

Assignment – 9.5

Ht.no : 2303A510H4

Batch : 23

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.

```
1 # (a) Docstring  
2 def reverse_string(text):  
3     """  
4         This function takes a string as input and returns the reversed version of that string.  
5         Parameters:  
6             text (str): The string to be reversed.  
7         Returns:  
8             str: The reversed version of the input string.  
9         """  
10        return text[::-1]  
11  
12 # (b) Inline comments  
13 def reverse_string(text):  
14     # This function takes a string as input and returns the reversed version of that string.  
15     # The input parameter 'text' is expected to be a string.  
16     # The function uses slicing to reverse the string. The syntax text[::-1] creates a new string that is a reversed version of "text".  
17     return text[::-1]  
18  
19 # (c) Google-style documentation  
20 def reverse_string(text):  
21     """  
22         Reverses the input string.  
23         Args:  
24             text (str): The string to be reversed.  
25         Returns:  
26             str: The reversed version of the input string.  
27         """  
28        return text[::-1]  
29
```

```

PS C:\Users\sriva\OneDrive\Documents\AI Assisted Code> python -m pydoc assignment
Help on module assignment:

Help on module assignment:

NAME
    assignment - # (a) Docstring

NAME
    assignment - # (a) Docstring
NAME
    assignment - # (a) Docstring

FUNCTIONS

FUNCTIONS
    reverse_string(text)
    reverse_string(text)
        Reverses the input string.
        Args:
            text (str): The string to be reversed.
        Returns:
            str: The reversed version of the input string.

FILE
c:\users\sriva\onedrive\documents\ai assisted code\assignment.py

PS C:\Users\sriva\OneDrive\Documents\AI Assisted Code>

```

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.

```

1  # (a) Docstring
2  def check_strength(password):
3      """
4          This function checks the strength of a password by verifying if it is at least 8 characters long.
5          Parameters:
6          password (str): The password to be checked.
7          Returns:
8          bool: True if the password is strong (at least 8 characters), False otherwise.
9          """
10         return len(password) >= 8
11
12     # (b) Inline comments
13     def check_strength(password):
14         # This function checks the strength of a password by verifying if it is at least 8 characters long.
15         # The input parameter 'password' is expected to be a string.
16         # The function returns True if the length of the password is greater than or equal to 8, indicating that it is strong. Otherwise, it returns False.
17         return len(password) >= 8
18
19     # (c) Google-style documentation
20     def check_strength(password):
21         """
22             Checks the strength of a password.
23             Args:
24                 password (str): The password to be checked.
25             Returns:
26                 bool: True if the password is strong (at least 8 characters), False otherwise.
27             """
28         return len(password) >= 8

```

```
PS C:\Users\sriva\OneDrive\Documents\AI Assisted Code> python -m pydoc assignment
Help on module assignment:

NAME
    assignment - # (a) Docstring

FUNCTIONS
    check_strength(password)
        Checks the strength of a password.
        Args:
            password (str): The password to be checked.
        Returns:
            bool: True if the password is strong (at least 8 characters), False otherwise.
```

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.

```
1 def square(x):
2     """
3         Returns the square of a number.
4         parameter x: The number to be squared.
5         return: The square of x.
6         int or float: The number to be squared.
7         """
8     return x * x
9 def cube(x):
10    """
11        Returns the cube of a number.
12        parameter x: The number to be cubed.
13        return: The cube of x.
14        int or float: The number to be cubed.
15        """
16    return x * x * x
17 def factorial(n):
18    """
19        Returns the factorial of a number.
20        parameter n: The number to compute the factorial of.
21        return: The factorial of n.
22        """
23    if n == 0:
24        return 1
25    else:
26        return n * factorial(n - 1)
27 print(square.__doc__)
28 print(cube.__doc__)
29 print(factorial.__doc__)
30
```

The screenshot shows a browser window displaying the Pydoc Index of Modules for Python 3.14.0. The title bar reads "Pydoc: Index of Modules" and the address bar shows "localhost:10013". The page content includes a search bar at the top right with "Module Index : Topics : Keywords" and search buttons "Get" and "Search". Below the search bar, the text "Python 3.14.0 [tags/v3.14.0:ebf955d, MSC v.1944 64 bit (AMD64)] Windows-11" is displayed. The main content area is titled "Index of Modules" and contains two sections: "Built-in Modules" and "C:\Users\sriva\OneDrive\Documents\AI Assisted Code".

