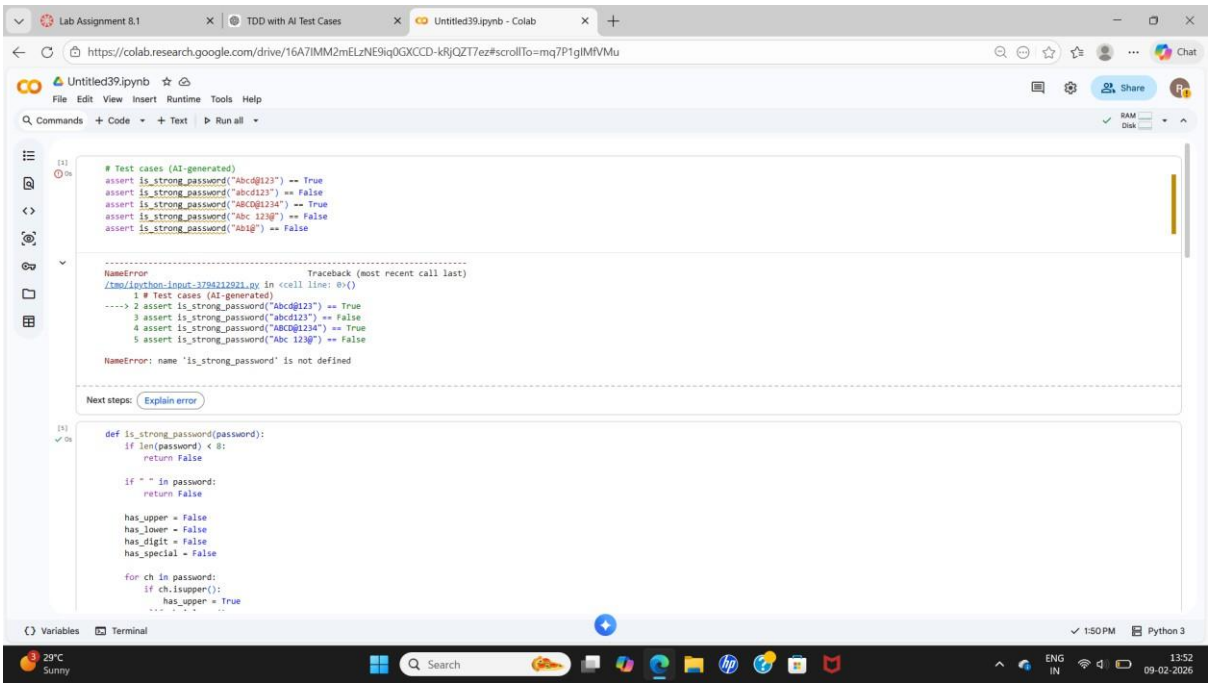
LAB ASSIGNMENT 8.1

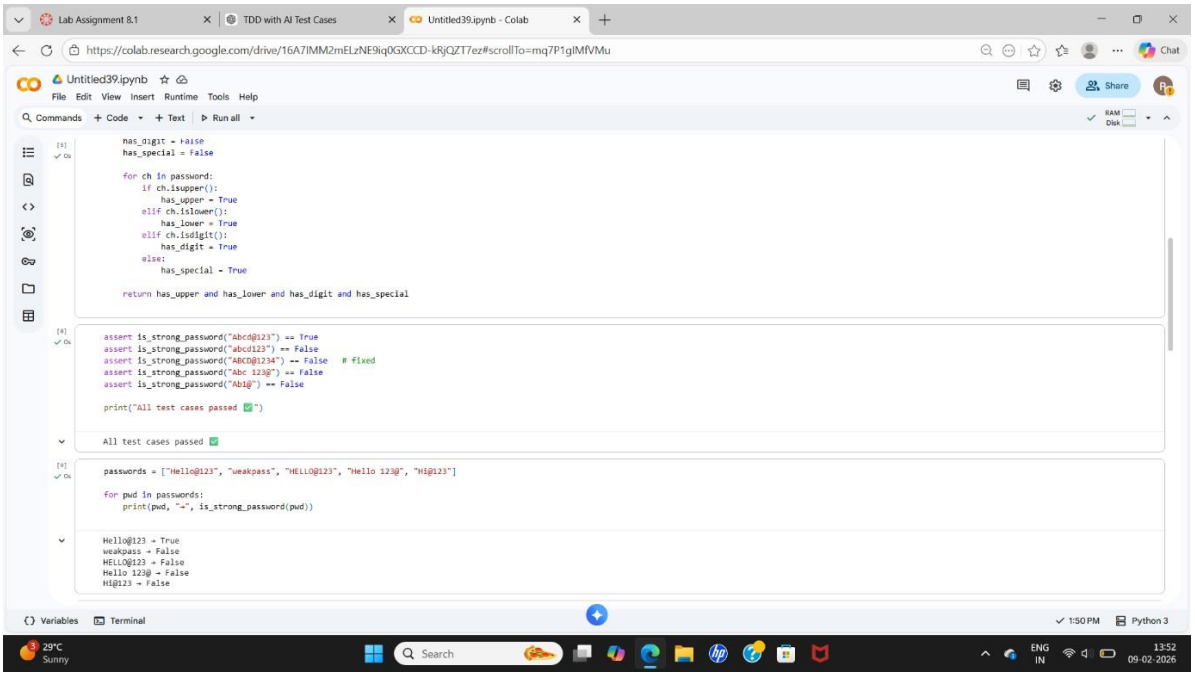
NAME: A.Harshita

ID.NO: 2303A52211

