Assignment-8.5

2303A51272

A. Srivani

Batch-23

Task Description #1 (Username Validator – Apply AI in Authentication Context)

- Task: Use AI to generate at least 3 assert test cases for a function is_valid_username(username) and then implement the function using Test-Driven Development principles.
- Requirements:
  - Username length must be between 5 and 15 characters.
  - Must contain only alphabets and digits. o Must not start with a digit.
  - No spaces allowed. Example Assert Test Cases:

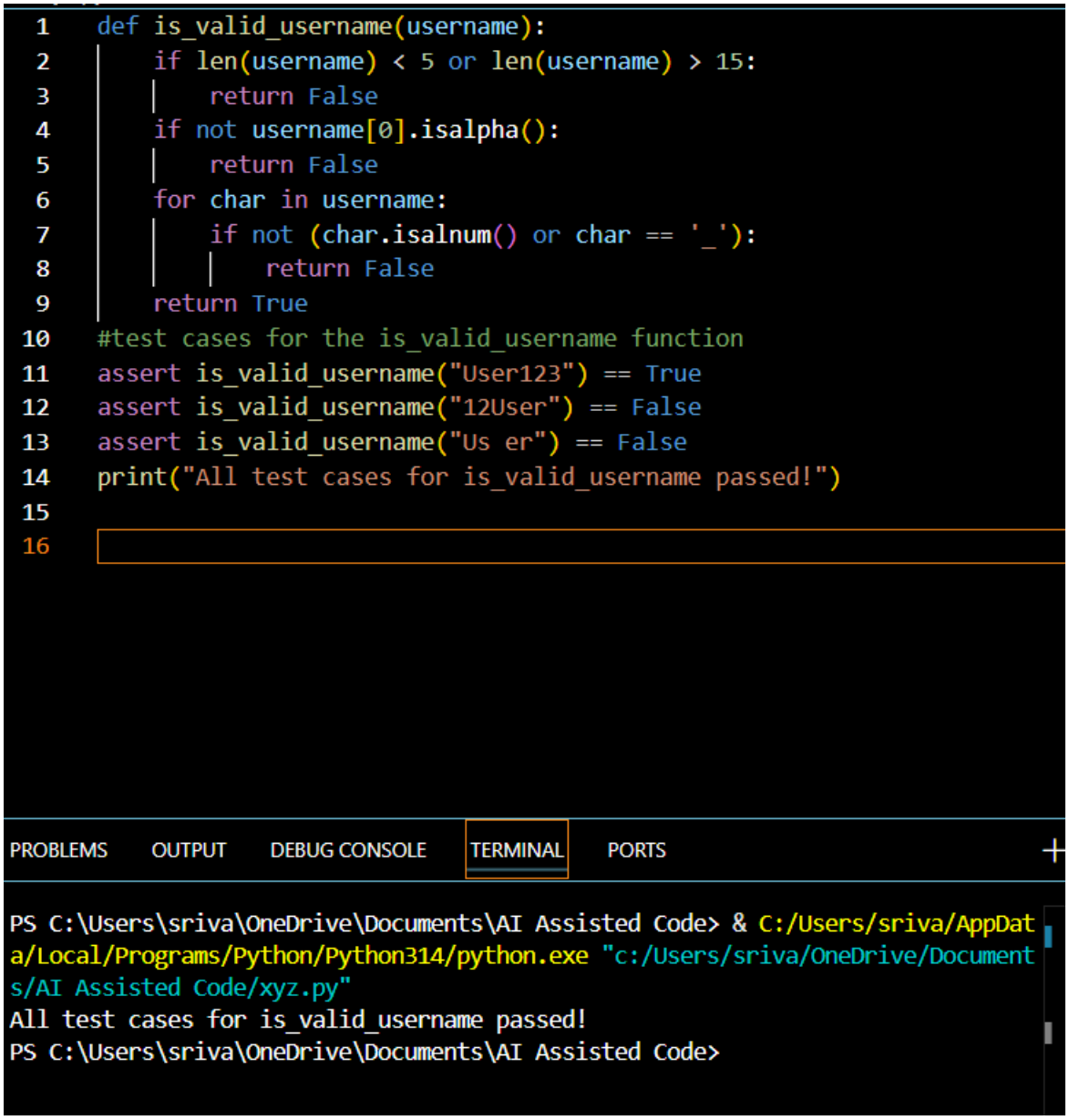
assert is_valid_username("User123") == True

