Assignment-9.5

Ht. No: 2303A51923

Name: V.Sravani

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

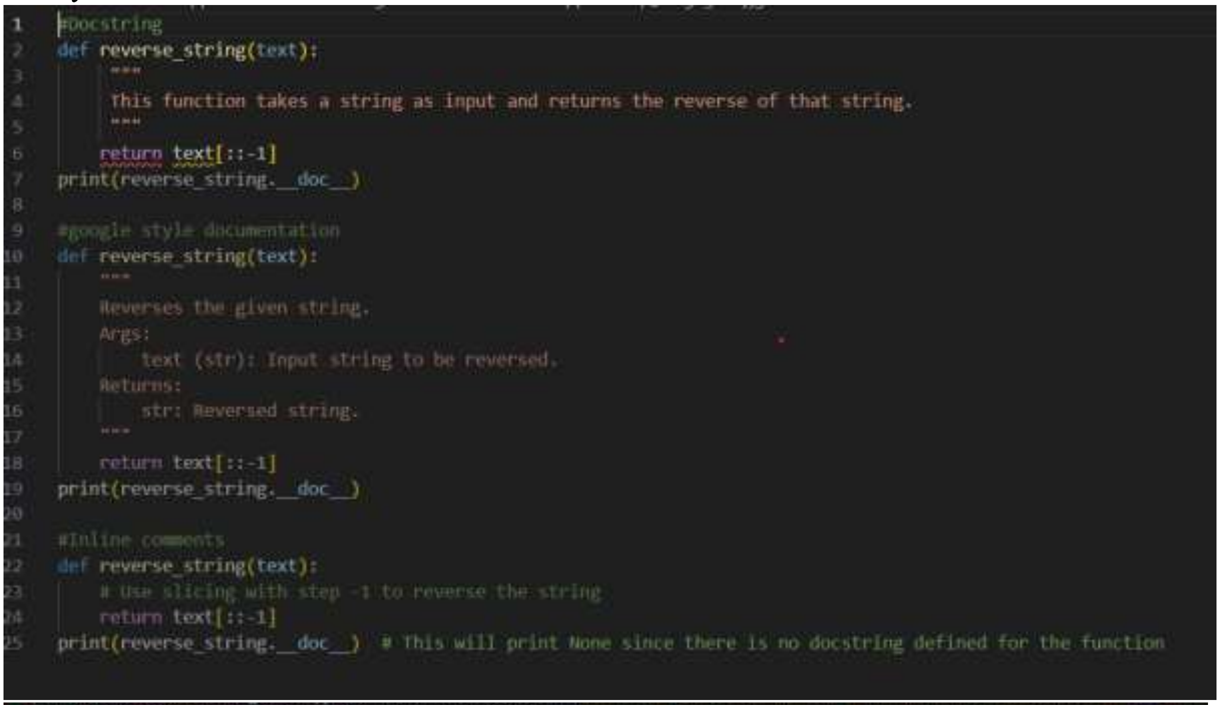
comments o (c) Google-style

documentation

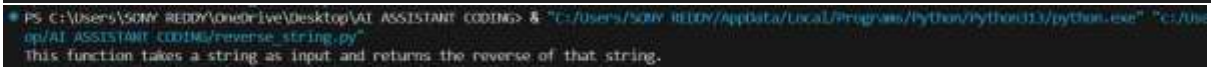
2. Compare the three documentation styles.

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

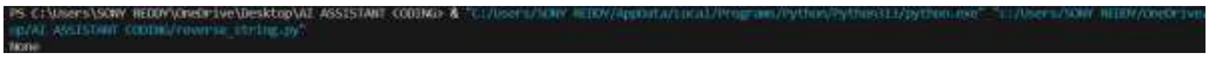
library.



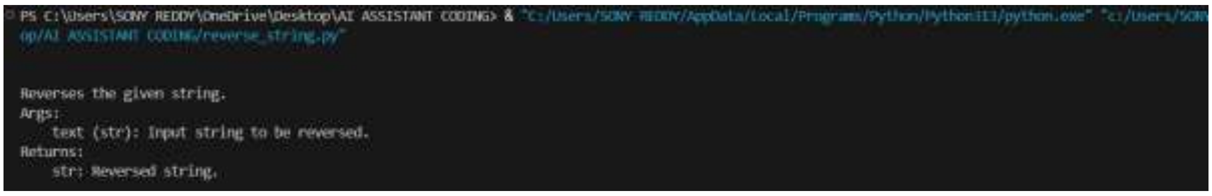
```
#Docstring
def reverse_string(text):
    """
    This function takes a string as input and returns the reverse of that string.
    """
    return text[::-1]
print(reverse_string.__doc__)

#google style documentation
def reverse_string(text):
    """
    Reverses the given string.
    Args:
        text (str): Input string to be reversed.
    Returns:
        str: Reversed string.
    """
    return text[::-1]
print(reverse_string.__doc__)

#Inline comments
def reverse_string(text):
    # Use slicing with step -1 to reverse the string
    return text[::-1]
print(reverse_string.__doc__)  # This will print None since there is no docstring defined for the function
```



```
PS C:\Users\SONY REDDY\OneDrive\Desktop\AI ASSISTANT CODING> & "C:/Users/SONY REDDY/AppData/Local/Programs/Python/Python313/python.exe" "c:/Use
op/AI ASSISTANT CODING/reverse_string.py"
This function takes a string as input and returns the reverse of that string.
```



```
PS C:\Users\SONY REDDY\OneDrive\Desktop\AI ASSISTANT CODING> & "C:/Users/SONY REDDY/AppData/Local/Programs/Python/Python313/python.exe" "c:/Users/SONY REDDY/OneDrive
op/AI ASSISTANT CODING/reverse_string.py"
None
```



```
PS C:\Users\SONY REDDY\OneDrive\Desktop\AI ASSISTANT CODING> & "C:/Users/SONY REDDY/AppData/Local/Programs/Python/Python313/python.exe" "c:/Users/SON
op/AI ASSISTANT CODING/reverse_string.py"


Reverses the given string.
Args:
    text (str): Input string to be reversed.
Returns:
    str: Reversed string.
```

