AI Assiginment

Assignment -9.5

Hallticket no:2303A510A6

Batch no:02

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

Documentation:

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

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

    Returns:
        str: The reversed string.
    """
    return text[::-1]

# Example usage
input_string = "Hello, World!"
reversed_string = reverse_string(input_string)

print(f"Original string: {input_string}")
print(f"Reversed string: {reversed_string}")

def reverse_string(text):
    # Reverse the string using slicing
    return text[::-1]

# Example usage
input_string = "Hello, World!"
reversed_string = reverse_string(input_string)
print("Original string:", input_string)
print("Reversed string:", reversed_string)

def reverse_string(text)
    """
    Reverse the given string.

    Args:
        text (str): The input string to reverse.

    Returns:
        str: The reversed string.
    """
    return text[::-1]
```

```
print(f"Original string: {input_string}")
print(f"Reversed string: {reversed_string}")

def reverse_string(text):
    # Reverse the string using slicing
    return text[::-1]

# Example usage
input_string = "Hello, World!"
reversed_string = reverse_string(input_string)
print("Original string:", input_string)
print("Reversed string:", reversed_string)

def reverse_string(text)
    """
    Reverse the given string.

    Args:
        text (str): The input string to reverse.

    Returns:
        str: The reversed string.
    """
    return text[::-1]
# Example usage
input_string = "Hello, World!"
reversed_string = reverse_string(input_string)

print(f"Original string: {input_string}")
print(f"Reversed string: {reversed_string}")
```

```
PS C:\Users\shyam\ai assistes code> python -m pydoc ass67
Help on module ass67:

NAME
    ass67

DESCRIPTION
    Module: reverse_string_examples
    This file demonstrates three documentation styles.

FUNCTIONS
    reverse_string_docstring(text)
        Takes a string as input and returns the reversed string.

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

        Returns:
            str: The reversed string.

    reverse_string_google(text)
        Reverses the given string.

        Args:
            text (str): The string to reverse.
-- More  --
```

index

ass67 c:\users\shyam\ai assistes code\ass67.py

Module: reverse_string_examples
This file demonstrates three documentation styles.

Functions

reverse_string_docstring(text)
Takes a string as input and returns the reversed string.

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

Returns:
str: The reversed string.

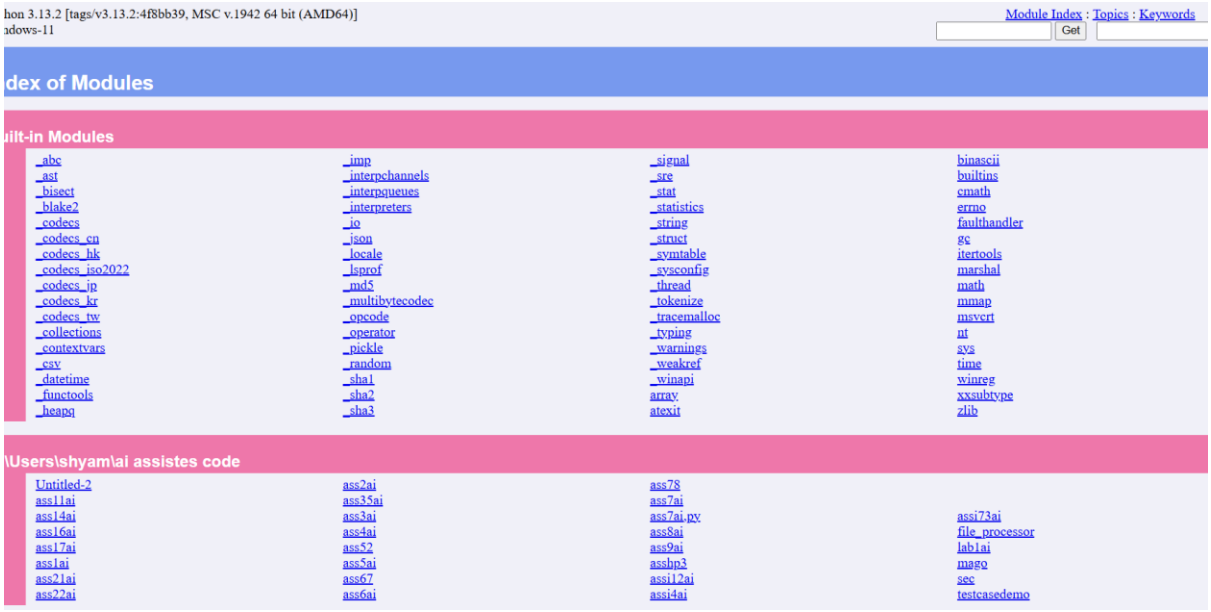
reverse_string_google(text)
Reverses the given string.

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

Returns:
str: The reversed string.

Example:
>>> reverse_string_google("hello")
'olleh'

reverse_string_inline(text)



Comparison of the Three Documentation Styles

1. Docstring Style

   Uses a clear description inside triple quotes.

   Explains parameters and return values.

   Easy to read and understand.

2. Inline Comments Style

   Uses comments inside the code.

   Explains small parts of the program.

   Useful for simple explanations but not detailed.

3. Google-Style Documentation

   Structured and well-formatted.

   Clearly separates Args, Returns, and Examples.

   Very professional and easy to maintain.

Recommendation

For a utility-based string library, the Google-style documentation is the most suitable.
It is clear, organized, and gives complete information, which is important for reusable code in libra

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.

