

# **AI Assignment**

## **Assignment -9.5**

Hallticket no:2303A510A6

Batch no: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

Documentation:

```

Users / shyam / ar-assistes code / ass9.3.py
1 def reverse_string(text):
2     """
3     Reverse the given string.
4
5     Parameters:
6     text (str): The string to be reversed.
7
8     Returns:
9     str: The reversed string.
10    """
11    return text[::-1]
12
13 # Example usage
14 input_string = "Hello, World!"
15 reversed_string = reverse_string(input_string)
16
17 print(f"Original string: {input_string}")
18 print(f"Reversed string: {reversed_string}")
19
20 def reverse_string(text):
21     # Reverse the string using slicing
22     return text[::-1]
23
24 # Example usage
25 input_string = "Hello, World!"
26 reversed_string = reverse_string(input_string)
27 print("Original string:", input_string)
28 print("Reversed string:", reversed_string)
29
30 def reverse_string(text)
31     """
32     Reverse the given string.
33
34     Args:
35     text (str): The input string to reverse.
36
37     Returns:
38     str: The reversed string.
39     """
40     return text[::-1]

```

```
print(f"Original string: {input_string}")
print(f"Reversed string: {reversed_string}")

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

# Example usage
input_string = "Hello, World!"
reversed_string = reverse_string(input_string)
print("Original string:", input_string)
print("Reversed string:", reversed_string)

def reverse_string(text)
    """
    Reverse the given string.

    Args:
        text (str): The input string to reverse.

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

# Example usage
input_string = "Hello, World!"
reversed_string = reverse_string(input_string)

print(f"Original string: {input_string}")
print(f"Reversed string: {reversed_string}")
```

```
PS C:\Users\shyam\ai assistes code> python -m pydoc ass67
Help on module ass67:
```

NAME

ass67

DESCRIPTION

Module: reverse\_string\_examples  
This file demonstrates three documentation styles.

FUNCTIONS

reverse\_string\_docstring(text)  
Takes a string as input and returns the reversed string.

Parameters:

text (str): The string to be reversed.

Returns:

str: The reversed string.

reverse\_string\_google(text)  
Reverses the given string.

Args:

text (str): The string to reverse.

-- More --

[index](#)

ass67 [c:\users\shyam\ai assistes code\ass67.py](#)

Module: reverse\_string\_examples

This file demonstrates three documentation styles.

## Functions

**reverse\_string\_docstring(text)**

Takes a string as input and returns the reversed string.

Parameters:

text (str): The string to be reversed.

Returns:

str: The reversed string.

**reverse\_string\_google(text)**

Reverses the given string.

Args:

text (str): The string to reverse.

Returns:

str: The reversed string.

Example:

```
>>> reverse\_string\_google("hello")  
'olleh'
```

**reverse\_string\_inline(text)**

hon 3.13.2 [tags/v3.13.2:4f8bb39, MSC v.1942 64 bit (AMD64)]  
adows-11

Module Index : Topics : Keywords

Index of Modules

Built-in Modules

<a href="#">_abc</a>	<a href="#">_imp</a>	<a href="#">_signal</a>	<a href="#">binascii</a>
<a href="#">_ast</a>	<a href="#">_interpchannels</a>	<a href="#">_sre</a>	<a href="#">builtins</a>
<a href="#">_bisect</a>	<a href="#">_interpqueues</a>	<a href="#">_stat</a>	<a href="#">cmath</a>
<a href="#">_blake2</a>	<a href="#">_interpreters</a>	<a href="#">_statistics</a>	<a href="#">errno</a>
<a href="#">_codecs</a>	<a href="#">_io</a>	<a href="#">_string</a>	<a href="#">faulthandler</a>
<a href="#">_codecs_cn</a>	<a href="#">_json</a>	<a href="#">_struct</a>	<a href="#">gc</a>
<a href="#">_codecs_hk</a>	<a href="#">_locale</a>	<a href="#">_symtable</a>	<a href="#">itertools</a>
<a href="#">_codecs_iso2022</a>	<a href="#">_lsprof</a>	<a href="#">_sysconfig</a>	<a href="#">marshal</a>
<a href="#">_codecs_jp</a>	<a href="#">_md5</a>	<a href="#">_thread</a>	<a href="#">math</a>
<a href="#">_codecs_kr</a>	<a href="#">_multibytecodec</a>	<a href="#">_tokenize</a>	<a href="#">mmap</a>
<a href="#">_codecs_tw</a>	<a href="#">_opcode</a>	<a href="#">_tracemalloc</a>	<a href="#">mmap</a>
<a href="#">_collections</a>	<a href="#">_operator</a>	<a href="#">_typing</a>	<a href="#">mmap</a>
<a href="#">_contextvars</a>	<a href="#">_pickle</a>	<a href="#">_warnings</a>	<a href="#">mmap</a>
<a href="#">_cav</a>	<a href="#">_random</a>	<a href="#">_weakref</a>	<a href="#">nt</a>
<a href="#">_datetime</a>	<a href="#">_sha1</a>	<a href="#">_winapi</a>	<a href="#">os</a>
<a href="#">_functools</a>	<a href="#">_sha2</a>	<a href="#">array</a>	<a href="#">time</a>
<a href="#">_heapq</a>	<a href="#">_sha3</a>	<a href="#">atexit</a>	<a href="#">xxsubtype</a>
			<a href="#">zlib</a>

