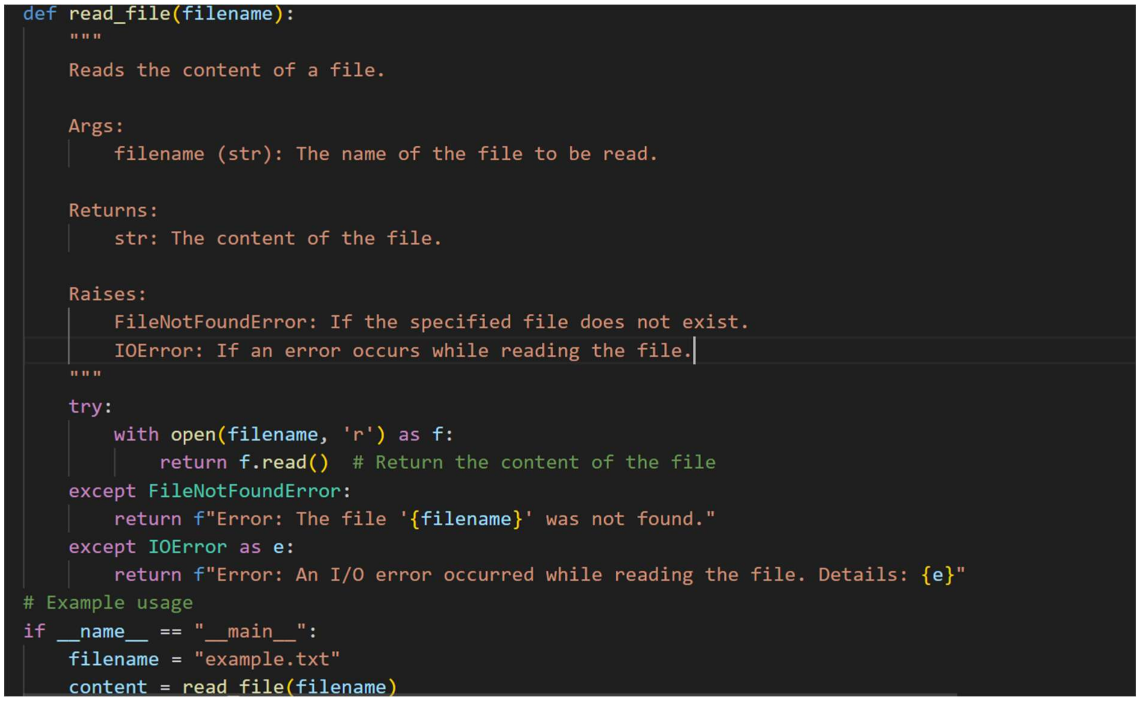
return f.read()

**Task:**

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

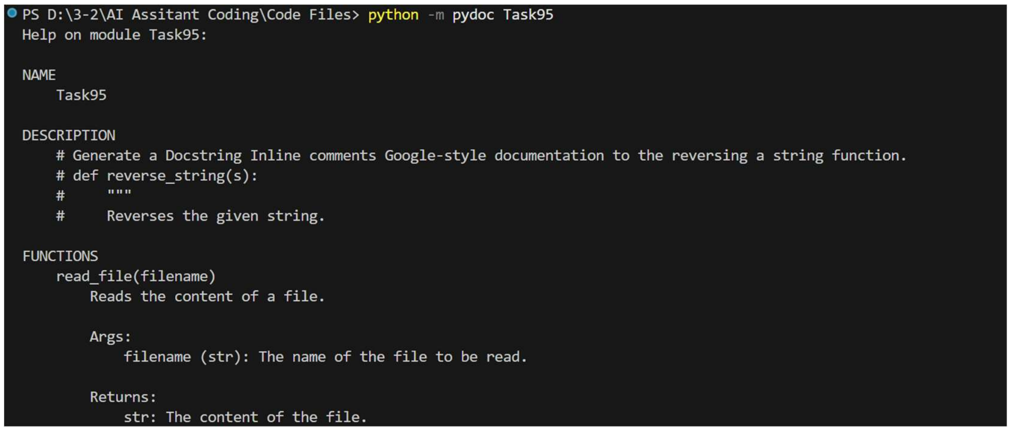
Prompt: Generate a code with def read_file(filename):,with open(filename, 'r') as f:,return f.read(). Generate documentation using all three formats.2. Identify which style best explains exception handling.

