Assignment – 9.5

Name : G.Charanya

Ht.no : 2303A51oG6

Batch : 23

Problem 1: String Utilities Function

Consider the following Python function:

def reverse_string(text):

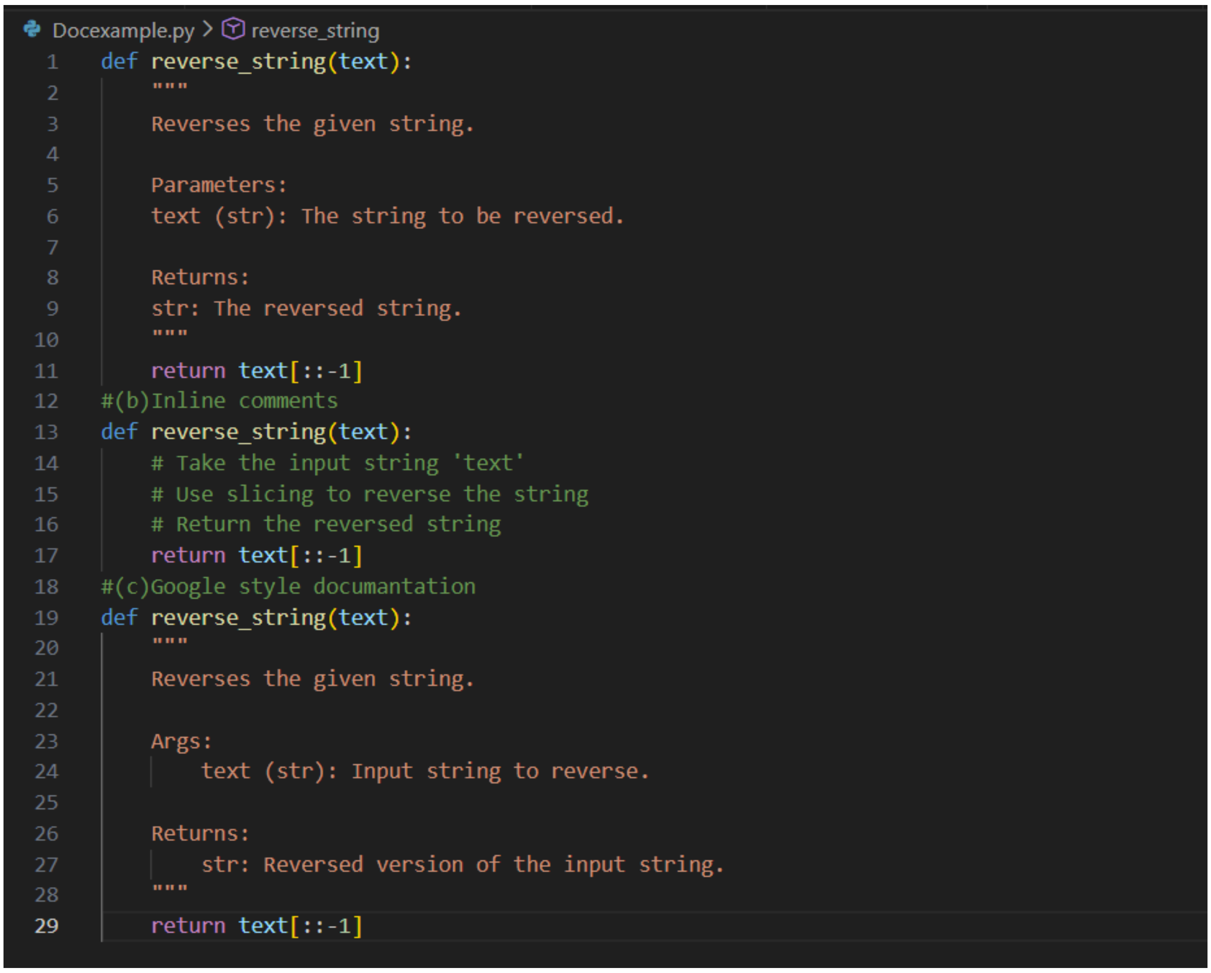
return text[::-1]

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.



