Assignment 8.1

Name : M.Varsha

Hall.No : 2403A51311

Batch : 13

Task :1   
• Task: Apply AI to generate at least 3 assert test cases for  
is\_strong\_password(password) and implement the validator  
function.  
• Requirements:  
o Password must have at least 8 characters.  
o Must include uppercase, lowercase, digit, and special  
character.  
o Must not contain spaces.  
Example Assert Test Cases:  
assert is\_strong\_password("Abcd@123") == True  
assert is\_strong\_password("abcd123") == False  
assert is\_strong\_password("ABCD@1234") == True  
Expected Output #1:  
• Password validation logic passing all AI-generated test cases.

Prompt :

write a python function that has 3 assert test cases for is\_strong\_password and implement the validator function.i Password must have at least 8 characters. must include uppercase, lowercase, digit, and special character.Must not contain spaces.with one example

CODE AND OUTPUT :

A screen shot of a computer

AI-generated content may be incorrect.

A screen shot of a computer

AI-generated content may be incorrect.

Task : 2, • Task: Use AI to generate at least 3 assert test cases for a  
classify\_number(n) function. Implement using loops.  
• Requirements:  
o Classify numbers as Positive, Negative or Zero.  
o Handle invalid inputs like strings and None.

o Include boundary conditions (-1, 0, 1).  
Example Assert Test Cases:  
assert classify\_number(10) == "Positive"  
assert classify\_number(-5) == "Negative"  
assert classify\_number(0) == "Zero"  
Expected Output #2:  
• Classification logic passing all assert tests.

PROMPT :

write a python of at least 3 assert test cases for a classify\_number(n) function. Implement using loops. Classify numbers as Positive, Negative, or Zero. Handle invalid inputs like strings and None.Include boundary conditions (-1, 0, 1).

OUTPUT AND CODE :



A screen shot of a computer

AI-generated content may be incorrect.

Task : 3

• Task: Use AI to generate at least 3 assert test cases for  
is\_anagram(str1, str2) and implement the function.  
• Requirements:  
o Ignore case, spaces, and punctuation.  
o Handle edge cases (empty strings, identical words).  
Example Assert Test Cases:  
assert is\_anagram("listen", "silent") == True  
assert is\_anagram("hello", "world") == False  
assert is\_anagram("Dormitory", "Dirty Room") == True  
Expected Output #3:  
• Function correctly identifying anagrams and passing all AI-  
generated tests.

PROMPT :

write a python fuction that has least 3 assert test cases for is\_anagram(str1, str2) and implement the function. Ignore case, spaces, and punctuation.Handle edge cases (empty strings, identical words)

CODE AND OUTPUT :

A screenshot of a computer program

AI-generated content may be incorrect.

A screen shot of a computer code

AI-generated content may be incorrect.

Task : 4

• Task: Ask AI to generate at least 3 assert-based tests for an  
Inventory class with stock management.  
• Methods:  
o add\_item(name, quantity)  
o remove\_item(name, quantity)  
o get\_stock(name)  
Example Assert Test Cases:  
inv = Inventory()  
inv.add\_item("Pen", 10)  
assert inv.get\_stock("Pen") == 10  
inv.remove\_item("Pen", 5)  
assert inv.get\_stock("Pen") == 5  
inv.add\_item("Book", 3)  
assert inv.get\_stock("Book") == 3  
Expected Output #4:  
• Fully functional class passing all assertions.

PROMPT :

write a python function that have at least 3 assert-based tests for an Inventory class with stock management.add\_item(name, quantity)remove\_item(name, quantity)get\_stock(name)

CODE AND OUTPUT :

A screenshot of a computer program

AI-generated content may be incorrect.

A computer screen shot of a program

AI-generated content may be incorrect.

A screen shot of a computer program

AI-generated content may be incorrect.

Task 5 :

• Task: Use AI to generate at least 3 assert test cases for  
validate\_and\_format\_date(date\_str) to check and convert dates.  
• Requirements:  
o Validate "MM/DD/YYYY" format.  
o Handle invalid dates.  
o Convert valid dates to "YYYY-MM-DD".  
Example Assert Test Cases:  
assert validate\_and\_format\_date("10/15/2023") == "2023-10-15"  
assert validate\_and\_format\_date("02/30/2023") == "Invalid Date"  
assert validate\_and\_format\_date("01/01/2024") == "2024-01-01"  
Expected Output #5:  
• Function passes all AI-generated assertions and handles edge  
cases.

PROMPT :

write a python function that has at least 3 assert test cases forvalidate\_and\_format\_date(date\_str) to check and convert dates.Validate "MM/DD/YYYY" format.Handle invalid dates.Convert valid dates to "YYYY-MM-DD"

CODE AND OUTPUT :

A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.