

# Lab Assignment-9.5

Name: P.Chandra Vardhan Reddy

Hall cket:2303A51034

Batch-01

Problem 1: String U li es Func on

Consider the following Python func on:

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

Task:

1. Write documenta on in:

o (a) Docstring o (b) Inline

comments o (c) Google-style

documenta on

2. Compare the three documenta on styles.

3. Recommend the most suitable style for a u lity-based string library.

Docstring:

```
1  def reverse_string(s):  
2      """  
3      Docstring for reverse_string  
4      :param: s (str)  
5      :return: str representing the reversed input string  
6      :exceptions: ValueError for invalid input types  
7      :error handling: Catches ValueError and prompts user to enter valid input  
8      :side effects: None  
9      :description: Reverses the input string. Validates the input to ensure that it is a string.  
10     :example usage:  
11         reverse_string("hello")  
12         Returns: "olleh"  
13     """  
14     if not isinstance(s, str):  
15         raise ValueError("Input must be a string.")  
16  
17     return s[::-1]
```

```

Help on module assg_09_5:

NAME
  assg_09_5

FUNCTIONS
  reverse_string(s)
    Docstring for reverse_string
    :param: s (str)
    :return: str representing the reversed input string
    :exceptions: ValueError for invalid input types
    :error handling: Catches ValueError and prompts user to enter valid input
    :side effects: None
    :description: Reverses the input string. Validates the input to ensure that it is a string.
    :example usage:
        reverse_string("hello")
-- More --

```

Incline comments:

```

assg_09_5.py > ...
1  def reverse_string(s):# defining a function that takes a string as input
2      if not isinstance(s, str):# checking if the input is not a string
3          raise ValueError("Input must be a string.")# raising a ValueError if the input is not a string
4      return s[::-1]# returning the reversed string using slicing
5  s="Hello, World!"# defining a string variable
6  print(reverse_string(s))# calling the reverse_string function and printing the result

```

Google style :

```

assg_09_5.py > ...
1  def reverse_string(s:str) -> str:
2      """
3      Docstring for reverse_string
4      :param: s (str)
5      :return: str representing the reversed input string
6      :exceptions: ValueError for invalid input types
7      :error handling: Catches ValueError and prompts user to enter valid input
8      :side effects: None
9      :description: Reverses the input string. Validates the input to ensure that it is a string.
10     :example usage:
11         reverse_string("hello")
12         Returns: "olleh"
13     """
14     if not isinstance(s, str):
15         raise ValueError("Input must be a string.")
16
17     return s[::-1]
18 string_to_reverse = "hello"
19 print(reverse_string(string_to_reverse))

```

## Problem 2: Password Strength Checker

Consider the func on:

```

def check_strength(password):

    return len(password) >= 8

```

Task:

1. Document the func on using docstring, inline comments, and

Google style.

2. Compare documenta on styles for security-related code.

3. Recommend the most appropriate style.

Doc\_string and google\_style:

```
9
10 def password_strength_checker(password:str) -> str:
11     """
12     Docstring for password_strength_checker
13     :param password (str)
14     :return: str indicating the strength of the password ("Weak", "Moderate", "Strong")
15     :exceptions: ValueError for invalid input types or values
16     :error handling: Catches ValueError and prompts user to enter valid input
17     :side effects: None
18     :description: Evaluates the strength of a given password based on its length and character composition.
19     | Validates the input to ensure that it is a string and meets certain criteria for strength.
20     :example usage:
21     | password_strength_checker("P@ssw0rd")
22     | Returns: "Strong"
23     """
24     if not isinstance(password, str):
25         raise ValueError("Password must be a string.")
26
27     length = len(password)
28     has_upper = any(c.isupper() for c in password)
29     has_lower = any(c.islower() for c in password)
30     has_digit = any(c.isdigit() for c in password)
31     has_special = any(not c.isalnum() for c in password)
32
33     if length < 6:
34         return "Weak"
35     elif length < 12:
36         if has_upper and has_lower and has_digit:
37             return "Moderate"
38         else:
39             return "Weak"
40     else:
41         if has_upper and has_lower and has_digit and has_special:
42             return "Strong"
43         else:
44             return "Moderate"
45 password = "P@ssw0rd"
46 print(password_strength_checker(password))
47
```

