

ASSIGNMENT – 9.5

HT.NO: 2303A52283

BATCH: 45

TASK – 01:

Question:

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.

Prompt:

Generate a python function to reverse a string and include doc string documentation Include assert test cases to validate the function's correctness.

Generated Code:

```
#Generate a python function to reverse a string and include doc string documentation Include  
assert test cases to validate the function's correctness.
```

```
def reverse_string(s):
```

```
    """
```

This function takes a string as input and returns the reversed version of that string.

Parameters:

s (str): The string to be reversed.

Returns:

str: The reversed string.

"""

```
return s[::-1]
```

```
# Assert test cases to validate the function's correctness
```

```
assert reverse_string("hello") == "olleh", "Test case 1 failed"
```

```
assert reverse_string("Python") == "nohtyP", "Test case 2 failed"
```

```
assert reverse_string("") == "", "Test case 3 failed"
```

```
assert reverse_string("a") == "a", "Test case 4 failed"
```

```
assert reverse_string("12345") == "54321", "Test case 5 failed"
```

```
print("All test cases passed!")
```

Output:

```
python.exe -c:/users/RIMSF
All test cases passed!
```

Explanation:

This function uses Python slicing `[::-1]` to reverse the string easily.

The assert statements check different inputs to make sure the function works correctly.

TASK – 02:

Question:

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.

Prompt:

Generate a python function check_strength to the check the strength of a password and include doc string documentation. Include assert test cases to validate the function's correctness. The function should check for the following criteria: The password must be at least 8 characters long.

Generated Code:

```
def check_strength(password):
```

```
    """
```

This function checks the strength of a password based on specific criteria.

Parameters:

password (str): The password to be checked.

Returns:

str: A message indicating whether the password is strong or weak.

```
    """
```

```
if len(password) < 8:
```

```
    return "Weak password: Password must be at least 8 characters long."
```

```
else:
```

```
    return "Strong password."
```

```
# Assert test cases to validate the function's correctness
```

```
assert check_strength("pass") == "Weak password: Password must be at least 8 characters long.", "Test case 1 failed"
```

```
assert check_strength("password") == "Strong password.", "Test case 2 failed"
```

```
assert check_strength("1234567") == "Weak password: Password must be at least 8 characters long.", "Test case 3 failed"
```

```
assert check_strength("12345678") == "Strong password.", "Test case 4 failed"
```

```
assert check_strength("abc") == "Weak password: Password must be at least 8 characters long.", "Test case 5 failed"
```

```
print("All test cases passed!")
```

Output:

```
All test cases passed!
All test cases passed!
```

Explanation:

This function checks the length of the password. If it has fewer than 8 characters, it returns a weak password message.

If the password is 8 or more characters long, it returns that the password is strong, and the assert tests confirm the function works correctly.

```
PS C:\Users\RIMSHA NAAZ\OneDrive\Documents\AAC-PY> python -m pydoc AAC-9-5
All test cases passed!
All test cases passed!
Help on module AAC-9-5:

NAME
    AAC-9-5

DESCRIPTION
    #Task - 01:
    #Generate a python function to reverse a string and include doc string documentation Include assert test cases to validate the function's correctness.

FUNCTIONS
    check_strength(password)
        This function checks the strength of a password based on specific criteria.

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

        Returns:
            str: A message indicating whether the password is strong or weak.

    reverse_string(s)
        This function takes a string as input and returns the reversed version of that string.

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

        Returns:
            str: The reversed string.

FILE
    c:\users\rimsha naaz\onedrive\documents\aac-py\aac-9-5.py
```

[index](#)
AAC-9-5 <c:\users\rimsha naaz\onedrive\documents\aac-py\aac-9-5.py>

#generate a docstring for the factorial function

Functions

```
factorial(n)
    This function calculates the factorial of a given non-negative integer n.

    Parameters:
        n (int): A non-negative integer for which the factorial is to be calculated.

    Returns:
        int: The factorial of the input integer n.

    Raises:
        ValueError: If n is a negative integer.

    Example Usage:
    >>> factorial\(5\)
    120
    >>> factorial\(0\)
    1
    >>> factorial\(-1\)
    ValueError: Input must be a non-negative integer.
```

TASK – 03:

Question:

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.

Prompt:

Create a module `math_utils.py` and include a function to calculate the `square(n)`, `q(n)` and `factorial(n)` and doc string documentation.

Generated Code:

```
# math_utils.py

def square(n):
    """
```

This function takes a number as input and returns its square.

Parameters:

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

Returns:

int or float: The square of the input number.

```
"""
```

```
return n * n
```

```
def q(n):
```

```
    """This function takes a number as input and returns its cube.
```

Parameters:

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

Returns:

int or float: The cube of the input number.

```
"""
return n * n * n

def factorial(n):
    """This function takes a number as input and returns its factorial.
```

Parameters:

n (int): The number to calculate the factorial of.

