Assignment-8.5

Ht. No: 2303A51923

Name: V.Sravani

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:

o Username length must be between 5 and 15 characters.

o Must contain only alphabets and digits.

o Must not start with a digit.

o No spaces allowed.

Example Assert Test Cases:

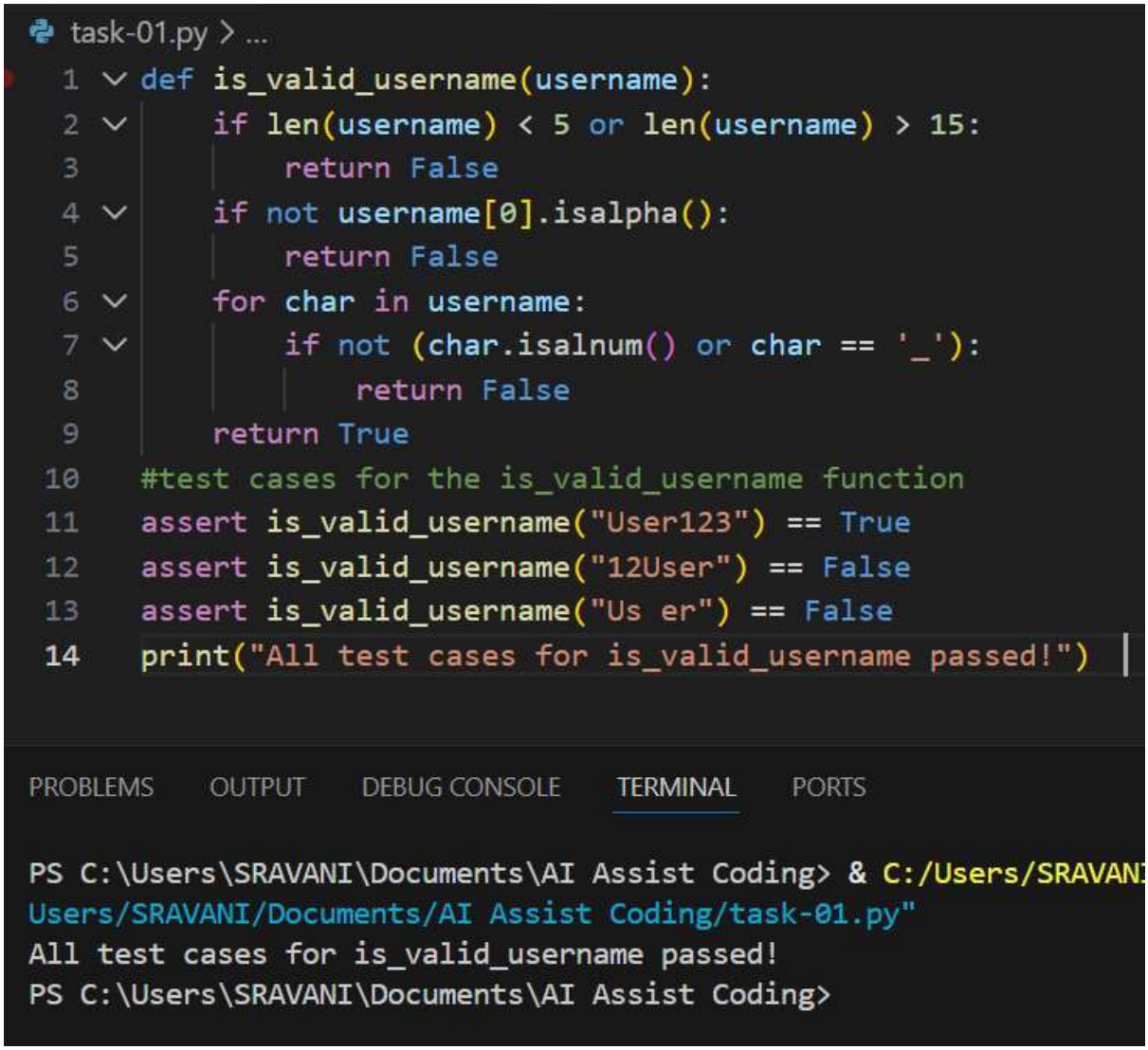
assert is_valid_username("User123") == True

