Lab Assignment_9.5

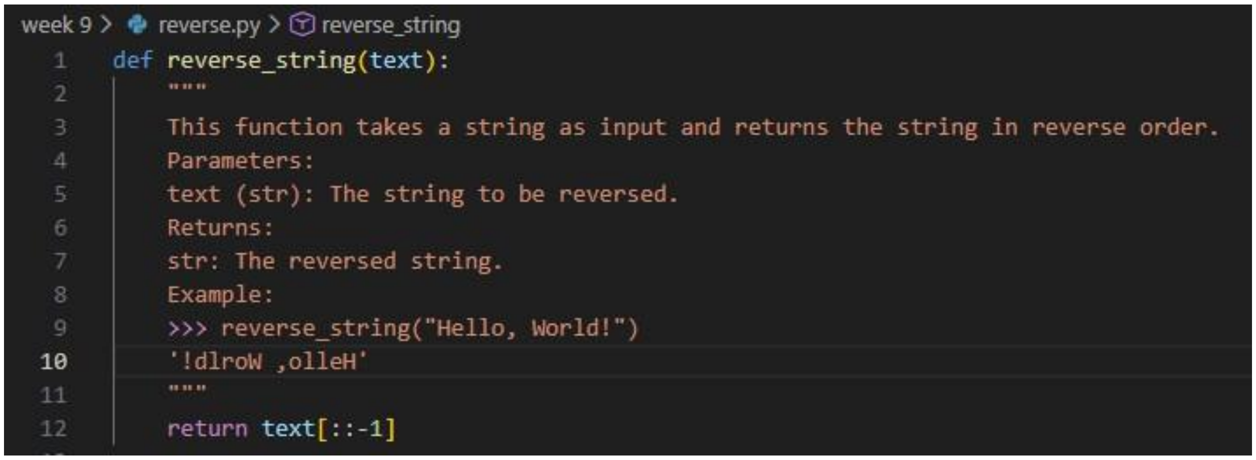
HT. No: 2303A51109
Name: M.Hasini
Batch-02

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.

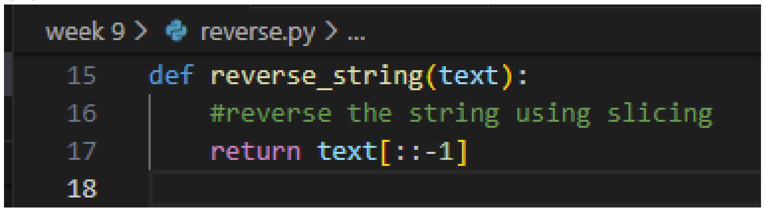
(a) Docstring Documentation:



week 9 > reverse.py > reverse_string

```
def reverse_string(text):
    """
    This function takes a string as input and returns the string in reverse order.
    Parameters:
    text (str): The string to be reversed.
    Returns:
    str: The reversed string.
    Example:
    >>> reverse_string("Hello, World!")
    '!dlroW ,olleH'
    """
    return text[::-1]
```

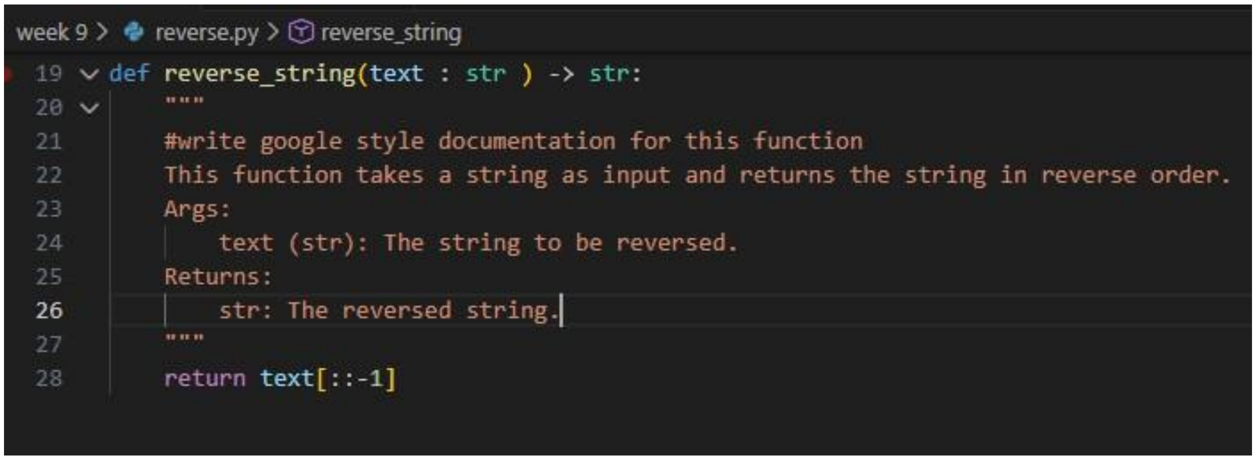
(b) Inline Comments



week 9 > reverse.py > ...

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

c) Google-Style Documentation



week 9 > reverse.py > reverse_string

```python
def reverse_string(text : str ) -> str:
    """
    #write google style documentation for this function
    This function takes a string as input and returns the string in reverse order.
    Args:
        text (str): The string to be reversed.
    Returns:
        str: The reversed string.
    """
    return text[::-1]
```

Comparison of Documentation Styles :

Style	Pros	Cons
Docstring	Simple, readable	Less standardized
Inline Comments	Easy to write	Limited explanation
Google-Style	Clear & professional	Slightly verbose

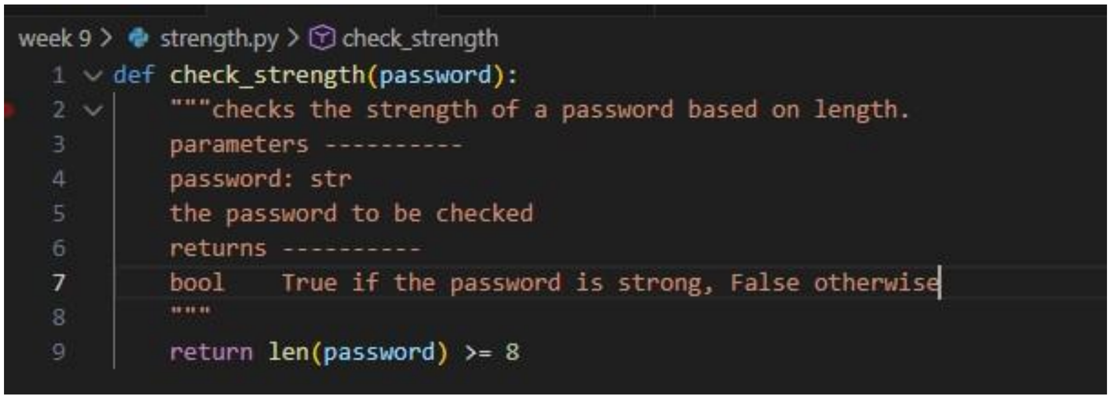
Google-style documentation is best for **string utility libraries** because it is structured and widely accepted in professional projects

2. Password Strength Checker
   Consider the function:

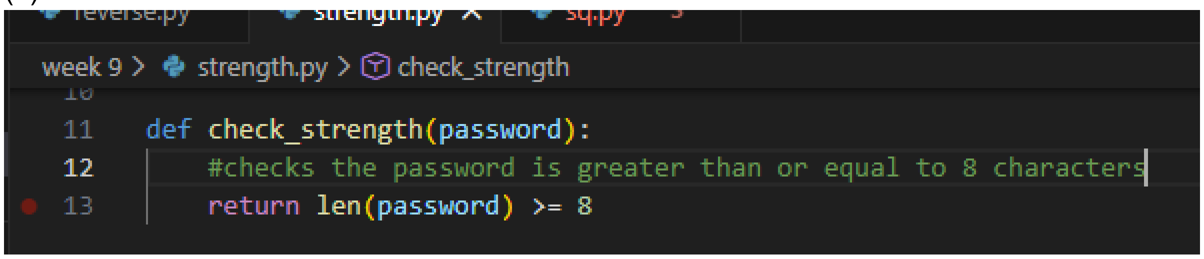
```python
def check_strength(password):
    return len(password) >= 8
```

   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.

(a) Docstring

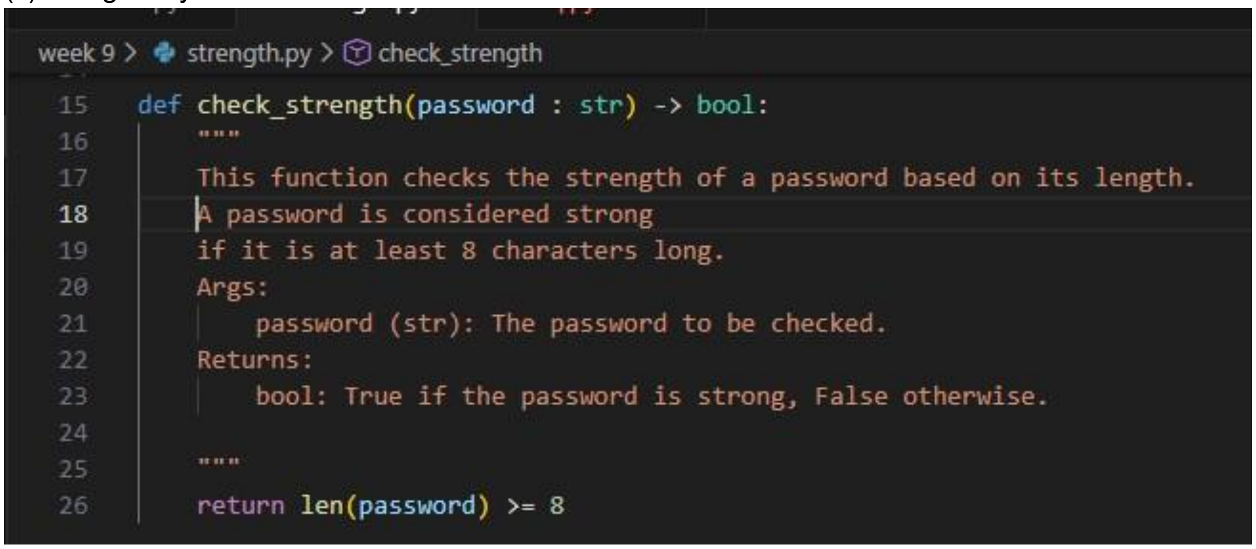

week 9 > strength.py > check_strength
```
def check_strength(password):
    """checks the strength of a password based on length.
    parameters ----------
    password: str
    the password to be checked
    returns ----------
    bool    True if the password is strong, False otherwise
    """
    return len(password) >= 8
```

(b) Inline Comments


reverse.py   strength.py   sq.py

week 9 > strength.py > check_strength
```
def check_strength(password):
    #checks the password is greater than or equal to 8 characters
    return len(password) >= 8
```

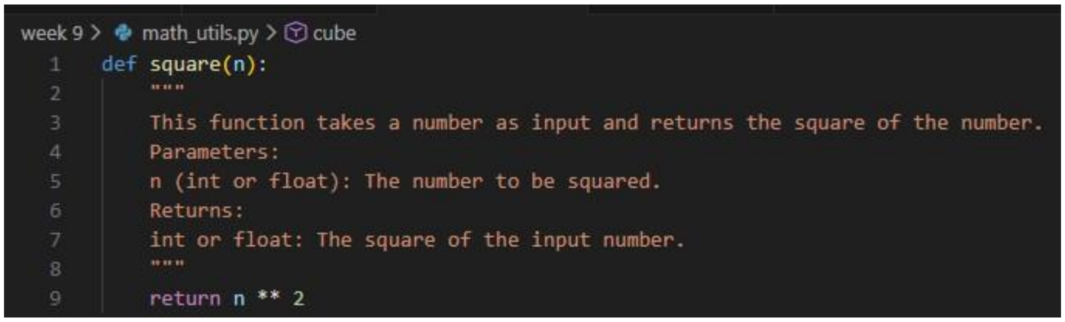
(c) Google-Style Documentation


week 9 > strength.py > check_strength
```
def check_strength(password : str) -> bool:
    """
    This function checks the strength of a password based on its length.
    A password is considered strong
    if it is at least 8 characters long.
    Args:
        password (str): The password to be checked.
    Returns:
        bool: True if the password is strong, False otherwise.

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

Google-style documentation is best for **security-related code**, as it clearly explains validation rules and return values

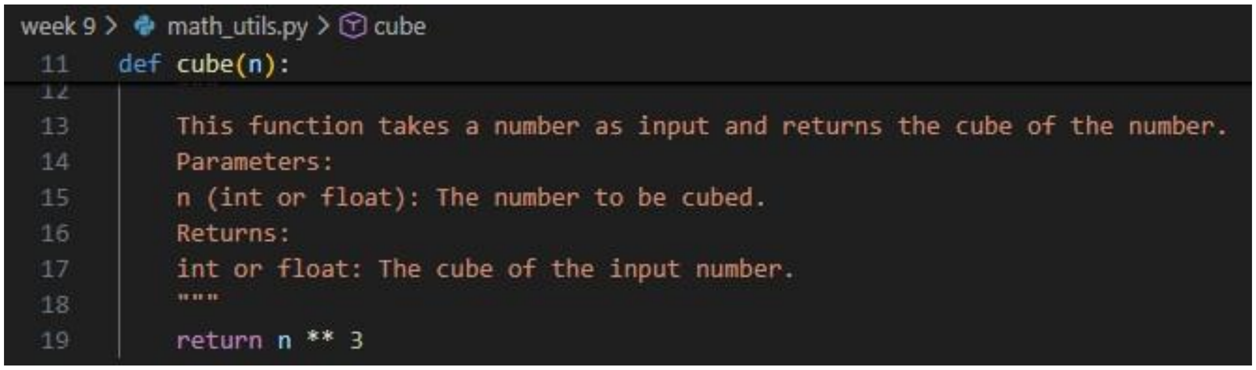
3. Math Utilities Module
   Task:
   1. Create a module math_utils.py with functions:
      a. square(n)
      b. cube(n)
      c. factorial(n)
   2. Generate docstrings automatically using AI tools.
   3. Export documentation as an HTML file.

> square(n)



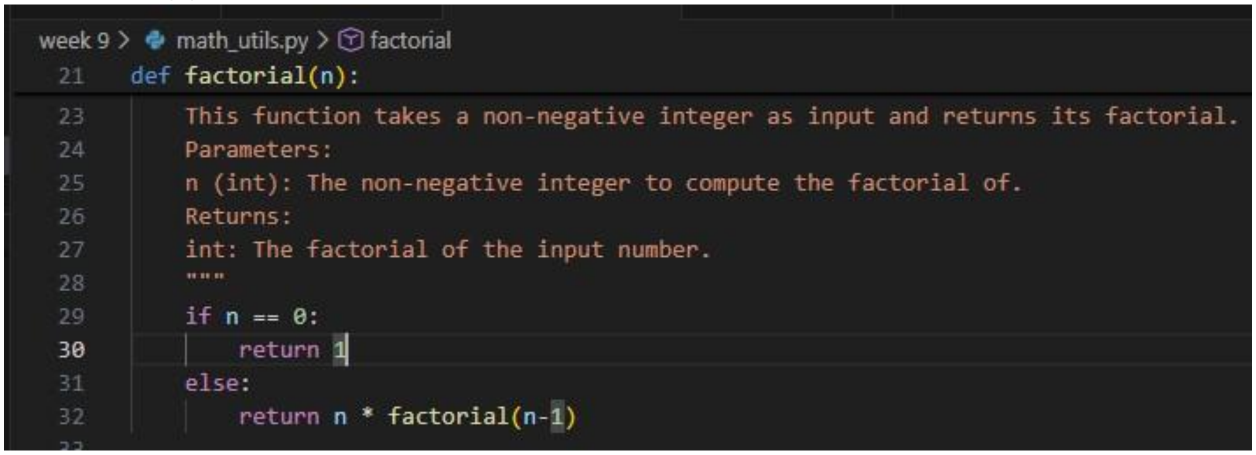
```
week 9 > math_utils.py > cube
def square(n):
    """
    This function takes a number as input and returns the square of the number.
    Parameters:
    n (int or float): The number to be squared.
    Returns:
    int or float: The square of the input number.
    """
    return n ** 2
```

>>> cube(n)



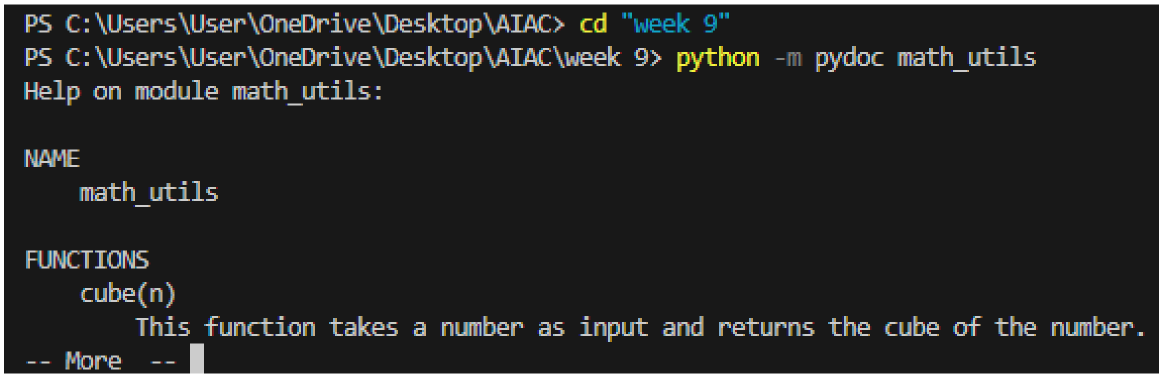
```
week 9 > math_utils.py > cube
def cube(n):

    This function takes a number as input and returns the cube of the number.
    Parameters:
    n (int or float): The number to be cubed.
    Returns:
    int or float: The cube of the input number.
    """
    return n ** 3
```

>>> factorial(n)



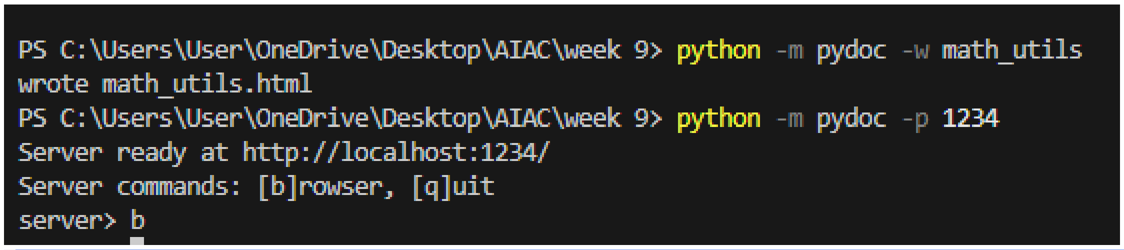
```
week 9 > math_utils.py > factorial
def factorial(n):
    This function takes a non-negative integer as input and returns its factorial.
    Parameters:
    n (int): The non-negative integer to compute the factorial of.
    Returns:
    int: The factorial of the input number.
    """
    if n == 0:
        return 1
    else:
        return n * factorial(n-1)
```

>>

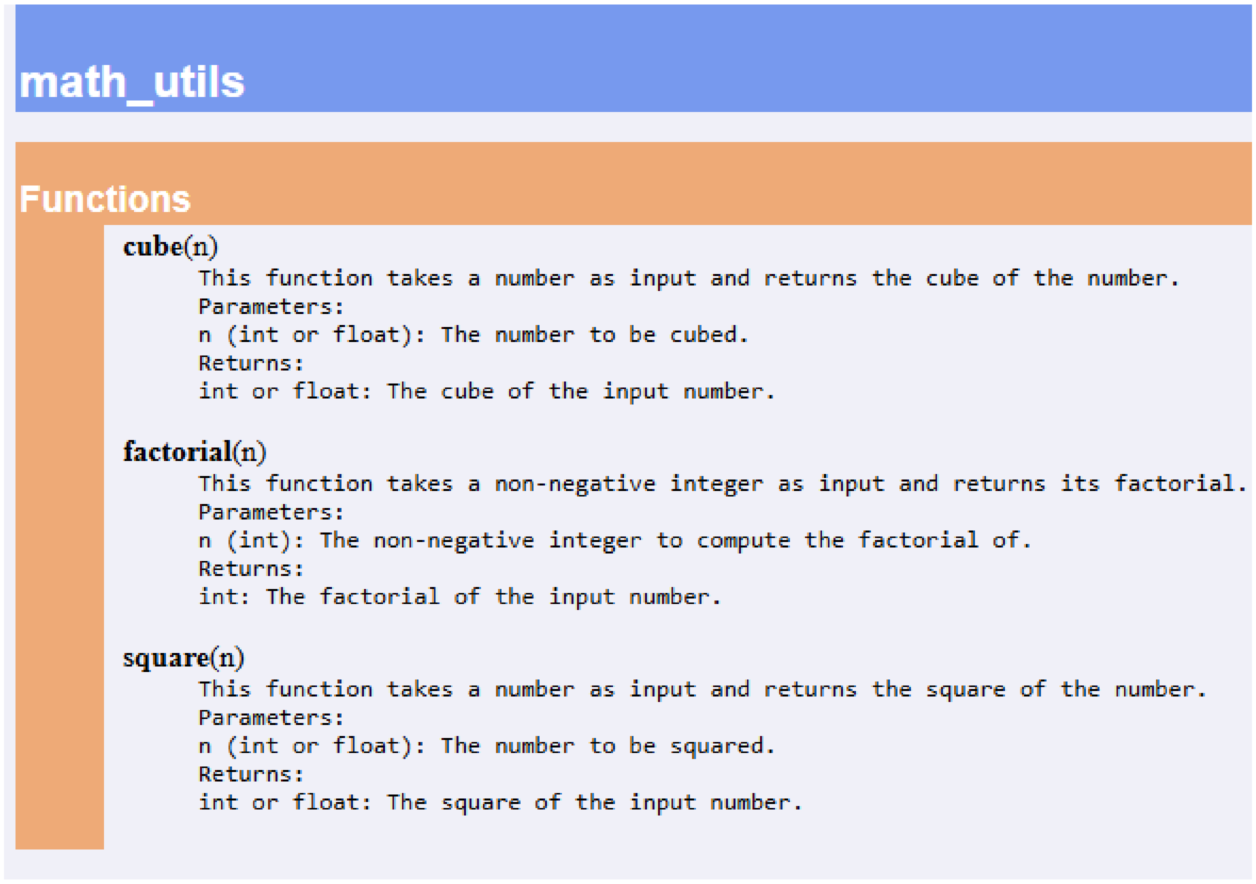


```
PS C:\Users\User\OneDrive\Desktop\AIAC> cd "week 9"
PS C:\Users\User\OneDrive\Desktop\AIAC\week 9> python -m pydoc math_utils
Help on module math_utils:

NAME
    math_utils

FUNCTIONS
    cube(n)
        This function takes a number as input and returns the cube of the number.
-- More  --
```

>>



```
PS C:\Users\User\OneDrive\Desktop\AIAC\week 9> python -m pydoc -w math_utils
wrote math_utils.html
PS C:\Users\User\OneDrive\Desktop\AIAC\week 9> python -m pydoc -p 1234
Server ready at http://localhost:1234/
Server commands: [b]rowser, [q]uit
server> b
```



math_utils

Functions

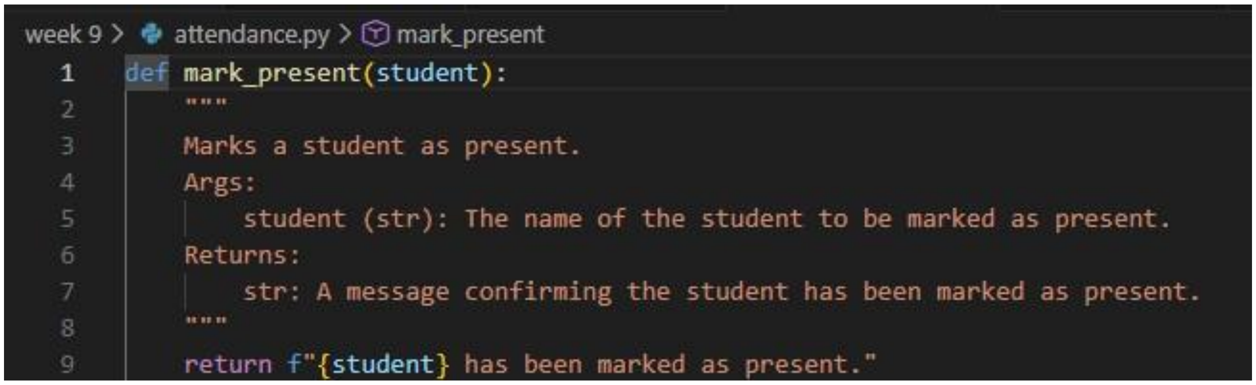
cube(n)
```
    This function takes a number as input and returns the cube of the number.
    Parameters:
    n (int or float): The number to be cubed.
    Returns:
    int or float: The cube of the input number.
```

factorial(n)
```
    This function takes a non-negative integer as input and returns its factorial.
    Parameters:
    n (int): The non-negative integer to compute the factorial of.
    Returns:
    int: The factorial of the input number.
```

square(n)
```
    This function takes a number as input and returns the square of the number.
    Parameters:
    n (int or float): The number to be squared.
    Returns:
    int or float: The square of the input number.
```

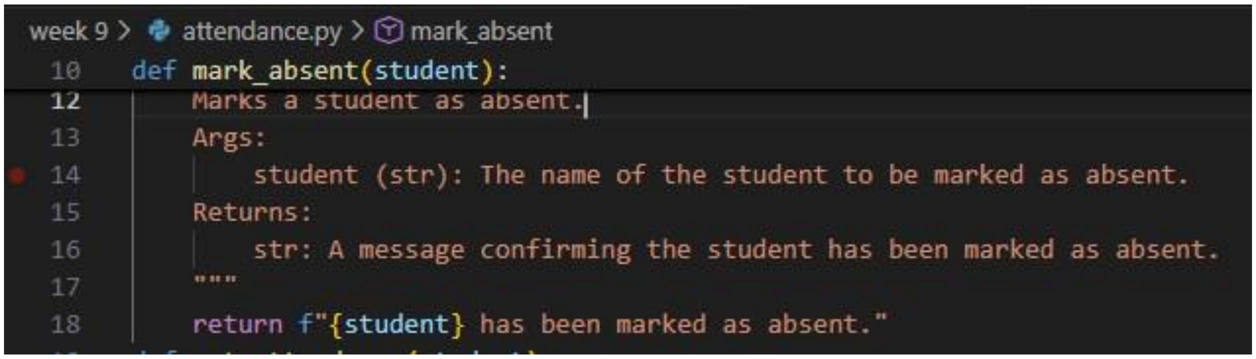
4. Attendance Management Module
   Task:
   1. Create a module attendance.py with functions:
      a. mark_present(student)
      b. mark_absent(student)
      c. get_attendance(student)
   2. Add proper docstrings.
   3. Generate and view documentation in terminal and browse

>>> mark_present(student)