Problem 2: Password Strength

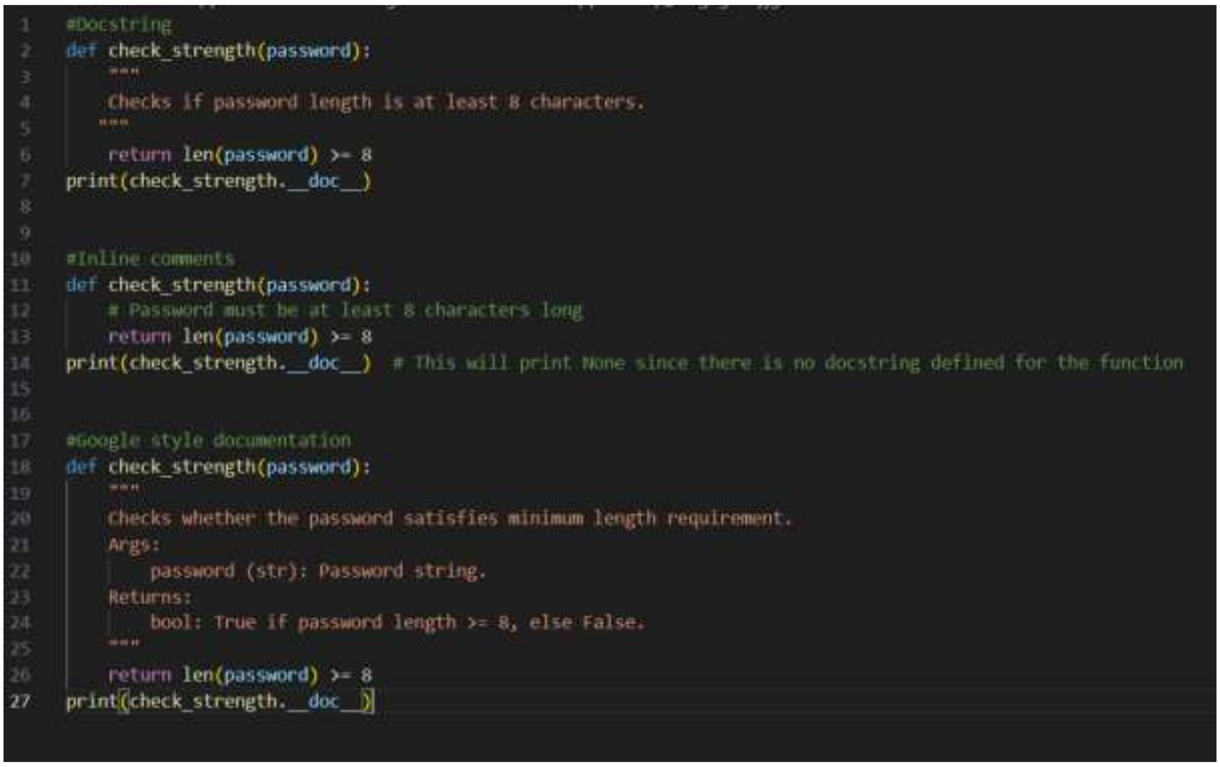
Checker Consider the function: def

check_strength(password):