Inline comments:

```
9
10 def password_strength_checker(password:str) -> str: #defining the function with type hints
11
12     if not isinstance(password, str):# if password is not a string, raise an error
13         raise ValueError("Password must be a string.")# raise an error if the password is not a string
14     if len(password) < 8:# if the password is less than 8 characters, return weak
15         return "Weak" #return weak if the password is less than 8 characters
16     if not any(char.isupper() for char in password):# if the password does not contain an uppercase letter, return weak
17         return "Weak" #return weak if the password does not contain an uppercase letter
18     if not any(char.islower() for char in password):# if the password does not contain a lowercase letter, return weak
19         return "Weak" #return weak if the password does not contain a lowercase letter
20     if not any(char.isdigit() for char in password):# if the password does not contain a digit, return weak
21         return "Weak" #return weak if the password does not contain a digit
22     if not any(char in "!@#$%^&*()-_+=[]{}|;:'\".,<>?/" for char in password):# if the password does not contain a special character, return weak
23         return "Weak" #return weak if the password does not contain a special character
24     return "Strong" #return strong if the password meets all the criteria
25 password_to_check = "P@ssw0rd"
26 print(password_strength_checker(password_to_check))
```

Problem 3: Math U li es Module

Task:

1. Create a module math\_u ls.py with func ons:

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 > ...

```
1  def factorial(n):
2      """
3      Calculate the factorial of a non-negative integer n.
4      :param n: A non-negative integer
5      :return: The factorial of n
6      :raises ValueError: If n is negative or not an integer
7
8
9      Example usage:
10     print(factorial(5)) # Output: 120
11
12     """
13     if not isinstance(n, int):
14         raise ValueError("Input must be an integer.")
15     if n < 0:
16         raise ValueError("Input must be a non-negative integer.")
17
18     result = 1
19     for i in range(2, n + 1):
20         result *= i
21
22     return result
23 def square(x):
24     """
25     Calculate the square of a number x.
26     :param x: A number (int or float)
27     :return: The square of x
28     :raises ValueError: If x is not a number
29
30
31     Example usage:
32     print(square(4)) # Output: 16
33     print(square(2.5)) # Output: 6.25
34
35     """
36     if not isinstance(x, (int, float)):
37         raise ValueError("Input must be a number.")
38
39     return x * x
40 def cube(x):
41     """
42     Calculate the cube of a number x.
43     :param x: A number (int or float)
44     :return: The cube of x
45     :raises ValueError: If x is not a number
46
47
48     Example usage:
49     print(cube(3)) # Output: 27
50     print(cube(1.5)) # Output: 3.375
51
52     """
53     if not isinstance(x, (int, float)):
54         raise ValueError("Input must be a number.")
55
56     return x * x * x
57 print(factorial(5))
58 print(square(4))
59 print(cube(3))
60
```

Output:

[index](#)  
**math\_utils** [c:\users\arell\music\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\arell\Music\aiac> python -m pydoc -w C:\Users\arell\Music\aiac\math_utils.py  
120  
16  
27  
wrote math_utils.html  
PS C:\Users\arell\Music\aiac> █
```

## Problem 4: A endance Management Module

Task:

1. Create a module a endance.py with func ons:

- 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 > ...
1 class Attendance:
2     def __init__(self, student_name, date, status):
3         """Docstring for Attendance class
4         :param: student_name (str), date (str), status (str)
5         :return: None
6         :exceptions: ValueError for invalid input types or values
7         :error handling: Catches ValueError and prompts user to enter valid input
8         :side effects: None
9         :description: Initializes an Attendance object with the given student name, date, and status.
10        | Validates the input to ensure that student_name and date are strings and status is either "Present" or "Absent".
11        :example usage:
12        | attendance = Attendance("John Doe", "2024-06-01", "Present")
13        """
14        if not isinstance(student_name, str) or not isinstance(date, str):
15            raise ValueError("Student name and date must be strings.")
16        if status not in ["Present", "Absent"]:
17            raise ValueError("Status must be either 'Present' or 'Absent'.")
18
19        self.student_name = student_name
20        self.date = date
21        self.status = status
22    def __str__(self):
23        """Docstring for __str__ method
24        :param: None
25        :return: str representation of the Attendance object
26        :exceptions: None
27        :error handling: None
28        :side effects: None
29        :description: Returns a string representation of the Attendance object in the format "Student Name:
30        | Date: YYYY-MM-DD, Status: Present/Absent".
31        :example usage:
32        | attendance = Attendance("John Doe", "2024-06-01", "Present")
33        | print(attendance) # Output: Student Name: John Doe, Date: 2024-06-01, Status: Present
34        """
35        return f"Student Name: {self.student_name}, Date: {self.date}, Status: {self.status}"
36 attendance = Attendance("John Doe", "2024-06-01", "Present")
37 print(attendance)
38
```

