

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/sriyam/ai-assistants-code/└ ass5.5.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)

Dex of Modules

Built-in Modules

abc	_imp	_signal	binascii
_ast	_interchannels	_src	builtins
_bisect	_interqueues	_stat	cmath
_blake2	_interpreters	_statistics	errno
_codecs	_io	_string	faulthandler
_codecs_cn	_json	_struct	gc
_codecs_hk	_locale	_syntable	_iterools
_codecs_iso2022	_liprof	_sysconfig	_marshal
_codecs_jp	_md5	_thread	_math
_codecs_kr	_multibytecodec	_tokenize	_mmmap
_codecs_tw	_opcode	_tracemalloc	_mwavef
_collections	_operator	_typing	_nt
_contextvars	_pickle	_warnings	_sys
_csv	_random	_weakref	_time
_datetime	_sha1	_winapi	_winreg
_functools	_sha2	_array	_xxsubtypes
_heapq	_sha3	_atexit	_zlib

\Users\shyam\ai assistes code

Untitled-2	ass2ai	ass78	assi73ai
assi1ai	ass35ai	ass7ai	file_processor
assi1ai	ass3ai	ass7ai.py	lab1ai
assi1ai	ass4ai	ass8ai	magq
assi1ai	ass5ai	ass9ai	sec
assi1ai	ass6ai	asshp3	testasedemo
assi2ai	ass67	ass12ai	
assi2ai	ass6ai	ass14ai	

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)

The screenshot shows a web-based Python module browser. At the top, it displays system information: Python 3.13.2 [tags/v3.13.2:4f8bb39, MSC v.1942 64 bit (AMD64)] on Windows-11. The header includes links for Module Index, Topics, and Keywords, with a search bar and a 'Get' button. Below the header, a blue bar says 'Index of Modules'. A pink bar labeled 'Built-in Modules' contains a list of standard library modules: abc, ast, bisect, blake2, codecs, codecs_cn, codecs_hk, codecs_iso2022, codecs_jp, codecs_kr, codecs_tw, collections, contextvars, csv, datetime, functools, heapq, imp, interpcodes, interpreters, io, json, locale, lprof, md5, multibytescodec, opcode, operator, pickle, random, sha1, sha2, sha3, signal, sre, stat, statistics, string, struct, symtable, sysconfig, thread, tokenize, tracemalloc, typing, warnings, weakref, winapi, array, atexit, binascii, builtins, cmath, errno, faulthandler, gc, iterools, marshal, math, mmap, msvcrt, nt, sys, time, winreg, xxsubtype, zlib.

Below the built-in modules, another pink bar says 'C:\Users\shyamal assistes code'. It lists several user-created files: Untitled-2, ass11ai, ass14ai, ass16ai, ass17ai, ass1ai, ass21ai, ass22ai, ass2ai, ass35ai, ass3ai, ass4ai, ass52, ass5ai, ass66, ass67, ass6ai, ass78, ass7ai, ass7ai.py, ass8ai, ass9ai, asshp3, ass112ai, assi4ai, assi73ai, file_processor, lab1ai, mago, sec, testcasedemo.

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:
```

```
use help(str) for help on the str class.
$ 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>

math_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.

Index of Modules

Built-in Modules

abc	_imp	signal	binascii
ast	_interpcchannels	_sre	builtins
bisect	_interqueues	_stat	cmath
blake2	_interpreters	_statistics	errno
codecs	_io	_string	faulthandler
codecs_cn	_json	_struct	gc
codecs_hk	_locale	_syntable	itertools
codecs_iso2022	_lsprof	_sysconfig	marshal
codecs_jp	_md5	_thread	math
codecs_kr	_multibytecodec	_tokenize	mmap
codecs_tw	_opcode	_tracemalloc	msvcrt
collections	_operator	_typing	nt
contextvars	_pickle	_warnings	sys
csv	_random	_weakref	time
datetime	_sha1	_winapi	winreg
functools	_sha2	array	xxsubtype
heapq	_sha3	atexit	zlib

C:\Users\shyamal assistes code

Untitled-2	ass2ai	ass6ai	assi4ai
ass1ai	ass35ai	ass78	assi73ai
ass14ai	ass3ai	ass7ai	file_processor
ass16ai	ass4ai	ass7ai.py	jab1ai
ass17ai	ass52	ass8ai	mag0
ass1ai	ass5ai	ass9ai	sec
ass21ai	ass66	asshp3	testeasedemo
ass22ai	ass67	assi12ai	

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")
```

```
elp 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			
abc	imp	signal	binascii
ast	interchannels	_sre	builtins
bisect	interqueues	_stat	cmath
blake2	interpreters	statistics	errno
codecs	_io	_string	faulthandler
codecs_en	json	_struct	gc
codecs_hk	locale	_symtable	itertools
codecs_iso2022	liprof	_sysconfig	marshal
codecs_jp	_md5	_thread	math
codecs_kr	multibytecodec	_tokenize	 mmap
codecs_tw	opcode	_tracemalloc	msvcrt
collections	operator	_typing	nt
contextvars	pickle	_warnings	sys
csv	random	_weakref	time
datetime	sha1	_winapi	winreg
functools	sha2	array	xxsubtype
heapq	sha3	atexit	zlib

C:\Users\shyam\ai assistes code			
Untitled-2	ass2ai	ass6ai	assi4ai
ass1ai	ass35ai	ass78	assi73ai
ass14ai	ass3ai	ass7ai	file_processor
ass16ai	ass4ai	ass7ai.py	lab1ai
ass17ai	ass52	ass8ai	mago
ass18ai	ass5ai	ass9ai	sec
ass21ai	ass66	asshp3	testcasedemo
ass22ai	ass67	assi12ai	

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.

```
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:

FileNotFoundException: 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

abc	_imp	signal	binascii
ast	_interpcchannels	_sre	builtins
bisect	_interqueues	_stat	cmath
blake2	_interpreters	_statistics	errno
codecs	_io	_string	faulthandler
codecs_cn	_json	_struct	gc
codecs_hk	_locale	_svmtable	gdb
codecs_iso2022	_lprof	_sysconfig	itertools
codecs_jp	_md5	_thread	marshal
codecs_kr	_multibytecodec	_tokenize	math
codecs_tw	_opcode	_tracemalloc	mmap
collections	_operator	_typing	msvcrt
contextvars	_pickle	_warnings	nt
csv	_random	_weakref	sys
datetime	_sha1	_winapi	time
functools	_sha2	array	winreg
heapq	_sha3	atexit	xxsubtype
			zlib

C:\Users\shyam\ai assisites code

Untitled-2	ass2ai	ass6ai	assi4ai
ass11ai	ass35ai	ass78	assi73ai
ass14ai	ass3ai	ass7ai	file_processor
ass16ai	ass4ai	ass7ai.py	lab1ai
ass17ai	ass52	ass8ai	mag0
ass1ai	ass5ai	ass9ai	sec
ass21ai	ass66	asshp3	testcasesdemo
ass22ai	ass67	ass12ai	

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.