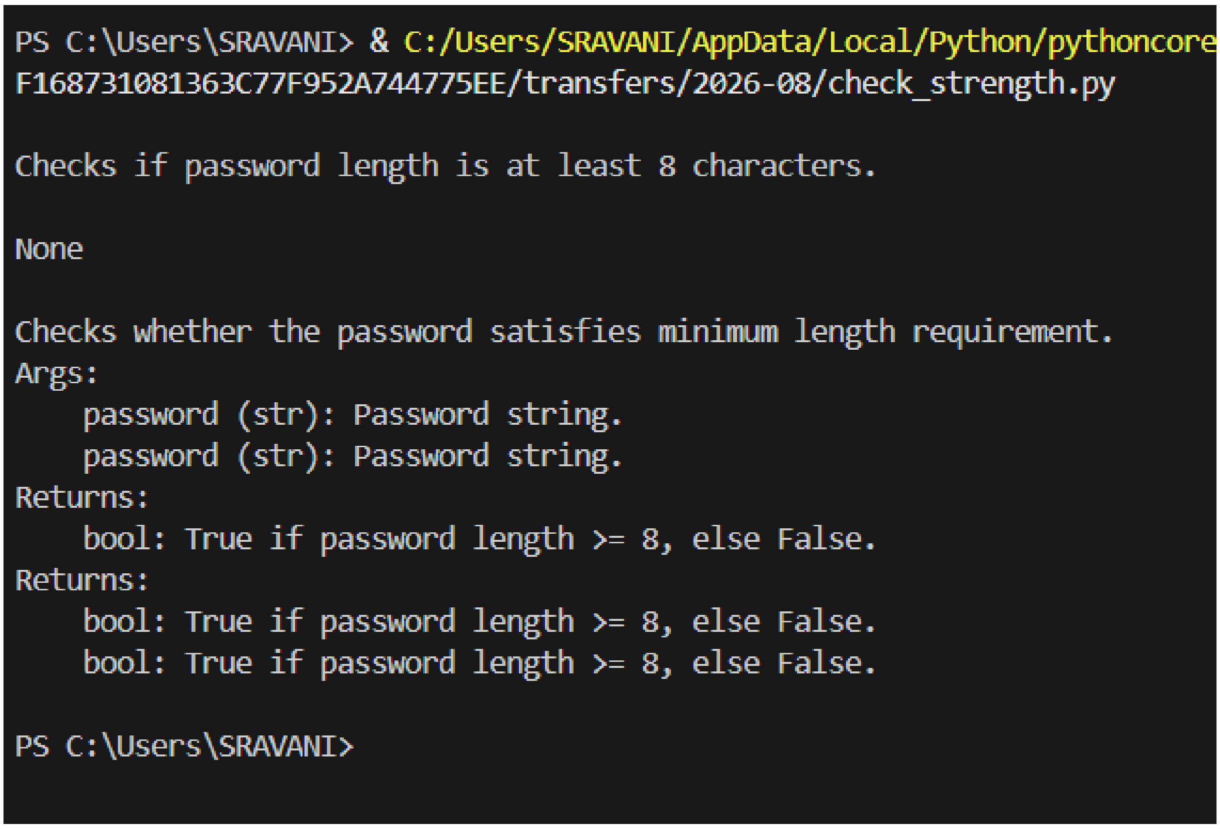
return len(password) >= 8

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.



```
#Docstring
def check_strength(password):
    """
    Checks if password length is at least 8 characters.
   """
    return len(password) >= 8
print(check_strength.__doc__)


#Inline comments
def check_strength(password):
    # Password must be at least 8 characters long
    return len(password) >= 8
print(check_strength.__doc__)  # This will print None since there is no docstring defined for the function


#Google style documentation
def check_strength(password):
    """
    Checks whether the password satisfies minimum length requirement.
    Args:
        password (str): Password string.
    Returns:
        bool: True if password length >= 8, else False.
    """
    return len(password) >= 8
print(check_strength.__doc__)
```


```
PS C:\Users\SRAVANI> & C:/Users/SRAVANI/AppData/Local/Python/pythoncore
F168731081363C77F952A744775EE/transfers/2026-08/check_strength.py

Checks if password length is at least 8 characters.

None

Checks whether the password satisfies minimum length requirement.
Args:
    password (str): Password string.
    password (str): Password string.
Returns:
    bool: True if password length >= 8, else False.
Returns:
    bool: True if password length >= 8, else False.
    bool: True if password length >= 8, else False.

PS C:\Users\SRAVANI>
```

Problem 3: Math Utilities Module

Task:

1. Create a module math_utils.py with functions:

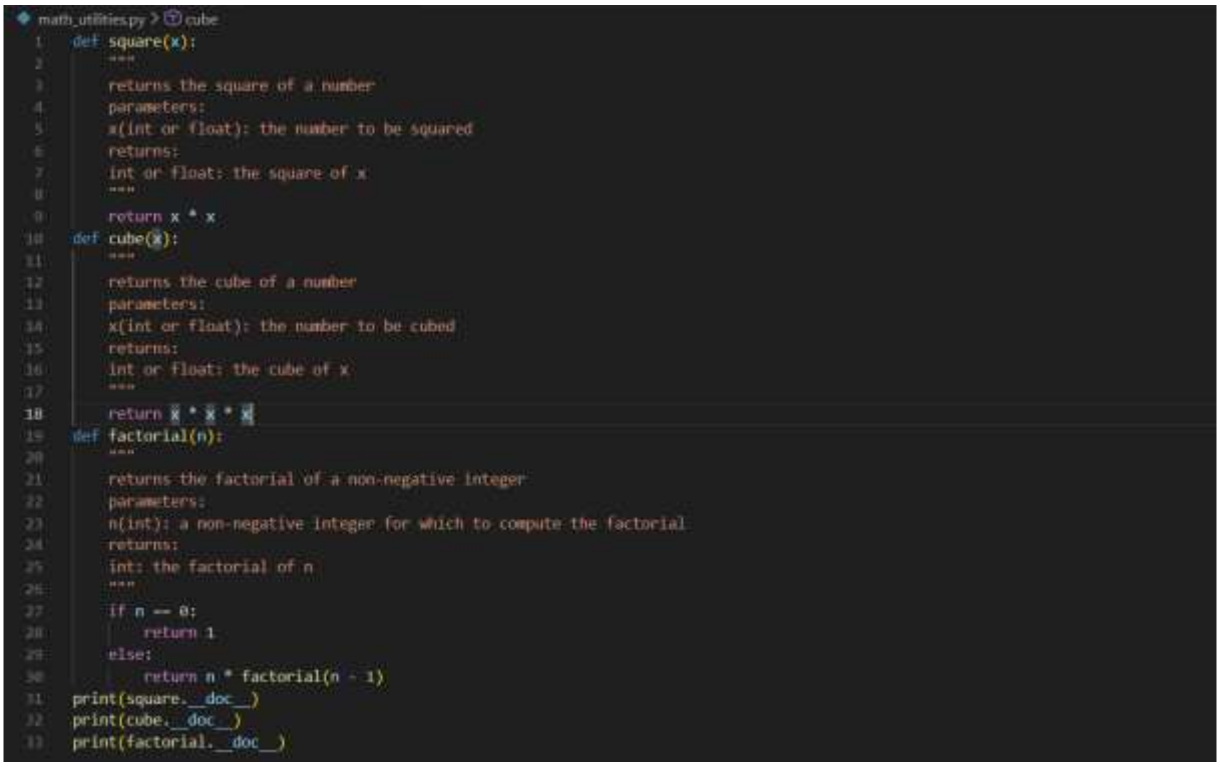
o square(n)

