Assignment 8.1

Name: N.sidhartha

HT No: 2303A51728

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:

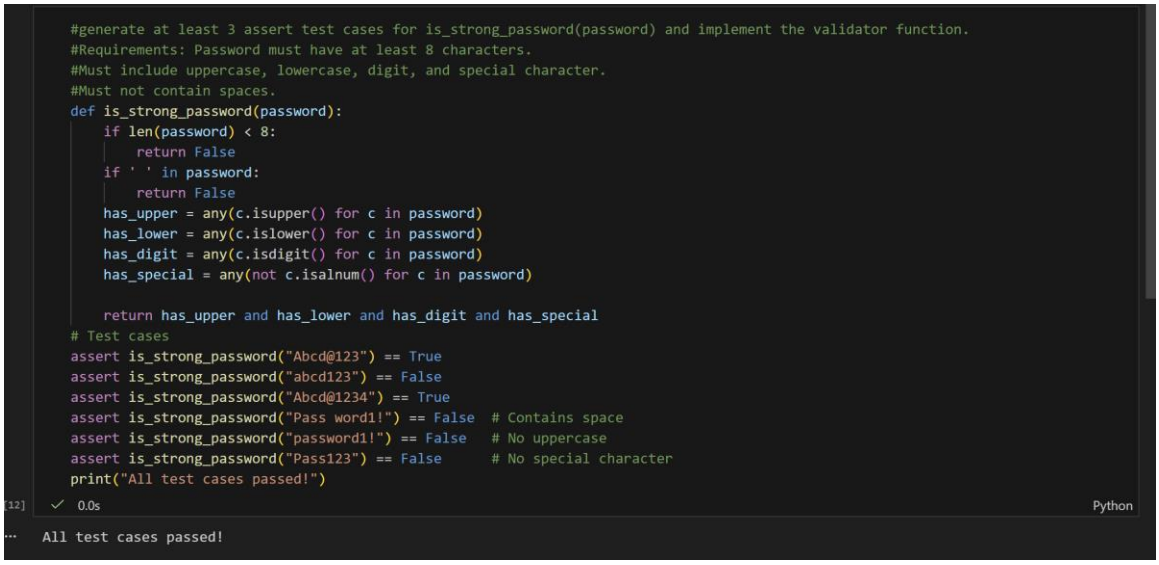
assert is_strong_password("Abcd@123") == True

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.



```python
#generate at least 3 assert test cases for is_strong_password(password) and implement the validator function.
#Requirements: Password must have at least 8 characters.
#Must include uppercase, lowercase, digit, and special character.
#Must not contain spaces.
def is_strong_password(password):
    if len(password) < 8:
        return False
    if ' ' in password:
        return False
    has_upper = any(c.isupper() for c in password)
    has_lower = any(c.islower() for c in password)
    has_digit = any(c.isdigit() for c in password)
    has_special = any(not c.isalnum() for c in password)

    return has_upper and has_lower and has_digit and has_special
# Test cases
assert is_strong_password("Abcd@123") == True
assert is_strong_password("abcd123") == False
assert is_strong_password("Abcd@1234") == True
assert is_strong_password("Pass word1!") == False  # Contains space
assert is_strong_password("password1!") == False   # No uppercase
assert is_strong_password("Pass123") == False      # No special character
print("All test cases passed!")
```

[12] ✓ 0.0s Python

```
All test cases passed!
```

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:

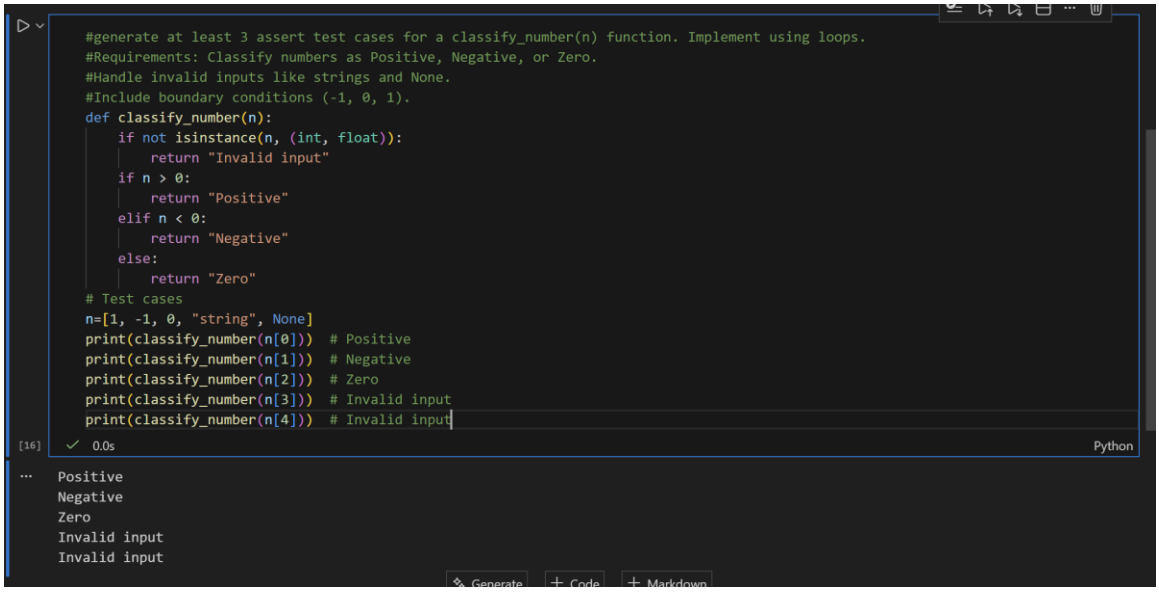
assert classify_number(10) == "Positive"