assert is_valid_username("12User") == False assert

is_valid_username("Us er") == False

Expected Output #1:

- Username validation logic successfully passing all AI-generated test cases.



Task Description #2 (Even–Odd & Type Classification – Apply AI for Robust Input Handling)

- Task: Use AI to generate at least 3 assert test cases for a function classify_value(x) and implement it using conditional logic and loops.
- Requirements:

o If input is an integer, classify as "Even" or "Odd".

o If input is 0, return "Zero".

o If input is non-numeric, return "Invalid Input".

Example Assert Test Cases:

assert classify_value(8) == "Even" assert

classify_value(7) == "Odd" assert

classify_value("abc") == "Invalid Input"

Expected Output #2:

• Function correctly classifying values and passing all test cases

```python
def classify_value(x):
    if x < 0:
        return "Negative"
    elif x == 0:
        return "Zero"
    elif x%2 == 0:
        return "Even"
    else:
        return "Odd"
# Test cases for the classify_value function
assert classify_value(8) == "Even"
assert classify_value(7) == "Odd"
assert classify_value("abc") == "Invalid Input"

```

```
ROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS   Python

/AI Assisted Code/xyz.py"
raceback (most recent call last):
 File "c:\Users\sriva\OneDrive\Documents\AI Assisted Code\xyz.py", line 13, in <module>
   assert classify_value("abc") == "Invalid Input"
          ~~~~~~~~~~~~~~~~~~~~^^^^^^^^^^
 File "c:\Users\sriva\OneDrive\Documents\AI Assisted Code\xyz.py", line 2, in classify_value
   if x < 0:
      ^^^^^
ypeError: '<' not supported between instances of 'str' and 'int'
S C:\Users\sriva\OneDrive\Documents\AI Assisted Code>
```

Task Description #3 (Palindrome Checker – Apply AI for String Normalization)

• Task: Use AI to generate at least 3 assert test cases for a function is_palindrome(text) and implement the function.

• Requirements:

o Ignore case, spaces, and punctuation.

o Handle edge cases such as empty strings and single characters.

Example Assert Test Cases:

assert is_palindrome("Madam") == True assert

is_palindrome("A man a plan a canal Panama") == True

assert is_palindrome("Python") == False

Expected Output #3:

- Function correctly identifying palindromes and passing all AI-generated tests

```
1  def is_palindrome(text):
2      cleaned_text = ''.join(char.lower() for char in text if char.isalnum()
3      return cleaned_text == cleaned_text[::-1]
4  # Test cases for the is_palindrome function
5  assert is_palindrome("Madam") == True
6  assert is_palindrome("A man a plan a canal Panama") ==True
7  assert is_palindrome("Python") == False
8  print("All test cases for is_palindrome passed!")
9
```

```
ROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS   Python
S C:\Users\sriva\OneDrive\Documents\AI Assisted Code> & C:/Users/sriva/AppData/Local/Programs
ython/Python314/python.exe "c:/Users/sriva/OneDrive/Documents/AI Assisted Code/xyz.py"
ll test cases for is_palindrome passed!
S C:\Users\sriva\OneDrive\Documents\AI Assisted Code>
```

Task Description #4 (BankAccount Class – Apply AI for Object-Oriented Test-Driven Development)

- Task: Ask AI to generate at least 3 assert-based test cases for a BankAccount class and then implement the class.
- Methods:

o deposit(amount) o

withdraw(amount) o

get_balance()