Docexample.py > reverse_string

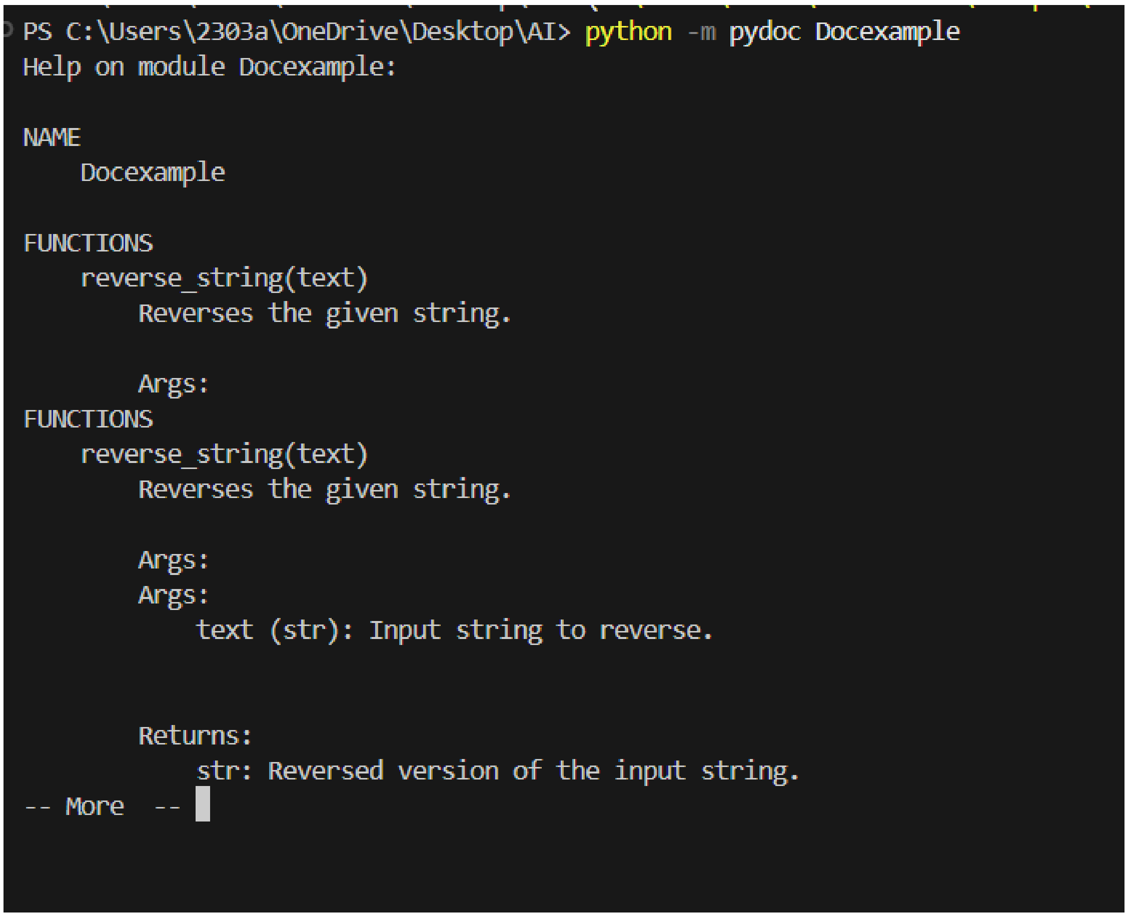
```python
def reverse_string(text):
    """
    Reverses the given string.

    Parameters:
    text (str): The string to be reversed.

    Returns:
    str: The reversed string.
    """
    return text[::-1]
#(b)Inline comments
def reverse_string(text):
    # Take the input string 'text'
    # Use slicing to reverse the string
    # Return the reversed string
    return text[::-1]
#(c)Google style documantation
def reverse_string(text):
    """
    Reverses the given string.

    Args:
        text (str): Input string to reverse.

    Returns:
        str: Reversed version of the input string.
    """
    return text[::-1]
```



```
PS C:\Users\2303a\OneDrive\Desktop\AI> python -m pydoc Docexample
Help on module Docexample:

NAME
    Docexample

FUNCTIONS
    reverse_string(text)
        Reverses the given string.

        Args:
FUNCTIONS
    reverse_string(text)
        Reverses the given string.

        Args:
        Args:
            text (str): Input string to reverse.


        Returns:
            str: Reversed version of the input string.
-- More  --
```

Problem 2: Password Strength Checker

Consider the function:

def check_strength(password):