Built-in Modules

abc	imp	_sre	bisect
ast	interchannels	stat	builtins
bisect	interqueues	statistics	cmath
blake2	interpreters	string	errno
codecs	io	struct	faulthandler
codecs_en	locale	suggestions	gc
codecs_hk	lprof	_symbolic	itertools
codecs_iso2022	md5	_sysconfig	marshal
codecs_jp	multibytecodec	thread	math
codecs_kr	opcode	_tokenize	mmap
codecs_tw	operator	_tracemalloc	operator
collections	pickle	_types	at
contextvars	random	warnings	ava
csv	sha1	weakref	time
datetime	sha2	_winapi	wings
functools	sha3	array	xxsubtype
heapq	signal	atexit	zlib
hmac			

C:\Users\sriva\OneDrive\Documents\AI Assisted Code

AI_lab2	documentationEx	lab9
assignment	lab10	labtest
calculator	lab6	math_utils

C:\Users\sriva\AppData\Local\Programs\Python\Python314\python314.zip

Problem 4: Attendance Management Module

Task:

1. Create a module attendance.py with functions:
 - o mark_present(student)
 - mark_absent(student) o get_attendance(student)
2. Add proper docstrings.
3. Generate and view documentation in terminal and browse

The screenshot shows a code editor window with the file "attendance.py" open. The code defines three functions: mark_present, mark_absent, and get_attendance. Each function has a docstring providing a brief description, parameters, and returns. The code editor interface includes tabs for other files like "calculator.py", "assignment.py", "calculator.html", and "documentationEx.py". The status bar at the bottom shows tabs for "HOME", "OUTPUT", "DEBUG CONSOLE", "TERMINAL", and "PORTS".

```

attendance = {}
def mark_present(student):
    """
    Marks a student as present in the attendance record.
    Parameters:
    student (str): The name of the student to be marked as present.
    """
    attendance[student] = 'Present'
def mark_absent(student):
    """
    Marks a student as absent in the attendance record.
    Parameters:
    student (str): The name of the student to be marked as absent.
    """
    attendance[student] = 'Absent'
def get_attendance(student):
    """
    Returns the attendance status of a student.
    Parameters:
    student (str): The name of the student whose attendance is to be retrieved.
    Returns:
    str: The attendance status of the student.
    """
    return attendance.get(student, 'Not Found')

```

The screenshot shows a browser window displaying the Pydoc documentation for a module named `math_utils`. The page is titled "math_utils" and contains code snippets and docstrings for various functions:

```

def square(x):
    """
    Returns the square of a number.
    parameters: The number to be squared.
    returns: The square of x.
    int or float: The number to be squared.
    """
    return x * x

def cube(x):
    """
    Returns the cube of a number.
    parameters: The number to be cubed.
    returns: The cube of x.
    int or float: The number to be cubed.
    """
    return x * x * x

def factorial(n):
    """
    Returns the factorial of a number.
    parameters: The number to compute the factorial of.
    returns: The factorial of n.
    """
    if n == 0:
        return 1
    else:
        return n * factorial(n - 1)
print(square.__doc__)
print(cube.__doc__)
print(factorial.__doc__)

Functions
get_attendance(student)
    Returns the attendance status of a student.
    Parameters:
        student (str): The name of the student whose attendance is to be retrieved.
    Returns:
        str: The attendance status of the student.

mark_absent(student)
    Marks a student as absent in the attendance record.
    Parameters:
        student (str): The name of the student to be marked as absent.

mark_present(student)
    Marks a student as present in the attendance record.
    Parameters:
        student (str): The name of the student to be marked as present.

Data
attendance = {}

```

The "Functions" section is highlighted with an orange background, and the "Data" section is highlighted with a green background.

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.

The screenshot shows a code editor window titled "AI Assisted Code". The main area displays the content of "math_utils.py". The code defines a function `read_file` which reads the content of a file specified by the argument `filename`. It includes error handling for `FileNotFoundError` and `IOError`.

```
# DocString style:  
def read_file(filename):  
    """  
    Reads the content of a file and returns it as a string.  
    Parameters:  
        filename (str): The name of the file to be read.  
    Returns:  
        str: The content of the file.  
    Raises:  
        FileNotFoundError: If the specified file does not exist.  
        IOError: If an I/O error occurs while reading the file.  
    """  
  
    try:  
        with open(filename, 'r') as f:  
            return f.read()  
    except FileNotFoundError:  
        print(f"Error: The file '{filename}' was not found.")  
    except IOError as e:  
        print(f"An I/O error occurred: {e}")  
  
# Google style Docstring:  
def read_file(filename):  
    """  
    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.  
    Raises:  
        FileNotFoundError: If the specified file does not exist.  
        IOError: If an I/O error occurs while reading the file.  
    """  
  
    try:  
        with open(filename, 'r') as f:  
            return f.read()  
    except FileNotFoundError:  
        print(f"Error: The file '{filename}' was not found.")  
    except IOError as e:  
        print(f"An I/O error occurred: {e}")  
    raise
```

```
use help(str) for help on the str class.  
PS C:\Users\sriva\OneDrive\Documents\AI Assisted Code> python -m pydoc math_utils  
Help on module math_utils:  
  
NAME  
    math_utils  
  
DESCRIPTION  
  
NAME  
    math_utils  
  
DESCRIPTION  
    def square(x):  
DESCRIPTION  
    def square(x):  
    def square(x):  
        """  
        Returns the square of a number.  
-- More --
```