Output:





## Problem 5: File Handling Func on

Consider the func on:

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

Task:

1. Write documenta on using all three formats.
2. Iden fy which style best explains excep on handling.
3. Jus fy your recommenda on.

Code:

```
27  
28 def read_file(filename:str) -> str:  
29     """  
30     Reads the contents of a file and returns it as a string.  
31     :param filename: The name of the file to read  
32     :return: The contents of the file as a string  
33     :raises ValueError: If the filename is not a string  
34     :raises FileNotFoundError: If the file does not exist  
35     :raises IOError: If there is an error reading the file  
36     Example usage:  
37     print(read_file("example.txt"))  
38  
39     """  
40     if not isinstance(filename, str):  
41         raise ValueError("Filename must be a string.")  
42  
43     try:  
44         with open(filename, 'r') as file:  
45             contents = file.read()  
46             return contents  
47     except FileNotFoundError:  
48         raise FileNotFoundError(f"The file '{filename}' does not exist.")  
49     except IOError as e:  
50         raise IOError(f"An error occurred while reading the file: {e}")  
51  
52     try:  
53         print(read_file("example.txt"))  
54     except ValueError as ve:  
55         print(f"ValueError: {ve}")  
56     except FileNotFoundError as fnfe:  
57         print(f"FileNotFoundError: {fnfe}")  
58     except IOError as ioe:  
59         print(f"IOError: {ioe}")  
60  
61     print(read_file("example.txt"))  
61
```

Output:

```

PS C:\Users\arell\Music\aiac> python -m pydoc C:\Users\arell\Music\aiac\assg_09_5.py
hello world!.
Help on module assg_09_5:

NAME
    assg_09_5

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\arell\music\aiac\assg_09_5.py

PS C:\Users\arell\Music\aiac>

```

[index](#)  
**assg\_09\_5** [c:\users\arell\music\aiac\assg\\_09\\_5.py](#)

## 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 3.14.2 [tags/v3.14.2:df79316, MSC v.1944 64 bit (AMD64)]  
 Windows-11

[Module Index](#) : [Topics](#) : [Keywords](#)

**assg\_09\_5**
[index](#)  
[c:\users\arell\music\aiac\assg\\_09\\_5.py](#)

**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 > ...

```
1  # Define a function that reads a file and returns its contents as a string
2  def read_file(filename: str) -> str:
3      """
4      Reads the contents of a file and returns it as a string.
5      :param filename: The name of the file to read
6      :return: The contents of the file as a string
7      :raises ValueError: If the filename is not a string
8      :raises FileNotFoundError: If the file does not exist
9      :raises IOError: If there is an error reading the file
10     """
11     # Check if the filename parameter is a string, raise ValueError if not
12     if not isinstance(filename, str):
13         raise ValueError("Filename must be a string.")
14
15     # Try to open and read the file
16     try:
17         # Open the file in read mode
18         with open(filename, 'r') as file:
19             # Read the entire file contents into a string variable
20             contents = file.read()
21             # Return the file contents
22             return contents
23     # Catch FileNotFoundError if the file doesn't exist
24     except FileNotFoundError:
25         raise FileNotFoundError(f"The file '{filename}' does not exist.")
26     # Catch IOError for any other file reading errors
27     except IOError as e:
28         raise IOError(f"An error occurred while reading the file: {e}")
29
30
31     # Attempt to read example.txt and print its contents
32     try:
33         print(read_file("example.txt"))
34     # Handle ValueError if filename is not a string
35     except ValueError as ve:
36         print(f"ValueError: {ve}")
37     # Handle FileNotFoundError if the file doesn't exist
38     except FileNotFoundError as fnfe:
39         print(f"FileNotFoundError: {fnfe}")
40     # Handle IOError for any file reading errors
41     except IOError as ioe:
42         print(f"IOError: {ioe}")
43
```