```
Help on module ass66:

NAME
    ass66

DESCRIPTION
    Module: password_strength_checker
    This module demonstrates three documentation styles for a function.

FUNCTIONS
    check_strength_docstring(password)
        Checks whether a password is strong based on its length.

        A password is considered strong if it has at least 8 characters.

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

index

ass66 c:\users\shyam\ai assistes code\ass66.py

Module: password_strength_checker
This module demonstrates three documentation styles for a function.

Functions

check_strength_docstring(password)
Checks whether a password is strong based on its length.

A password is considered strong if it has at least 8 characters.

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

Returns:
bool: True if password length is 8 or more, False otherwise.

check_strength_google(password)
Checks whether a password is strong based on its length.

Args:
password (str): The password to evaluate.

Returns:
bool: True if the password has at least 8 characters,
False otherwise.

Example:
>>> check_strength_google("mypassword")
True

check_strength_inline(password)

Comparison of Documentation Styles (Security Code)

Inline comments**:** Explain small parts of code, but too many can make code messy.

Docstrings**:** Describe what the function does clearly and are good for documentation.

Google style**:** Structured, clear, and shows parameters and return values.

Recommendation

For security-related code, Google style docstrings are best because they are clear, organized, and easy for teams to understand and review.

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.

Code:

```
math_utils.py

A simple math utilities module that provides basic mathematical
- square
- cube
- factorial
"""
def square(n):
    """
    Return the square of a number.

    Args:
        n (int or float): The number to be squared.

    Returns:
        int or float: The square of n.
    """
    return n * n

def cube(n):
    """
    Return the cube of a number.

    Args:
        n (int or float): The number to be cubed.

    Returns:
        int or float: The cube of n.
    """
    return n * n * n

def factorial(n):
    """
    Calculate the factorial of a non-negative integer.

    Args:
        n (int): A non-negative integer.

    Returns:
        int: The factorial of n.

    Raises:
        ValueError: If n is negative.
    """
    if n < 0:
```

```
se help(str) for help on the str class.
5 C:\Users\shyam\ai assistes code> python -m pydoc ass66
elp on module ass66:

AME
   ass66 - math_utils.py

ESCRIPTION
   A simple math utilities module that provides basic mathematical functions:
   - square
   - cube
   - factorial

UNCTIONS
   cube(n)
       Return the cube of a number.

       Args:
           n (int or float): The number to be cubed.

       Returns:
           int or float: The cube of n.

   factorial(n)
       Calculate the factorial of a non-negative integer.

       Args:
           n (int): A non-negative integer.

       Returns:
           int: The factorial of n.

       Raises:
           ValueError: If n is negative.

   square(n)
       Return the square of a number.

       Args:
           n (int or float): The number to be squared.

       Returns:
```

index

ass66 c:\users\shyam\ai assistes code\ass66.py

math_utils.py

A simple math utilities module that provides basic mathematical functions:
- square
- cube
- factorial

Functions

cube(n)
Return the cube of a number.