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"
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\SRAVANI\Documents\AI Assist Coding> & C:/Users/SRAVANI/AppDa
Users/SRAVANI/Documents/AI Assist Coding/task-01.py"
Traceback (most recent call last):
  File "c:\Users\SRAVANI\Documents\AI Assist Coding\task-01.py", line 13
    assert classify_value("abc") == "Invalid Input"
           ~~~~~~~~~~~~~~~~~~~^^^^^^^
  File "c:\Users\SRAVANI\Documents\AI Assist Coding\task-01.py", line 2,
    if x < 0:
       ^^^^^
TypeError: '<' not supported between instances of 'str' and 'int'
```

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") ==

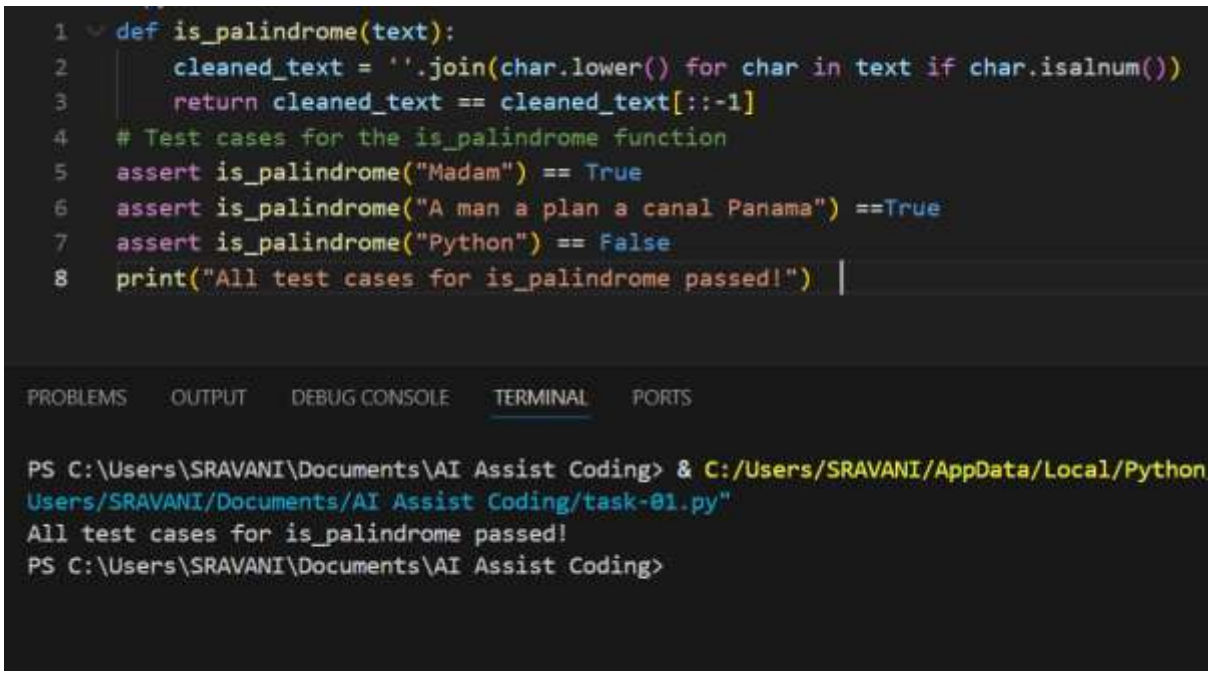
True

assert is_palindrome("Python") == False

Expected Output #3:

• Function correctly identifying palindromes and passing all

AI-generated tests.



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

```python
class BankAccount:
    def __init__(self, account_number, balance=0):
        self.account_number = account_number
        self.balance = balance

    def deposit(self, amount):
        if amount > 0:
            self.balance += amount
            return True
        return False

    def withdraw(self, amount):
        if 0 < amount <= self.balance:
            self.balance -= amount
            return True
        return False
    def get_balance(self):
        return self.balance
# Test cases for the BankAccount class
acc = BankAccount(1000)
acc.deposit(500)
assert acc.get_balance() == 1500
acc.withdraw(300)
assert acc.get_balance() == 1200
print("All test cases for BankAccount passed!")
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

Traceback (most recent call last):
  File "c:\Users\SRAVANI\Documents\AI Assist Coding\task-01.py", line 22, in <module>
    assert acc.get_balance() == 1500
           ^^^^^^^^^^^^^^^^^^^^^^^^^
AssertionError
PS C:\Users\SRAVANI\Documents\AI Assist Coding>
```

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.