o cube(n) o

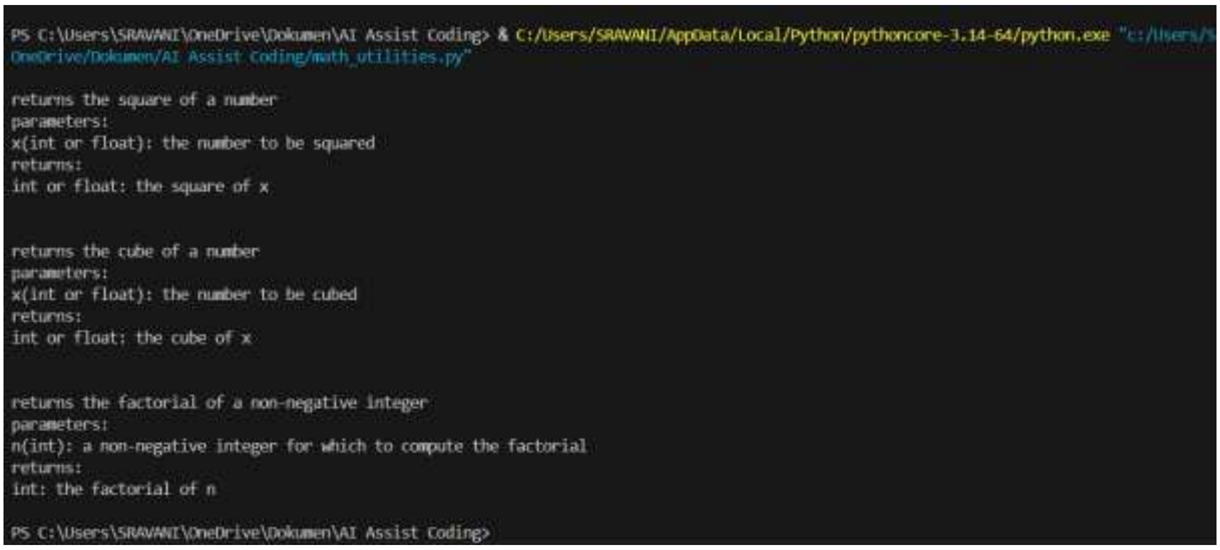
factorial(n)

2. Generate docstrings automatically using AI tools.

3. Export documentation as an HTML file.



```python
def square(x):
    """
    returns the square of a number
    parameters:
    x(int or float): the number to be squared
    returns:
    int or float: the square of x
    """
    return x * x
def cube(x):
    """
    returns the cube of a number
    parameters:
    x(int or float): the number to be cubed
    returns:
    int or float: the cube of x
    """
    return x * x * x
def factorial(n):
    """
    returns the factorial of a non-negative integer
    parameters:
    n(int): a non-negative integer for which to compute the factorial
    returns:
    int: the factorial of n
    """
    if n == 0:
        return 1
    else:
        return n * factorial(n - 1)
print(square.__doc__)
print(cube.__doc__)
print(factorial.__doc__)
```



```
PS C:\Users\SRAVANI\OneDrive\Dokumen\AI Assist Coding> & C:/Users/SRAVANI/AppData/Local/Python/pythoncore-3.14-64/python.exe "c:/Users/S
OneDrive/Dokumen/AI Assist Coding/math_utilities.py"

returns the square of a number
parameters:
x(int or float): the number to be squared
returns:
int or float: the square of x


returns the cube of a number
parameters:
x(int or float): the number to be cubed
returns:
int or float: the cube of x


returns the factorial of a non-negative integer
parameters:
n(int): a non-negative integer for which to compute the factorial
returns:
int: the factorial of n

PS C:\Users\SRAVANI\OneDrive\Dokumen\AI Assist Coding>
```

Problem 4: Attendance Management Module

Task:

1. Create a module attendance.py with functions:

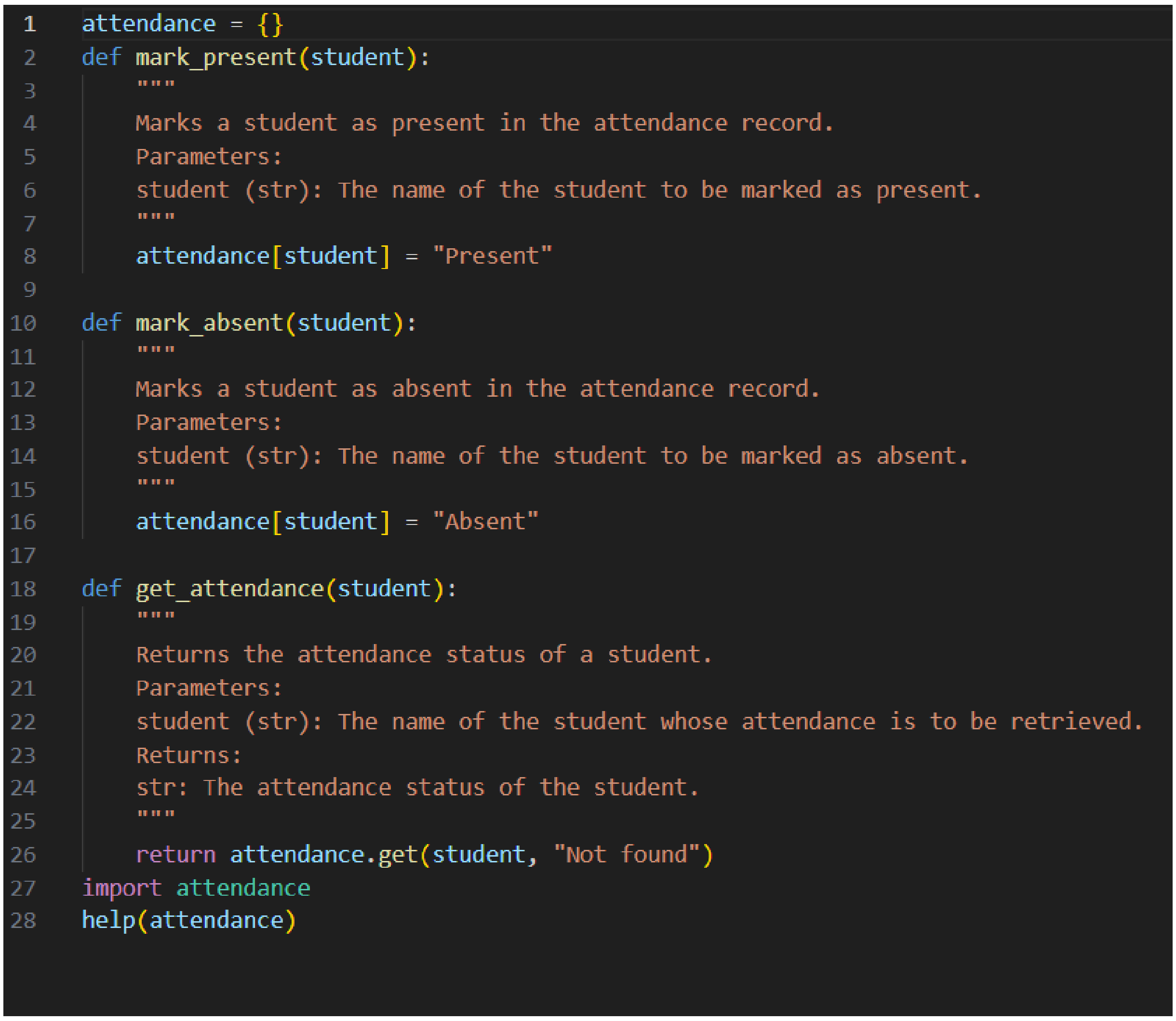
o mark_present(student) o

mark_absent(student) o

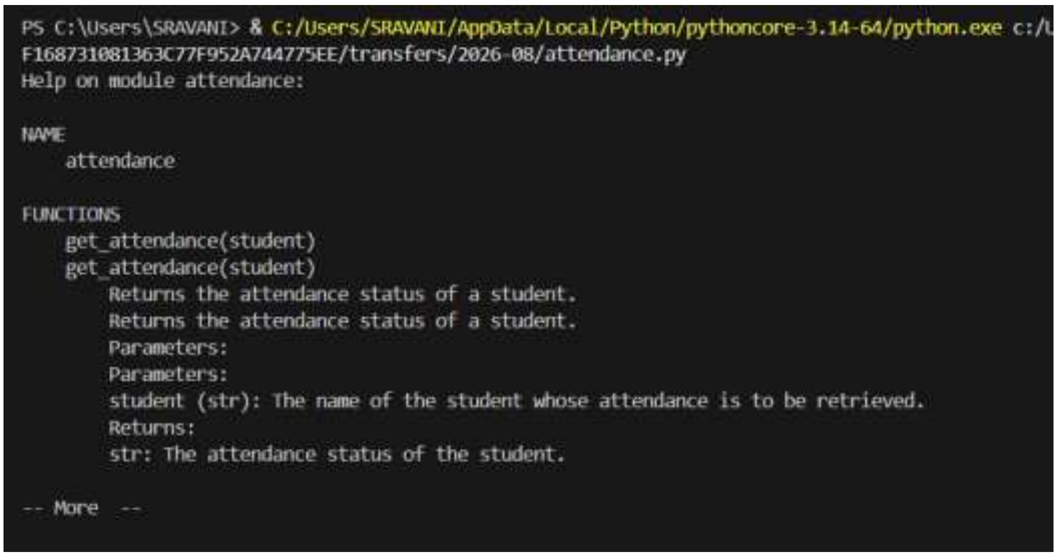
get_attendance(student)

