

AI Assisted Coding Assignment-9.5

2303A51543 BT: 29 VELDI.HRUTHIKA 20.02.2026



Problem 1: String Utilities Function

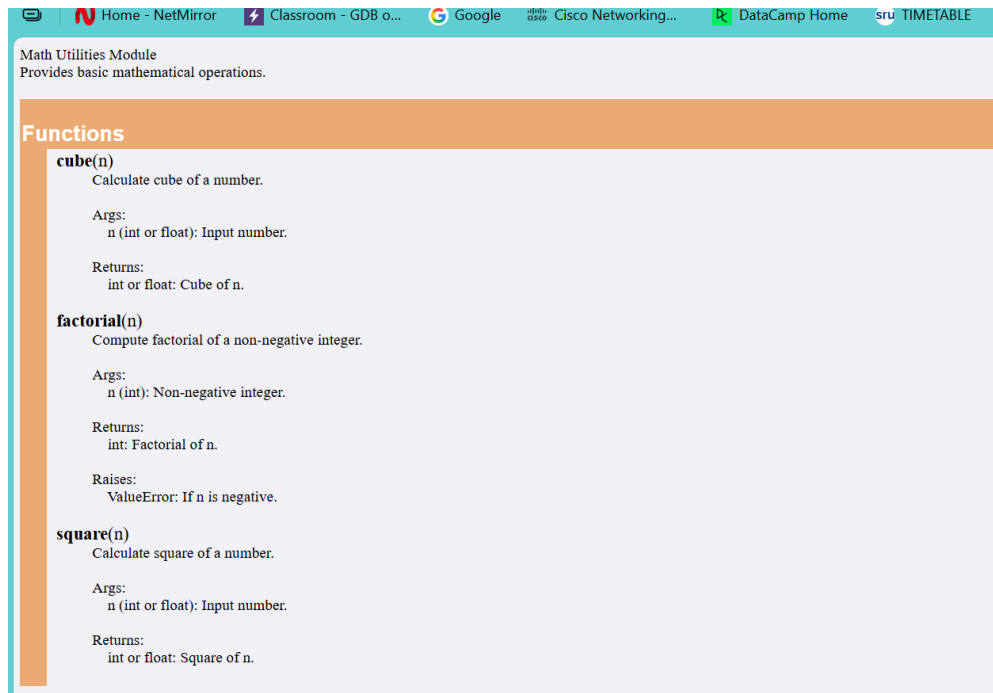
Consider the following Python function:

```
def reverse_string(text):  
    return text[::-1]
```

Task:

1. Write documentation in:
 - (a) Docstring
 - (b) Inline comments
 - (c) Google-style documentation
2. Compare the three documentation styles.
3. Recommend the most suitable style for a utility-based string library.

```
"""  
String Utilities Module  
Provides utility functions for string manipulation.  
"""  
  
def reverse_string(text):  
    """  
    Reverse the input string.  
  
    Args:  
        text (str): The string to reverse.  
  
    Returns:  
        str: The reversed string.  
    """  
    return text[::-1]
```



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.

```
"""  
Password Utilities Module  
Provides basic password validation functions.  
"""  
  
def check_strength(password):  
    """  
    Validate password strength.  
  
    Args:  
        password (str): The password string.  
  
    Returns:  
        bool: True if password length is at least 8 characters, else False.  
    """
```

```
"""
return len(password) >= 8
```

Password Utilities Module
Provides basic password validation functions.

Functions

check_strength(password)

Validate password strength.

Args:

password (str): The password string.

Returns:

bool: True if password length is at least 8 characters, else False.



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 Utilities Module
Provides basic mathematical operations.
"""

def square(n):
    """
    Calculate square of a number.

    Args:
        n (int or float): Input number.
```

```

Returns:
    int or float: Square of n.
"""
return n * n

def cube(n):
    """
    Calculate cube of a number.

    Args:
        n (int or float): Input number.

    Returns:
        int or float: Cube of n.
    """
    return n * n * n

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

    Args:
        n (int): Non-negative integer.

    Returns:
        int: Factorial of n.

    Raises:
        ValueError: If n is negative.
    """
    if n < 0:
        raise ValueError("Factorial not defined for negative numbers")

    if n == 0 or n == 1:
        return 1

    result = 1
    for i in range(2, n + 1):
        result *= i

    return result

```

Math Utilities Module
Provides basic mathematical operations.

Functions

cube(n)

Calculate cube of a number.

Args:

n (int or float): Input number.

Returns:

int or float: Cube of n.

factorial(n)

Compute factorial of a non-negative integer.

Args:

n (int): Non-negative integer.

Returns:

int: Factorial of n.

Raises:

ValueError: If n is negative.

square(n)

Calculate square of a number.

Args:

n (int or float): Input number.

Returns:

int or float: Square of n.



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

```
"""
Attendance Management Module
Handles student attendance records.
"""

attendance_record = {}

def mark_present(student):
    """
    Mark student as present.

    Args:
        student (str): Student name.
    """
    attendance_record[student] = "Present"

def mark_absent(student):
```

```

"""
Mark student as absent.

Args:
    student (str): Student name.
"""
attendance_record[student] = "Absent"

def get_attendance(student):
    """
    Get attendance status of a student.

    Args:
        student (str): Student name.

    Returns:
        str: Attendance status or 'No record found'.
    """
    return attendance_record.get(student, "No record found")

```

Attendance Management Module
Handles student attendance records.

Functions

get_attendance(student)
Get attendance status of a student.

Args:
student (str): Student name.

Returns:
str: Attendance status or 'No record found'.

mark_absent(student)
Mark student as absent.

Args:
student (str): Student name.

mark_present(student)
Mark student as present.

Args:
student (str): Student name.



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.

```

"""
File Utilities Module
Provides file handling operations.
"""

def read_file(filename):
    """
    Read content from a file.

    Args:
        filename (str): File path.

    Returns:
        str: File content.

    Raises:
        FileNotFoundError: If file is missing.
        IOError: If file cannot be accessed.
    """
    with open(filename, 'r') as f:
        return f.read()

```

File Utilities Module
Provides file handling operations.

Functions

read_file(filename)
Read content from a file.

Args:
filename (str): File path.

Returns:
str: File content.

Raises:
FileNotFoundError: If file is missing.
IOError: If file cannot be accessed.