SUBJECT: AI ASST CODING

Task 1: Password Strength Validator





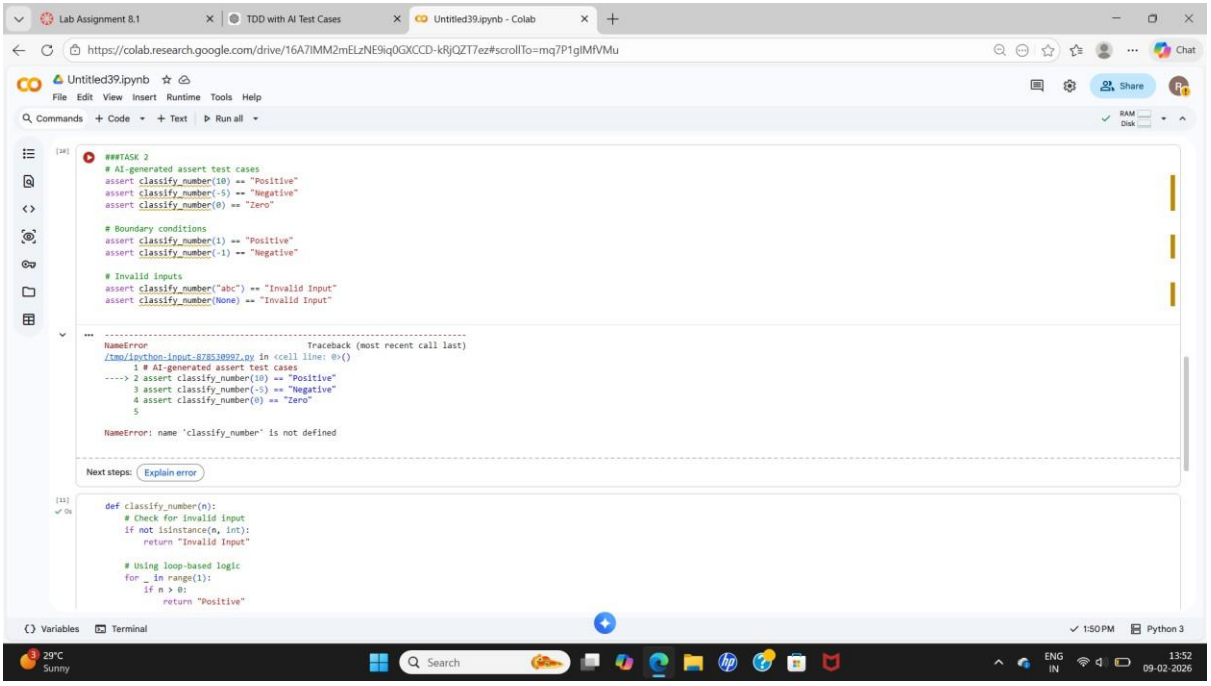
Code Explanation:

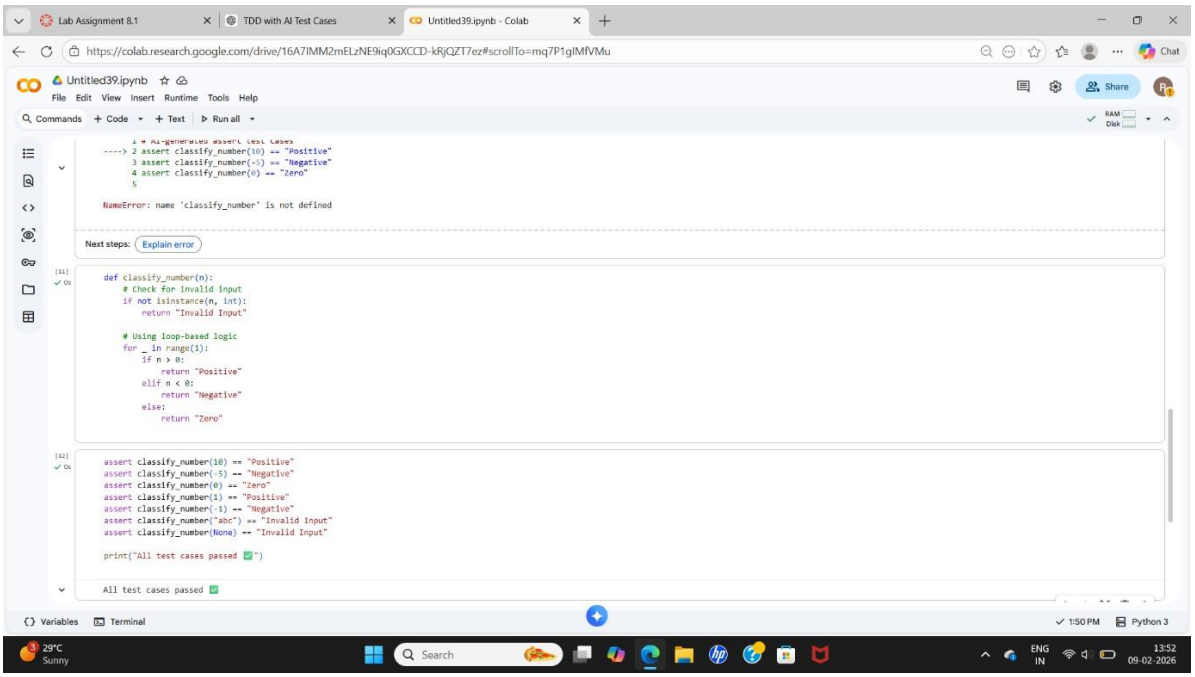
The function checks whether a password is strong by verifying length, presence of uppercase, lowercase, digit, special character, and absence of spaces. It returns True for strong passwords and False otherwise.

AI Explanation:

AI generated test cases for strong and weak passwords. These tests helped verify security rules and identify incorrect password patterns.