Users\shyam\ai assistes code

<a href="#">Untitled-2</a>	<a href="#">ass2ai</a>	<a href="#">ass78</a>	
<a href="#">ass11ai</a>	<a href="#">ass35ai</a>	<a href="#">ass7ai.py</a>	
<a href="#">ass14ai</a>	<a href="#">ass3ai</a>	<a href="#">ass8ai</a>	<a href="#">ass173ai</a>
<a href="#">ass16ai</a>	<a href="#">ass4ai</a>	<a href="#">ass9ai</a>	<a href="#">file_processor</a>
<a href="#">ass17ai</a>	<a href="#">ass52</a>	<a href="#">asshp3</a>	<a href="#">lab1ai</a>
<a href="#">ass1ai</a>	<a href="#">ass5ai</a>	<a href="#">ass112ai</a>	<a href="#">mmap</a>
<a href="#">ass21ai</a>	<a href="#">ass67</a>	<a href="#">ass14ai</a>	<a href="#">sec</a>
<a href="#">ass22ai</a>	<a href="#">ass6ai</a>		<a href="#">testcasedemo</a>

## Comparison of the Three Documentation Styles

### 1. Docstring Style

Uses a clear description inside triple quotes.

Explains parameters and return values.

Easy to read and understand.

### 2. Inline Comments Style

Uses comments inside the code.

Explains small parts of the program.

Useful for simple explanations but not detailed.

### 3. Google-Style Documentation

Structured and well-formatted.

Clearly separates Args, Returns, and Examples.

Very professional and easy to maintain.

## Recommendation

For a utility-based string library, the Google-style documentation is the most suitable.

It is clear, organized, and gives complete information, which is important for reusable code in libra

## 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.

```
Help on module ass66:  
  
NAME  
    ass66  
  
DESCRIPTION  
    Module: password_strength_checker  
    This module demonstrates three documentation styles for a function.  
  
FUNCTIONS  
    check_strength_docstring(password)  
        Checks whether a password is strong based on its length.  
  
        A password is considered strong if it has at least 8 characters.  
  
    Parameters:  
        password (str): The password to check.
```

---

[index](#)

ass66 [c:\users\shyam\ai assistes code\ass66.py](#)

Module: password\_strength\_checker

This module demonstrates three documentation styles for a function.

## Functions

**check\_strength\_docstring**(password)

Checks whether a password is strong based on its length.

A password is considered strong if it has at least 8 characters.

Parameters:

password (str): The password to check.

Returns:

bool: True if password length is 8 or more, False otherwise.

**check\_strength\_google**(password)

Checks whether a password is strong based on its length.

Args:

password (str): The password to evaluate.

Returns:

bool: True if the password has at least 8 characters,  
False otherwise.

Example:

```
>>> check\_strength\_google("mypassword")  
True
```

**check\_strength\_inline**(password)





## Comparison of Documentation Styles (Security Code)

Inline comments: Explain small parts of code, but too many can make code messy.

Docstrings: Describe what the function does clearly and are good for documentation.

Google style: Structured, clear, and shows parameters and return values.

## Recommendation

For security-related code, Google style docstrings are best because they are clear, organized, and easy for teams to understand and review.

## 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:

```
math_utils.py

A simple math utilities module that provides basic mathematical
- square
- cube
- factorial
"""
def square(n):
    """
    Return the square of a number.

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

    Returns:
        int or float: The square of n.
    """
    return n * n

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

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

    Returns:
        int or float: The cube of n.
    """
    return n * n * n

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

    Args:
        n (int): A non-negative integer.

    Returns:
        int: The factorial of n.

    Raises:
        ValueError: If n is negative.
    """
    if n < 0:
```

se help(str) for help on the str class.

S C:\Users\shyam\ai assistes code> python -m pydoc ass66  
help on module ass66:

NAME

ass66 - math\_utils.py

DESCRIPTION

A simple math utilities module that provides basic mathematical functions:

- square
- cube
- factorial

FUNCTIONS

cube(n)

Return the cube of a number.

Args:

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

Returns:

int or float: The cube of n.

factorial(n)

Calculate the factorial of a non-negative integer.

Args:

n (int): A non-negative integer.

Returns:

int: The factorial of n.

Raises:

ValueError: If n is negative.

square(n)

Return the square of a number.

Args:

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

Returns:

[index](#)

ass66 [c:\users\shyam\ai assistes code\ass66.py](#)

nath\_utils.py

A simple math utilities module that provides basic mathematical functions:

- square
- cube
- factorial

## Functions

### **cube(n)**

Return the cube of a number.

Args:

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

Returns:

int or float: The cube of n.

### **factorial(n)**

Calculate the factorial of a non-negative integer.

Args:

n (int): A non-negative integer.

Returns:

int: The factorial of n.

Raises:

ValueError: If n is negative.

### **square(n)**

Return the square of a number.

Args:

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



## 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
- Documentation:

```

"""
# Dictionary to store attendance data
attendance_record = {}
def mark_present(student):
    """
    Mark a student as present.

    Args:
        student (str): Name of the student.

    Returns:
        None
    """
    attendance_record[student] = "Present"
def mark_absent(student):
    """
    Mark a student as absent.

    Args:
        student (str): Name of the student.

    Returns:
        None
    """
    attendance_record[student] = "Absent"
def get_attendance(student):
    """
    Get the attendance status of a student.

    Args:
        student (str): Name of the student.

    Returns:
        str: Attendance status ("Present", "Absent", or "Not Recorded").
    """
    return attendance_record.get(student, "Not Recorded")

```

help on module ass66:

NAME

ass66 - attendance.py

DESCRIPTION

A simple Attendance Management Module.

Provides functions to mark student attendance and check status.

FUNCTIONS

get\_attendance(student)

Get the attendance status of a student.

Args:

student (str): Name of the student.

Returns:

str: Attendance status ("Present", "Absent", or "Not Recorded").

mark\_absent(student)

Mark a student as absent.

Args:

student (str): Name of the student.

Returns:

None

mark\_present(student)

Mark a student as present.

Args:

student (str): Name of the student.

Returns:

None

DATA

attendance\_record = {}

FILE

[index](#)

ass66 [c:\users\shyam\ai assistes code\ass66.py](#)

attendance.py

A simple Attendance Management Module.

Provides functions to mark student attendance and check status.

## Functions

**get\_attendance**(student)

Get the attendance status of a student.

Args:

student (str): Name of the student.

Returns:

str: Attendance status ("Present", "Absent", or "Not Recorded").

**mark\_absent**(student)

Mark a student as absent.

Args:

student (str): Name of the student.

Returns:

None

**mark\_present**(student)

Mark a student as present.

Args:

student (str): Name of the student.

Returns:

None



Index Of Modules			
Built-in Modules			
<a href="#">_abc</a>	<a href="#">_imp</a>	<a href="#">_signal</a>	<a href="#">binascii</a>
<a href="#">_ast</a>	<a href="#">interpchannels</a>	<a href="#">_sre</a>	<a href="#">builtins</a>
<a href="#">_bisect</a>	<a href="#">interpqueues</a>	<a href="#">_stat</a>	<a href="#">cmath</a>
<a href="#">_blake2</a>	<a href="#">interpreters</a>	<a href="#">statistics</a>	<a href="#">errno</a>
<a href="#">_codecs</a>	<a href="#">_io</a>	<a href="#">string</a>	<a href="#">faulthandler</a>
<a href="#">_codecs_cn</a>	<a href="#">_json</a>	<a href="#">struct</a>	<a href="#">gc</a>
<a href="#">_codecs_hk</a>	<a href="#">_locale</a>	<a href="#">symtable</a>	<a href="#">itertools</a>
<a href="#">_codecs_iso2022</a>	<a href="#">_lsprof</a>	<a href="#">sysconfig</a>	<a href="#">marshal</a>
<a href="#">_codecs_jp</a>	<a href="#">md5</a>	<a href="#">thread</a>	<a href="#">math</a>
<a href="#">_codecs_kr</a>	<a href="#">multibytecodec</a>	<a href="#">tokenize</a>	<a href="#">mmap</a>
<a href="#">_codecs_tw</a>	<a href="#">opcode</a>	<a href="#">tracemalloc</a>	<a href="#">msvert</a>
<a href="#">_collections</a>	<a href="#">operator</a>	<a href="#">typing</a>	<a href="#">nt</a>
<a href="#">_contextvars</a>	<a href="#">pickle</a>	<a href="#">warnings</a>	<a href="#">sys</a>
<a href="#">_csv</a>	<a href="#">random</a>	<a href="#">weakref</a>	<a href="#">time</a>
<a href="#">_datetime</a>	<a href="#">_sha1</a>	<a href="#">winapi</a>	<a href="#">winreg</a>
<a href="#">_functools</a>	<a href="#">_sha2</a>	<a href="#">array</a>	<a href="#">xxsubtype</a>
<a href="#">_heapq</a>	<a href="#">_sha3</a>	<a href="#">atexit</a>	<a href="#">zlib</a>
C:\Users\shyam\ai assistes code			
<a href="#">Untitled-2</a>	<a href="#">ass2ai</a>	<a href="#">ass6ai</a>	
<a href="#">ass11ai</a>	<a href="#">ass35ai</a>	<a href="#">ass78</a>	<a href="#">ass14ai</a>
<a href="#">ass14ai</a>	<a href="#">ass3ai</a>	<a href="#">ass7ai.py</a>	<a href="#">ass173ai</a>
<a href="#">ass16ai</a>	<a href="#">ass4ai</a>		<a href="#">file_processor</a>
<a href="#">ass17ai</a>	<a href="#">ass52</a>	<a href="#">ass8ai</a>	<a href="#">lnb1ai</a>
<a href="#">ass1ai</a>	<a href="#">ass5ai</a>	<a href="#">ass9ai</a>	<a href="#">mago</a>
<a href="#">ass21ai</a>	<a href="#">ass66</a>	<a href="#">asshp3</a>	<a href="#">sec</a>
<a href="#">ass22ai</a>	<a href="#">ass67</a>	<a href="#">ass112ai</a>	<a href="#">testcasedemo</a>

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

```

1  """
2  read_file.py
3
4  This module contains a function to read the contents of a file.
5  It demonstrates different documentation styles.
6  """
7
8  def read_file_inline(filename):
9      # Open the file in read mode
10     with open(filename, 'r') as f:
11         # Read and return file content
12         return f.read()
13
14 def read_file_basic(filename):
15     """
16     Read and return the contents of a file.
17
18     Parameters:
19         filename (str): Name of the file to read.
20
21     Returns:
22         str: Content of the file.
23     """
24     with open(filename, 'r') as f:
25         return f.read()
26
27 def read_file_google(filename):
28     """
29     Read and return the contents of a file.
30
31     Args:
32         filename (str): The path to the file.
33
34     Returns:
35         str: The content of the file.
36
37     Raises:
38         FileNotFoundError: If the file does not exist.
39         IOError: If an error occurs while reading the file.
40     """
41     with open(filename, 'r') as f:
42         return f.read()

```

```
PS C:\Users\shyam\ai assistes code> python -m pydoc ass66
Help on module ass66:
```

#### NAME

ass66 - read\_file.py

#### DESCRIPTION

This module contains a function to read the contents of a file.  
It demonstrates different documentation styles.

#### FUNCTIONS

read\_file\_basic(filename)

Read and return the contents of a file.

Parameters:

filename (str): Name of the file to read.

Returns:

str: Content of the file.

read\_file\_google(filename)

Read and return the contents of a file.

Args:

filename (str): The path to the file.

Returns:

str: The content of the file.

Raises:

FileNotFoundError: If the file does not exist.

IOError: If an error occurs while reading the file.

read\_file\_inline(filename)

#### FILE

[index](#)

ss66 [c:\users\shyam\ai assistes code\ass66.py](#)

ad\_file.py

This module contains a function to read the contents of a file.  
It demonstrates different documentation styles.

