AI Assisted Coding

ASSIGNMENT 9.5

Name: T.Akshaya

HT No:2303A52017

Batch: 31

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.

Code:

(a) Docstring

```python
# 1. Docstring Type Documentation
def reverse_string(text):
    """
    Reverses the given string.
    Parameters:
    text: The string to be reversed.
    Returns:
    The reversed string.
    """
    return text[::-1]
print(reverse_string.__doc__)
```

```
Reverses the given string.
Parameters:
text (str): The string to be reversed.
Returns:
str: The reversed string.
Example:
>>> reverse_string("hello")
'olleh'
```

(b) Inline comments

```python
# Inline Type Documentation
def reverse_string(text):
    return text[::-1] # Reverses the given string
#------------------------------
# Reverse the string "Hello, World!" and print the result
#------------------------------
```

(c) Google–style documentation

```python
# Google Style Documentation
def reverse_string(text: str) -> str:
    """Reverses the given string.

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

    Returns:
        str: The reversed string.

    """

    return text[::-1]
input_text = input("Enter a string to reverse: ")
if not isinstance(input_text, str):
    raise ValueError("Input must be a string.")
print(reverse_string(input_text))
print(reverse_string.__doc__)
```

```
Enter a string to reverse: hello
olleh
Reverses the given string.

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

Returns:
    str: The reversed string.
```

2. Compare the three documentation styles.

Aspect	Docstring	Inline Comments	Google–Style
Scope	Function	Line	Function

Structure	Basic	None	Structured
Readability	Good	Simple	Clear
Detail Level	Medium	Low	High
Tool Support	Yes	No	Yes
Professional Use	Moderate	Low	High
Best For	Scripts	Logic	Libraries

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

Google-style documentation is most suitable for a utility-based string library because it reflects professional coding standards and industry practices.

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.

```python
# Inline Comment
# This function checks if the password is strong enough by ensuring it has at least 8 characters.
def check_strength(password: str) -> bool:
    # Doctype Comment
    """
    Parameters:
        password (str): The password to be checked for strength.
    Returns:
        bool: True if the password is strong (at least 8 characters), False otherwise.
    """
    # Google Docstring Comment
    """
    A strong password is defined as one that has at least 8 characters.
    This function checks the length of the password and returns True
    if it meets the requirement, otherwise it returns False.
    Args:
        password (str): The password to be evaluated for strength.
    Returns:
        bool: True if the password is strong, False if it is weak.
    Example:
    >>> check_strength("password123")
    True
    """
    return len(password) >= 8 # This is the minimum length requirement for a strong password.
print(check_strength.__doc__)
print(check_strength("password123"))  # This should return True since the password is strong.
```

```
Parameters:
    password (str): The password to be checked for strength.
Returns:
    bool: True if the password is strong (at least 8 characters), False otherwise.

True
```

2. Compare documentation styles for security-related code.

Aspect	Docstring	Inline Comments	Google-Style
Clarity	Good	Basic	Very Clear
Structure	Simple	None	Structured
Security Explanation	Limited	Minimal	Detailed
Professional Use	Moderate	Low	High
Maintainability	Medium	Low	High
Industry Preference	Acceptable	Rare	Strong

3. Recommend the most appropriate style.

For security-related code like password validation, Google-style documentation is most appropriate because it ensures clarity, professionalism, and maintainability.

Problem 3: Math Utilities Module

Task:

1. Create a module math_utils.py with functions:

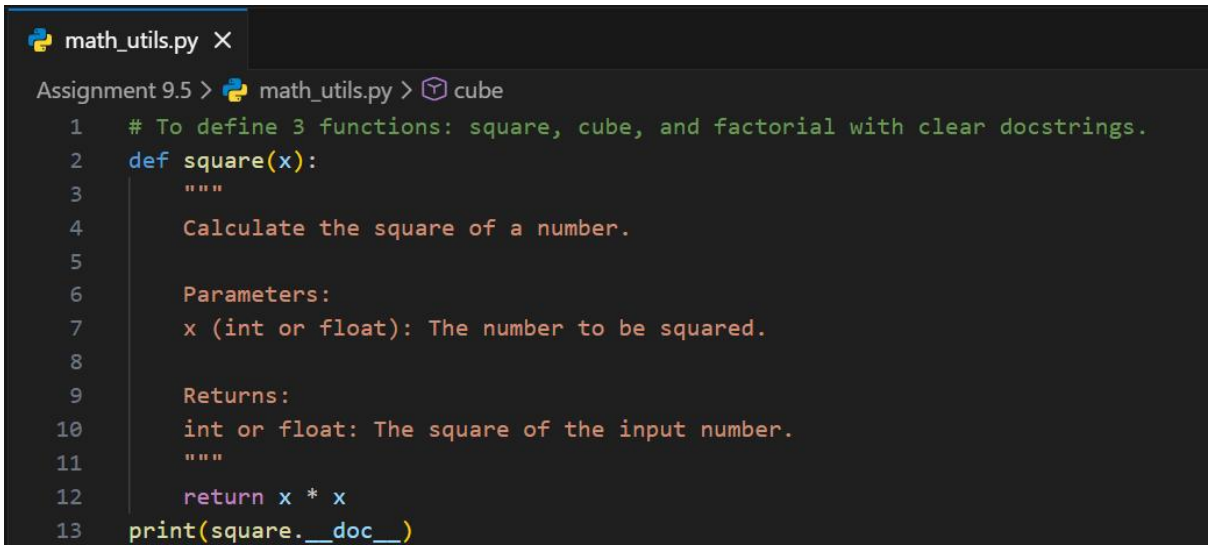
o square(n)

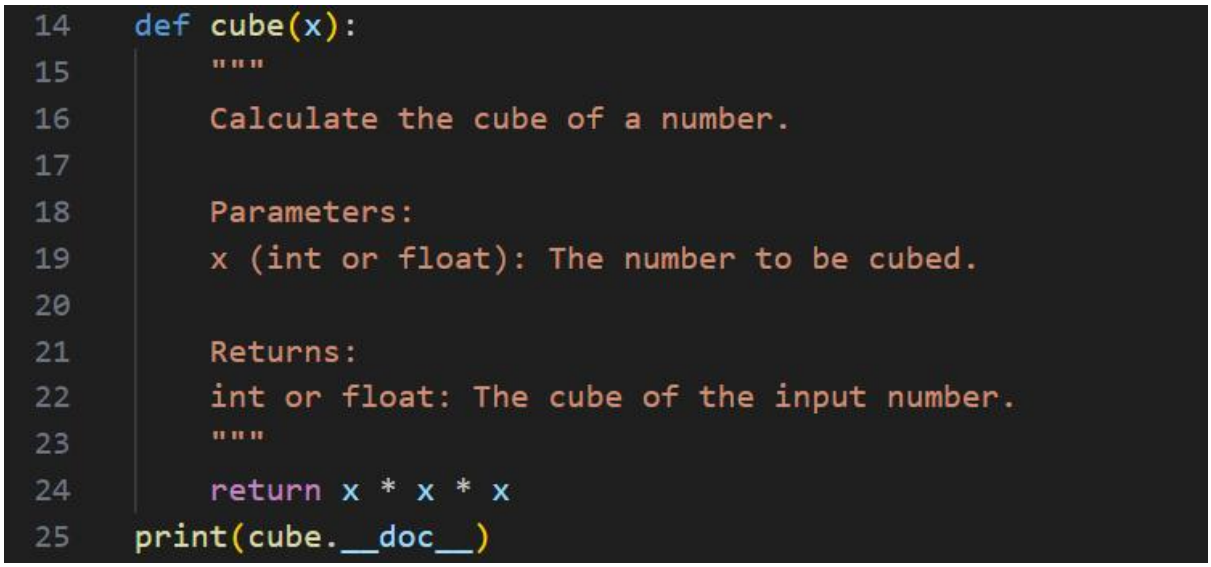
o cube(n)

o factorial(n)

2. Generate docstrings automatically using AI tools.

3. Export documentation as an HTML file.





```python
def factorial(n):
    """
    Calculate the factorial of a non-negative integer.
    Parameters:
    n (int): The non-negative integer for which to calculate the factorial.
    Returns:
    int: The factorial of the input number.
    Raises:
    ValueError: If n is negative.
    """
    if n < 0:
        raise ValueError("Factorial is not defined for negative numbers.")
    elif n == 0 or n == 1:
        return 1
    else:
        result = 1
        for i in range(2, n + 1):
            result *= i
        return result
print(factorial.__doc__)
```

3. Export documentation as an HTML file.

```
PS Z:\AIAC\Assignment 9.5> python -m pydoc -w .\math_utils.py

  Calculate the square of a number.

  Parameters:
  x (int or float): The number to be squared.

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


  Calculate the cube of a number.

  Parameters:
  x (int or float): The number to be cubed.

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


  Calculate the factorial of a non-negative integer.
  Parameters:
  n (int): The non-negative integer for which to calculate the factorial.
  Returns:
  int: The factorial of the input number.
  Raises:
  ValueError: If n is negative.

  wrote math_utils.html
```

