AI Assisted Coding

Assignment – 9.5

Name : T.Phanindra

Ht.no : 2303A51703

Batch : 24

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

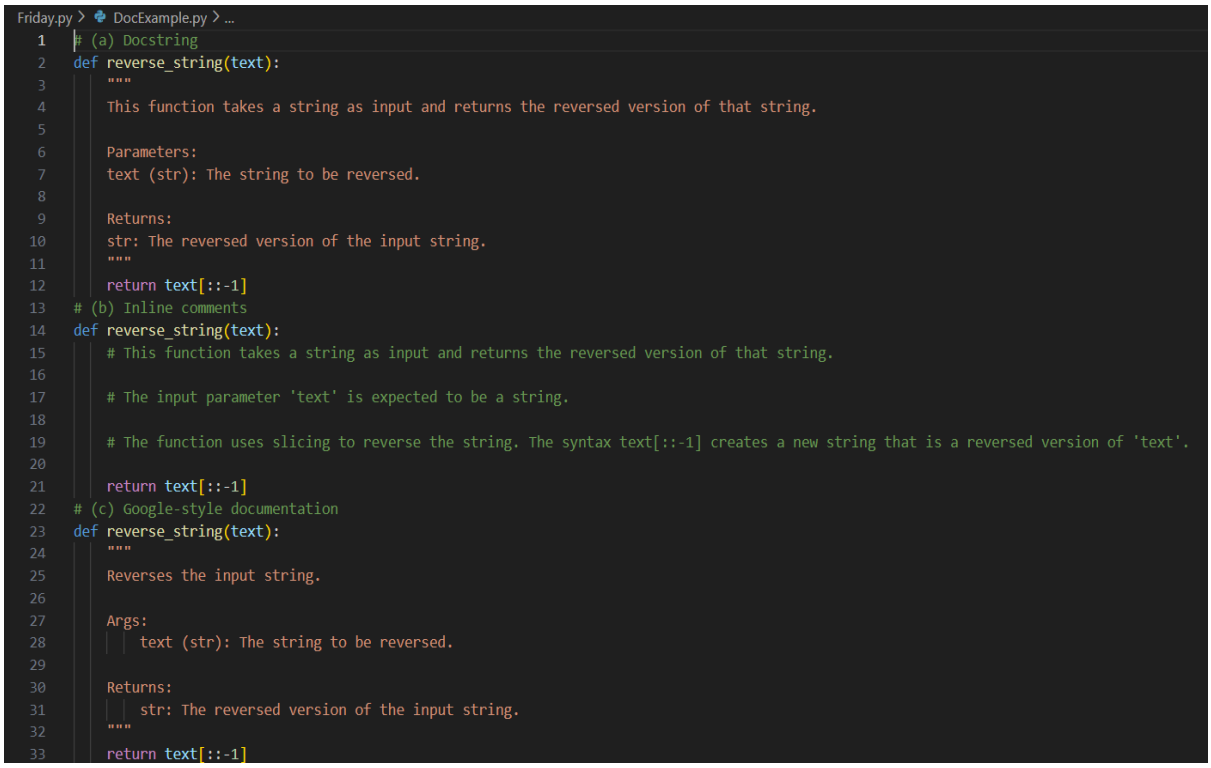
o (b) Inline comments

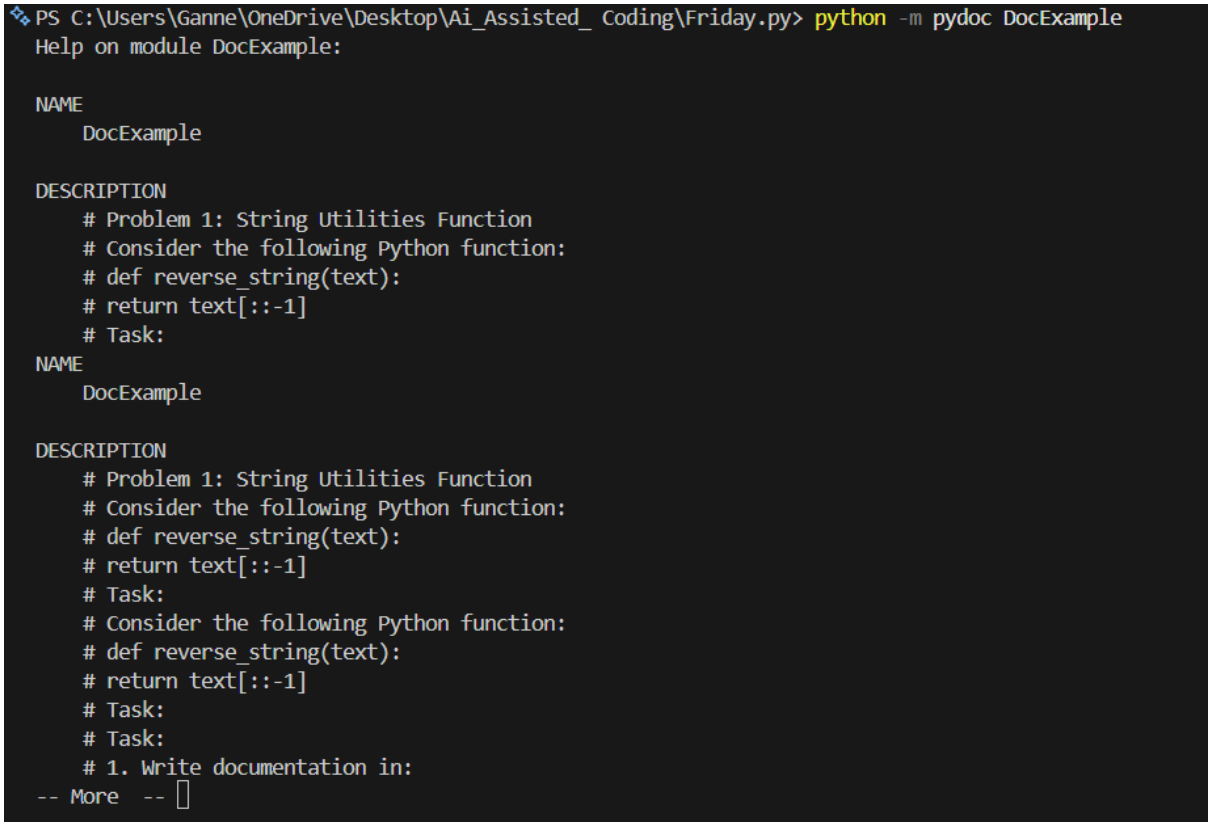
o (c) Google-style documentation

2. Compare the three documentation styles.

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

library.





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 > ...

```python
# (a) Docstring
def check_strength(password):
    """
    This function checks the strength of a password by verifying if it is at least 8 characters long.

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

    Returns:
    bool: True if the password is strong (at least 8 characters), False otherwise.
    """
    return len(password) >= 8
# (b) Inline comments
def check_strength(password):
    # This function checks the strength of a password by verifying if it is at least 8 characters long.

    # The input parameter 'password' is expected to be a string.

    # 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.

    return len(password) >= 8
# (c) Google-style documentation
def check_strength(password):
    """
    Checks the strength of a password.

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

    Returns:
        bool: True if the password is strong (at least 8 characters), False otherwise.
    """
    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 > ...