## Functions

**read\_file\_basic(filename)**

Read and return the contents of a file.

Parameters:

filename (str): Name of the file to read.

Returns:

str: Content of the file.

**read\_file\_google(filename)**

Read and return the contents of a file.

Args:

filename (str): The path to the file.

Returns:

str: The content of the file.

Raises:

FileNotFoundError: If the file does not exist.

IOError: If an error occurs while reading the file.

**read\_file\_inline(filename)**

## Index of Modules

### Built-in Modules

<a href="#">_abc</a>	<a href="#">_imp</a>	<a href="#">_signal</a>	<a href="#">binascii</a>
<a href="#">_ast</a>	<a href="#">_interchannels</a>	<a href="#">_sre</a>	<a href="#">builtins</a>
<a href="#">_bisect</a>	<a href="#">_intersplices</a>	<a href="#">_stat</a>	<a href="#">cmath</a>
<a href="#">_blake2</a>	<a href="#">_interpreters</a>	<a href="#">_statistics</a>	<a href="#">errno</a>
<a href="#">_codecs</a>	<a href="#">_io</a>	<a href="#">_string</a>	<a href="#">faulthandler</a>
<a href="#">_codecs_cn</a>	<a href="#">_json</a>	<a href="#">_struct</a>	<a href="#">gc</a>
<a href="#">_codecs_hk</a>	<a href="#">_locale</a>	<a href="#">_symtable</a>	<a href="#">itertools</a>
<a href="#">_codecs_iso2022</a>	<a href="#">_lprof</a>	<a href="#">_sysconfig</a>	<a href="#">marshal</a>
<a href="#">_codecs_jp</a>	<a href="#">_md5</a>	<a href="#">_thread</a>	<a href="#">math</a>
<a href="#">_codecs_kr</a>	<a href="#">_multibytecodec</a>	<a href="#">_tokenize</a>	<a href="#">mmap</a>
<a href="#">_codecs_tw</a>	<a href="#">_opcode</a>	<a href="#">_tracemalloc</a>	<a href="#">mmap</a>
<a href="#">_collections</a>	<a href="#">_operator</a>	<a href="#">_typing</a>	<a href="#">mmap</a>
<a href="#">_contextvars</a>	<a href="#">_pickle</a>	<a href="#">_warnings</a>	<a href="#">nt</a>
<a href="#">_csv</a>	<a href="#">_random</a>	<a href="#">_weakref</a>	<a href="#">os</a>
<a href="#">_datetime</a>	<a href="#">_sha1</a>	<a href="#">_winapi</a>	<a href="#">time</a>
<a href="#">_functools</a>	<a href="#">_sha2</a>	<a href="#">array</a>	<a href="#">winreg</a>
<a href="#">_heapq</a>	<a href="#">_sha3</a>	<a href="#">atexit</a>	<a href="#">xxsubtype</a>
			<a href="#">zlib</a>

### C:\Users\shyam\ai assistes code

<a href="#">Untitled-2</a>	<a href="#">ass2ai</a>	<a href="#">ass6ai</a>	<a href="#">ass4ai</a>
<a href="#">ass11ai</a>	<a href="#">ass35ai</a>	<a href="#">ass78</a>	<a href="#">ass173ai</a>
<a href="#">ass14ai</a>	<a href="#">ass3ai</a>	<a href="#">ass7ai</a>	<a href="#">file_processor</a>
<a href="#">ass16ai</a>	<a href="#">ass4ai</a>	<a href="#">ass7ai.py</a>	<a href="#">lab1ai</a>
<a href="#">ass17ai</a>	<a href="#">ass52</a>	<a href="#">ass8ai</a>	<a href="#">magq</a>
<a href="#">ass1ai</a>	<a href="#">ass5ai</a>	<a href="#">ass9ai</a>	<a href="#">sec</a>
<a href="#">ass21ai</a>	<a href="#">ass66</a>	<a href="#">asshp3</a>	<a href="#">testcasedemo</a>
<a href="#">ass22ai</a>	<a href="#">ass67</a>	<a href="#">ass12ai</a>	

## Best Style for Exception Handling

Google Style Docstring is the best style.

### Justification

It has a “Raises” section to clearly explain errors.

It makes exception handling easy to understand.

It improves code readability.

It is useful for team projects and professional coding.

It clearly shows how the function handles possible problems.