Code:

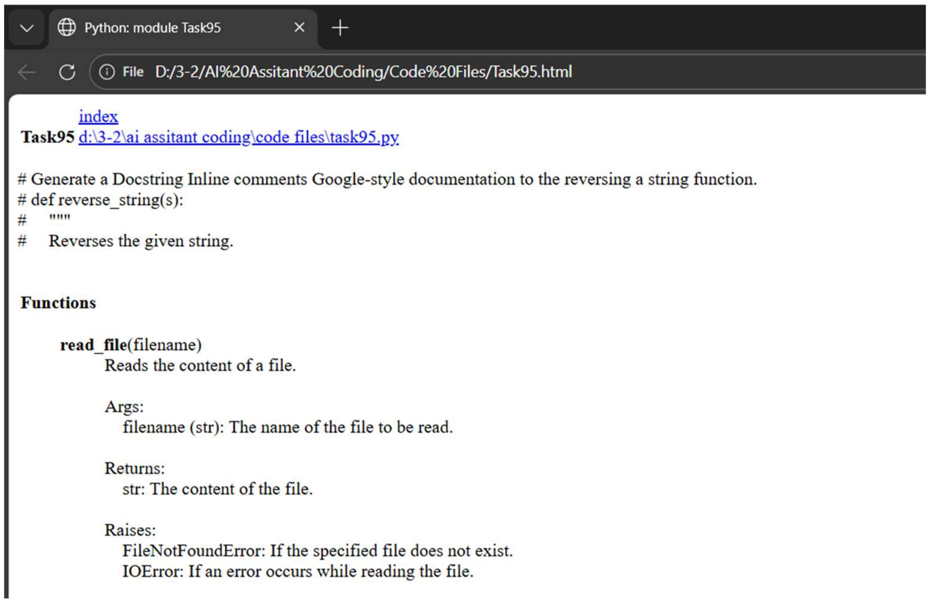


```
def read_file(filename):
    """
    Reads the content of a file.

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

    Returns:
        str: The content of the file.

    Raises:
        FileNotFoundError: If the specified file does not exist.
        IOError: If an error occurs while reading the file.
    """
    try:
        with open(filename, 'r') as f:
            return f.read()  # Return the content of the file
    except FileNotFoundError:
        return f"Error: The file '{filename}' was not found."
    except IOError as e:
        return f"Error: An I/O error occurred while reading the file. Details: {e}"
# Example usage
if __name__ == "__main__":
    filename = "example.txt"
    content = read_file(filename)
```

Output:



```
PS D:\3-2\AI Assitant Coding\Code Files> python -m pydoc Task95
Help on module Task95:

NAME
    Task95

DESCRIPTION
    # Generate a Docstring Inline comments Google-style documentation to the reversing a string function.
    # def reverse_string(s):
    #     """
    #     Reverses the given string.

FUNCTIONS
    read_file(filename)
        Reads the content of a file.

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

        Returns:
            str: The content of the file.
```



Python: module Task95

File D:/3-2/AI%20Assitant%20Coding/Code%20Files/Task95.html

index
**Task95** d:\3-2\ai assitant coding\code files\task95.py

# Generate a Docstring Inline comments Google-style documentation to the reversing a string function.
# def reverse_string(s):
#     """
#     Reverses the given string.

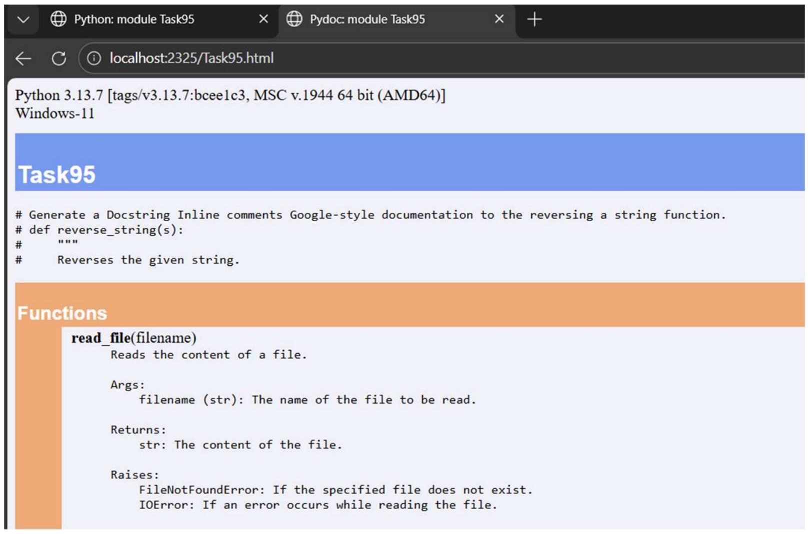
**Functions**

**read_file**(filename)
Reads the content of a file.

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

Returns:
str: The content of the file.

Raises:
FileNotFoundError: If the specified file does not exist.
IOError: If an error occurs while reading the file.



Python: module Task95 | Pydoc: module Task95

localhost:2325/Task95.html

Python 3.13.7 [tags/v3.13.7:bcee1c3, MSC v.1944 64 bit (AMD64)]
Windows-11

**Task95**

```
# Generate a Docstring Inline comments Google-style documentation to the reversing a string function.
# def reverse_string(s):
#     """
#     Reverses the given string.
```

**Functions**

**read_file**(filename)

```
Reads the content of a file.

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

Returns:
    str: The content of the file.

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