AI Assisted Coding

Assignment-8.1

Roll Number:2303A51247

Batch:18

Lab 8: Test-Driven Development with AI – Generating and Working with Test Cases

Lab Objectives:

• To introduce students to test-driven development (TDD) using AI code generation tools.

• To enable the generation of test cases before writing code implementations.

• To reinforce the importance of testing, validation, and error handling.

• To encourage writing clean and reliable code based on AI- generated test expectations.

Lab Outcomes (LOs):

After completing this lab, students will be able to:

• Use AI tools to write test cases for Python functions and classes.

• Implement functions based on test cases in a test-first development style.

• Use unittest or pytest to validate code correctness.

• Analyze the completeness and coverage of AI-generated tests.

• Compare AI-generated and manually written test cases for quality and logic

Task Description #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.

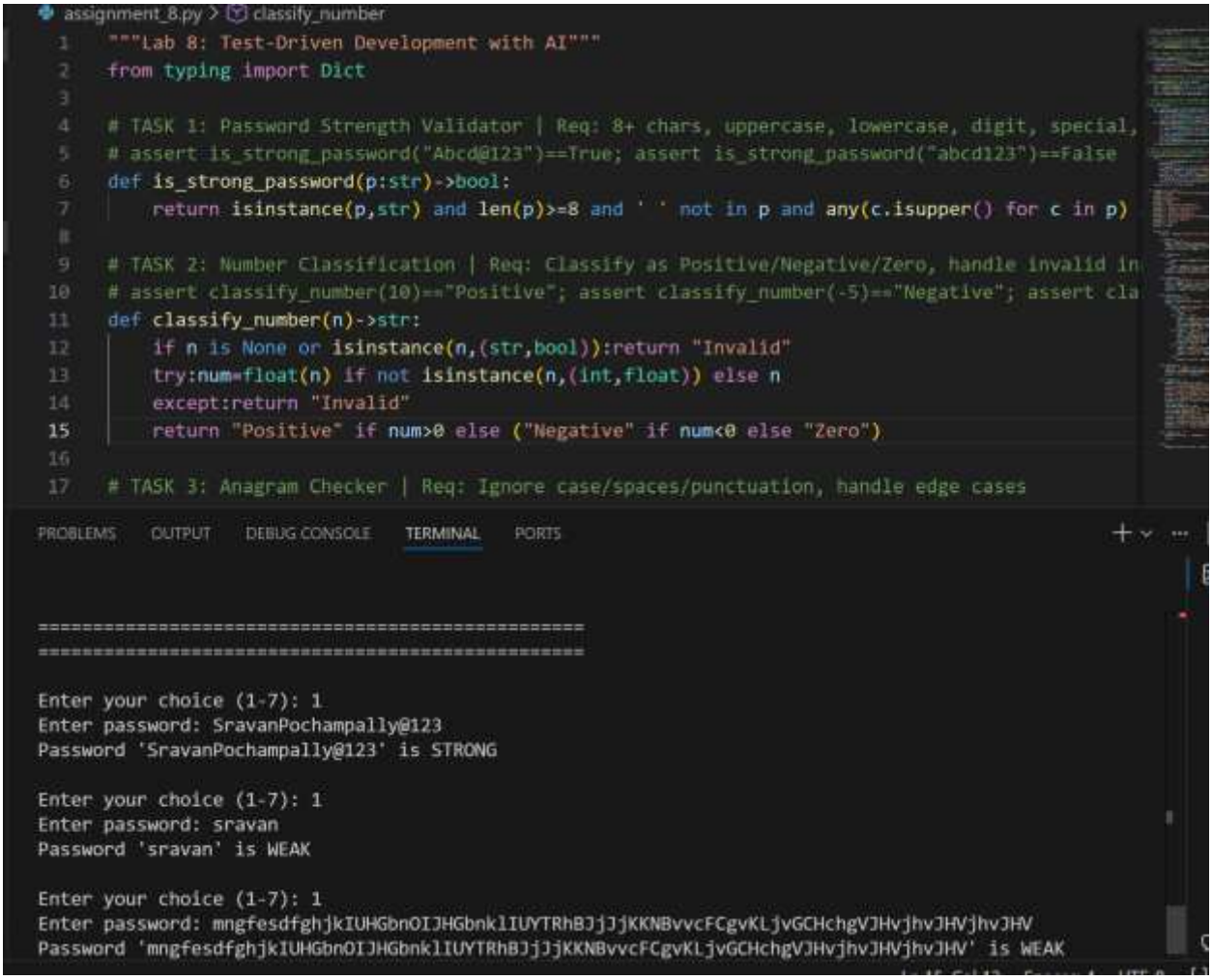
Example Assert Test Cases:

assert is_strong_password("Abcd@123") == True

assert is_strong_password("abcd123") == False

assert is_strong_password("ABCD@1234") == True

Code:



Expected Output #1:

• Password validation logic passing all AI-generated test cases.

Task Description #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"

Code:

assignment_8.py > classify_number

```
    return isinstance(p,str) and len(p)>=8 and ' ' not in p and any(c.isupper() for c i

# TASK 2: Number Classification | Req: Classify as Positive/Negative/Zero, handle inval
# assert classify_number(10)=="Positive"; assert classify_number(-5)=="Negative"; asser
def classify_number(n)->str:
    if n is None or isinstance(n,(str,bool)):return "Invalid"
    try:num=float(n) if not isinstance(n,(int,float)) else n
    except:return "Invalid"
    return "Positive" if num>0 else ("Negative" if num<0 else "Zero")

# TASK 3: Anagram Checker | Req: Ignore case/spaces/punctuation, handle edge cases
# assert is_anagram("listen","silent")==True; assert is_anagram("hello","world")==False
def is_anagram(s1:str,s2:str)->bool:
    if not isinstance(s1,str) or not isinstance(s2,str):return False
    c1=''.join(x.lower() for x in s1 if x.isalnum())
    c2=''.join(x.lower() for x in s2 if x.isalnum())
    return (len(c1)==0 and len(c2)==0) or (len(c1)>0 and len(c2)>0 and sorted(c1)==sort
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
Enter your choice (1-7): 2
Enter a number: 3456789
Number classification: Positive

Enter your choice (1-7): 2
Enter a number: -23456
Number classification: Negative

Enter your choice (1-7): 2
Enter a number: 00000000000
Number classification: Zero
```

Expected Output #2:

• Classification logic passing all assert tests.

Task Description #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).

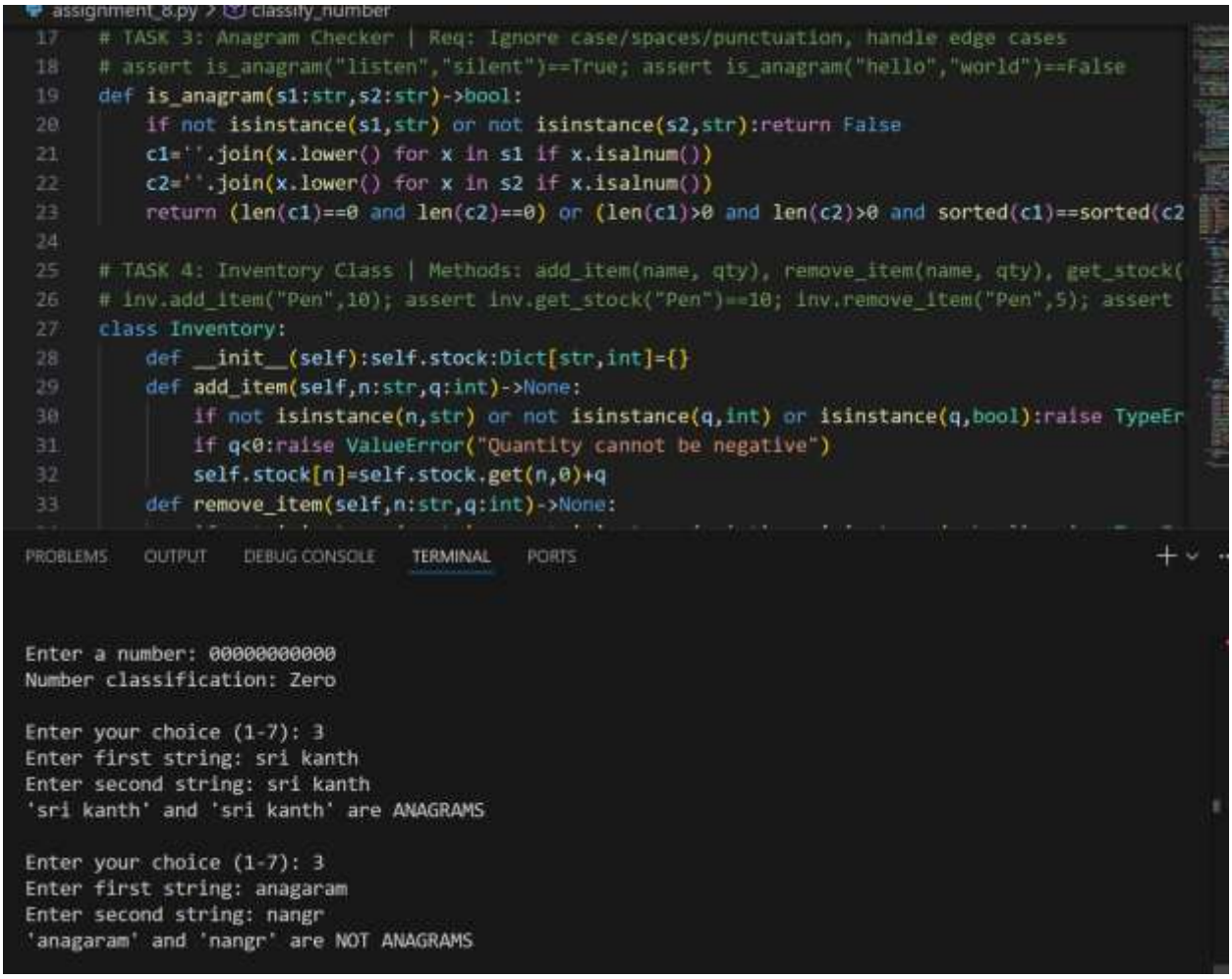
Example Assert Test Cases:

assert is_anagram("listen", "silent") == True

assert is_anagram("hello", "world") == False

assert is_anagram("Dormitory", "Dirty Room") == True

Code:



Expected Output #3:

• Function correctly identifying anagrams and passing all AI- generated tests.

Task Description #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

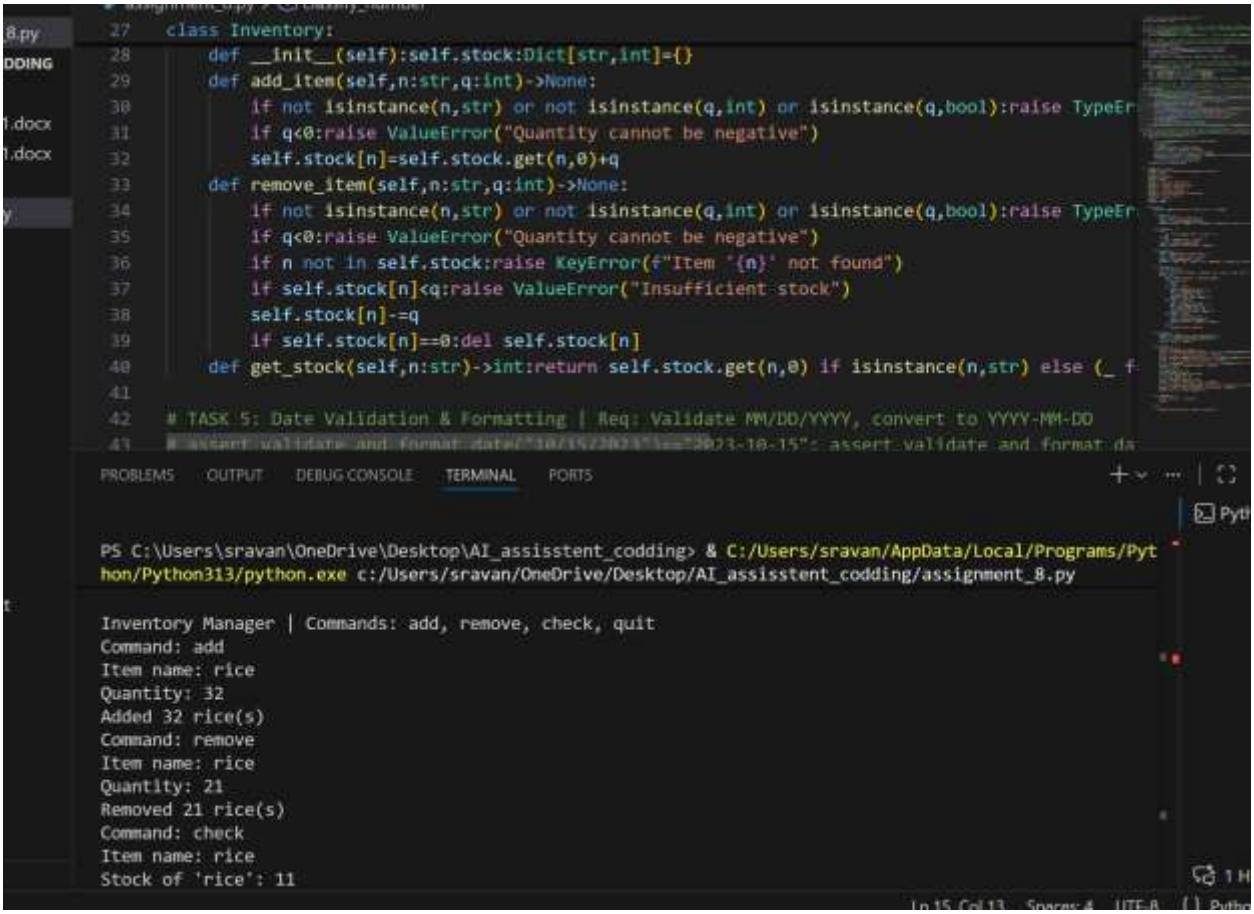
inv.remove_item("Pen", 5)

assert inv.get_stock("Pen") == 5

inv.add_item("Book", 3)

assert inv.get_stock("Book") == 3

Code:



Expected Output #4:

• Fully functional class passing all assertions.

Task Description #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".

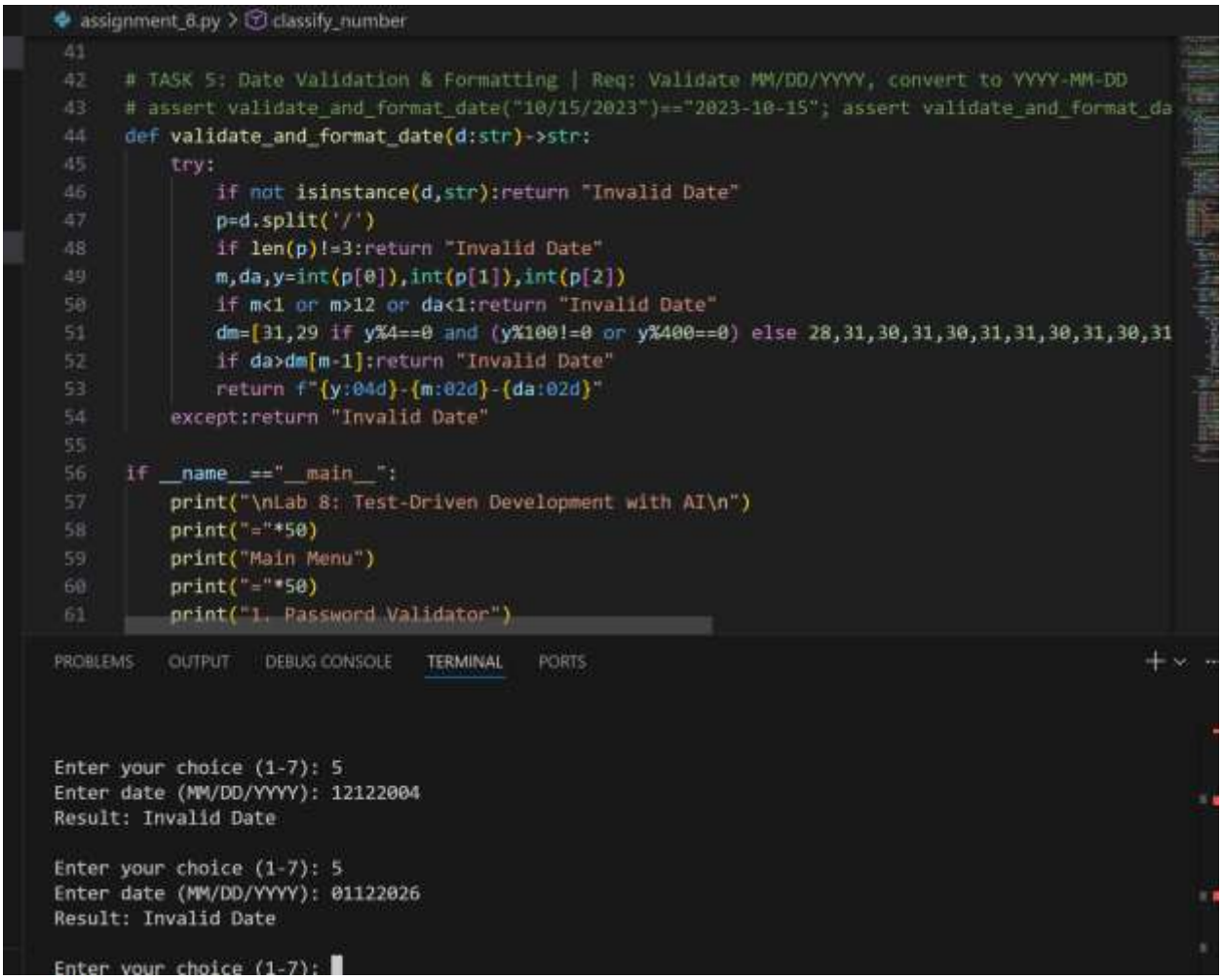
Example Assert Test Cases:

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"

Code:



Expected Output #5:

• Function passes all AI-generated assertions and handles edge cases.