Args:
n (int or float): The number to be cubed.

Returns:
int or float: The cube of n.

factorial(n)
Calculate the factorial of a non-negative integer.

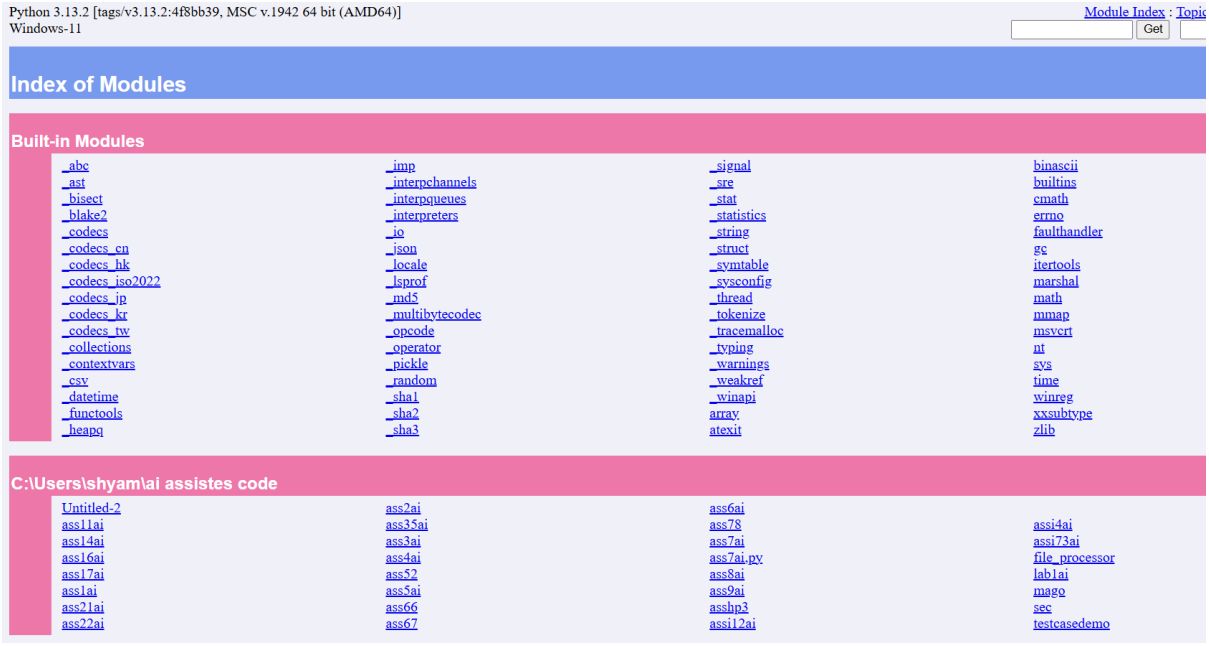
Args:
n (int): A non-negative integer.

Returns:
int: The factorial of n.

Raises:
ValueError: If n is negative.

square(n)
Return the square of a number.

Args:
n (int or float): The number to be squared.



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

Documentation:

```
    """
    # Dictionary to store attendance data
    attendance_record = {}
    def mark_present(student):
        """
        Mark a student as present.

        Args:
            student (str): Name of the student.

        Returns:
            None
        """
        attendance_record[student] = "Present"
    def mark_absent(student):
        """
        Mark a student as absent.

        Args:
            student (str): Name of the student.

        Returns:
            None
        """
        attendance_record[student] = "Absent"
    def get_attendance(student):
        """
        Get the attendance status of a student.

        Args:
            student (str): Name of the student.

        Returns:
            str: Attendance status ("Present", "Absent", or "Not Recorded").
        """
        return attendance_record.get(student, "Not Recorded")
```

```
elp on module ass66:

AME
   ass66 - attendance.py

ESCRIPTION
   A simple Attendance Management Module.
   Provides functions to mark student attendance and check status.

UNCTIONS
   get_attendance(student)
       Get the attendance status of a student.

       Args:
           student (str): Name of the student.

       Returns:
           str: Attendance status ("Present", "Absent", or "Not Recorded").

   mark_absent(student)
       Mark a student as absent.

       Args:
           student (str): Name of the student.

       Returns:
           None

   mark_present(student)
       Mark a student as present.

       Args:
           student (str): Name of the student.

       Returns:
           None

ATA
   attendance_record = {}

ILE
```