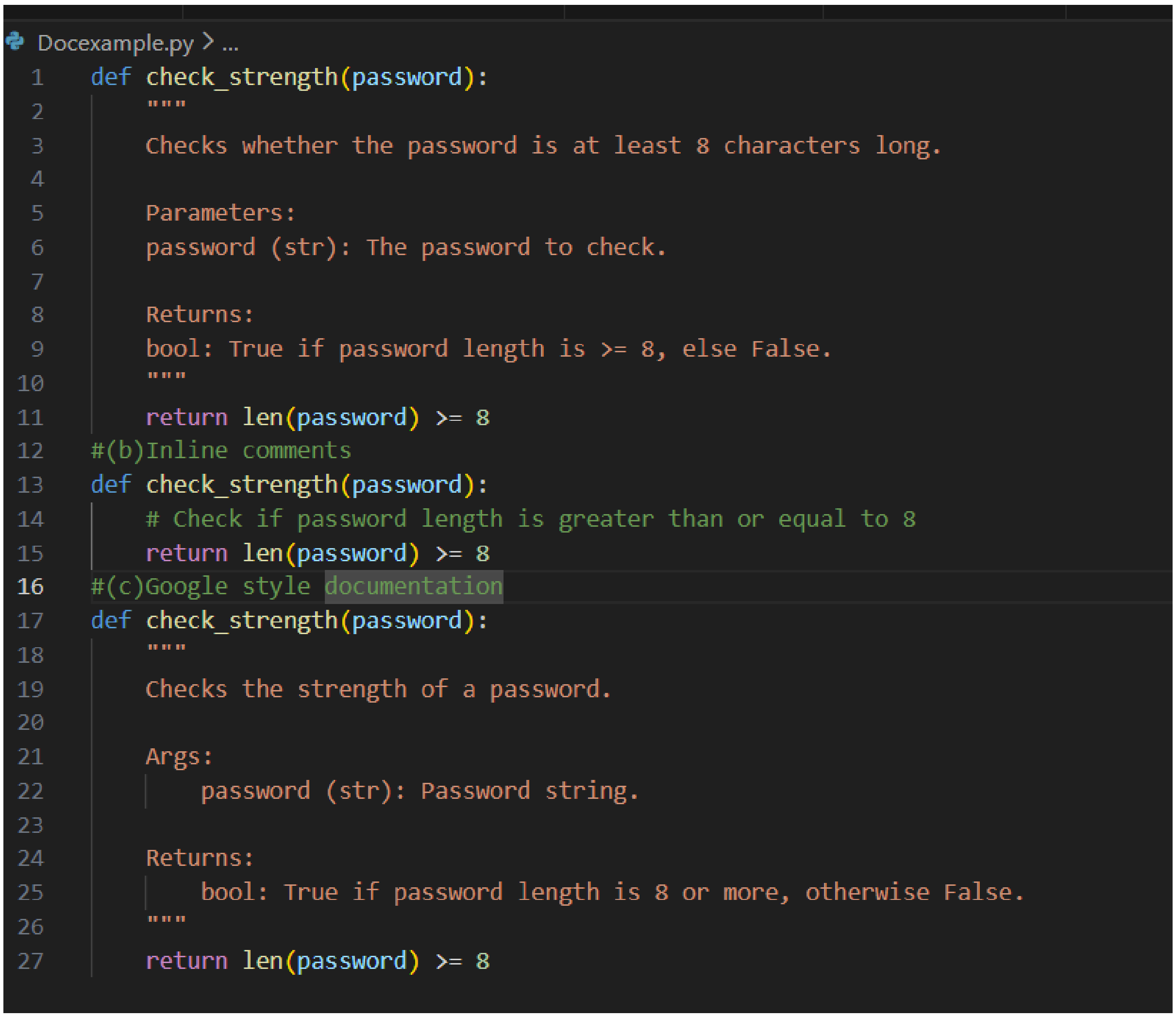
return len(password) >= 8

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.



Docexample.py > ...

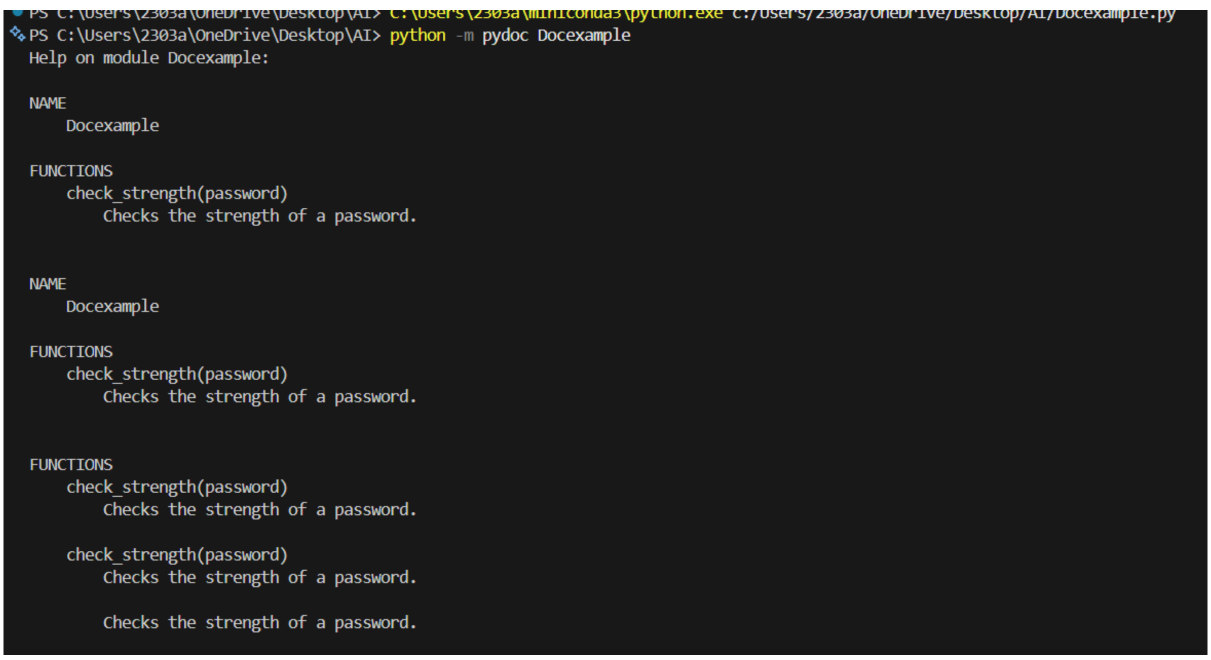
```python
def check_strength(password):
    """
    Checks whether the password is at least 8 characters long.

    Parameters:
    password (str): The password to check.

    Returns:
    bool: True if password length is >= 8, else False.
    """
    return len(password) >= 8
#(b)Inline comments
def check_strength(password):
    # Check if password length is greater than or equal to 8
    return len(password) >= 8
#(c)Google style documentation
def check_strength(password):
    """
    Checks the strength of a password.

    Args:
        password (str): Password string.

    Returns:
        bool: True if password length is 8 or more, otherwise False.
    """
    return len(password) >= 8
```



```
PS C:\Users\2303a\OneDrive\Desktop\AI> C:\Users\2303a\miniconda3\python.exe c:/Users/2303a/OneDrive/Desktop/AI/Docexample.py
PS C:\Users\2303a\OneDrive\Desktop\AI> python -m pydoc Docexample
Help on module Docexample:

NAME
    Docexample

FUNCTIONS
    check_strength(password)
        Checks the strength of a password.


NAME
    Docexample

FUNCTIONS
    check_strength(password)
        Checks the strength of a password.


FUNCTIONS
    check_strength(password)
        Checks the strength of a password.

    check_strength(password)
        Checks the strength of a password.

        Checks the strength of a password.
```