2. Add proper docstrings.

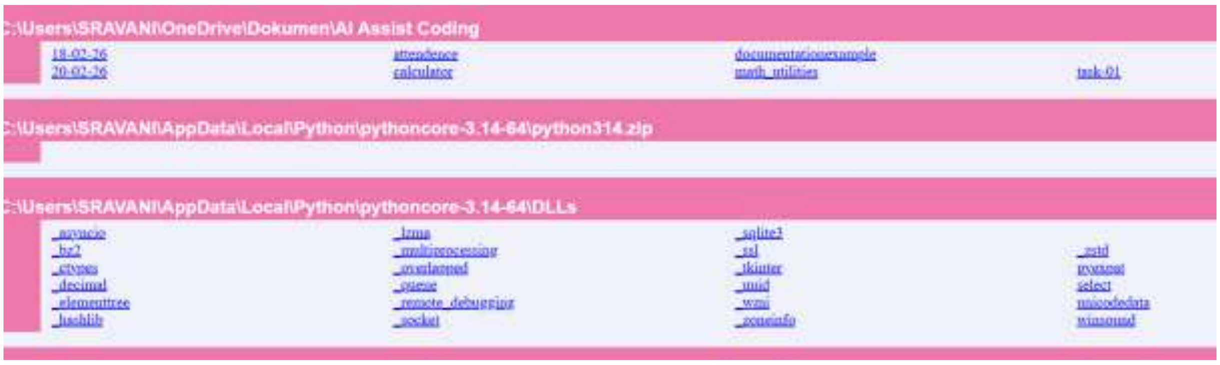
3. Generate and view documentation in terminal and browse



```python
attendance = {}
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.
    """
    attendance[student] = "Present"

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.
    """
    attendance[student] = "Absent"

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.
    """
    return attendance.get(student, "Not found")
import attendance
help(attendance)
```



```
PS C:\Users\SRAVANI> & C:/Users/SRAVANI/AppData/Local/Python/pythoncore-3.14-64/python.exe c:/U
F168731081363C77F952A744775EE/transfers/2026-08/attendance.py
Help on module attendance:

NAME
    attendance

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

-- More  --
```



```
C:\Users\SRAVANI\OneDrive\Dokumen\AI Assist Coding
18-02-26          attendence              documentationexample
20-02-26          calculator              math_utilities            task-01

C:\Users\SRAVANI\AppData\Local\Python\pythoncore-3.14-64\python314.zip

C:\Users\SRAVANI\AppData\Local\Python\pythoncore-3.14-64\DLLs
_asyncio          _lzma                   _sqlite3
_bz2              _multiprocessing        _ssl                      _zstd
_ctypes           _overlapped             _tkinter                  pyexpat
_decimal          _queue                  _uuid                     select
_elementtree      _remote_debugging       _wmi                      unicodedata
_hashlib          _socket                 _zoneinfo                 winsound
```

Problem 5: File Handling

Function Consider the function:

def read_file(filename): with

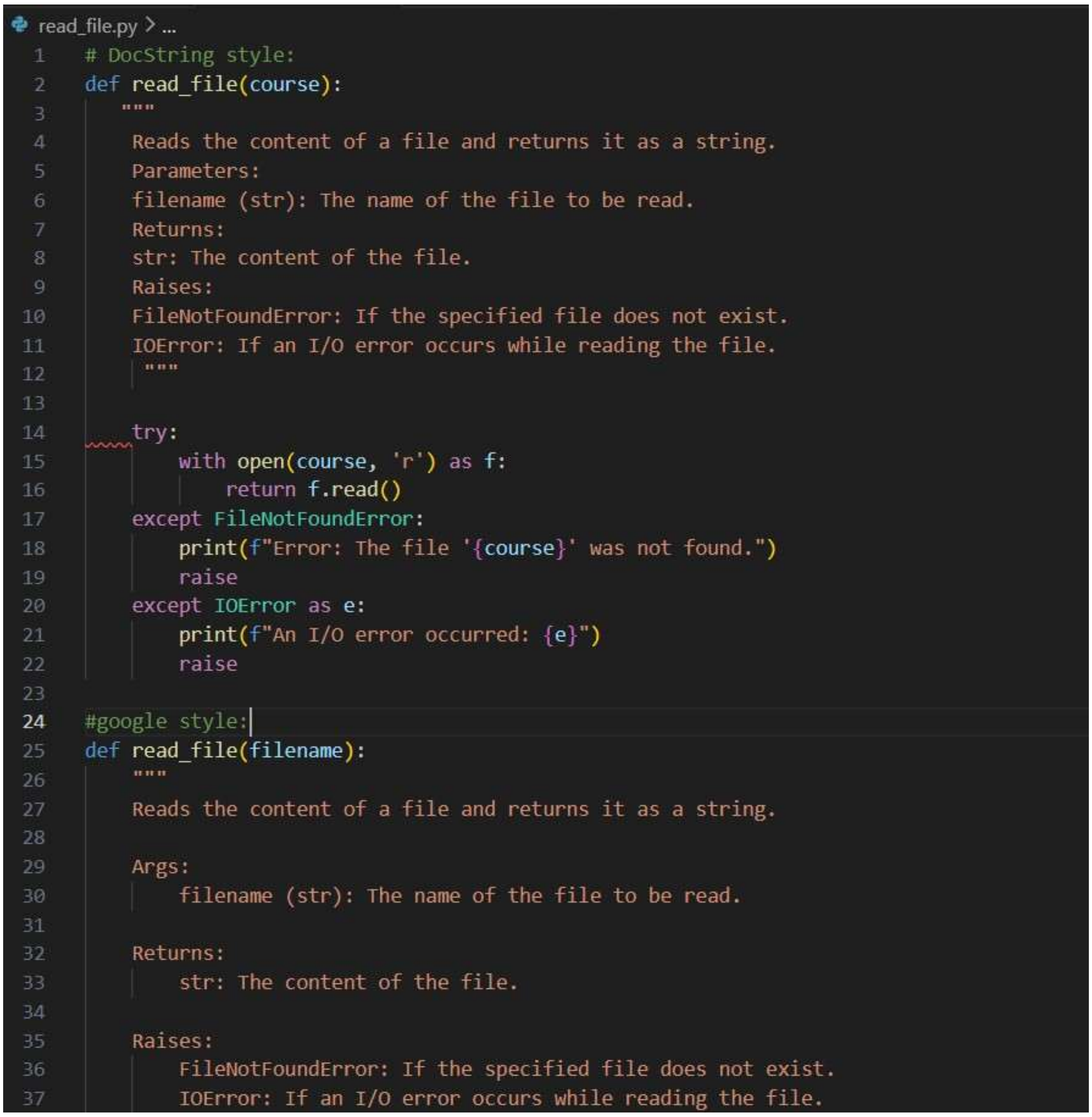
open(filename, 'r') as f:

return f.read() Task:

1. Write documentation using all three formats.

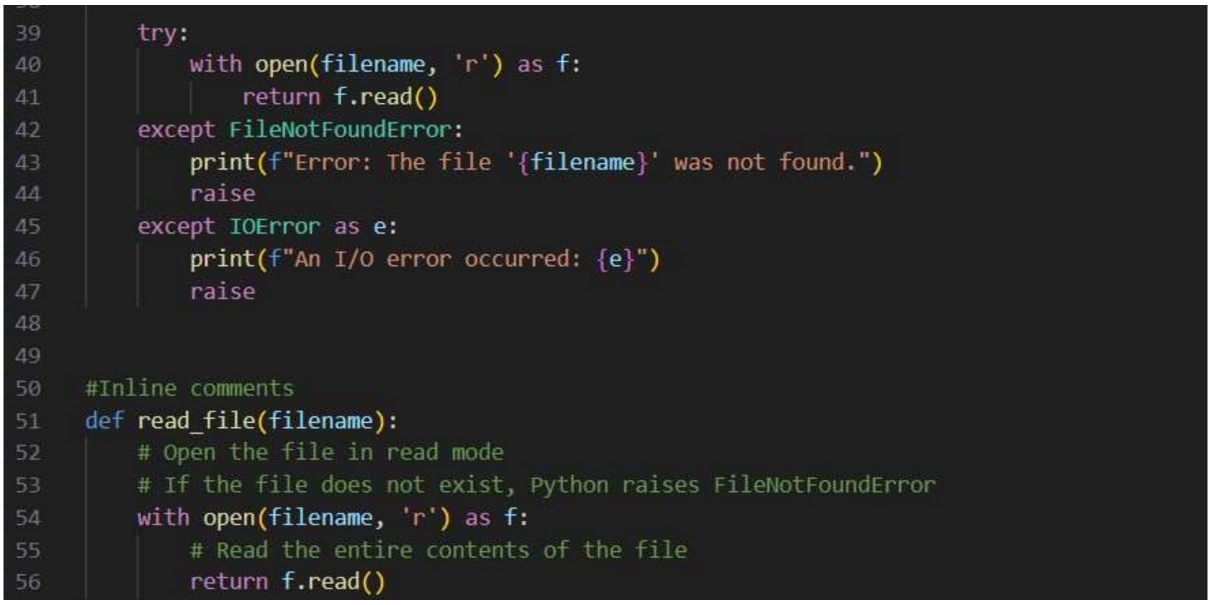
2. Identify which style best explains exception handling.

3. Justify your recommendation.

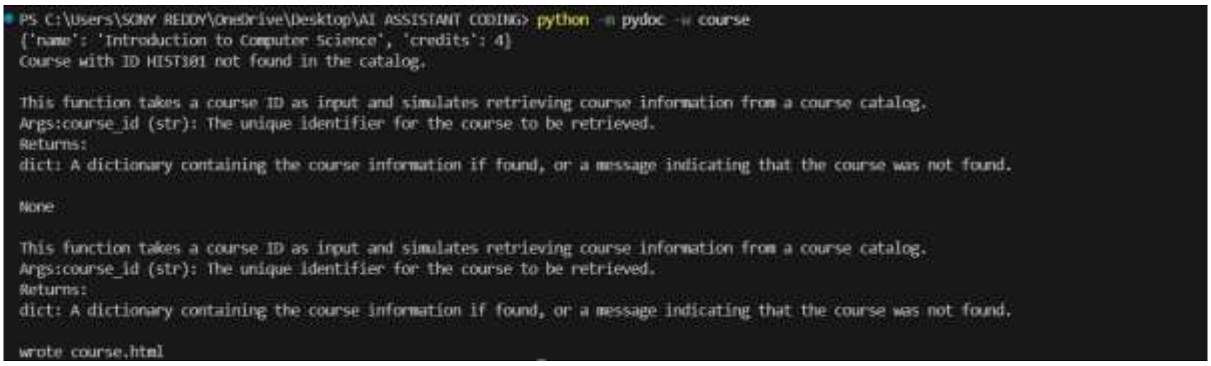


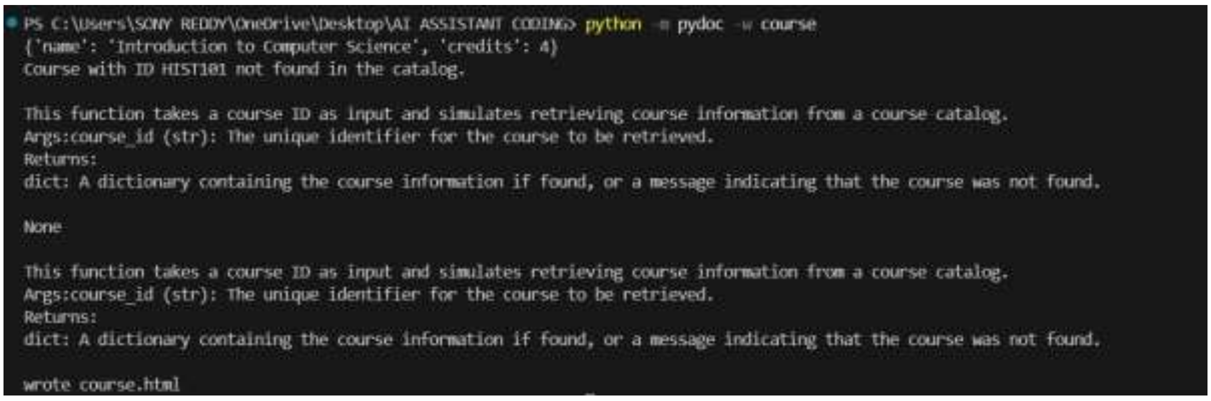
read_file.py > ...

```python
# DocString style:
def read_file(course):
    """
    Reads the content of a file and returns it as a string.
    Parameters:
    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 I/O error occurs while reading the file.
    """

    try:
        with open(course, 'r') as f:
            return f.read()
    except FileNotFoundError:
        print(f"Error: The file '{course}' was not found.")
        raise
    except IOError as e:
        print(f"An I/O error occurred: {e}")
        raise

#google style:
def read_file(filename):
    """
    Reads the content of a file and returns it as a string.

    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 I/O error occurs while reading the file.
```



```python
    try:
        with open(filename, 'r') as f:
            return f.read()
    except FileNotFoundError:
        print(f"Error: The file '{filename}' was not found.")
        raise
    except IOError as e:
        print(f"An I/O error occurred: {e}")
        raise


#Inline comments
def read_file(filename):
    # Open the file in read mode
    # If the file does not exist, Python raises FileNotFoundError
    with open(filename, 'r') as f:
        # Read the entire contents of the file
        return f.read()
```


```
PS C:\Users\SONY REDDY\OneDrive\Desktop\AI ASSISTANT CODING> python -m pydoc -w course
{'name': 'Introduction to Computer Science', 'credits': 4}
Course with ID HIST101 not found in the catalog.

This function takes a course ID as input and simulates retrieving course information from a course catalog.
Args:course_id (str): The unique identifier for the course to be retrieved.
Returns:
dict: A dictionary containing the course information if found, or a message indicating that the course was not found.

None

This function takes a course ID as input and simulates retrieving course information from a course catalog.
Args:course_id (str): The unique identifier for the course to be retrieved.
Returns:
dict: A dictionary containing the course information if found, or a message indicating that the course was not found.

wrote course.html
```



```
PS C:\Users\SONY REDDY\OneDrive\Desktop\AI ASSISTANT CODING> python -m pydoc -w course
{'name': 'Introduction to Computer Science', 'credits': 4}
Course with ID HIST101 not found in the catalog.

This function takes a course ID as input and simulates retrieving course information from a course catalog.
Args:course_id (str): The unique identifier for the course to be retrieved.
Returns:
dict: A dictionary containing the course information if found, or a message indicating that the course was not found.

None

This function takes a course ID as input and simulates retrieving course information from a course catalog.
Args:course_id (str): The unique identifier for the course to be retrieved.
Returns:
dict: A dictionary containing the course information if found, or a message indicating that the course was not found.

wrote course.html
```