index
ass66 c:\users\shyam\ai assistes code\ass66.py

attendance.py

A simple Attendance Management Module.
Provides functions to mark student attendance and check status.

Functions

get_attendance(student)
Get the attendance status of a student.

Args:
student (str): Name of the student.

Returns:
str: Attendance status ("Present", "Absent", or "Not Recorded").

mark_absent(student)
Mark a student as absent.

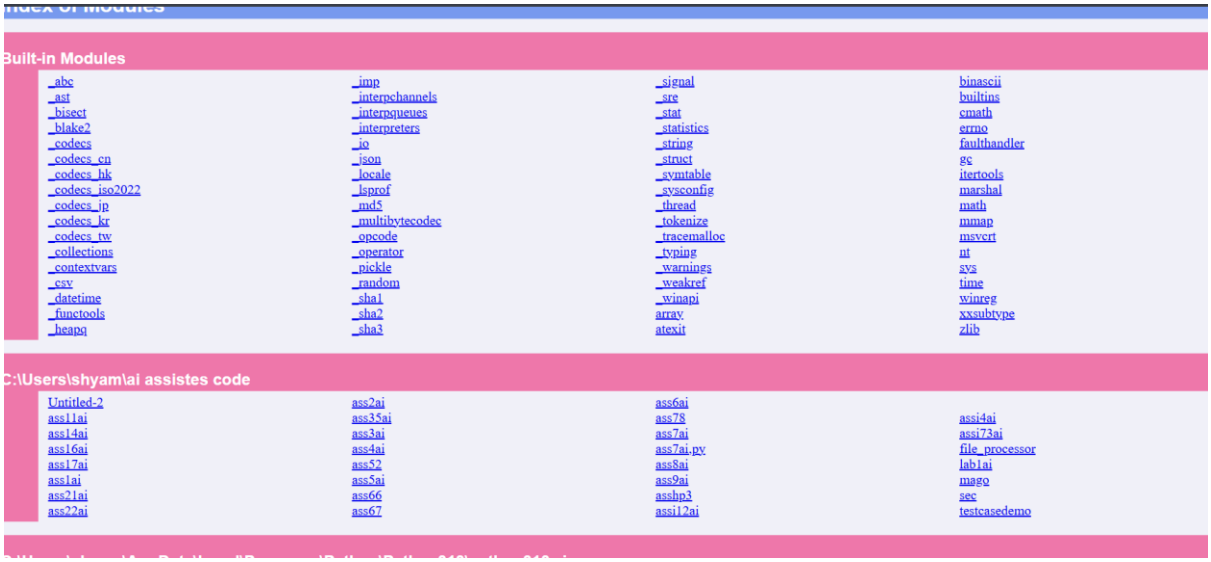
Args:
student (str): Name of the student.

Returns:
None

mark_present(student)
Mark a student as present.

Args:
student (str): Name of the student.

Returns:
None



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
"""
read_file.py

This module contains a function to read the contents of a file.
It demonstrates different documentation styles.
"""

def read_file_inline(filename):
    # Open the file in read mode
    with open(filename, 'r') as f:
        # Read and return file content
        return f.read()

def read_file_basic(filename):
    """
    Read and return the contents of a file.

    Parameters:
        filename (str): Name of the file to read.

    Returns:
        str: Content of the file.
    """
    with open(filename, 'r') as f:
        return f.read()

def read_file_google(filename):
    """
    Read and return the contents of a file.