Returns:

int: The factorial of the input number.

```
"""

result = 1
```

```
for i in range(1, n + 1):
```

```
    result = result * i
```

```
return result
```

Output:

```
int or float. The square of the input number.

FILE
c:\users\rimsha naaz\onedrive\documents\aac-py\aac-9-5.py

PS C:\Users\RIMSHA NAAZ\OneDrive\Documents\AAC-PY> python -m pydoc -w AAC-9-S
No Python documentation found for 'AAC-9-S'.
Use help() to get the interactive help utility.
Use help(str) for help on the str class.
PS C:\Users\RIMSHA NAAZ\OneDrive\Documents\AAC-PY> python -m pydoc -w AAC-9-5
All test cases passed!
All test cases passed!
wrote AAC-9-5.html
PS C:\Users\RIMSHA NAAZ\OneDrive\Documents\AAC-PY> python -m pydoc -p 1234
Server ready at http://localhost:1234/
Server commands: [b]rowser, [q]uit
server>
Server commands: [b]rowser, [q]uit
server> All test cases passed!
All test cases passed!

Server commands: [b]rowser, [q]uit
server> q
Server stopped
PS C:\Users\RIMSHA NAAZ\OneDrive\Documents\AAC-PY>
```

← → ⌂ ⌂ localhost:1234

Python 3.11.0 [main, MSC v.1933 64 bit (AMD64)]
Windows-10

Module Index : Topics : Keywords
Get Search

Index of Modules

Built-in Modules

abc	imp	stat	builtins
ast	io	statistics	cmath
bisect	json	string	errno
blake2	_locale	struct	faulthandler
codecs	_lsprof	symtable	gc
codecs_cn	_md5	thread	itertools
codecs_hk	_multibytecodec	 tokenize	marshal
codecs_iso2022	_opcode	tracemalloc	math
codecs_jp	_operator	typing	mmap
codecs_kr	_pickle	warnings	msvcrt
codecs_tw	_random	weakref	nt
collections	_sha1	_winapi	sys
contextvars	_sha256	_xsubinterpreters	time
csv	_sha3	array	winreg
datetime	_sha512	atexit	_xxsubtype
functools	_signal	audioop	zlib
heapq	_sre	binascii	

C:\Users\RIMSHA NAAZ\OneDrive\Documents\AAC-PY
[AAC-04](#) [AAC-10-5](#) [AAC-9-5](#) [AAC-LAB-02](#)

C:\Users\RIMSHA NAAZ\AppData\Local\Programs\Python\Python311\python311.zip

Python 3.11.0 [main, MSC v.1933 64 bit (AMD64)]
Windows-10

Module Index : Topics : Keywords
Get Search

AAC-9-5

index
<c:/users/rimsha naaz/onedrive/documents/aac-py/aac-9-5.py>

#Task - 01:
#Generate a python function to reverse a string and include doc string documentation. Include assert test cases to validate the function's correctness.

Functions

check_strength(password)
This function checks the strength of a password based on specific criteria.
Parameters:
password (str): The password to be checked.
Returns:
str: A message indicating whether the password is strong or weak.

factorial(n)
This function takes a number as input and returns its factorial.
Parameters:
n (int): The number to calculate the factorial of.
Returns:
int: The factorial of the input number.

q(n)
This function takes a number as input and returns its cube.
Parameters:
n (int or float): The number to be cubed.
Returns:
int or float: The cube of the input number.

reverse_string(s)
This function takes a string as input and returns the reversed version of that string.
Parameters:
s (str): The string to be reversed.
Returns:
str: The reversed string.

reverse_string(s)
This function takes a string as input and returns the reversed version of that string.
Parameters:
s (str): The string to be reversed.
Returns:
str: The reversed string.

square(n)
This function takes a number as input and returns its square.
Parameters:
n (int or float): The number to be squared.
Returns:
int or float: The square of the input number.

Explanation:

square(n) returns the square of a number by multiplying it with itself, and q(n) returns the cube by multiplying the number three times.

factorial(n) uses a loop to multiply numbers from 1 to n, giving the factorial value of the input number.

TASK – 04:

Question:

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

Prompt:

create a module attendance.py and include functions mark_present(student),

mark_absent(student),get_attendance(student). Include docstring documentation in the function.

attendance.py and add assert test cases to validate the functions correctness

Generated Code:

class Attendance:

```
def __init__(self):  
    self.attendance_record = {}  
  
def mark_present(self, student):
```

""""

Marks a student as present in the attendance record.

Parameters:

student (str): The name of the student to be marked as present.

""""

```
    self.attendance_record[student] = "Present"
```

```
def mark_absent(self, student):
```

"""

Marks a student as absent in the attendance record.