Problem 4: Attendance Management Module

Task:

1. Create a module attendance.py with functions:

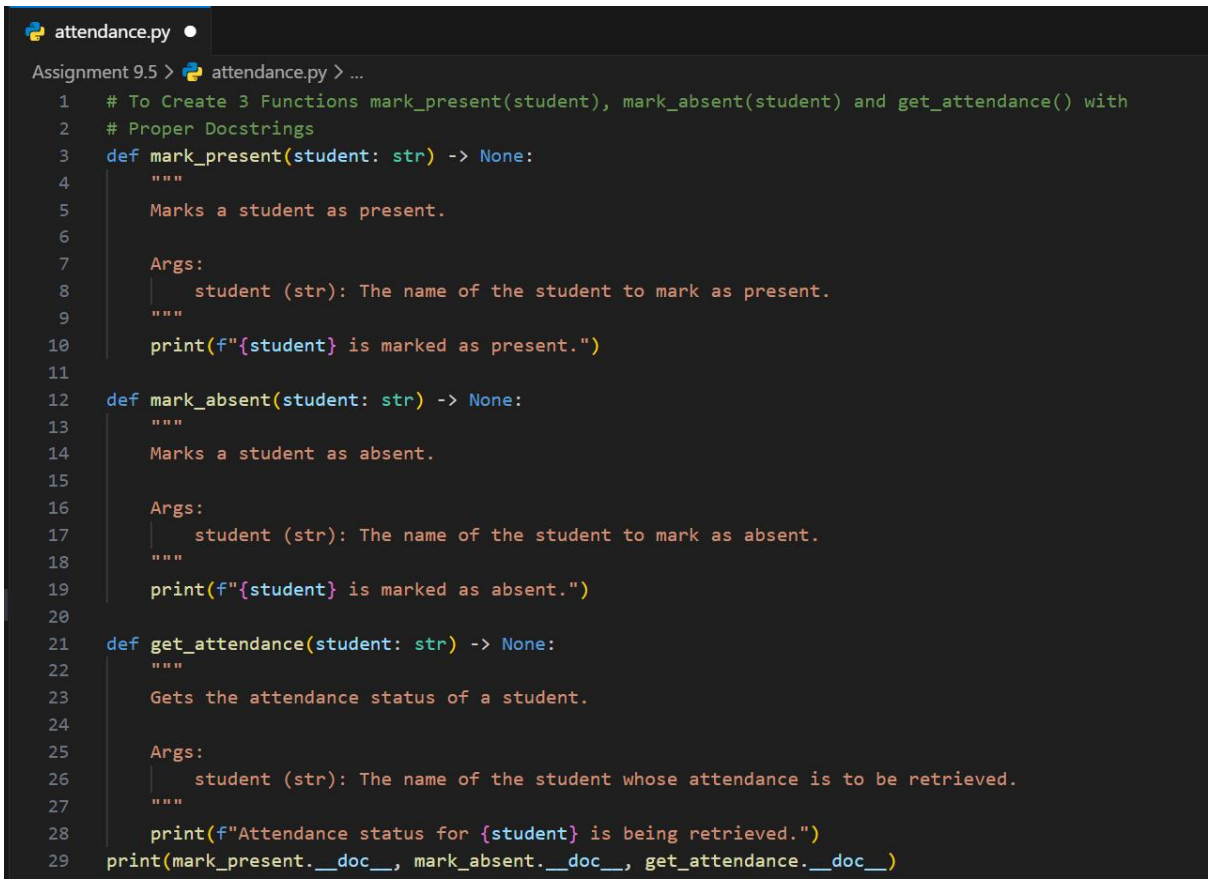
o mark_present(student)

o mark_absent(student)

o get_attendance(student)

2. Add proper docstrings.

3. Generate and view documentation in terminal and browse



```
PS Z:\AIAC\Assignment 9.5> python -m pydoc -p 8080
Server ready at http://localhost:8080/
Server commands: [b]rowser, [q]uit
server> b
server>
Marks a student as present.

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

Marks a student as absent.

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

Gets the attendance status of a student.

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

```
Marks a student as present.

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

Marks a student as absent.

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

Gets the attendance status of a student.