Task 2: Number Classification Using Loops





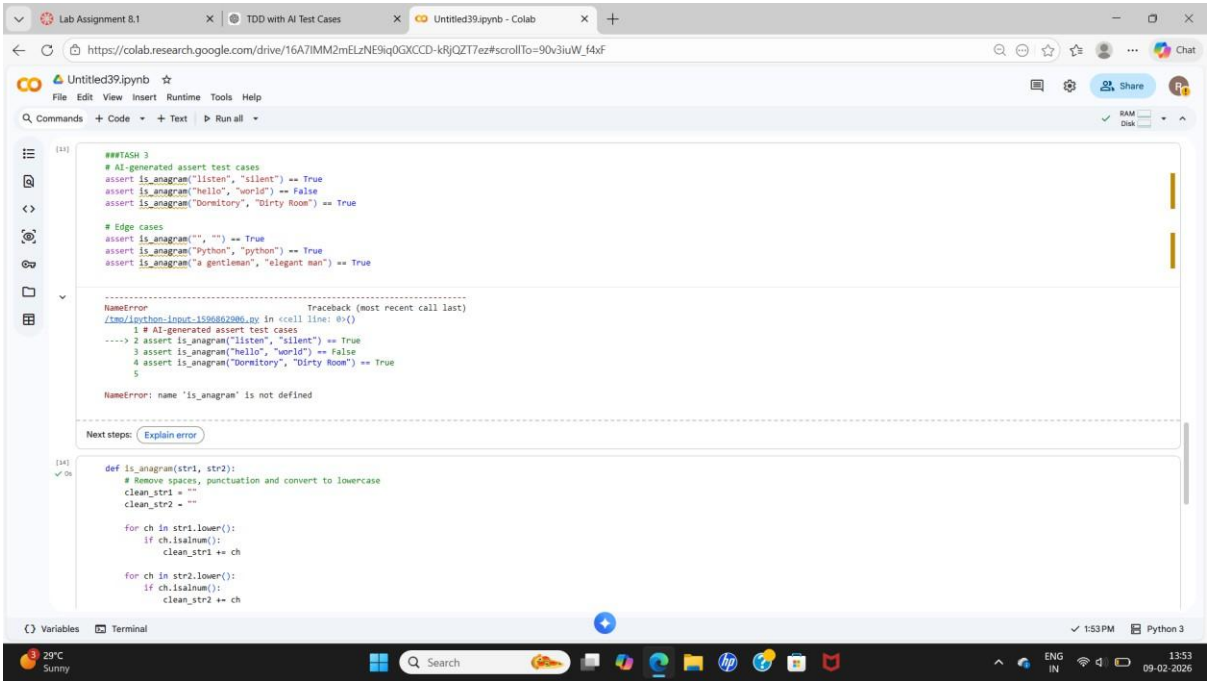
Code Explanation:

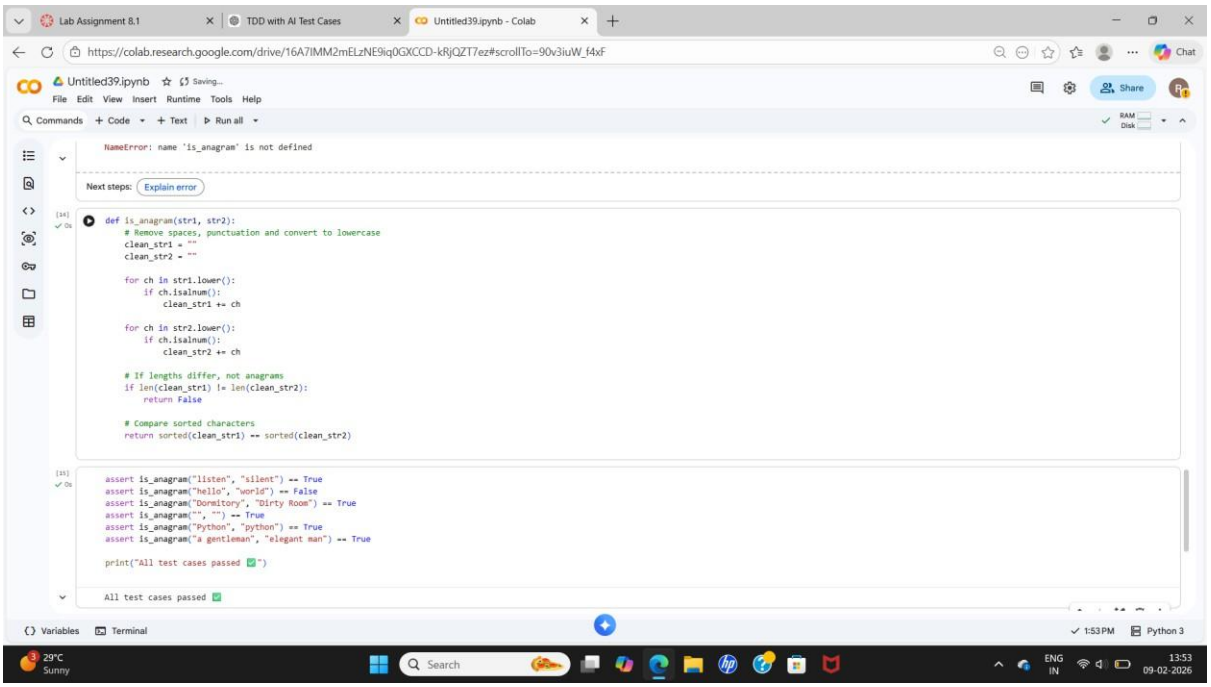
The function classifies a number as Positive, Negative, or Zero. It first checks for invalid inputs like strings or None and then uses logic inside a loop to return the correct classification.

AI Explanation:

AI generated test cases including boundary values (-1, 0, 1) and invalid inputs, ensuring correct classification.

Task 3: Anagram Checker





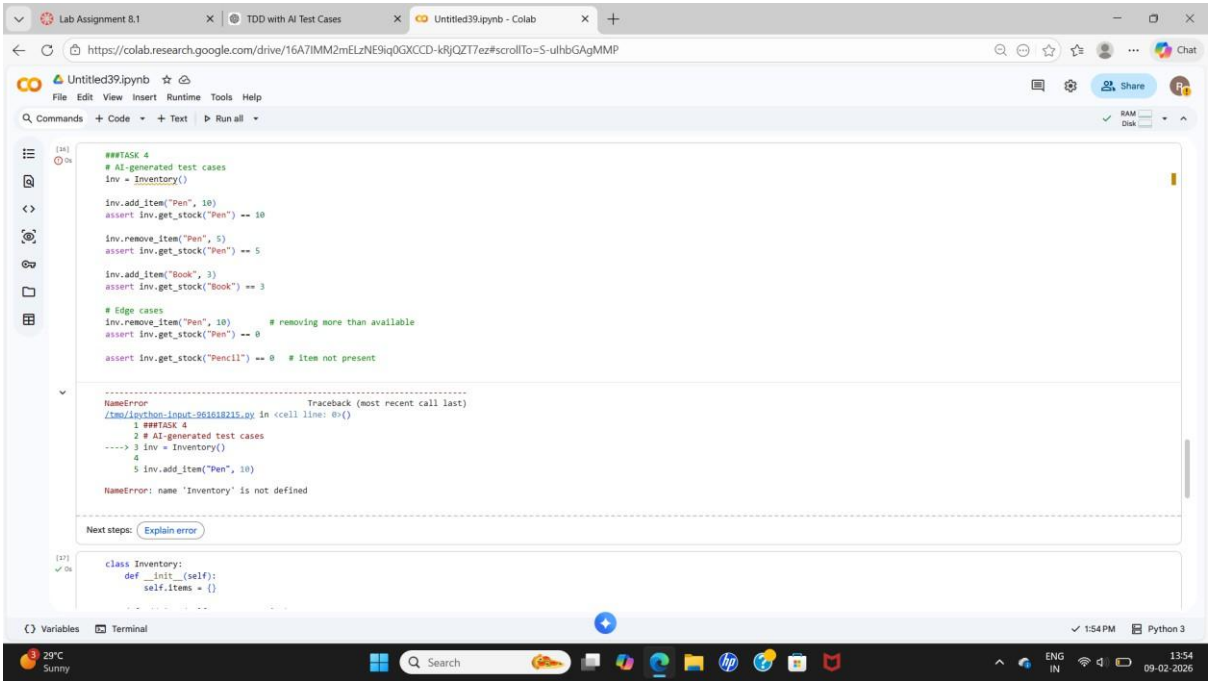
Code Explanation:

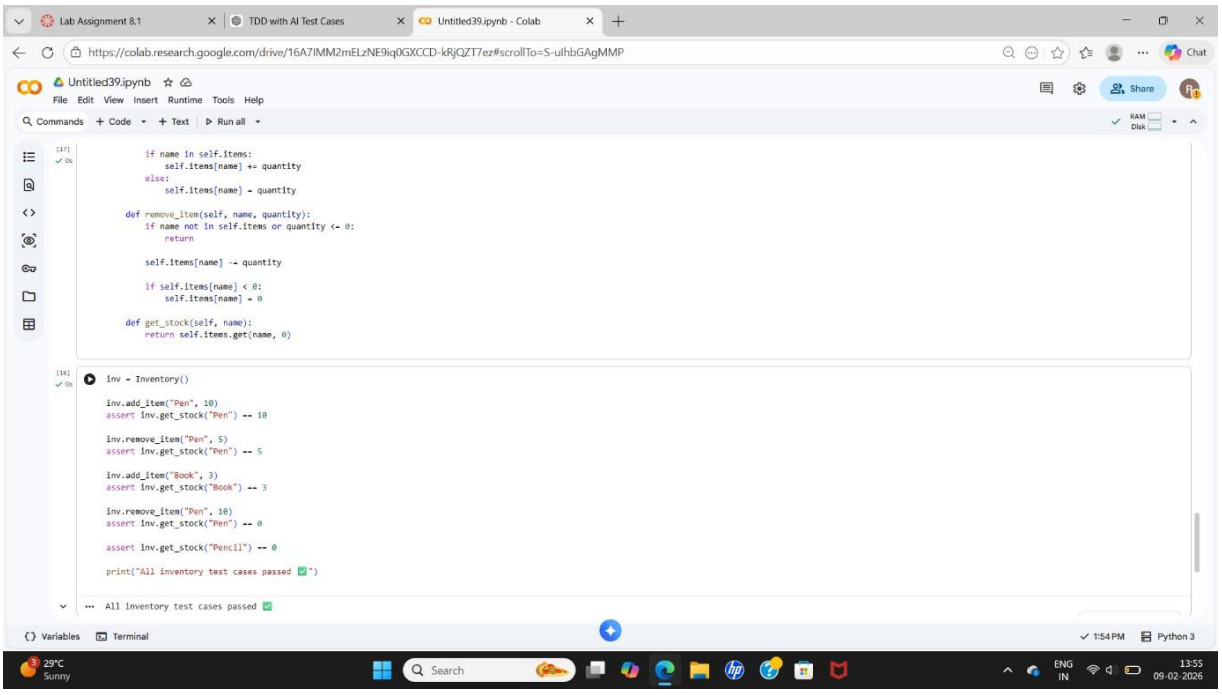
The function removes spaces and punctuation, converts strings to lowercase, and compares characters to check if two strings are anagrams.

AI Explanation:

AI-generated tests helped cover case differences, spaces, and empty strings for accurate string comparison.

Task 4: Inventory Class (Real-World Simulation)





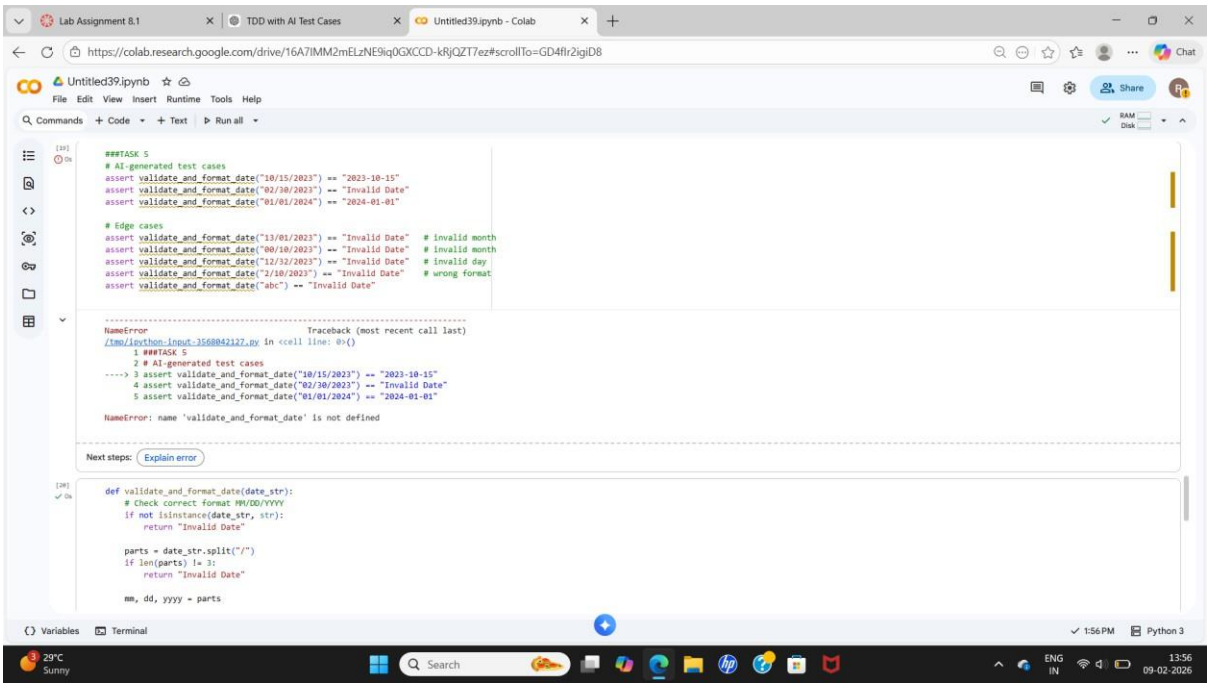
Code Explanation:

The Inventory class manages item stock. It allows adding items, removing items safely, and checking stock quantity.

AI Explanation:

AI provided test cases simulating real inventory actions like adding, removing, and checking items.