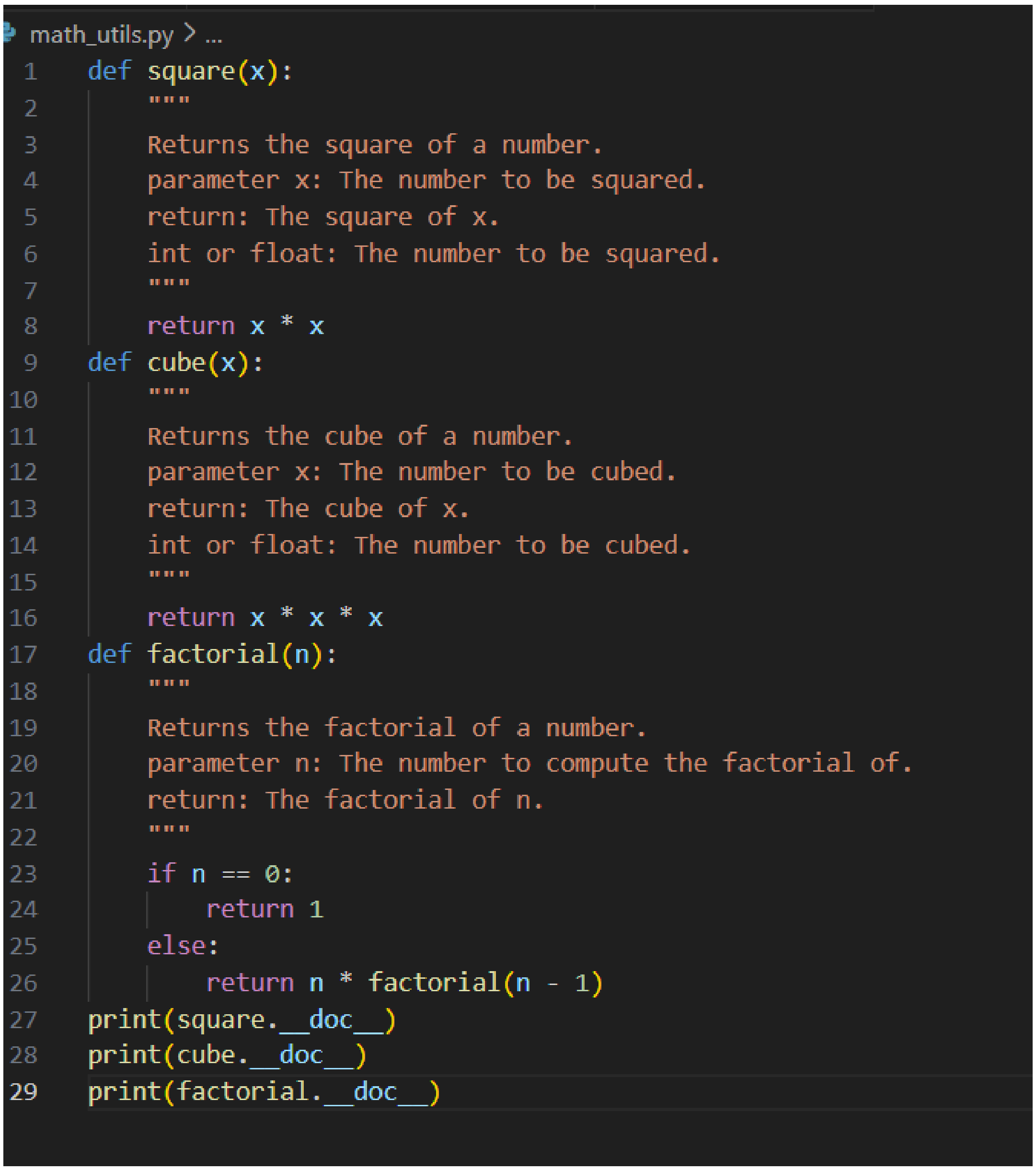
Problem 3: Math Utilities Module

Task:

1. Create a module math_utils.py with functions:
   o square(n)
   o cube(n)
   o factorial(n)

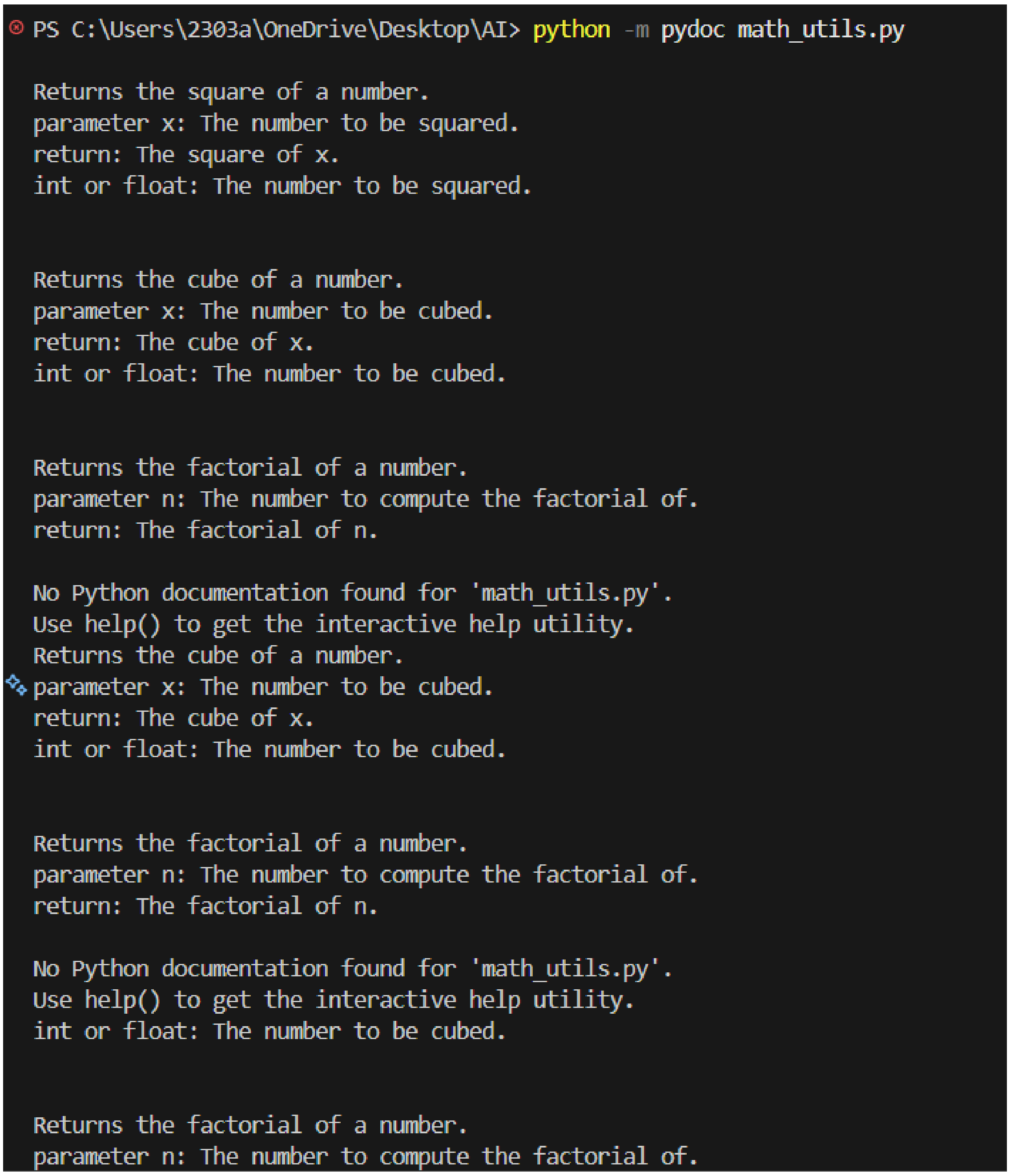
2. Generate docstrings automatically using AI tools.

3. Export documentation as an HTML file.



math_utils.py > ...

```python
def square(x):
    """
    Returns the square of a number.
    parameter x: The number to be squared.
    return: The square of x.
    int or float: The number to be squared.
    """
    return x * x
def cube(x):
    """
    Returns the cube of a number.
    parameter x: The number to be cubed.
    return: The cube of x.
    int or float: The number to be cubed.
    """
    return x * x * x
def factorial(n):
    """
    Returns the factorial of a number.
    parameter n: The number to compute the factorial of.
    return: 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\2303a\OneDrive\Desktop\AI> python -m pydoc math_utils.py

  Returns the square of a number.
  parameter x: The number to be squared.
  return: The square of x.
  int or float: The number to be squared.


  Returns the cube of a number.
  parameter x: The number to be cubed.
  return: The cube of x.
  int or float: The number to be cubed.


  Returns the factorial of a number.
  parameter n: The number to compute the factorial of.
  return: The factorial of n.

  No Python documentation found for 'math_utils.py'.
  Use help() to get the interactive help utility.
  Returns the cube of a number.
  parameter x: The number to be cubed.
  return: The cube of x.
  int or float: The number to be cubed.


  Returns the factorial of a number.
  parameter n: The number to compute the factorial of.
  return: The factorial of n.

  No Python documentation found for 'math_utils.py'.
  Use help() to get the interactive help utility.
  int or float: The number to be cubed.


  Returns the factorial of a number.
  parameter n: The number to compute the factorial of.
```

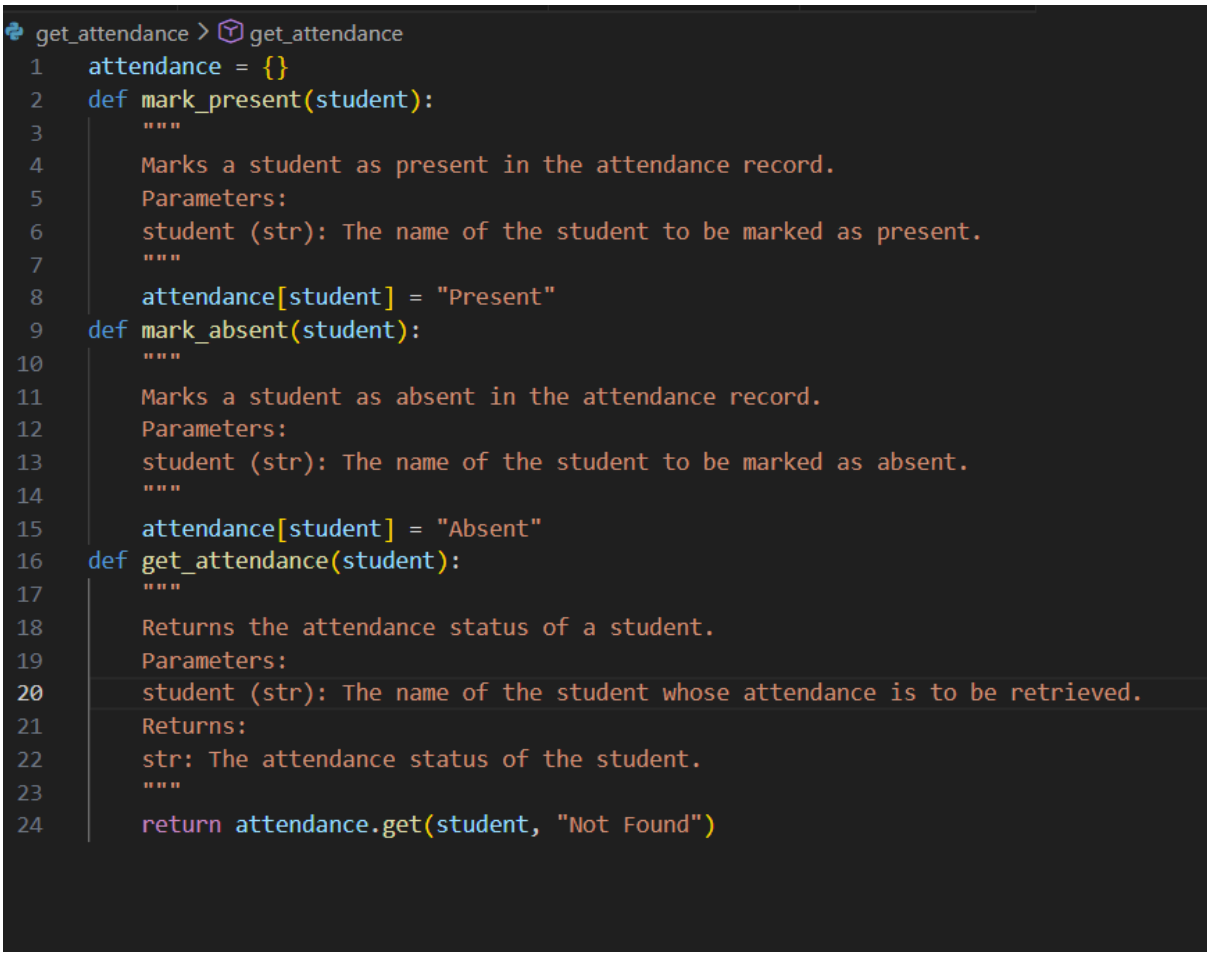
Problem 4: Attendance Management Module

Task:

1. Create a module attendance.py with functions:
   o mark_present(student)
   o mark_absent(student)
   o get_attendance(student)

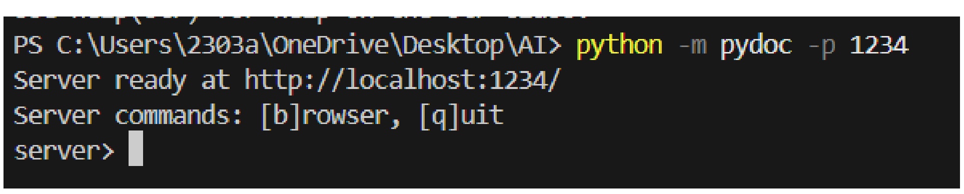
2. Add proper docstrings.

3. Generate and view documentation in terminal and browse

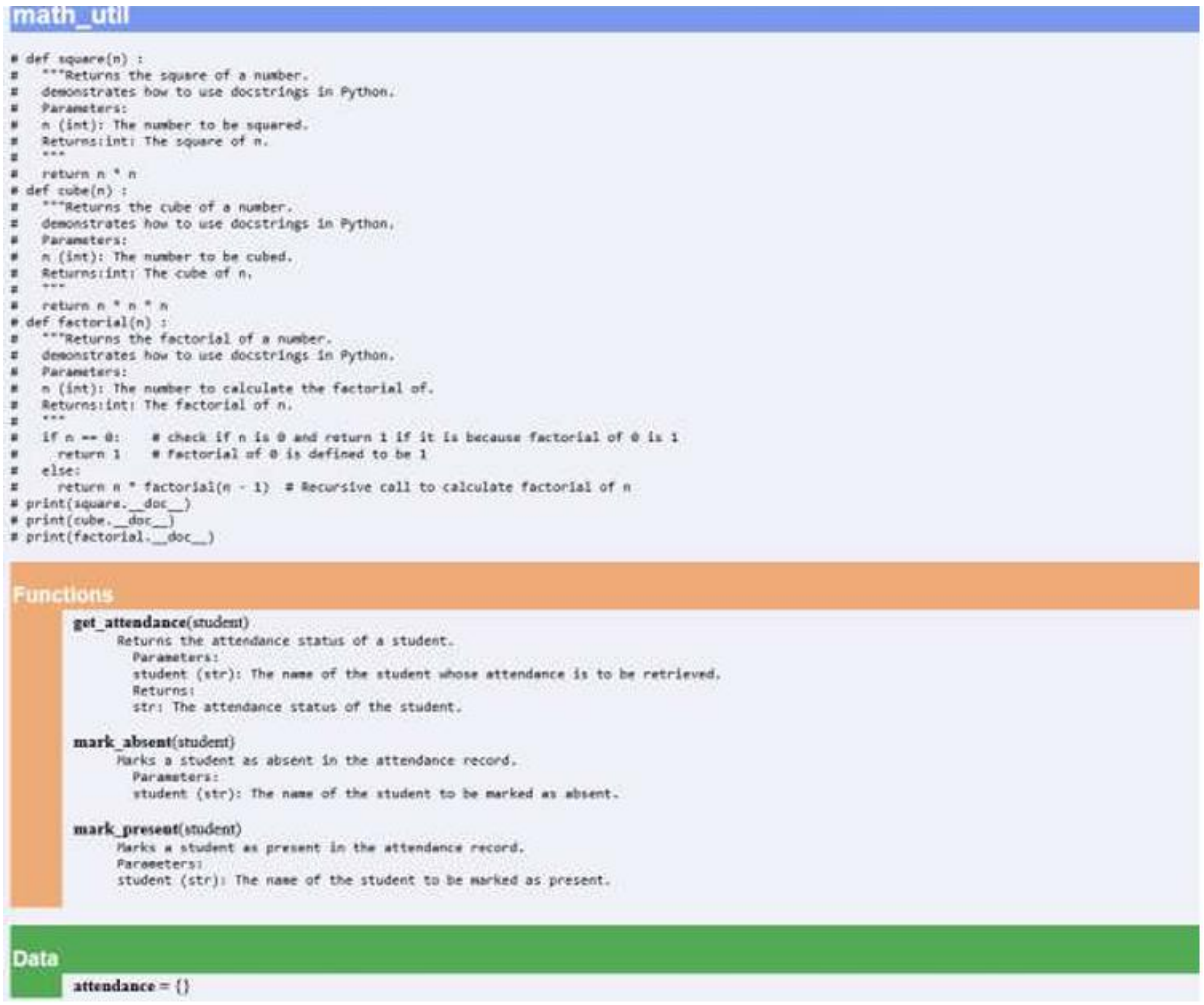


get_attendance > get_attendance

```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")
```



```
PS C:\Users\2303a\OneDrive\Desktop\AI> python -m pydoc -p 1234
Server ready at http://localhost:1234/
Server commands: [b]rowser, [q]uit
server>
```



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

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.
```

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.
```

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:

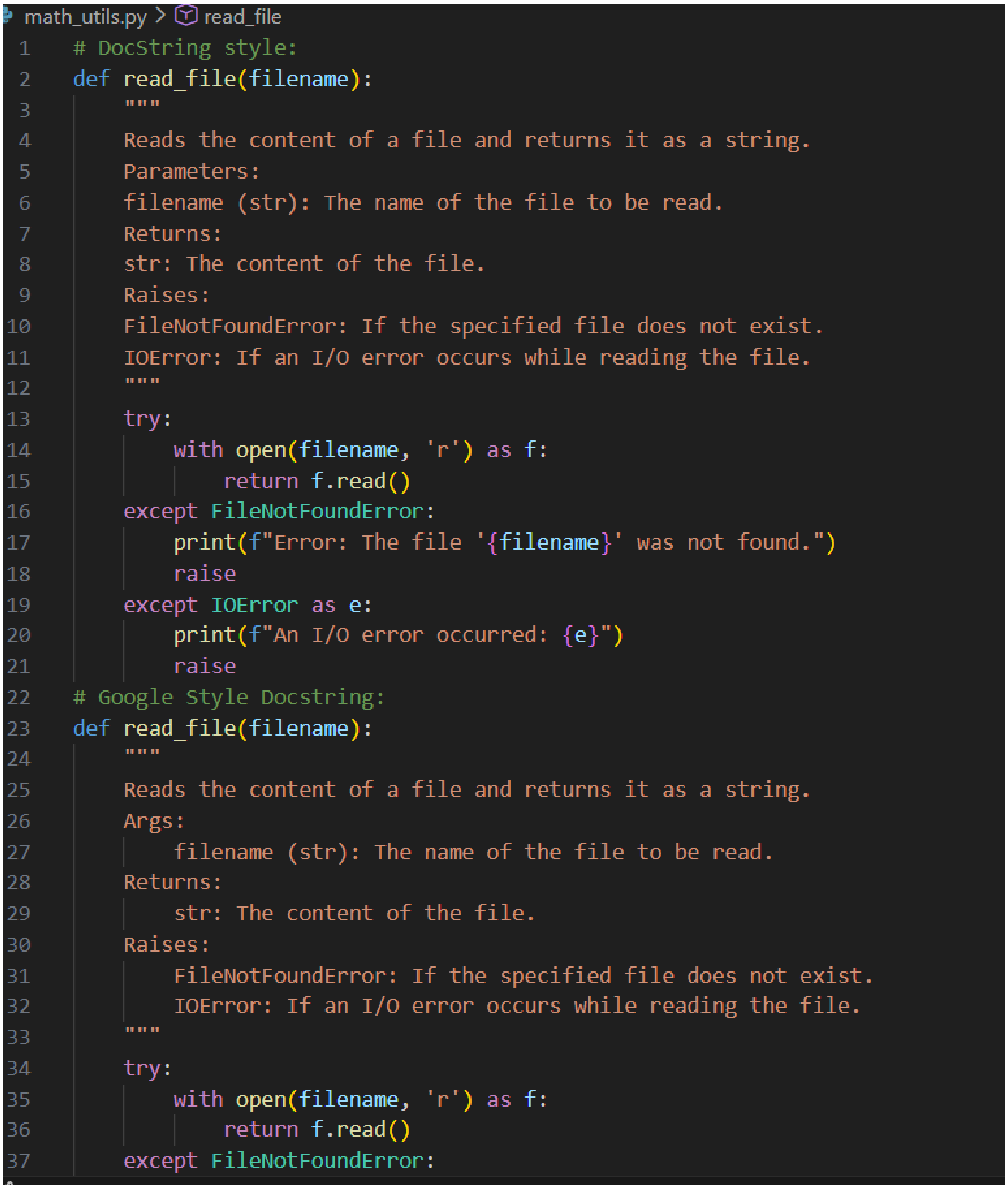
return f.read()

Task:

1. Write documentation using all three formats.

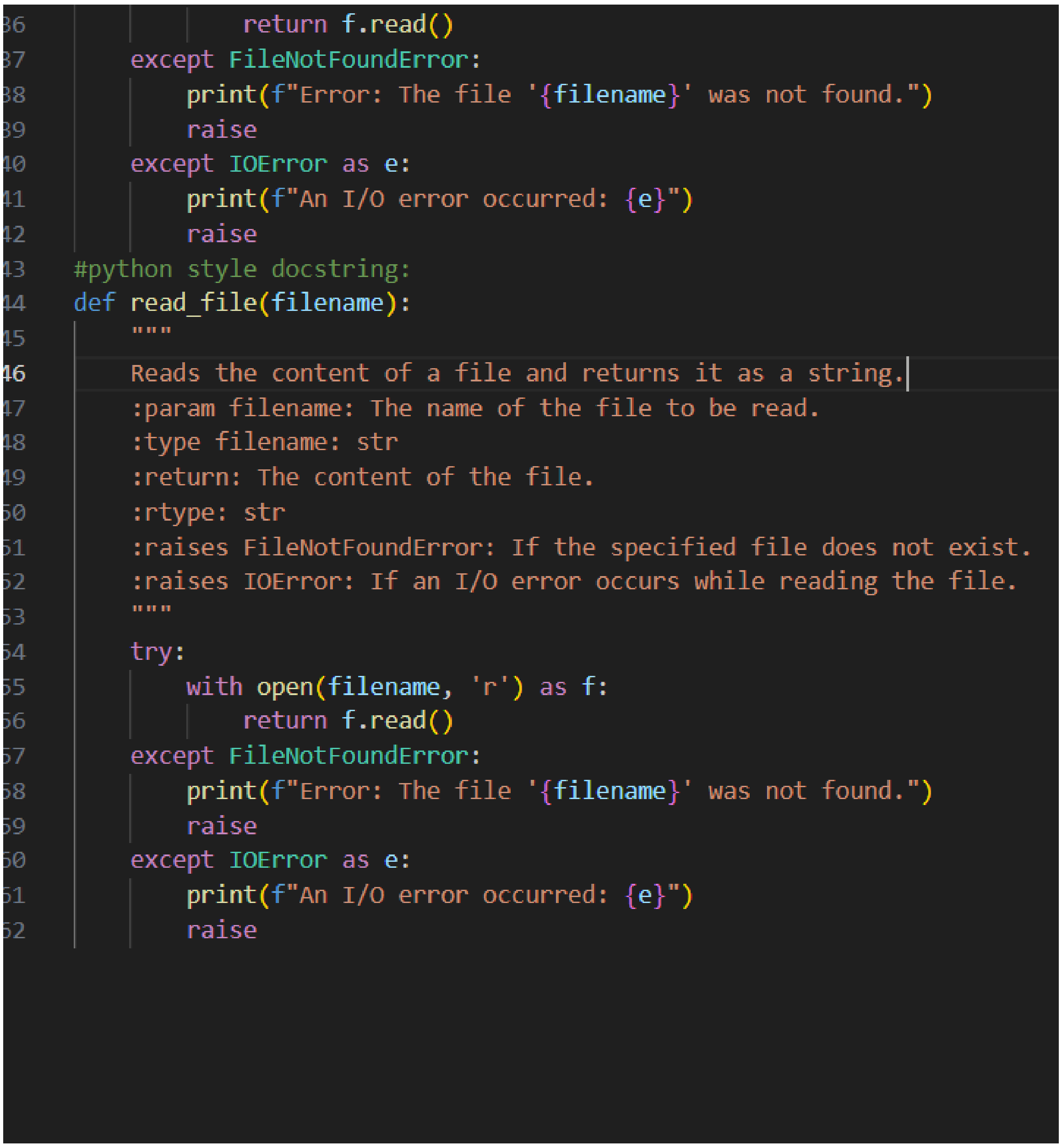
2. Identify which style best explains exception handling.

3. Justify your recommendation.

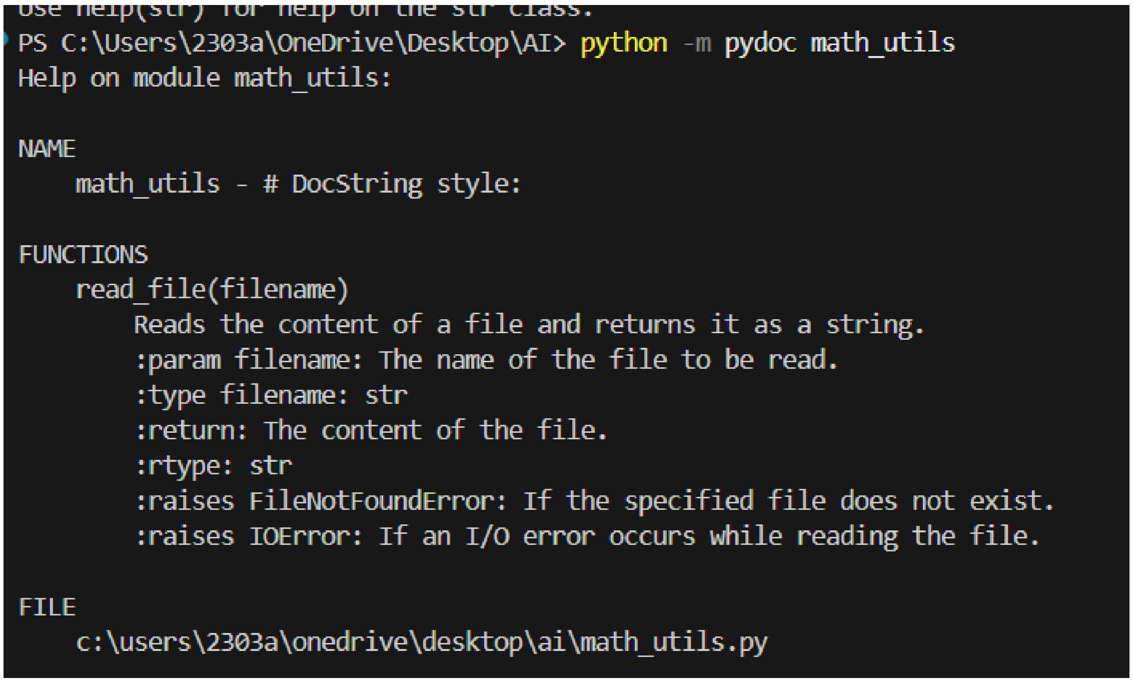


math_utils.py > read_file

```python
# DocString style:
def read_file(filename):
    """
    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(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
# Google Style Docstring:
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.
    """
    try:
        with open(filename, 'r') as f:
            return f.read()
    except FileNotFoundError:
```



```python
            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
#python style docstring:
def read_file(filename):
    """
    Reads the content of a file and returns it as a string.
    :param filename: The name of the file to be read.
    :type filename: str
    :return: The content of the file.
    :rtype: str
    :raises FileNotFoundError: If the specified file does not exist.
    :raises IOError: If an I/O error occurs while reading the file.
    """
    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
```



```
Use help(str) for help on the str class.
PS C:\Users\2303a\OneDrive\Desktop\AI> python -m pydoc math_utils
Help on module math_utils:

NAME
    math_utils - # DocString style:

FUNCTIONS
    read_file(filename)
        Reads the content of a file and returns it as a string.
        :param filename: The name of the file to be read.
        :type filename: str
        :return: The content of the file.
        :rtype: str
        :raises FileNotFoundError: If the specified file does not exist.
        :raises IOError: If an I/O error occurs while reading the file.

FILE
    c:\users\2303a\onedrive\desktop\ai\math_utils.py
```