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

Problem 5: File Handling Function

Consider the function:

def read_file(filename):

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

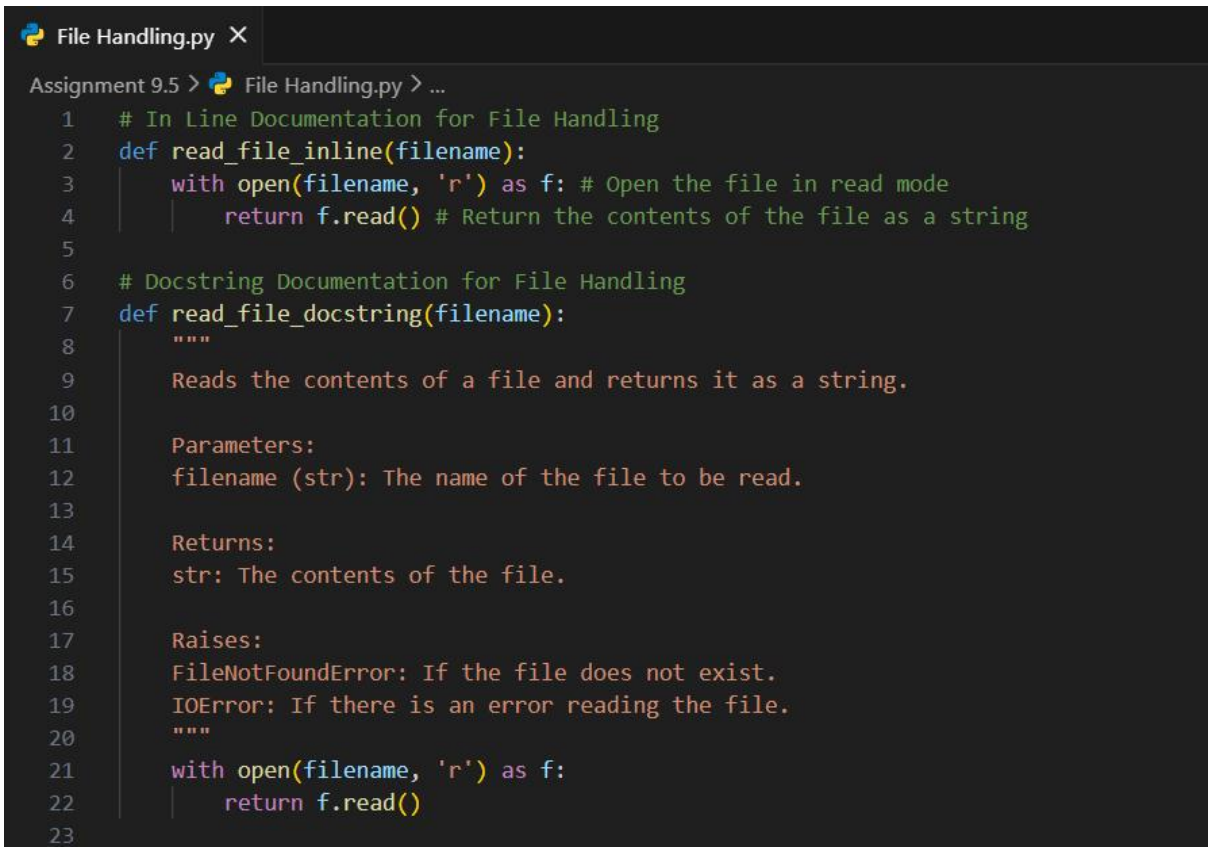
return f.read()

Task:

1. Write documentation using all three formats.

2. Identify which style best explains exception handling.

3. Justify your recommendation.



```python
# Google Style Docstring Documentation for File Handling
def read_file_google_style(filename: str) -> str:
    """
    Reads the contents of a file and returns it as a string.

    Args:
        filename (str): The name of the file to be read.
    Returns:
        str: The contents of the file.
    Raises:
        FileNotFoundError: If the file does not exist.
        IOError: If there is an error reading the file.
    """
    with open(filename, 'r') as f:
        return f.read()


print(read_file_inline.__doc__)
print(read_file_docstring.__doc__)
print(read_file_google_style.__doc__)
```

```
None

Reads the contents of a file and returns it as a string.

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

Returns:
str: The contents of the file.

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


Reads the contents of a file and returns it as a string.

Args:
    filename (str): The name of the file to be read.
Returns:
    str: The contents of the file.
Raises:
    FileNotFoundError: If the file does not exist.
    IOError: If there is an error reading the file.
```

Identify which style best explains exception handling, Justify your recommendation.

Google Style Docstring

It clearly separates exceptions under a dedicated "Raises" section, making error handling easy to understand and professionally structured.