```python
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__)

```

```
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

```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\Ganne\OneDrive\Desktop\Ai_Assisted_ Coding\Friday.py> python -m pydoc -w math_util
wrote 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_ut
    os.system(cmd + ' "' + filename + '"')
KeyboardInterrupt
PS C:\Users\Ganne\OneDrive\Desktop\Ai_Assisted_ Coding\Friday.py> python -m pydoc -w math_util
wrote 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_ut
wrote 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_ut
il.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
server> b
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()

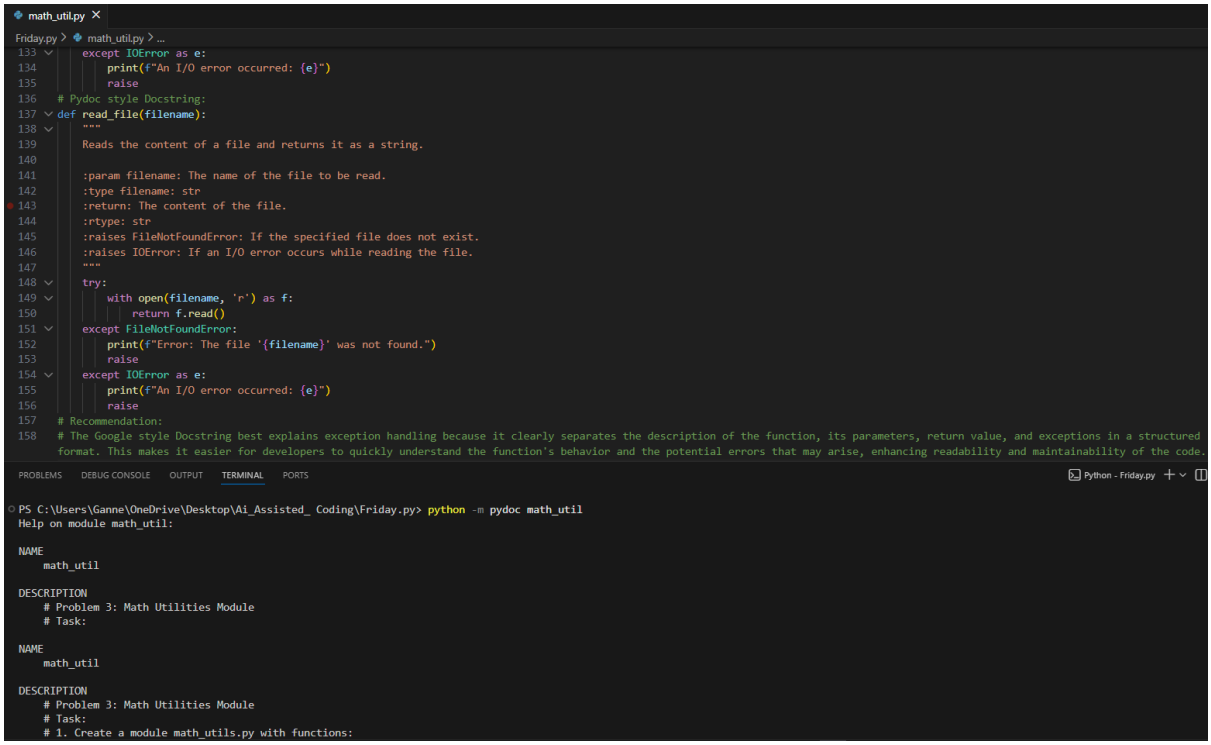
Task:

1. Write documentation using all three formats.

2. Identify which style best explains exception handling.

3. Justify your recommendation.

```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:
        print(f"Error: The file '{filename}' was not found.")
        raise
    except IOError as e:
```


math_util.py
Friday.py > math_util.py > ...
except IOError as e:
print(f"An I/O error occurred: {e}")
raise
# Pydoc 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
Recommendation:
The Google style Docstring best explains exception handling because it clearly separates the description of the function, its parameters, return value, and exceptions in a structured 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.
PROBLEMS
DEBUG CONSOLE
OUTPUT
TERMINAL
PORTS
Python - Friday.py
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
Problem 3: Math Utilities Module
Task:
NAME
math_util
DESCRIPTION
Problem 3: Math Utilities Module
Task:
1. Create a module math_utils.py with functions: