Lab Assignment-09.5

Name:D.BHARATH KUMAR

Hallticket:2303A51097

Batch-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

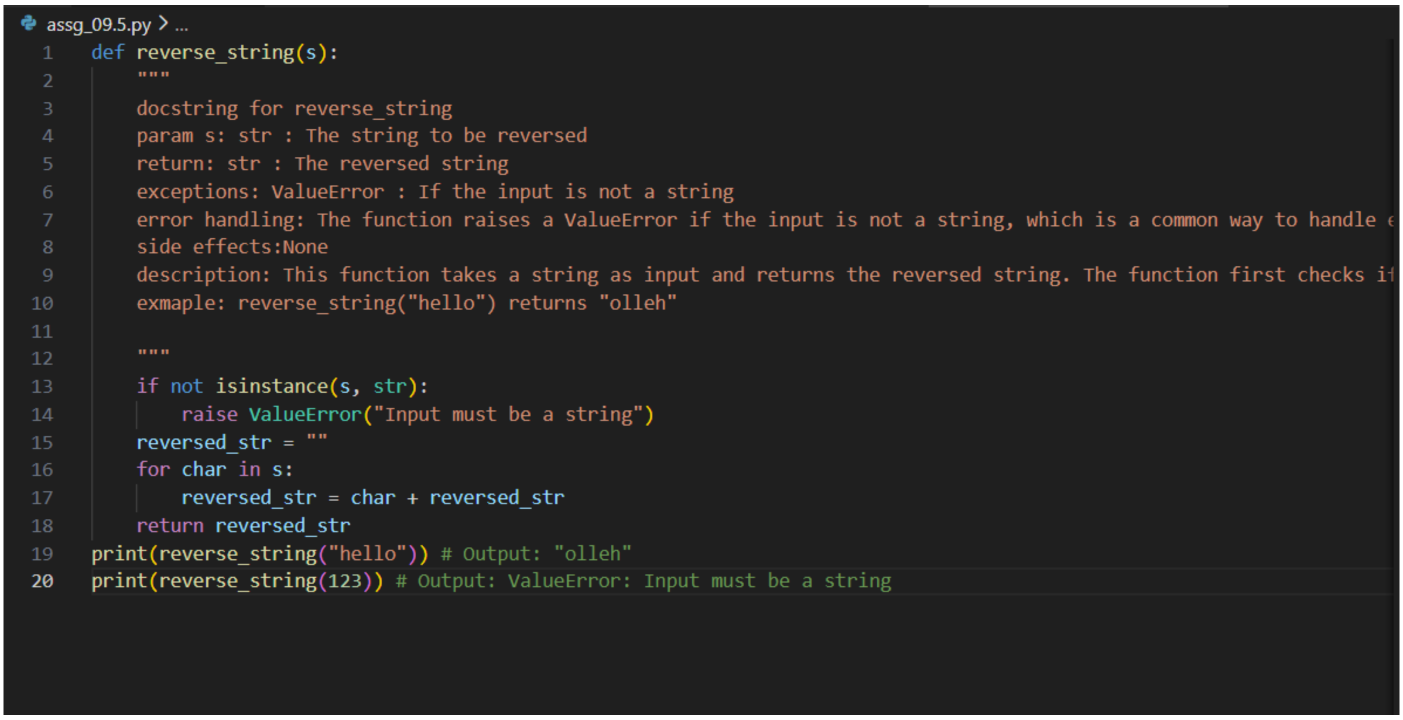
   o (c) Google-style documentation

2. Compare the three documentation styles.

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

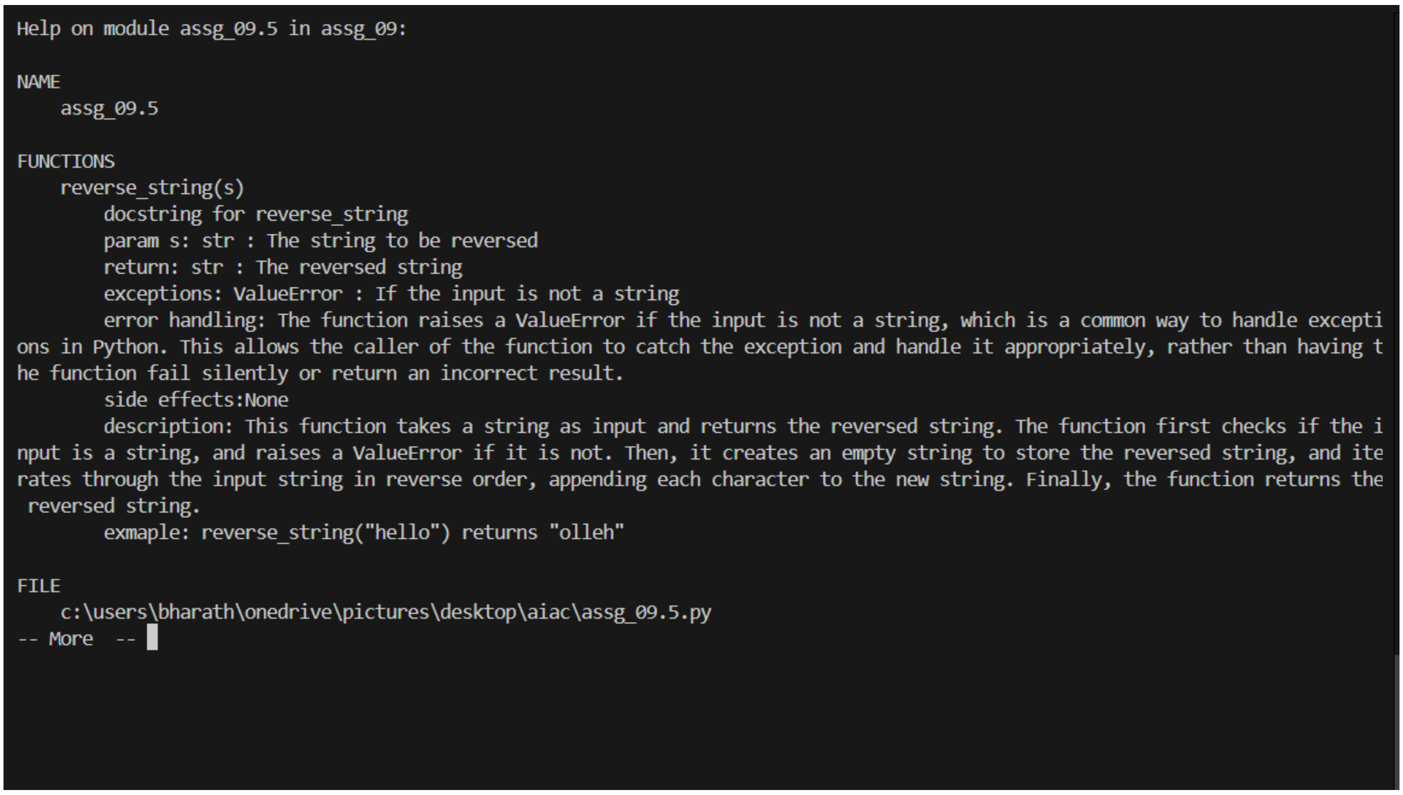
   library.

Docstring:



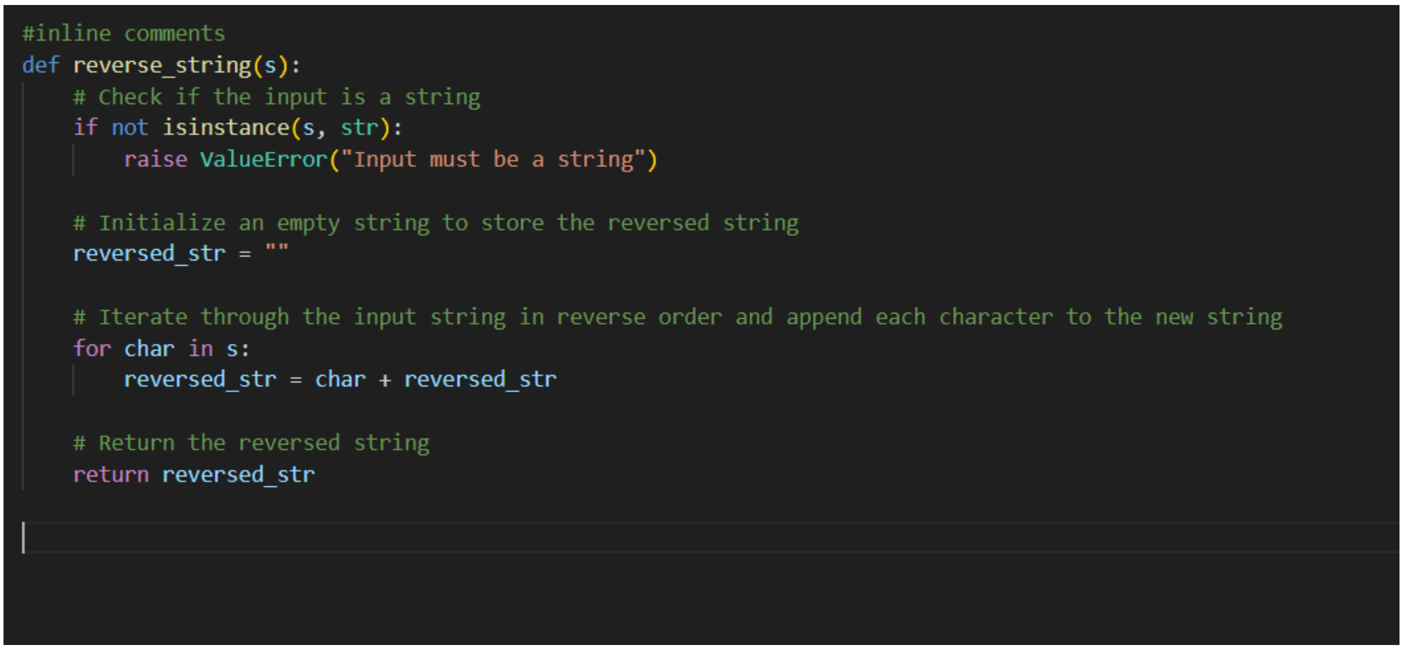
assg_09.5.py > ...

```python
def reverse_string(s):
    """
    docstring for reverse_string
    param s: str : The string to be reversed
    return: str : The reversed string
    exceptions: ValueError : If the input is not a string
    error handling: The function raises a ValueError if the input is not a string, which is a common way to handle e
    side effects:None
    description: This function takes a string as input and returns the reversed string. The function first checks if
    exmaple: reverse_string("hello") returns "olleh"

    """
    if not isinstance(s, str):
        raise ValueError("Input must be a string")
    reversed_str = ""
    for char in s:
        reversed_str = char + reversed_str
    return reversed_str
print(reverse_string("hello")) # Output: "olleh"
print(reverse_string(123)) # Output: ValueError: Input must be a string
```



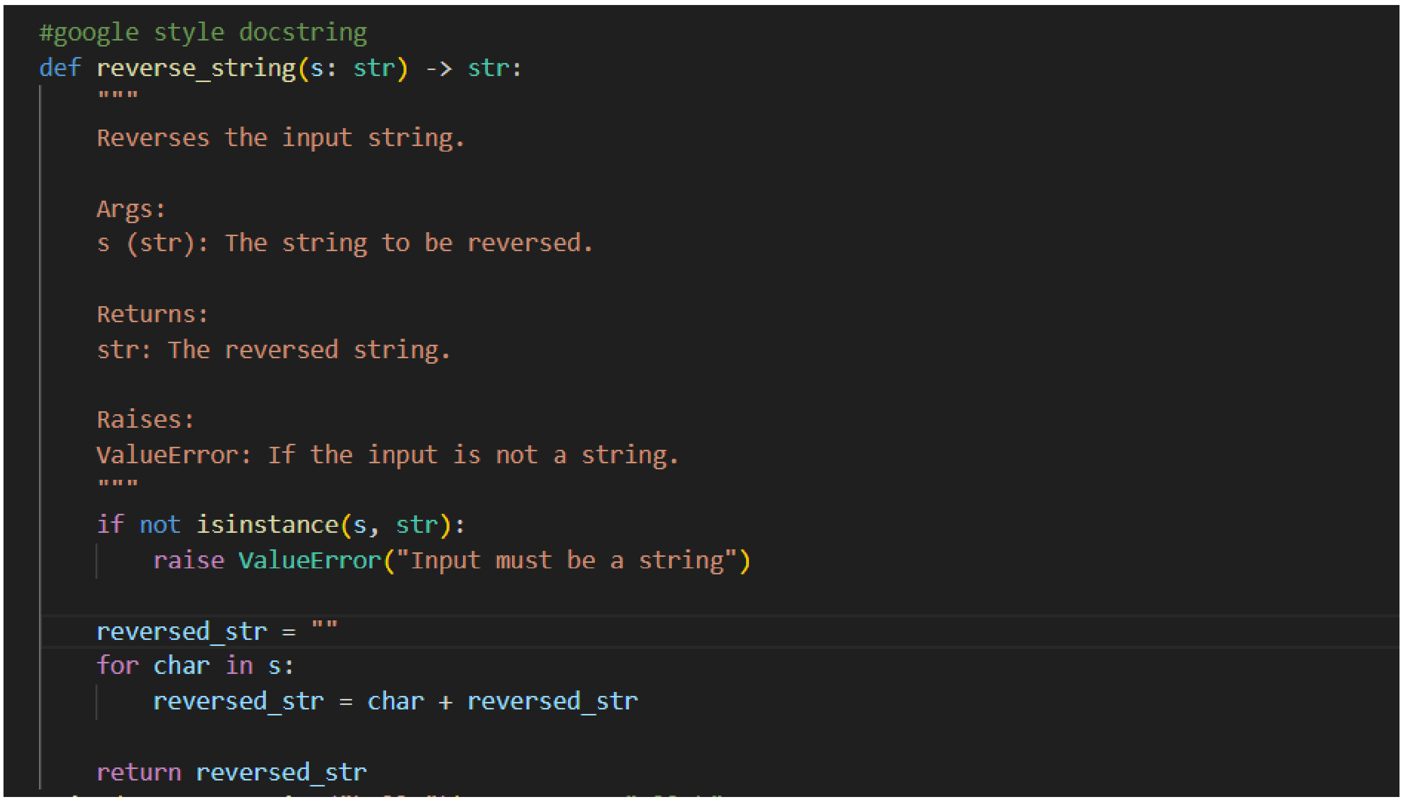
```
Help on module assg_09.5 in assg_09:

NAME
    assg_09.5

FUNCTIONS
    reverse_string(s)
        docstring for reverse_string
        param s: str : The string to be reversed
        return: str : The reversed string
        exceptions: ValueError : If the input is not a string
        error handling: The function raises a ValueError if the input is not a string, which is a common way to handle excepti
ons in Python. This allows the caller of the function to catch the exception and handle it appropriately, rather than having t
he function fail silently or return an incorrect result.
        side effects:None
        description: This function takes a string as input and returns the reversed string. The function first checks if the i
nput is a string, and raises a ValueError if it is not. Then, it creates an empty string to store the reversed string, and ite
rates through the input string in reverse order, appending each character to the new string. Finally, the function returns the
 reversed string.
        exmaple: reverse_string("hello") returns "olleh"

FILE
    c:\users\bharath\onedrive\pictures\desktop\aiac\assg_09.5.py
-- More  --
```

Inline comments:



```python
#inline comments
def reverse_string(s):
    # Check if the input is a string
    if not isinstance(s, str):
        raise ValueError("Input must be a string")

    # Initialize an empty string to store the reversed string
    reversed_str = ""

    # Iterate through the input string in reverse order and append each character to the new string
    for char in s:
        reversed_str = char + reversed_str

    # Return the reversed string
    return reversed_str
```

Google style :



```python
#google style docstring
def reverse_string(s: str) -> str:
    """
    Reverses the input string.

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

    Returns:
    str: The reversed string.

    Raises:
    ValueError: If the input is not a string.
    """
    if not isinstance(s, str):
        raise ValueError("Input must be a string")

    reversed_str = ""
    for char in s:
        reversed_str = char + reversed_str

    return reversed_str
```

Problem 2: Password Strength Checker

Consider the function:

def check_strength(password):

return len(password) >= 8

Task:

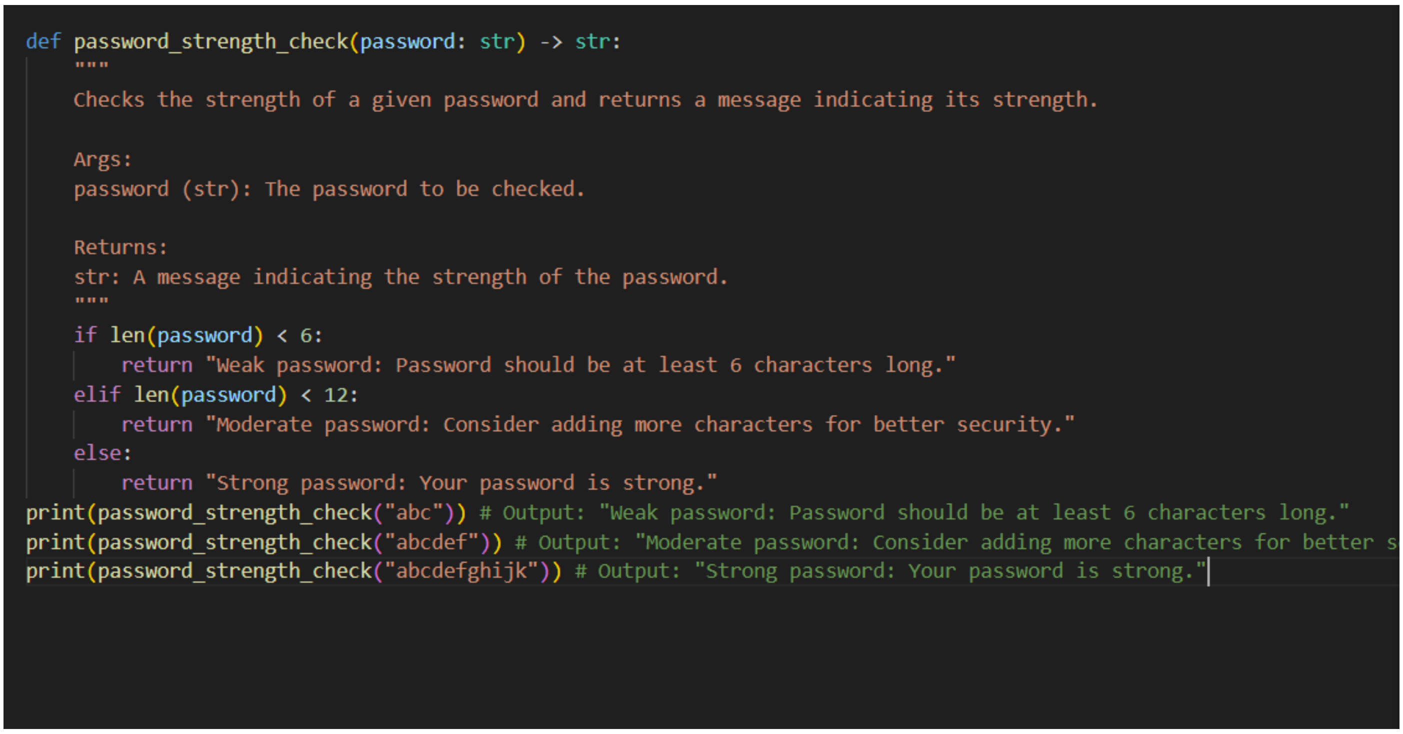
1. Document the function using docstring, inline comments, and

Google style.

2. Compare documentation styles for security-related code.

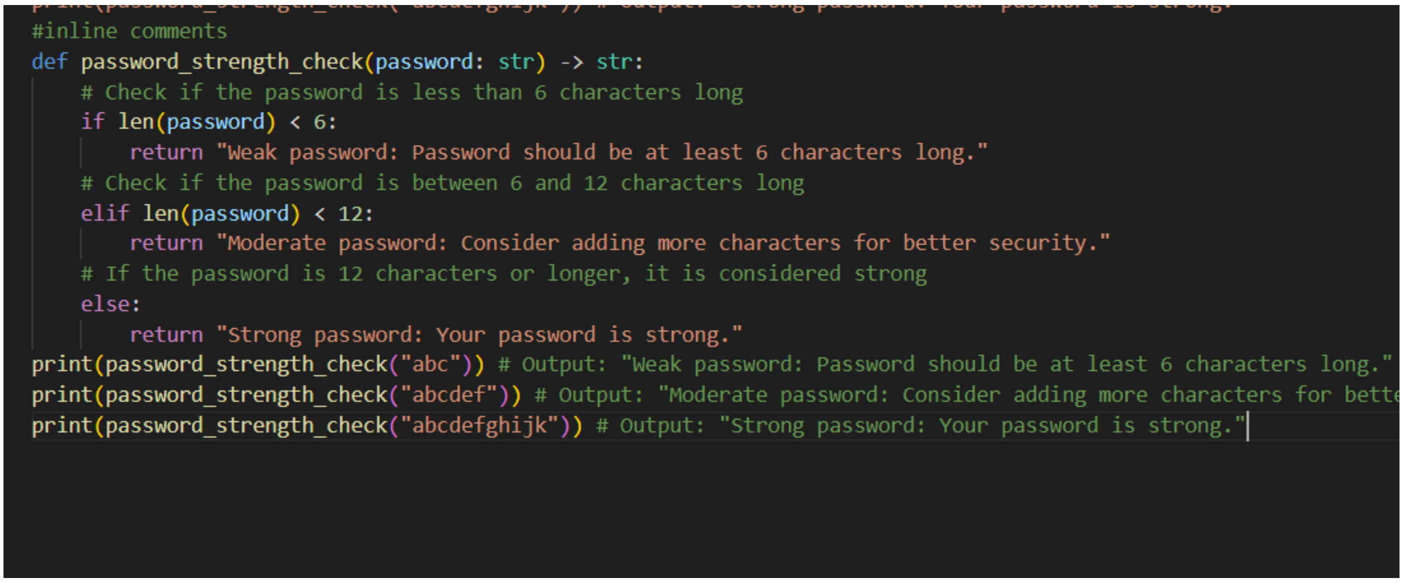
3. Recommend the most appropriate style.

Doc_string and google_style:



```
def password_strength_check(password: str) -> str:
    """
    Checks the strength of a given password and returns a message indicating its strength.

    Args:
    password (str): The password to be checked.

    Returns:
    str: A message indicating the strength of the password.
    """
    if len(password) < 6:
        return "Weak password: Password should be at least 6 characters long."
    elif len(password) < 12:
        return "Moderate password: Consider adding more characters for better security."
    else:
        return "Strong password: Your password is strong."
print(password_strength_check("abc")) # Output: "Weak password: Password should be at least 6 characters long."
print(password_strength_check("abcdef")) # Output: "Moderate password: Consider adding more characters for better s
print(password_strength_check("abcdefghijk")) # Output: "Strong password: Your password is strong."
```

Incline comments:



```
#inline comments
def password_strength_check(password: str) -> str:
    # Check if the password is less than 6 characters long
    if len(password) < 6:
        return "Weak password: Password should be at least 6 characters long."
    # Check if the password is between 6 and 12 characters long
    elif len(password) < 12:
        return "Moderate password: Consider adding more characters for better security."
    # If the password is 12 characters or longer, it is considered strong
    else:
        return "Strong password: Your password is strong."
print(password_strength_check("abc")) # Output: "Weak password: Password should be at least 6 characters long."
print(password_strength_check("abcdef")) # Output: "Moderate password: Consider adding more characters for bette
print(password_strength_check("abcdefghijk")) # Output: "Strong password: Your password is strong."
```

Problem 3: Math Utilities Module

Task:

1. Create a module math_utils.py with functions:

    o square(n)

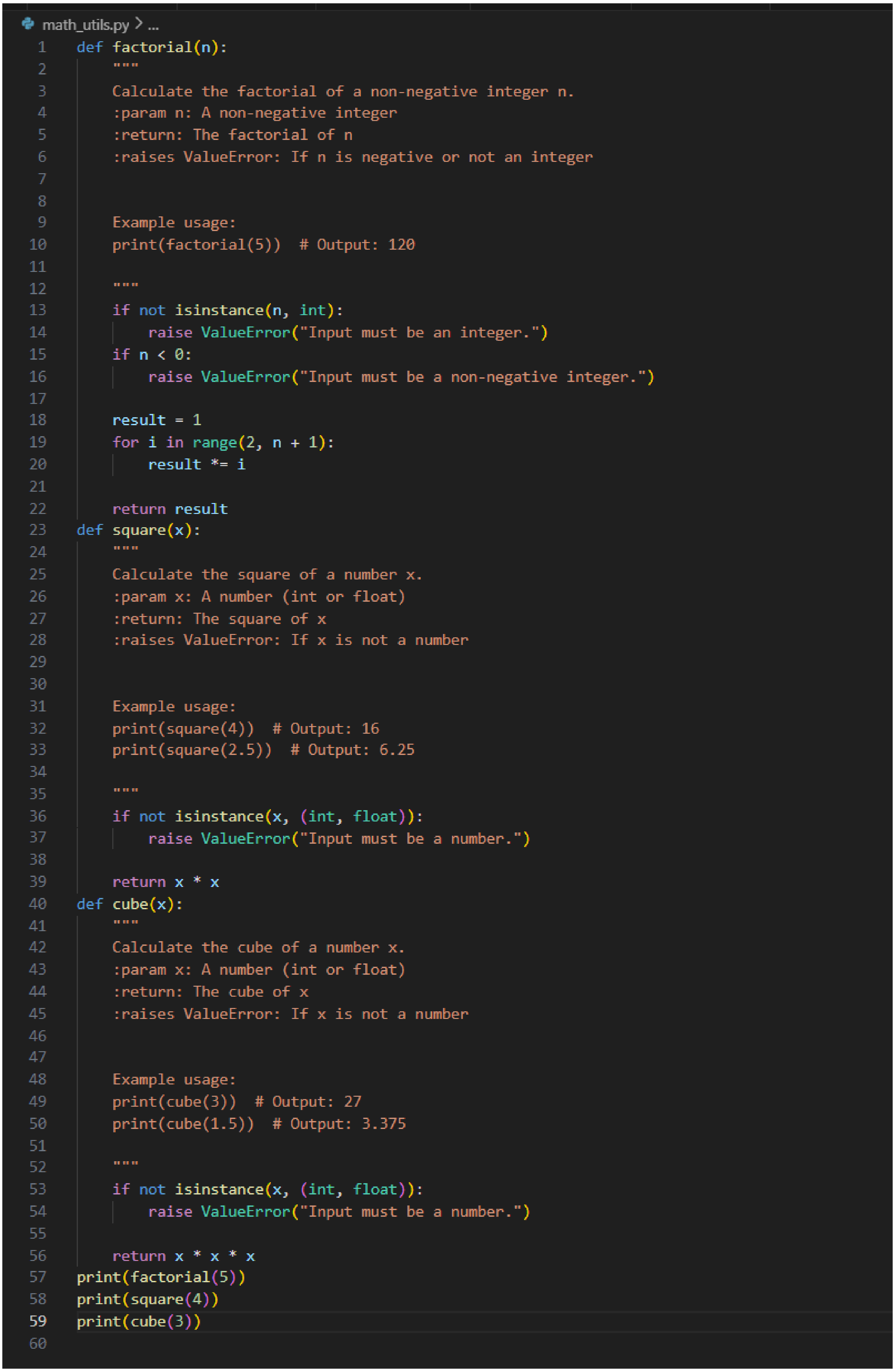
    o cube(n)

    o factorial(n)

2. Generate docstrings automatically using AI tools.

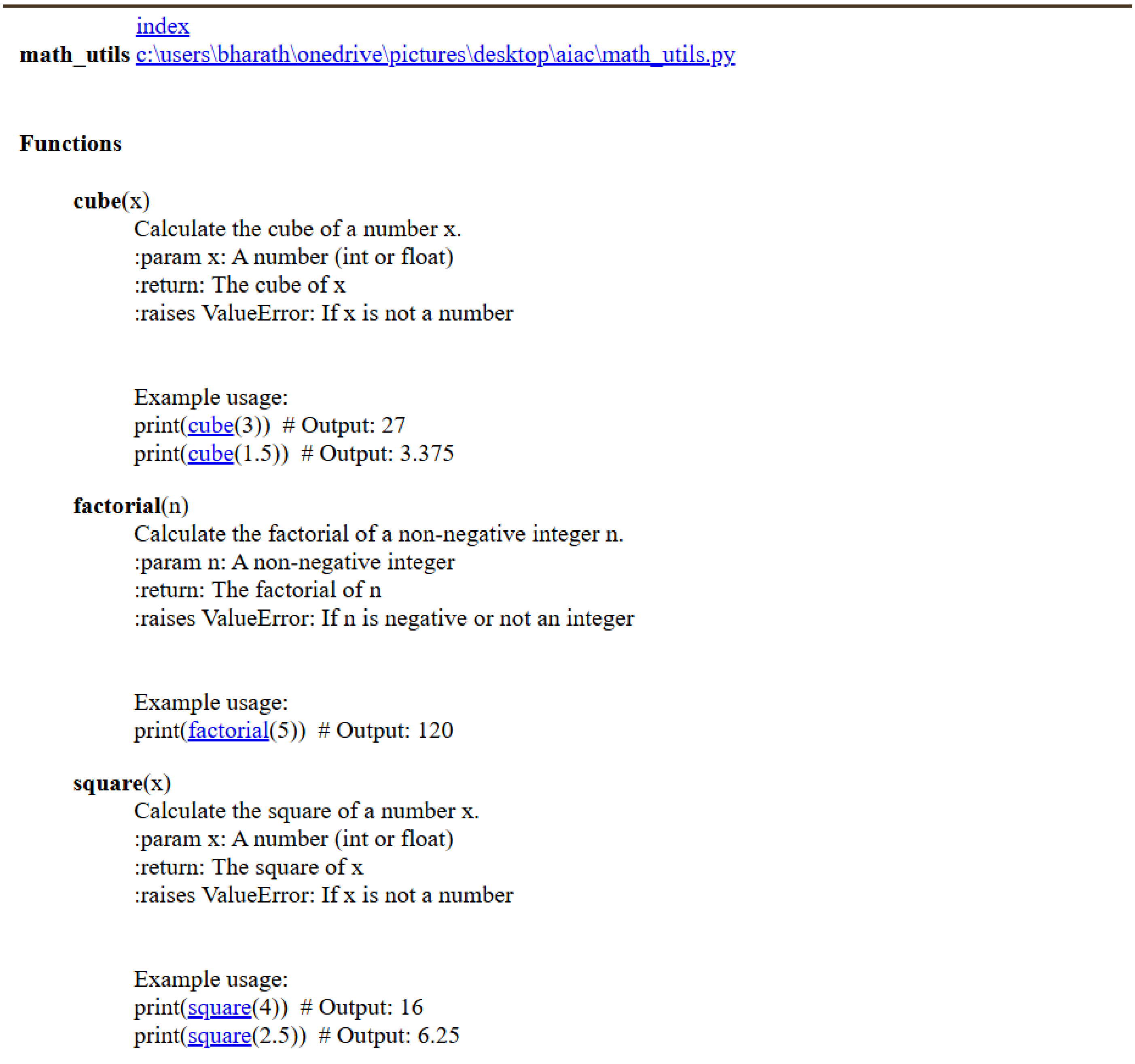
3. Export documentation as an HTML file.

Code:

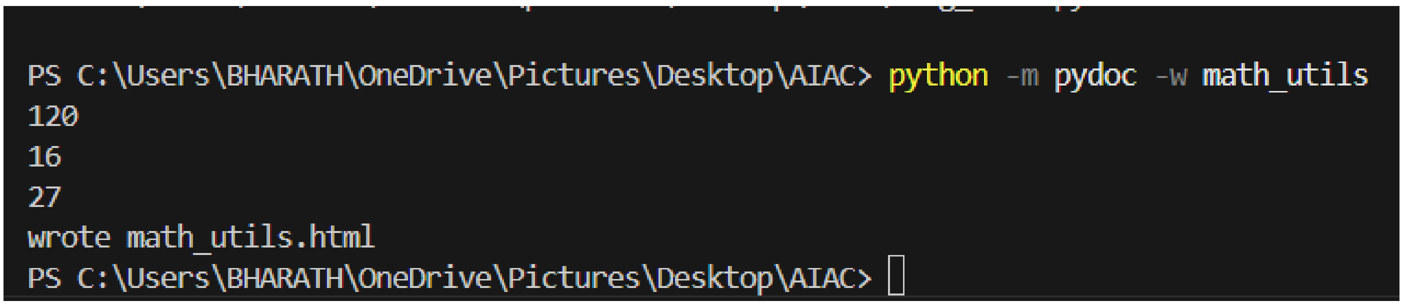


```python
def factorial(n):
    """
    Calculate the factorial of a non-negative integer n.
    :param n: A non-negative integer
    :return: The factorial of n
    :raises ValueError: If n is negative or not an integer


    Example usage:
    print(factorial(5))  # Output: 120

    """
    if not isinstance(n, int):
        raise ValueError("Input must be an integer.")
    if n < 0:
        raise ValueError("Input must be a non-negative integer.")

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

    return result
def square(x):
    """
    Calculate the square of a number x.
    :param x: A number (int or float)
    :return: The square of x
    :raises ValueError: If x is not a number


    Example usage:
    print(square(4))  # Output: 16
    print(square(2.5))  # Output: 6.25

    """
    if not isinstance(x, (int, float)):
        raise ValueError("Input must be a number.")

    return x * x
def cube(x):
    """
    Calculate the cube of a number x.
    :param x: A number (int or float)
    :return: The cube of x
    :raises ValueError: If x is not a number


    Example usage:
    print(cube(3))  # Output: 27
    print(cube(1.5))  # Output: 3.375

    """
    if not isinstance(x, (int, float)):
        raise ValueError("Input must be a number.")

    return x * x * x
print(factorial(5))
print(square(4))
print(cube(3))

```

Output:



```
             index
math_utils c:\users\bharath\onedrive\pictures\desktop\aiac\math_utils.py


Functions

    cube(x)
        Calculate the cube of a number x.
        :param x: A number (int or float)
        :return: The cube of x
        :raises ValueError: If x is not a number


        Example usage:
        print(cube(3))  # Output: 27
        print(cube(1.5))  # Output: 3.375

    factorial(n)
        Calculate the factorial of a non-negative integer n.
        :param n: A non-negative integer
        :return: The factorial of n
        :raises ValueError: If n is negative or not an integer


        Example usage:
        print(factorial(5))  # Output: 120

    square(x)
        Calculate the square of a number x.
        :param x: A number (int or float)
        :return: The square of x
        :raises ValueError: If x is not a number


        Example usage:
        print(square(4))  # Output: 16
        print(square(2.5))  # Output: 6.25
```



```
PS C:\Users\BHARATH\OneDrive\Pictures\Desktop\AIAC> python -m pydoc -w math_utils
120
16
27
wrote math_utils.html
PS C:\Users\BHARATH\OneDrive\Pictures\Desktop\AIAC>
```

Problem 4: Attendance Management Module

Task:

1. Create a module attendance.py with functions:

    o mark_present(student)

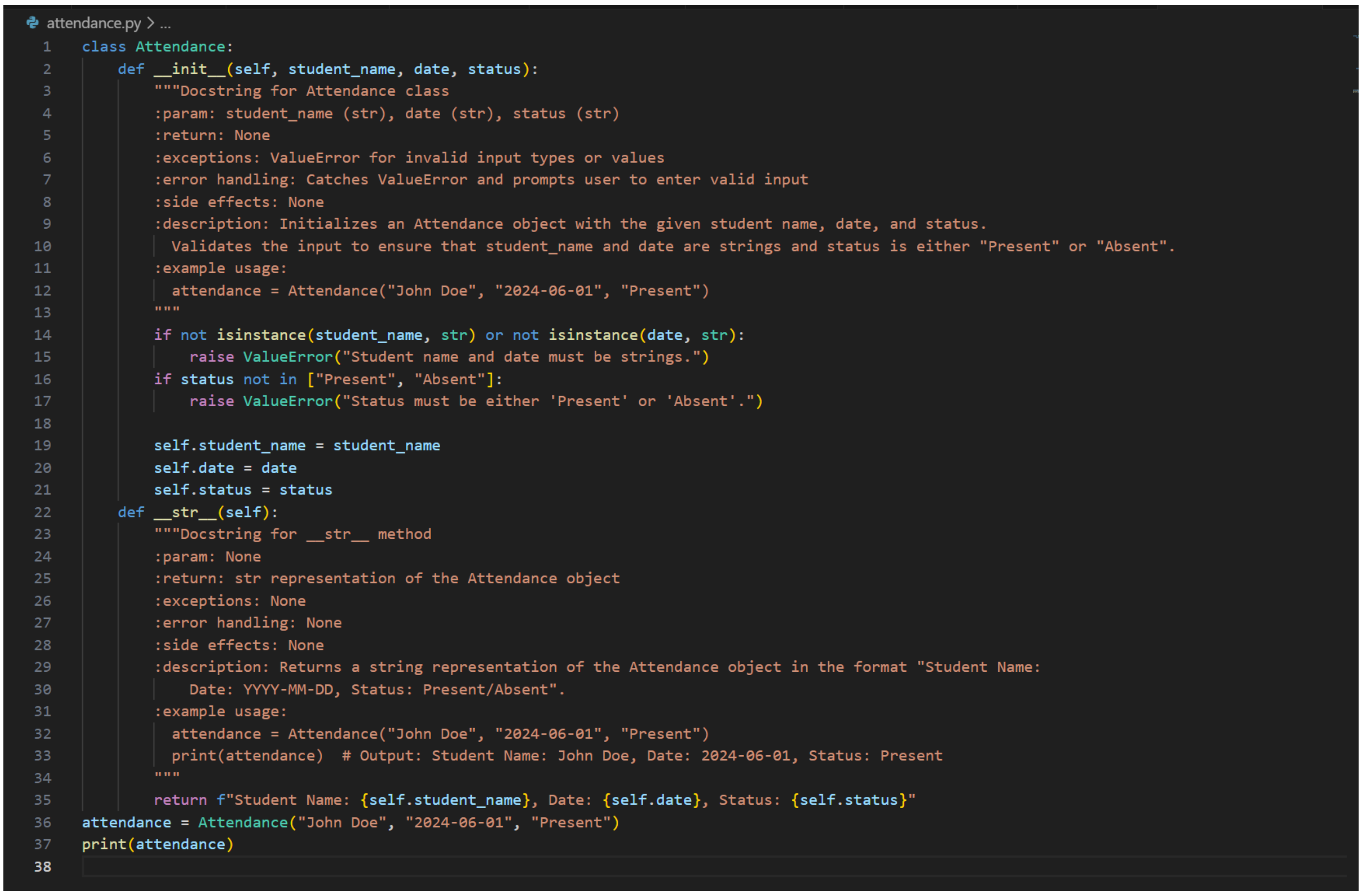
    o mark_absent(student)

o get_attendance(student)

2. Add proper docstrings.

3. Generate and view documentation in terminal and browse

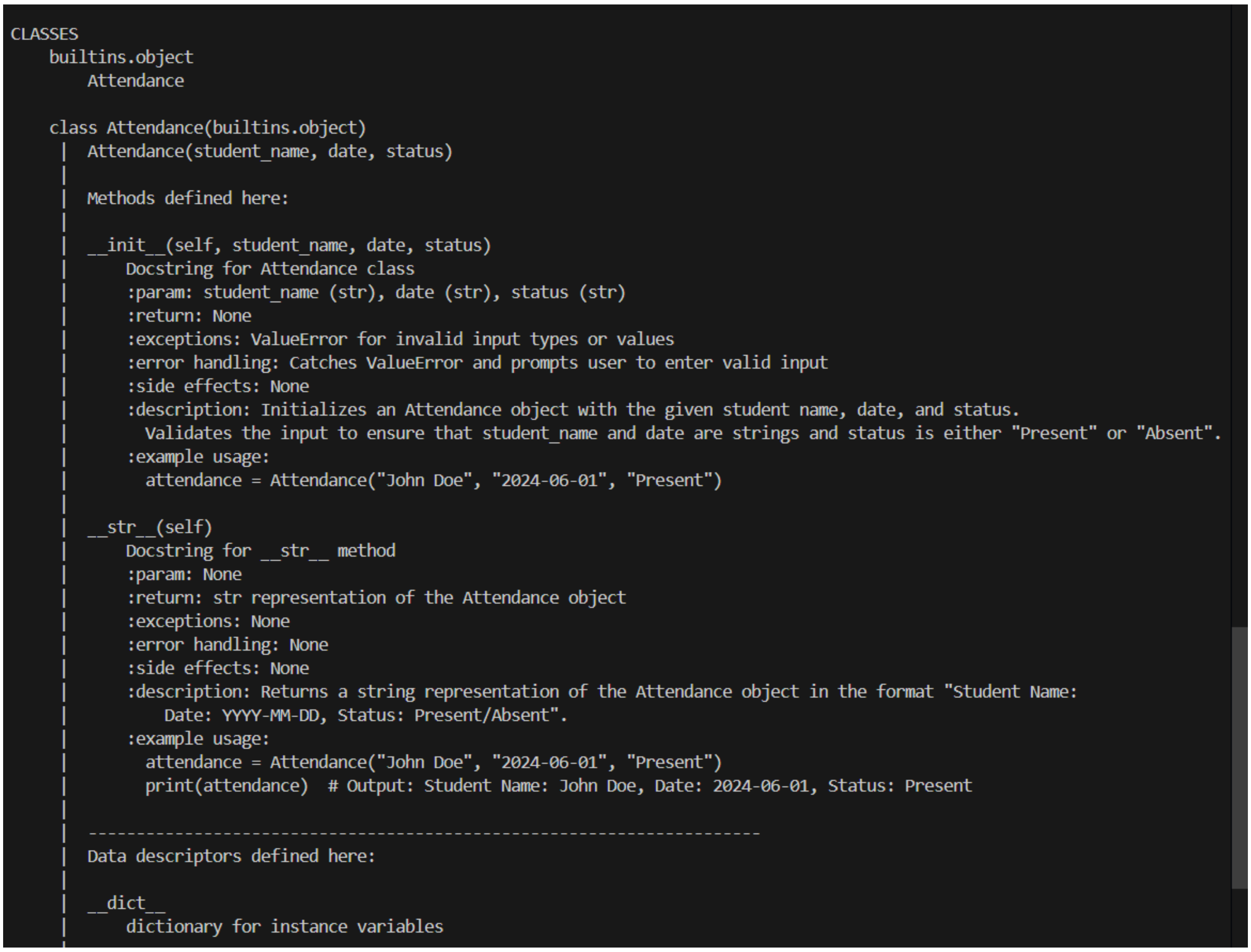
Code:



attendance.py > ...

```python
class Attendance:
    def __init__(self, student_name, date, status):
        """Docstring for Attendance class
        :param: student_name (str), date (str), status (str)
        :return: None
        :exceptions: ValueError for invalid input types or values
        :error handling: Catches ValueError and prompts user to enter valid input
        :side effects: None
        :description: Initializes an Attendance object with the given student name, date, and status.
          Validates the input to ensure that student_name and date are strings and status is either "Present" or "Absent".
        :example usage:
          attendance = Attendance("John Doe", "2024-06-01", "Present")
        """
        if not isinstance(student_name, str) or not isinstance(date, str):
            raise ValueError("Student name and date must be strings.")
        if status not in ["Present", "Absent"]:
            raise ValueError("Status must be either 'Present' or 'Absent'.")

        self.student_name = student_name
        self.date = date
        self.status = status
    def __str__(self):
        """Docstring for __str__ method
        :param: None
        :return: str representation of the Attendance object
        :exceptions: None
        :error handling: None
        :side effects: None
        :description: Returns a string representation of the Attendance object in the format "Student Name:
            Date: YYYY-MM-DD, Status: Present/Absent".
        :example usage:
          attendance = Attendance("John Doe", "2024-06-01", "Present")
          print(attendance)  # Output: Student Name: John Doe, Date: 2024-06-01, Status: Present
        """
        return f"Student Name: {self.student_name}, Date: {self.date}, Status: {self.status}"
attendance = Attendance("John Doe", "2024-06-01", "Present")
print(attendance)

```

Output:



```
CLASSES
    builtins.object
        Attendance

    class Attendance(builtins.object)
     |  Attendance(student_name, date, status)
     |
     |  Methods defined here:
     |
     |  __init__(self, student_name, date, status)
     |      Docstring for Attendance class
     |      :param: student_name (str), date (str), status (str)
     |      :return: None
     |      :exceptions: ValueError for invalid input types or values
     |      :error handling: Catches ValueError and prompts user to enter valid input
     |      :side effects: None
     |      :description: Initializes an Attendance object with the given student name, date, and status.
     |        Validates the input to ensure that student_name and date are strings and status is either "Present" or "Absent".
     |      :example usage:
     |        attendance = Attendance("John Doe", "2024-06-01", "Present")
     |
     |  __str__(self)
     |      Docstring for __str__ method
     |      :param: None
     |      :return: str representation of the Attendance object
     |      :exceptions: None
     |      :error handling: None
     |      :side effects: None
     |      :description: Returns a string representation of the Attendance object in the format "Student Name:
     |          Date: YYYY-MM-DD, Status: Present/Absent".
     |      :example usage:
     |        attendance = Attendance("John Doe", "2024-06-01", "Present")
     |        print(attendance)  # Output: Student Name: John Doe, Date: 2024-06-01, Status: Present
     |
     |  ----------------------------------------------------------------------
     |  Data descriptors defined here:
     |
     |  __dict__
     |      dictionary for instance variables
```

Problem 5: File Handling Function

Consider the function:

def read_file(filename):

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

return f.read()

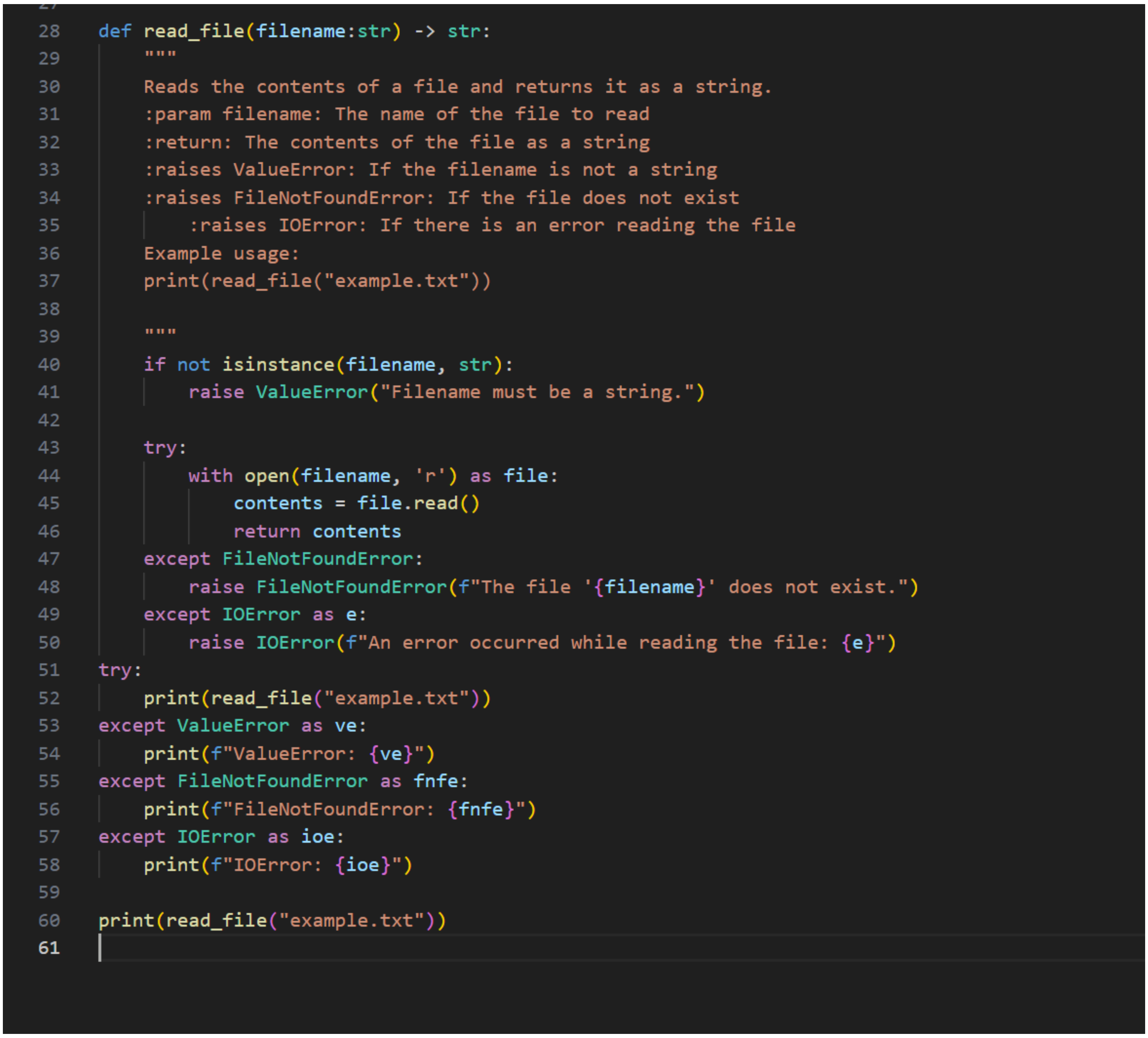
Task:

1. Write documentation using all three formats.

2. Identify which style best explains exception handling.

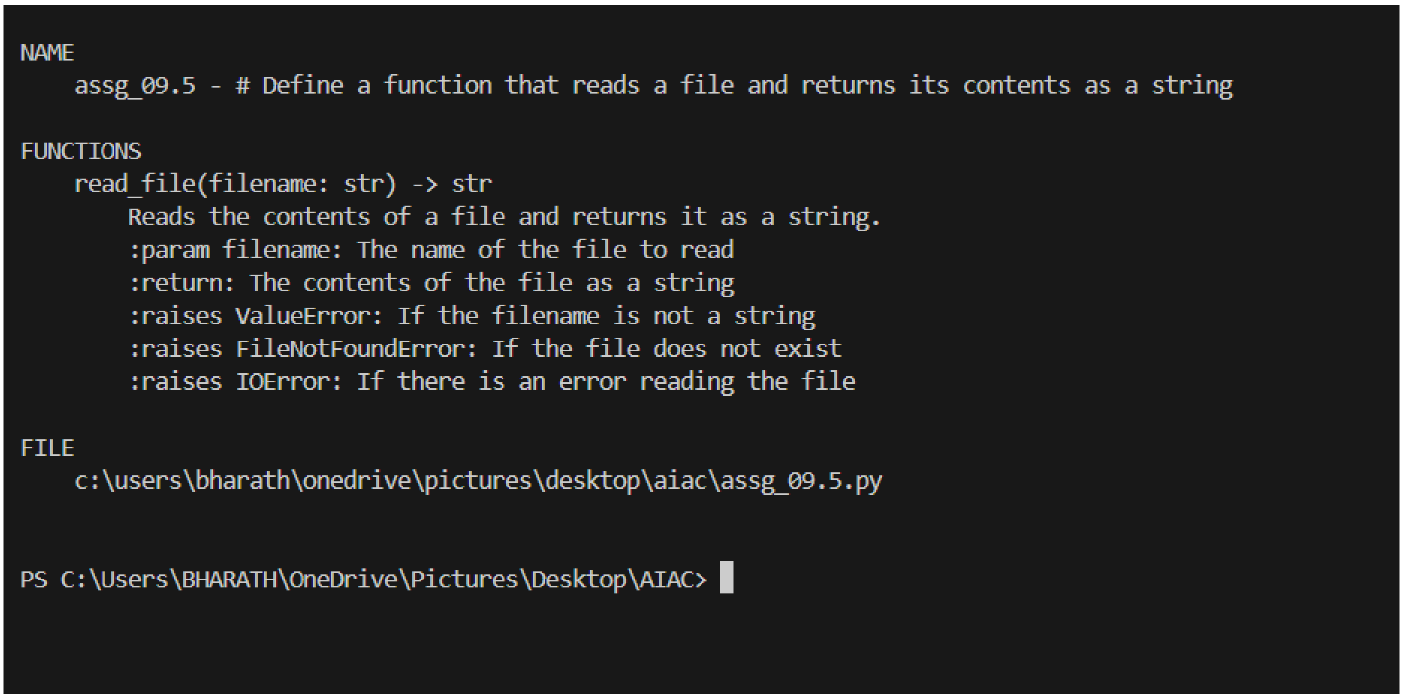
3. Justify your recommendation.

Code:



```python
def read_file(filename:str) -> str:
    """
    Reads the contents of a file and returns it as a string.
    :param filename: The name of the file to read
    :return: The contents of the file as a string
    :raises ValueError: If the filename is not a string
    :raises FileNotFoundError: If the file does not exist
        :raises IOError: If there is an error reading the file
    Example usage:
    print(read_file("example.txt"))

    """
    if not isinstance(filename, str):
        raise ValueError("Filename must be a string.")

    try:
        with open(filename, 'r') as file:
            contents = file.read()
            return contents
    except FileNotFoundError:
        raise FileNotFoundError(f"The file '{filename}' does not exist.")
    except IOError as e:
        raise IOError(f"An error occurred while reading the file: {e}")
try:
    print(read_file("example.txt"))
except ValueError as ve:
    print(f"ValueError: {ve}")
except FileNotFoundError as fnfe:
    print(f"FileNotFoundError: {fnfe}")
except IOError as ioe:
    print(f"IOError: {ioe}")

print(read_file("example.txt"))

```

Output:



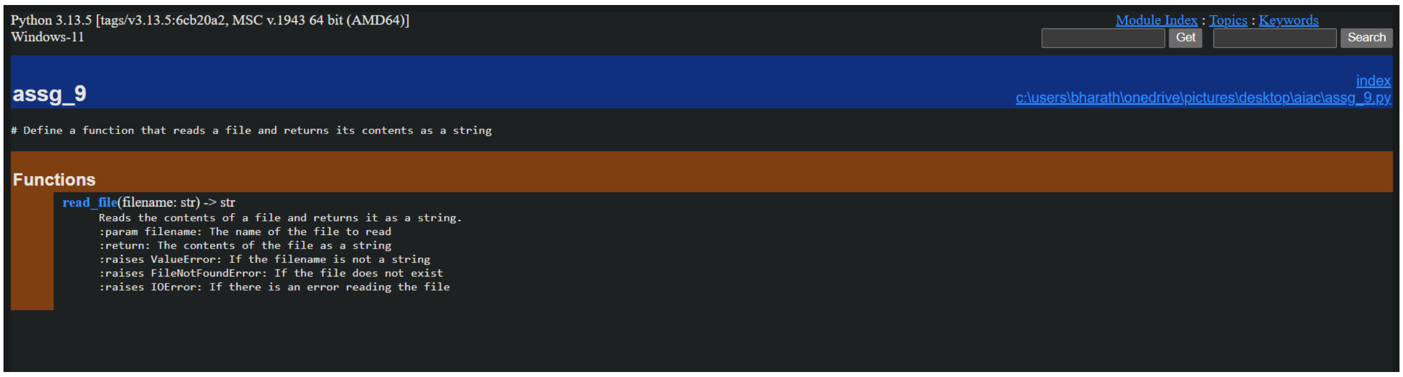
```
NAME
    assg_09.5 - # Define a function that reads a file and returns its contents as a string

FUNCTIONS
    read_file(filename: str) -> str
        Reads the contents of a file and returns it as a string.
        :param filename: The name of the file to read
        :return: The contents of the file as a string
        :raises ValueError: If the filename is not a string
        :raises FileNotFoundError: If the file does not exist
        :raises IOError: If there is an error reading the file

FILE
    c:\users\bharath\onedrive\pictures\desktop\aiac\assg_09.5.py


PS C:\Users\BHARATH\OneDrive\Pictures\Desktop\AIAC>
```

index
assg_09.5 c:\users\bharath\onedrive\pictures\desktop\aiac\assg_09.5.py

Define a function that reads a file and returns its contents as a string

Functions

read_file(filename: str) -> str
Reads the contents of a file and returns it as a string.
:param filename: The name of the file to read
:return: The contents of the file as a string
:raises ValueError: If the filename is not a string
:raises FileNotFoundError: If the file does not exist
:raises IOError: If there is an error reading the file


Python 3.13.5 [tags/v3.13.5:6cb20a2, MSC v.1943 64 bit (AMD64)]
Windows-11
Module Index : Topics : Keywords
Get
Search

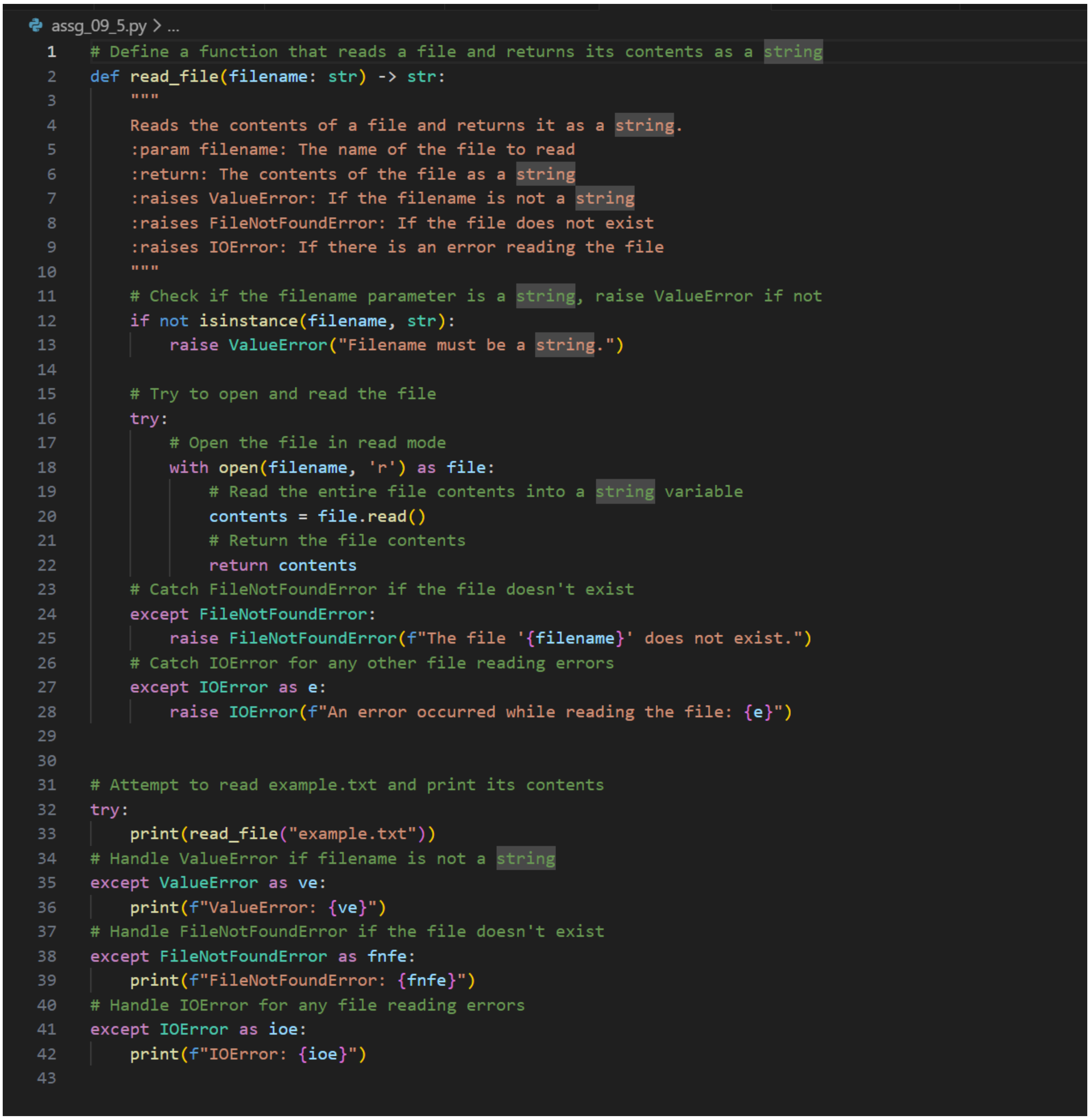
assg_9
index
c:\users\bharath\onedrive\pictures\desktop\aiac\assg_9.py

```
# Define a function that reads a file and returns its contents as a string
```

Functions

read_file(filename: str) -> str
```
Reads the contents of a file and returns it as a string.
:param filename: The name of the file to read
:return: The contents of the file as a string
:raises ValueError: If the filename is not a string
:raises FileNotFoundError: If the file does not exist
:raises IOError: If there is an error reading the file
```

Incline comments:


assg_09_5.py > ...

```python
# Define a function that reads a file and returns its contents as a string
def read_file(filename: str) -> str:
    """
    Reads the contents of a file and returns it as a string.
    :param filename: The name of the file to read
    :return: The contents of the file as a string
    :raises ValueError: If the filename is not a string
    :raises FileNotFoundError: If the file does not exist
    :raises IOError: If there is an error reading the file
    """
    # Check if the filename parameter is a string, raise ValueError if not
    if not isinstance(filename, str):
        raise ValueError("Filename must be a string.")

    # Try to open and read the file
    try:
        # Open the file in read mode
        with open(filename, 'r') as file:
            # Read the entire file contents into a string variable
            contents = file.read()
            # Return the file contents
            return contents
    # Catch FileNotFoundError if the file doesn't exist
    except FileNotFoundError:
        raise FileNotFoundError(f"The file '{filename}' does not exist.")
    # Catch IOError for any other file reading errors
    except IOError as e:
        raise IOError(f"An error occurred while reading the file: {e}")


# Attempt to read example.txt and print its contents
try:
    print(read_file("example.txt"))
# Handle ValueError if filename is not a string
except ValueError as ve:
    print(f"ValueError: {ve}")
# Handle FileNotFoundError if the file doesn't exist
except FileNotFoundError as fnfe:
    print(f"FileNotFoundError: {fnfe}")
# Handle IOError for any file reading errors
except IOError as ioe:
    print(f"IOError: {ioe}")
```