Example Assert Test Cases: acc

= BankAccount(1000)

acc.deposit(500) assert

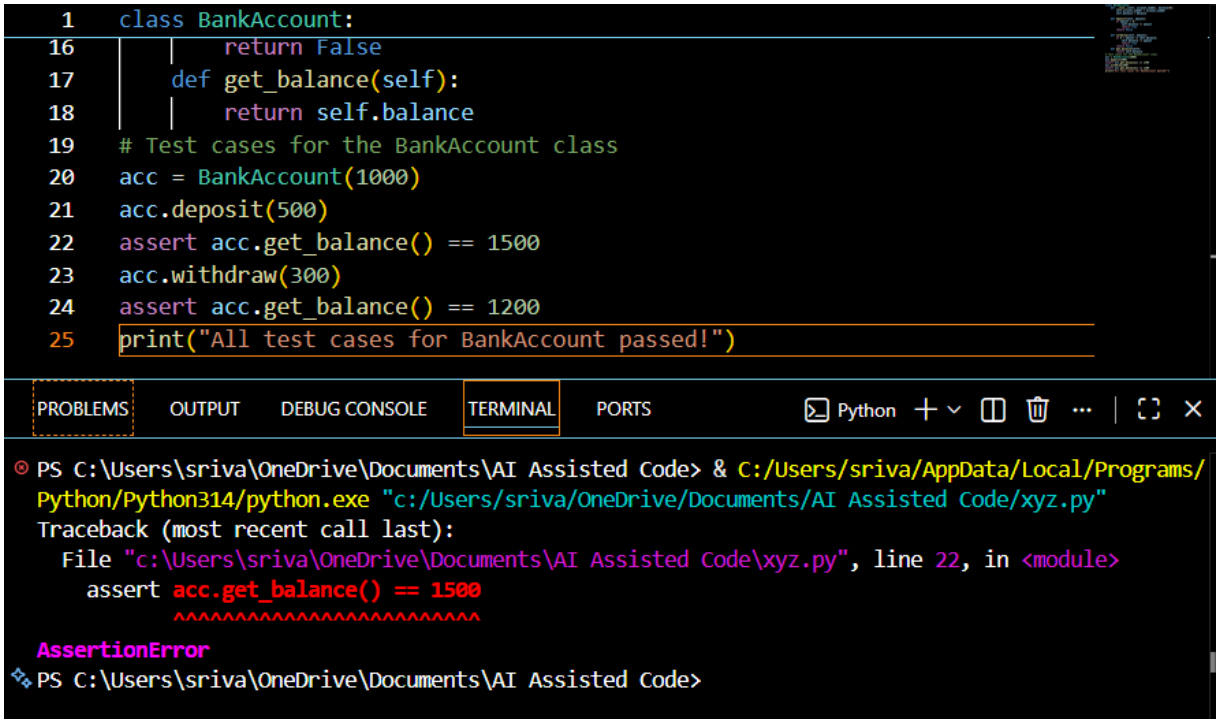
acc.get_balance() == 1500

acc.withdraw(300) assert

acc.get_balance() == 1200

Expected Output #4:

- Fully functional class that passes all AI-generated assertions.



Task Description #5 (Email ID Validation – Apply AI for Data Validation)

- Task: Use AI to generate at least 3 assert test cases for a function validate_email(email) and implement the function.
- Requirements:

o Must contain @ and . o Must not start or end

with special characters. o Should handle invalid

formats gracefully.

Example Assert Test Cases:

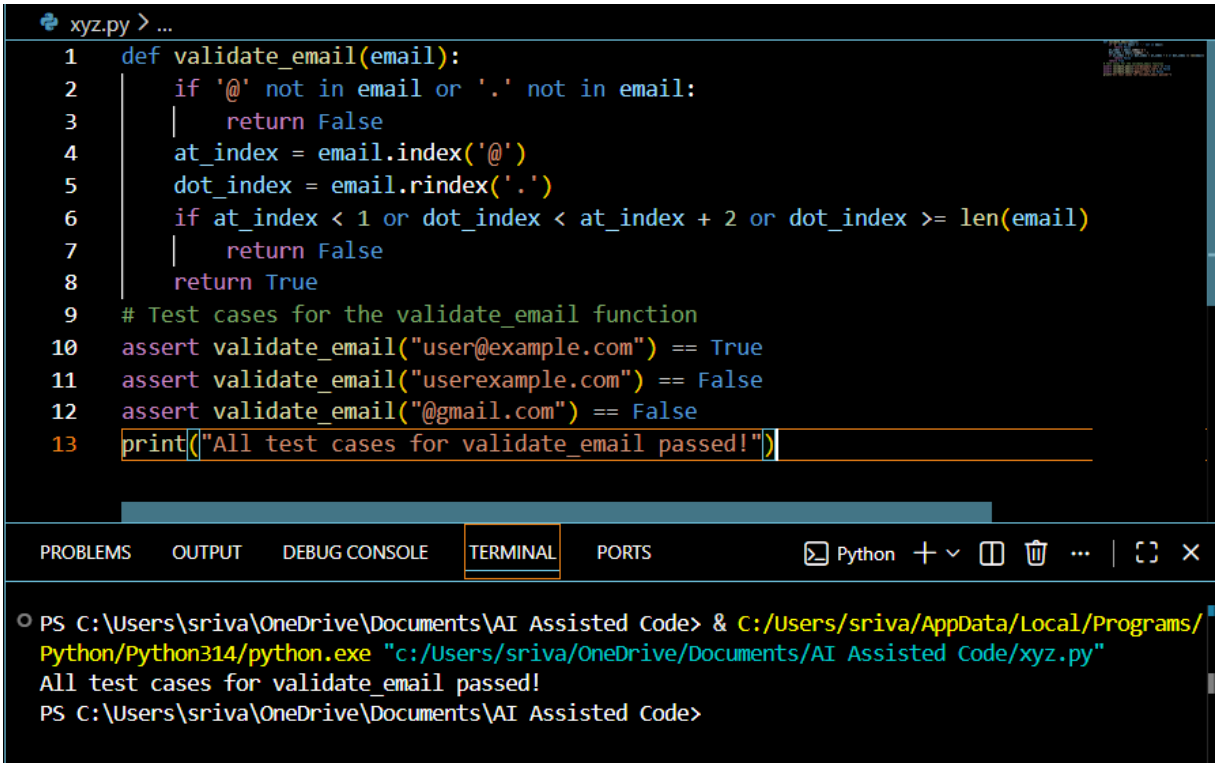
assert validate_email("user@example.com") == True

assert validate_email("userexample.com") == False assert

validate_email("@gmail.com") == False

Expected Output #5:

• Email validation function passing all AI-generated test cases and handling edge cases correctly.


xyz.py > ...
1 def validate_email(email):
2 if '@' not in email or '.' not in email:
3 return False
4 at_index = email.index('@')
5 dot_index = email.rindex('.')
6 if at_index < 1 or dot_index < at_index + 2 or dot_index >= len(email)
7 return False
8 return True
9 # Test cases for the validate_email function
10 assert validate_email("user@example.com") == True
11 assert validate_email("userexample.com") == False
12 assert validate_email("@gmail.com") == False
13 print("All test cases for validate_email passed!")
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
Python
PS C:\Users\sriva\OneDrive\Documents\AI Assisted Code> & C:/Users/sriva/AppData/Local/Programs/Python/Python314/python.exe "c:/Users/sriva/OneDrive/Documents/AI Assisted Code/xyz.py"
All test cases for validate_email passed!
PS C:\Users\sriva\OneDrive\Documents\AI Assisted Code>