# **Assignment:8.1**

**Task 1** (Password Strength Validator – Apply AI in  
Security Context)  
• 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

# **Prompt:**

Write a Python function called is\_strong\_password(password) that checks if a password is strong based on these requirements:

* At least 8 characters long
* Includes at least one uppercase letter, one lowercase letter, one digit, and one special character
* Does not contain spaces

Then, generate at least 3 assert test cases to validate the function. Example:

# **Code:**

A screen shot of a computer program

AI-generated content may be incorrect.

# **Output:** **A black background with yellow text AI-generated content may be incorrect.**

# **Observation:**

This Python code defines a function is\_strong\_password that checks if a password meets security rules: minimum length of 8, no spaces, at least one uppercase letter, one lowercase letter, one digit, and one special character. It uses regular expressions for validation and includes test cases to confirm correctness.

**Task 2** (Number Classification with Loops – Apply AI for  
Edge Case Handling)  
• 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"

# **Prompt:**

Write a Python function called classify\_number(n) that classifies a number as "Positive", "Negative", or "Zero".

* The function should handle invalid inputs such as strings and None by returning "Invalid input".
* Implement the function using loops (if needed).
* Generate at least 3 assert test cases, including boundary conditions like -1, 0, and 1, and edge cases for invalid input.

# **Code:**

A screenshot of a computer program

AI-generated content may be incorrect.

# **Output:** **A screenshot of a computer program AI-generated content may be incorrect.**

# **Observation:**

This Python function classify\_number(n) checks whether a number is positive, negative, or zero. It uses conditional statements to return the result. The script includes assert test cases to verify correctness for integers, strings, and None, marking invalid inputs. Finally, it prints confirmation. The shown error arises from a missing closing quotation mark in the print statement.

**Task 3** (Anagram Checker – Apply AI for String Analysis)  
• 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

# **Prompt:**

Write a Python function called is\_anagram(str1, str2) that checks if two strings are anagrams.

* Ignore case, spaces, and punctuation.
* Handle edge cases such as empty strings and identical words.
* Generate at least 3 assert test cases, including examples with spaces, punctuation, and different cases.

# **Code:**

A screen shot of a computer program

AI-generated content may be incorrect.

# **Output:**

A black background with yellow text

AI-generated content may be incorrect.

# **Observation:**

This code defines [is\_anagram(str1, str2)](vscode-file://vscode-app/c:/Users/vandhana/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html" \o ") to check if two strings are anagrams, ignoring case, spaces, and punctuation. It cleans each string by removing non-alphanumeric characters and converting to lowercase, then compares their sorted characters. Assert test cases cover various scenarios, including spaces, punctuation, empty strings, and identical words. If all tests pass, it prints a confirmation message.

**Task 4** (Inventory Class – Apply AI to Simulate Real-  
World Inventory System)  
• 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

# **Prompt:**

Write a Python class called Inventory to simulate a real-world inventory system.

* The class should have methods: add\_item(name, quantity), remove\_item(name, quantity), and get\_stock(name).
* Generate at least 3 assert-based test cases to validate the class, including adding, removing, and checking stock for different items.

# **Code:**

A screen shot of a computer program

AI-generated content may be incorrect.

# **Output:** **A black background with yellow lines AI-generated content may be incorrect.**

# **Observation:**

This code defines an [Inventory](vscode-file://vscode-app/c:/Users/vandhana/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html) class for managing stock items. It includes methods to add, remove, and check item quantities. The class uses a dictionary to store item names and their quantities. Assert-based test cases verify adding, removing, and retrieving stock for different items. If all assertions pass, it prints a success message, confirming correct inventory management functionality.

**Task 5** (Date Validation & Formatting – Apply AI for  
Data Validation)  
• 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"

# Prompt:

Write a Python function called validate\_and\_format\_date(date\_str) that checks if a date string is valid and converts it to the format "YYYY-MM-DD".

* Validate that the input is in "MM/DD/YYYY" format.
* Handle invalid dates by returning "Invalid Date".
* Convert valid dates to "YYYY-MM-DD".
* Generate at least 3 assert test cases, including valid dates, invalid dates, and edge cases.

# Code:

A screen shot of a computer program

AI-generated content may be incorrect.

# **Output:** **A black background with yellow text AI-generated content may be incorrect.**

# **Observation:**

This code defines [validate\_and\_format\_date(date\_str)](vscode-file://vscode-app/c:/Users/vandhana/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html" \o ") to check if a date string is valid in "MM/DD/YYYY" format. If valid, it converts the date to "YYYY-MM-DD"; otherwise, it returns "Invalid Date". Several assert test cases verify correct handling of valid dates, invalid dates, and leap years. The function prints a success message if all tests pass.