Parameters:

student (str): The name of the student to be marked as absent.

"""

```
self.attendance_record[student] = "Absent"
```

def get_attendance(self, student):

"""

Retrieves the attendance status of a student.

Parameters:

student (str): The name of the student whose attendance status is to be retrieved.

Returns:

str: The attendance status of the student ("Present", "Absent", or "Not Recorded").

"""

```
return self.attendance_record.get(student, "Not Recorded")
```

Assert test cases to validate the functions' correctness

```
attendance = Attendance()
```

```
attendance.mark_present("Alice")
```

```
attendance.mark_absent("Bob")
```

```
assert attendance.get_attendance("Alice") == "Present", "Test case 1 failed"
```

```
assert attendance.get_attendance("Bob") == "Absent", "Test case 2 failed"
```

```
assert attendance.get_attendance("Charlie") == "Not Recorded", "Test case 3 failed"
```

```
print("All test cases passed!")
```

Output:

AAC-9-5

c:\users\rimsha naaz\onedrive\documents\aac-py\aac

#Task - 01:
#Generate a python function to reverse a string and include doc string documentation. Include assert test cases to validate the function's correctness.

Classes

[builtins.object](#)
[Attendance](#)

```
class Attendance(builtins.object):
    # mark_absent(student),get_attendance(student). Include docstring documentation in the function.
    # attendance.py and add assert test cases to validate the functions correctness.""""
    # attendance.py

    Methods defined here:

    __init__(self)
        Initialize self. See help(type(self)) for accurate signature.

    get_attendance(self, student)
        Retrieves the attendance status of a student.

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

        Returns:
        str: The attendance status of the student ("Present", "Absent", or "Not Recorded").

    mark_absent(self, student)
        Marks a student as absent in the attendance record.
```

Parameters:
student (str): The name of the student whose attendance status is to be retrieved.
Returns:
str: The attendance status of the student ("Present", "Absent", or "Not Recorded").

mark_absent(self, 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(self, student)
Marks a student as present in the attendance record.

Parameters:
student (str): The name of the student to be marked as present.

Data descriptors defined here:

__dict__
dictionary for instance variables (if defined)
__weakref__
list of weak references to the object (if defined)

Functions

```
check_strength(password)
    This function checks the strength of a password based on specific criteria.

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

    Returns:
        str: A message indicating whether the password is strong or weak.

factorial(n)
    This function takes a number as input and returns its factorial.
    Parameters:
        n (int): The number to calculate the factorial of.
    Returns:
        int: The factorial of the input number.

q(n)
    This function takes a number as input and returns its cube.
    Parameters:
        n (int or float): The number to be cubed.
    Returns:
        int or float: The cube of the input number.

reverse_string(s)
    This function takes a string as input and returns the reversed version of that string.

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

    Returns:
        str: The reversed string.

square(n)
    This function takes a number as input and returns its square.
    Parameters:
        n (int or float): The number to be squared.
```

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

Returns:
int or float: The square of the input number.

Data

attendance = <AAC-9-5.Attendance object>

Explanation:

This class stores student attendance using a dictionary. The mark_present and mark_absent methods update a student's status.

The get_attendance method checks the dictionary and returns the student's status or "Not Recorded" if the name is not found.

TASK – 05:

Question:

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.

Prompt:

Consider the function: def read_file(filename): with open(filename, 'r') as f: return f.read() and include docstring documentation in the function. Include assert test cases to validate the function's correctness. The function should read the contents of a file and return it as a string.

Generated Code:

```
def read_file(filename):  
    """
```

This function reads the contents of a file and returns it as a string.

Parameters:

filename (str): The name of the file to be read.

Returns:

str: The contents of the file as a string.

```
"""
```

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

```
    return f.read()
```

```
# Assert test cases to validate the function's correctness
```

```
# Note: For the assert test cases to work, you need to create a file named "
```

#test_file.txt" with some content in it. Here, we will assume that the file contains the text "Hello, World!".

```
assert read_file("test_file.txt") == "Hello, World!", "Test case 1 failed"  
assert read_file("non_existent_file.txt") == "", "Test case 2 failed"  
print("All test cases passed!")
```

Output:

```
Traceback (most recent call last):  
  File "c:\Users\RIMSHA NAAZ\OneDrive\Documents\AAC-PY\AAC-9-5.py", line 143, in <module>  
    assert read_file("test_file.txt") == "Hello, World!", "Test case 1 failed"  
AssertionError: Test case 1 failed  
PS C:\Users\RIMSHA NAAZ\OneDrive\Documents\AAC-PY>
```

Explanation:

The function opens and reads a file using with open.

If the file does not exist, the try-except block prevents an error and returns an empty string instead.