Task 5: Date Validation & Formatting

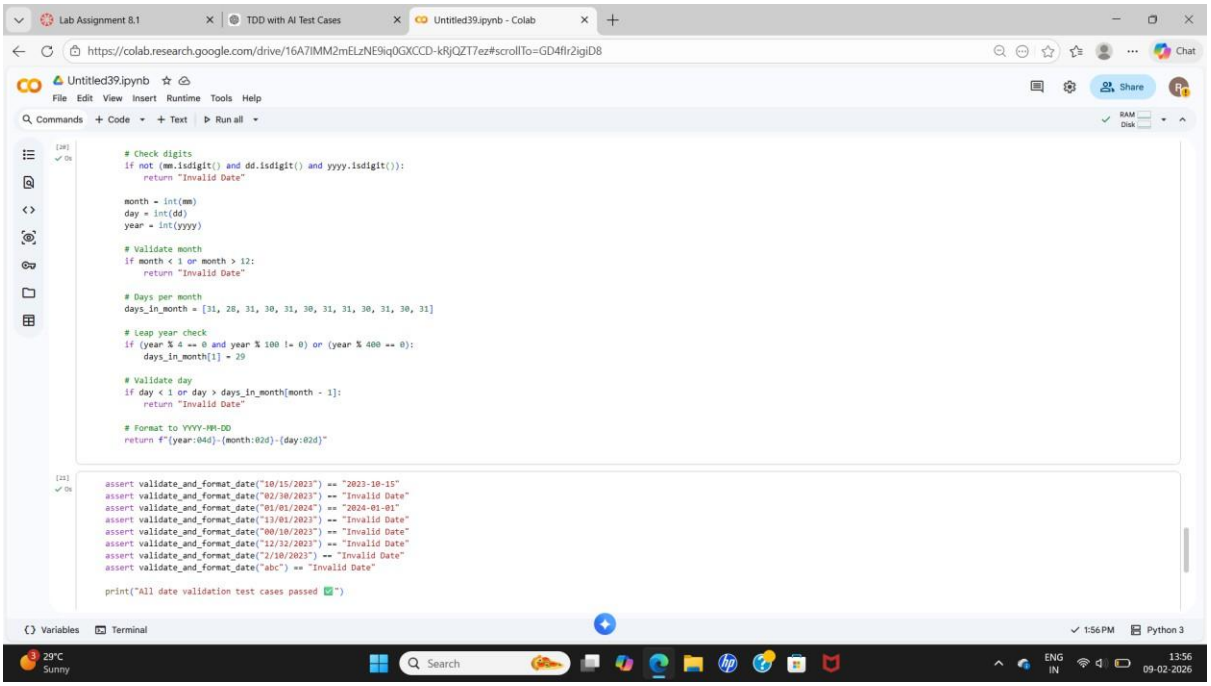


```
###TASK 5
# AI-generated test cases
assert validate_and_format_date("10/15/2023") == "2023-10-15"
assert validate_and_format_date("02/30/2023") == "Invalid Date"
assert validate_and_format_date("01/01/2024") == "2024-01-01"

# Edge cases
assert validate_and_format_date("13/01/2023") == "Invalid Date"   # invalid month
assert validate_and_format_date("00/10/2023") == "Invalid Date"   # invalid month
assert validate_and_format_date("12/32/2023") == "Invalid Date"   # invalid day
assert validate_and_format_date("2/10/2023") == "Invalid Date"    # wrong format
assert validate_and_format_date("abc") == "Invalid Date"
```

```
NameError                                 Traceback (most recent call last)
/tmp/ipython-input-3568042127.py in <cell line: 0>()
      1 ###TASK 5
      2 # AI-generated test cases
----> 3 assert validate_and_format_date("10/15/2023") == "2023-10-15"
      4 assert validate_and_format_date("02/30/2023") == "Invalid Date"
      5 assert validate_and_format_date("01/01/2024") == "2024-01-01"

NameError: name 'validate_and_format_date' is not defined
```

Next steps: Explain error

```
def validate_and_format_date(date_str):
    # Check correct format MM/DD/YYYY
    if not isinstance(date_str, str):
        return "Invalid Date"

    parts = date_str.split("/")
    if len(parts) != 3:
        return "Invalid Date"

    mm, dd, yyyy = parts
```



```
    # Check digits
    if not (mm.isdigit() and dd.isdigit() and yyyy.isdigit()):
        return "Invalid Date"

    month = int(mm)
    day = int(dd)
    year = int(yyyy)

    # Validate month
    if month < 1 or month > 12:
        return "Invalid Date"

    # Days per month
    days_in_month = [31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31]

    # Leap year check
    if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):
        days_in_month[1] = 29

    # Validate day
    if day < 1 or day > days_in_month[month - 1]:
        return "Invalid Date"

    # Format to YYYY-MM-DD
    return f"{year:04d}-{month:02d}-{day:02d}"
```

```
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"
assert validate_and_format_date("13/01/2023") == "Invalid Date"
assert validate_and_format_date("00/10/2023") == "Invalid Date"
assert validate_and_format_date("12/32/2023") == "Invalid Date"
assert validate_and_format_date("2/10/2023") == "Invalid Date"
assert validate_and_format_date("abc") == "Invalid Date"

print("All date validation test cases passed ✅")
```

Code Explanation:

The function checks whether a date is in MM/DD/YYYY format, validates the date, and converts valid dates to YYYY-MM-DD.

AI Explanation:

AI generated test cases for valid dates, invalid formats, and edge cases like leap years.