assert classify_number(-5) == "Negative"

assert classify_number(0) == "Zero"

Expected Output #2:

• Classification logic passing all assert tests.



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

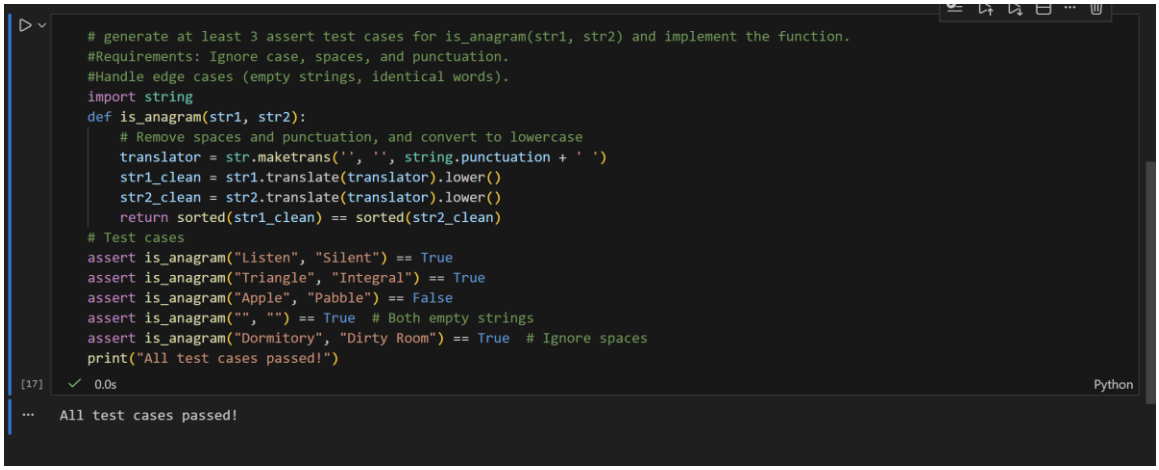
assert is_anagram("hello", "world") == False

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

Expected Output #3:

• Function correctly identifying anagrams and passing all AI-

generated tests.



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)

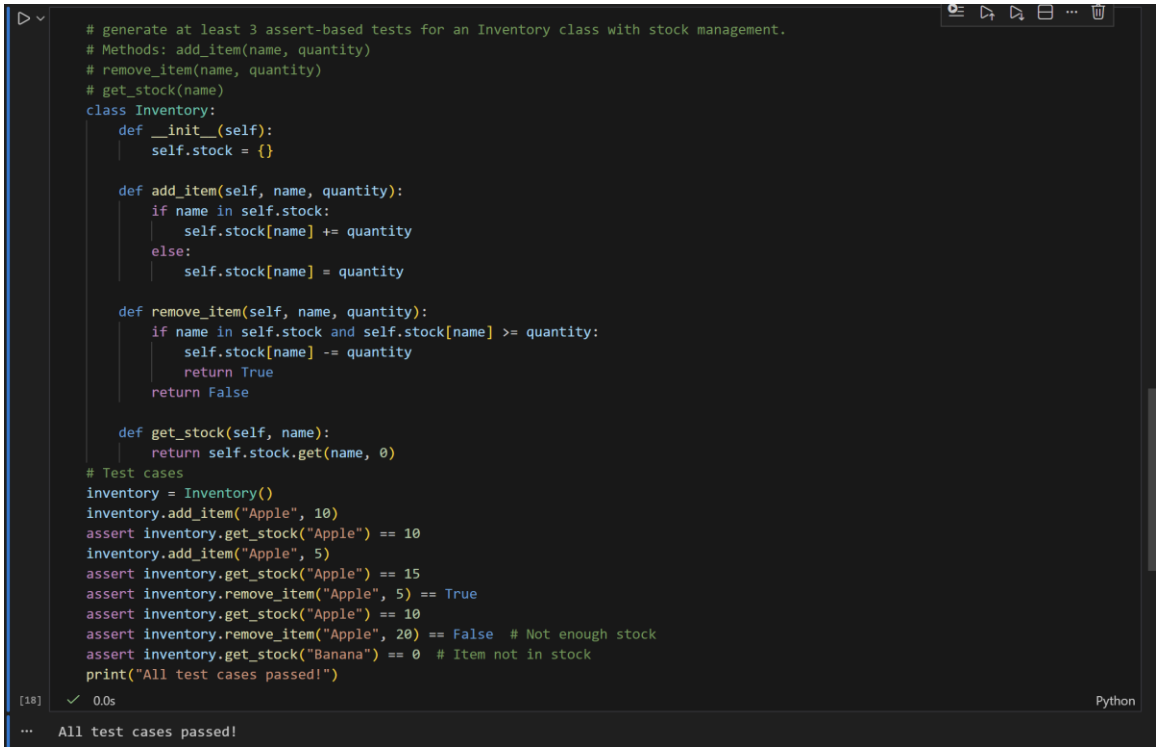
assert inv.get_stock("Pen") == 5

inv.add_item("Book", 3)

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

Expected Output #4:

• Fully functional class passing all assertions.



```python
# generate at least 3 assert-based tests for an Inventory class with stock management.
# Methods: add_item(name, quantity)
# remove_item(name, quantity)
# get_stock(name)
class Inventory:
    def __init__(self):
        self.stock = {}

    def add_item(self, name, quantity):
        if name in self.stock:
            self.stock[name] += quantity
        else:
            self.stock[name] = quantity

    def remove_item(self, name, quantity):
        if name in self.stock and self.stock[name] >= quantity:
            self.stock[name] -= quantity
            return True
        return False

    def get_stock(self, name):
        return self.stock.get(name, 0)
# Test cases
inventory = Inventory()
inventory.add_item("Apple", 10)
assert inventory.get_stock("Apple") == 10
inventory.add_item("Apple", 5)
assert inventory.get_stock("Apple") == 15
assert inventory.remove_item("Apple", 5) == True
assert inventory.get_stock("Apple") == 10
assert inventory.remove_item("Apple", 20) == False  # Not enough stock
assert inventory.get_stock("Banana") == 0  # Item not in stock
print("All test cases passed!")
```

[18] ✓ 0.0s Python

```
All test cases passed!
```

Task 5

(Date Validation & Formatting – Apply AI forData 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"

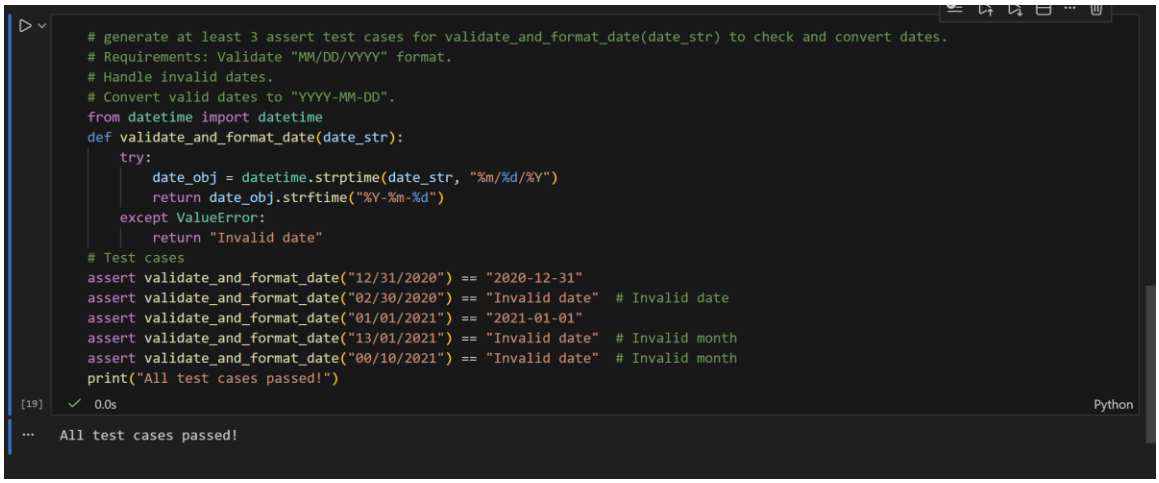
assert validate_and_format_date("02/30/2023") == "Invalid Date"

assert validate_and_format_date("01/01/2024") == "2024-01-01"

Expected Output #5:

• Function passes all AI-generated assertions and handles edge

cases.



```
# generate at least 3 assert test cases for validate_and_format_date(date_str) to check and convert dates.
# Requirements: Validate "MM/DD/YYYY" format.
# Handle invalid dates.
# Convert valid dates to "YYYY-MM-DD".
from datetime import datetime
def validate_and_format_date(date_str):
    try:
        date_obj = datetime.strptime(date_str, "%m/%d/%Y")
        return date_obj.strftime("%Y-%m-%d")
    except ValueError:
        return "Invalid date"
# Test cases
assert validate_and_format_date("12/31/2020") == "2020-12-31"
assert validate_and_format_date("02/30/2020") == "Invalid date"  # Invalid date
assert validate_and_format_date("01/01/2021") == "2021-01-01"
assert validate_and_format_date("13/01/2021") == "Invalid date"  # Invalid month
assert validate_and_format_date("00/10/2021") == "Invalid date"  # Invalid month
print("All test cases passed!")
```

[19] ✓ 0.0s Python

```
All test cases passed!
```