ASSIGNMENT - 9.1

Name : V Puneet Teja

Hall Ticket No : 2303a51084

Batch No : 02

Course Title : AI Assisted Coding

Instructor's Name : Mr. S Naresh Kumar

Problem - 1:

```
# (a) Docstring
def reverse_string(text):
    """
    Reverses the given string.

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

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

```
# (b) Inline comments
def reverse_string(text):
    # Reverse the input string using slicing
    return text[::-1]
```

```python
# (c) Google-style documentation
def reverse_string(text):
    """
    Reverses the given string.

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

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

# Comparison of documentation styles:

# 1. Docstring: This style provides a clear and structured way to document the function, including the purpose,
#    arguments, and return value. It is easily accessible through the help() function and is widely used in Python.
# 2. Inline comments: This style is less formal and may not provide as much detail as a docstring. It is useful for
#    explaining specific lines of code
# 3. Google-style documentation: This style is similar to the docstring but follows a specific format that is widely
#    recognized in the Python community. It provides a clear structure for documenting functions and is often preferred for
#    larger projects.

# Recommendation:
# For a utility-based string library, the most suitable style would be the Google-style documentation.
# This style provides a clear and consistent format for documenting functions, making it easier for users to understand
# the purpose and usage of each function in the library. Additionally, it is widely recognized and accepted in the Python
# community, which can enhance the readability and maintainability of the code.
```

Problem - 2:

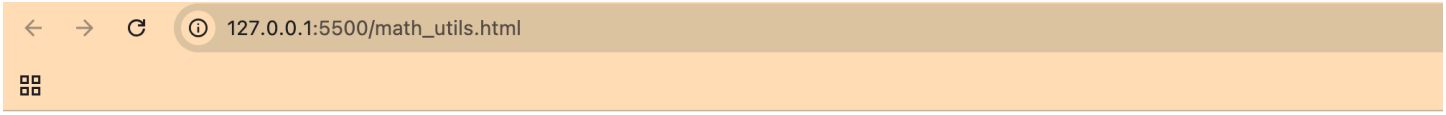
```python
#  (a) Docstring
def check_strength(password):
    """
    Checks if a password is strong.

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

    Returns:
        bool: True if the password is strong, False otherwise.
    """
    return len(password) >= 8

#  (b) Inline comments
def check_strength(password):
    # Check if the password length is at least 8 characters
    return len(password) >= 8
```

```python
#  (c) Google-style documentation
def check_strength(password):
    """
    Checks if a password is strong.

    Args:
        password (str): The password to check.
    Returns:
        bool: True if the password is strong, False otherwise.
    """
    return len(password) >= 8

# Comparison of documentation styles for security-related code:
# 1. Docstring: This style provides a clear and structured way to document the function, including the purpose,
#    arguments, and return value. It is easily accessible through the help() function and is
#    widely used in Python. However, it may not provide enough detail for security-related code.
# 2. Inline comments: This style is less formal and may not provide as much detail
#    as a docstring. It is useful for explaining specific lines of code, but it may not be sufficient for security-related code.
# 3. Google-style documentation: This style is similar to the docstring but follows a
#    specific format that is widely recognized in the Python community. It provides a clear structure for documenting functions
# Recommendation:
# For security-related code, the most appropriate style would be the Google-style documentation. This style
# provides a clear and consistent format for documenting functions, making it easier for users to understand the purpose and usa
```

Problem - 3:

```python
# math_utils.py
def square(n):
    """
    Returns the square of a number.

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

    Returns:
        int or float: The square of the input number.
    """
    return n ** 2

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

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

    Returns:
        int or float: The cube of the input number.
    """
    return n ** 3
```

```python
def factorial(n):
    """
    Returns the factorial of a number.

    Args:
        n (int): The number to calculate the factorial of. Must be a non-negative integer.

    Returns:
        int: The factorial of the input number.
    """
    if n < 0:
        raise ValueError("Input must be a non-negative integer.")
    elif n == 0 or n == 1:
        return 1
    else:
        result = 1
        for i in range(2, n + 1):
            result *= i
        return result

    # export documentation as an HTML file
    # This can be done using tools like Sphinx or pydoc. For example, you can use pydoc to generate HTML documentation:
    # pydoc -w math_utils
```



index
math_utils /Users/aravindreddy/Desktop/My-Information/College/AI Assissted Coding/Assignments/math_utils.py

math_utils.py

Functions

cube(n)
Returns the cube of a number.

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

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

factorial(n)
Returns the factorial of a number.

Args:
n (int): The number to calculate the factorial of. Must be a non-negative integer.

Returns:
int: The factorial of the input number.

square(n)
Returns the square of a number.

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

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

Problem - 4:

```python
# # attendance.py
class Attendance:
    def __init__(self):
        self.attendance_record = {}

    def mark_present(self, student):
        """
        Marks a student as present.

        Args:
            student (str): The name of the student to mark as present.
        """
        self.attendance_record[student] = 'Present'

    def mark_absent(self, student):
        """
        Marks a student as absent.

        Args:
            student (str): The name of the student to mark as absent.
        """
        self.attendance_record[student] = 'Absent'

    def get_attendance(self, student):
        """
        Retrieves the attendance status of a student.

        Args:
            student (str): The name of the student to check attendance for.

        Returns:
            str: The attendance status of the student ('Present', 'Absent', or 'Not Recorded').
        """
        return self.attendance_record.get(student, 'Not Recorded')

    # To generate and view documentation in the terminal, you can use the help() function:
    # help(Attendance)
```

Problem - 5:

```python
# (a) Docstring
def read_file(filename):
    """
    Reads the contents of a file.

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

    Returns:
        str: The contents of the file.

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

# (b) Inline comments
def read_file(filename):
    # Attempt to open the file and read its contents
    with open(filename, 'r') as f:
        return f.read()
```

```python
# (c) Google-style documentation
def read_file(filename):
    """
    Reads the contents of a file.

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

    Returns:
        str: The contents of the file.

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