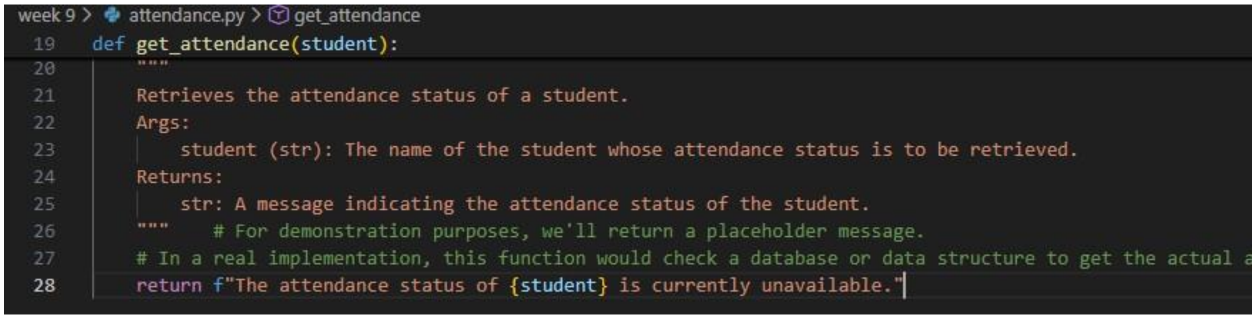
```python
def mark_present(student):
    """
    Marks a student as present.
    Args:
        student (str): The name of the student to be marked as present.
    Returns:
        str: A message confirming the student has been marked as present.
    """
    return f"{student} has been marked as present."
```

>>> mark_absent(student)



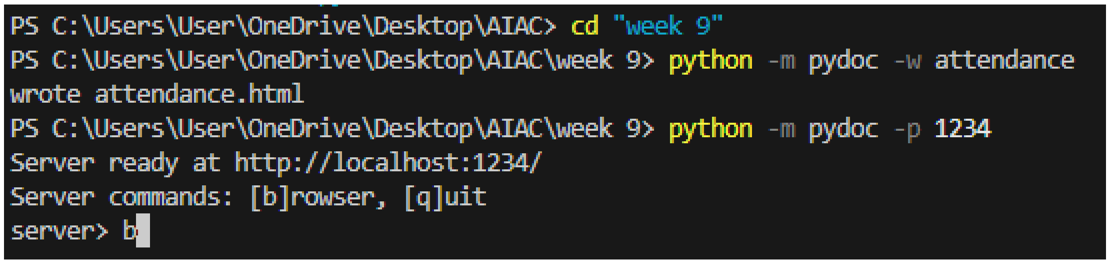
```python
def mark_absent(student):
    Marks a student as absent.
    Args:
        student (str): The name of the student to be marked as absent.
    Returns:
        str: A message confirming the student has been marked as absent.
    """
    return f"{student} has been marked as absent."
```

>>> get_attendance(student)



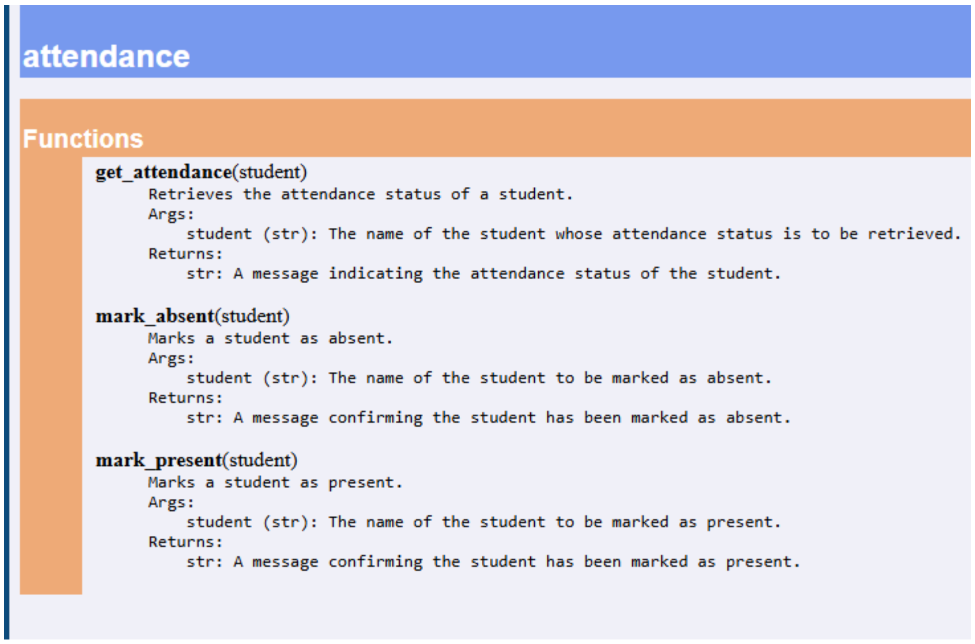
```python
def get_attendance(student):
    """
    Retrieves the attendance status of a student.
    Args:
        student (str): The name of the student whose attendance status is to be retrieved.
    Returns:
        str: A message indicating the attendance status of the student.
    """    # For demonstration purposes, we'll return a placeholder message.
    # In a real implementation, this function would check a database or data structure to get the actual a
    return f"The attendance status of {student} is currently unavailable."
```

>>>



```
PS C:\Users\User\OneDrive\Desktop\AIAC> cd "week 9"
PS C:\Users\User\OneDrive\Desktop\AIAC\week 9> python -m pydoc -w attendance
wrote attendance.html
PS C:\Users\User\OneDrive\Desktop\AIAC\week 9> python -m pydoc -p 1234
Server ready at http://localhost:1234/
Server commands: [b]rowser, [q]uit
server> b
```

>>>



attendance

Functions

```
get_attendance(student)
    Retrieves the attendance status of a student.
    Args:
        student (str): The name of the student whose attendance status is to be retrieved.
    Returns:
        str: A message indicating the attendance status of the student.

mark_absent(student)
    Marks a student as absent.
    Args:
        student (str): The name of the student to be marked as absent.
    Returns:
        str: A message confirming the student has been marked as absent.

mark_present(student)
    Marks a student as present.
    Args:
        student (str): The name of the student to be marked as present.
    Returns:
        str: A message confirming the student has been marked as present.
```

5. File Handling Function

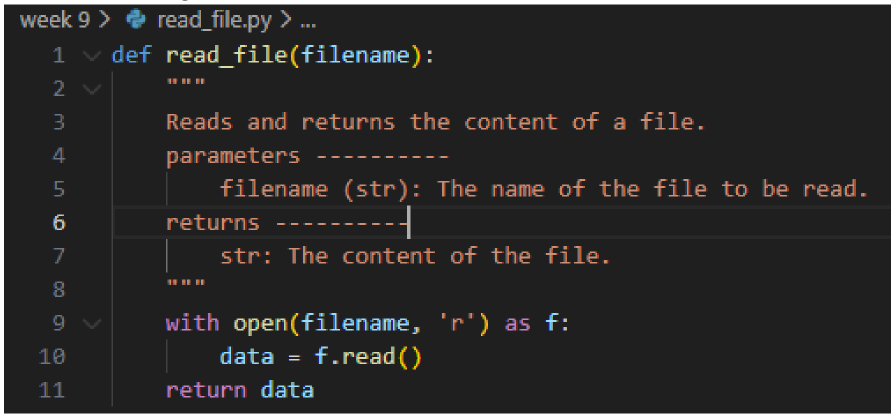
   Consider the function:

```
def read_file(filename):
    with open(filename, 'r') as f:
        return f.read()
```

Task:

- Write documentation using all three formats.
- Identify which style best explains exception handling.
- Justify your recommendation.

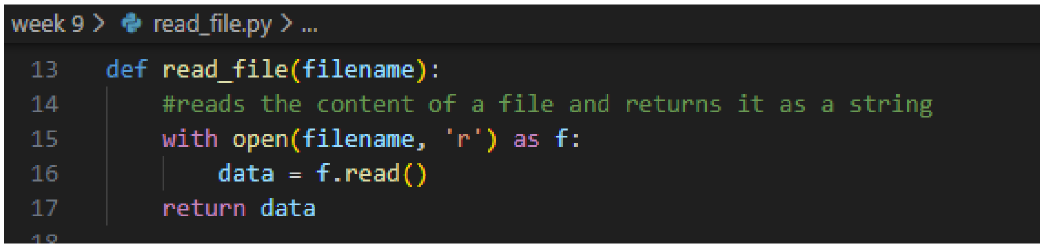
>>> Docstring



week 9 > read_file.py > ...

```
def read_file(filename):
    """
    Reads and returns the content of a file.
    parameters ----------
        filename (str): The name of the file to be read.
    returns ----------
        str: The content of the file.
    """
    with open(filename, 'r') as f:
        data = f.read()
    return data
```

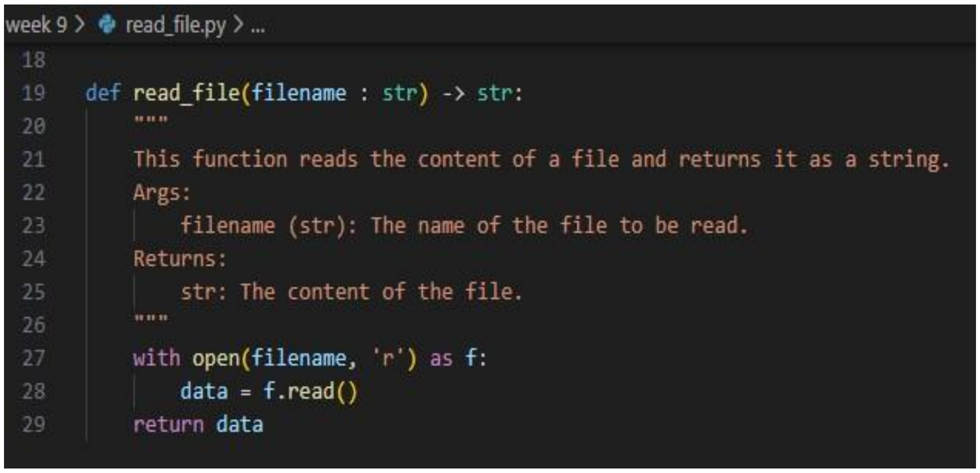
>>> Inline Comments



week 9 > read_file.py > ...

```python
def read_file(filename):
    #reads the content of a file and returns it as a string
    with open(filename, 'r') as f:
        data = f.read()
    return data
```

>>> Google-Style Documentation



week 9 > read_file.py > ...

```python

def read_file(filename : str) -> str:
    """
    This function 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.
    """
    with open(filename, 'r') as f:
        data = f.read()
    return data
```