    Args:
        filename (str): The path to the file.

    Returns:
        str: The content of the file.

    Raises:
        FileNotFoundError: If the file does not exist.
        IOError: If an error occurs while reading the file.
    """
    with open(filename, 'r') as f:
        return f.read()
```

```
PS C:\Users\shyam\ai assistes code> python -m pydoc ass66
Help on module ass66:

NAME
    ass66 - read_file.py

DESCRIPTION
    This module contains a function to read the contents of a file.
    It demonstrates different documentation styles.

FUNCTIONS
    read_file_basic(filename)
        Read and return the contents of a file.

        Parameters:
            filename (str): Name of the file to read.

        Returns:
            str: Content of the file.

    read_file_google(filename)
        Read and return the contents of a file.

        Args:
            filename (str): The path to the file.

        Returns:
            str: The content of the file.

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

    read_file_inline(filename)

FILE
```

index
ss66 c:\users\shyam\ai assistes code\ass66.py

ead_file.py

his module contains a function to read the contents of a file.
demonstrates different documentation styles.

unctions

read_file_basic(filename)
Read and return the contents of a file.

Parameters:
filename (str): Name of the file to read.

Returns:
str: Content of the file.

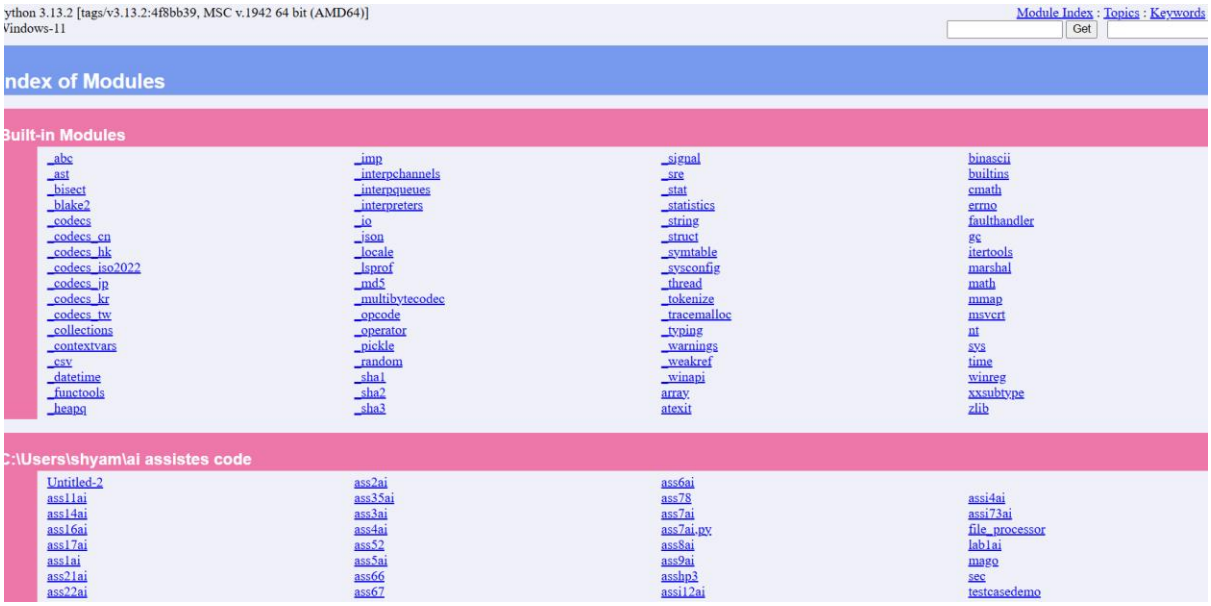
read_file_google(filename)
Read and return the contents of a file.

Args:
filename (str): The path to the file.

Returns:
str: The content of the file.

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

read_file_inline(filename)



Best Style for Exception Handling

Google Style Docstring is the best style.

Justification

It has a **“**Raises” section to clearly explain errors.

It makes exception handling easy to understand.

It improves code readability.

It is useful for team projects and professional coding.

It clearly shows how the function handles possible problems.