# **AI ASSISTED CODING**

***ASSIGNMENT*-8.1**

NAME:S.VRINDHA REDDY

HALL NUMBER:2403A51255

BATCH:11

**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  
**Expected Output -1:**• Password validation logic passing all AI-generated test cases.

**Prompt:** Write a Python function is\_strong\_password(password) that checks:

* ≥8 chars, includes uppercase, lowercase, digit, special char, no spaces.  
  Add at least 3 assert test cases (valid + invalid). Ensure all pass.

**Code:**

A screen shot of a computer program

AI-generated content may be incorrect.

**Output:**

A screen shot of a computer

AI-generated content may be incorrect.

**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"  
**Expected Output -2:**• Classification logic passing all assert tests.

**Prompt:** Write classify\_number(n) in Python using loops: return "Positive", "Negative", or "Zero". Handle invalid inputs (e.g., string, None). Add ≥3 assert tests, including -1, 0, 1. Ensure all pass.

**Code:**

A screen shot of a computer program

AI-generated content may be incorrect.

**Output:**

A screen shot of a computer

AI-generated content may be incorrect.

**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  
Expected Output #3:  
• Function correctly identifying anagrams and passing all AI-  
generated tests

**Prompt:** Write is\_anagram (str1, str2) in Python (ignore case, spaces, punctuation). Handle empty strings & identical words. Add ≥3 assert tests (True & False cases). Ensure all pass.

**Code:**

A computer screen shot of text

AI-generated content may be incorrect.

**Output:**

A screen shot of a computer

AI-generated content may be incorrect.

**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  
**Expected Output -4:**• Fully functional class passing all assertions.

**Prompt:** Write a Python Inventory class with methods add\_item (name, quantity), remove\_item (name, quantity), and get\_stock(name). Use AI to generate ≥3 assert test cases (adding, removing, checking stock). Ensure the class passes all assertions.

**Code:**

A computer screen shot of a program code

AI-generated content may be incorrect.

A screen shot of a computer program

AI-generated content may be incorrect.

**Output:**

A black screen with white text

AI-generated content may be incorrect.

**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"  
**Expected Output -5:**• Function passes all AI-generated assertions and handles edge  
cases.

**Prompt**: Write validate\_and\_format\_date(date\_str) in Python:

* Check "MM/DD/YYYY" format
* Return "Invalid Date" if invalid
* Convert valid dates to "YYYY-MM-DD"

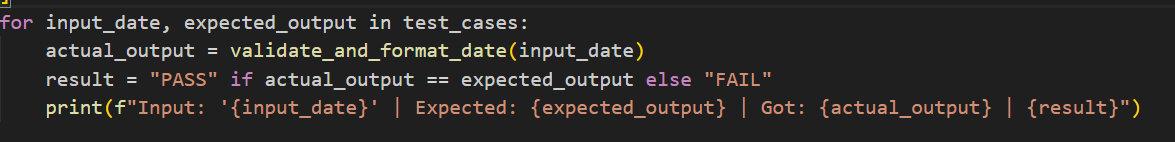
Add ≥3 assert tests (valid + invalid + edge).  
Print results as:

Input: '...' | Expected: ... | Got: ... | PASS/FAIL|

**Code:**

A screen shot of a computer program

AI-generated content may be incorrect.



**Output:**

A screen shot of a computer

AI-generated content may be incorrect.