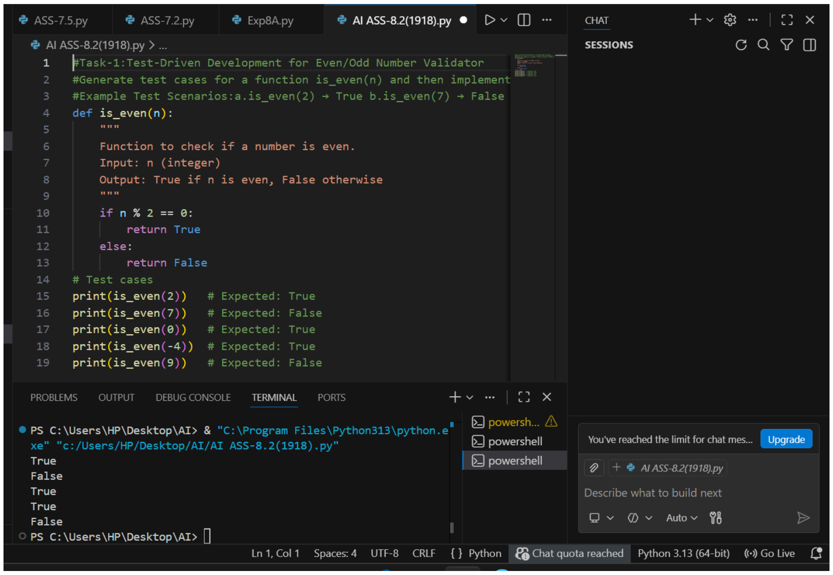
# AI Assisted Coding Assignment- 8.2

RUDROJU RUPA SRI

2303A51918

BATCH-30

Task 1 – Test-Driven Development for Even/Odd Number Validator
• Use AI tools to first generate test cases for a function is_even(n) and then implement the function so that it satisfies all generated tests.
Requirements:
• Input must be an integer
• Handle zero, negative numbers, and large integers
Example Test Scenarios:
is_even(2) → True
is_even(7) → False
is_even(0) → True
is_even(-4) → True
is_even(9) → False
Expected Output -1
• A correctly implemented is_even() function that passes all AI-generated test cases



```python
#Task-1:Test-Driven Development for Even/Odd Number Validator
#Generate test cases for a function is_even(n) and then implement
#Example Test Scenarios:a.is_even(2) → True b.is_even(7) → False
def is_even(n):
    """
    Function to check if a number is even.
    Input: n (integer)
    Output: True if n is even, False otherwise
    """
    if n % 2 == 0:
        return True
    else:
        return False
# Test cases
print(is_even(2))   # Expected: True
print(is_even(7))   # Expected: False
print(is_even(0))   # Expected: True
print(is_even(-4))  # Expected: True
print(is_even(9))   # Expected: False
```

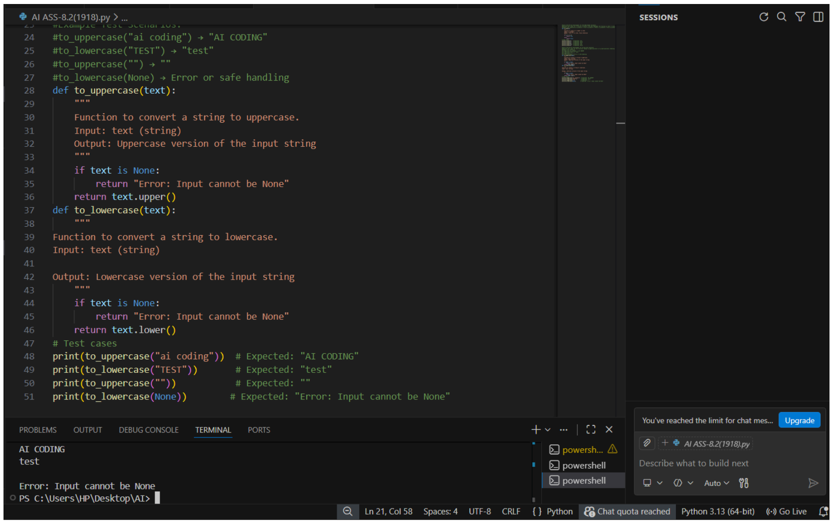
```
PS C:\Users\HP\Desktop\AI> & "C:\Program Files\Python313\python.exe" "c:/Users/HP/Desktop/AI/AI ASS-8.2(1918).py"
True
False
True
True
False
PS C:\Users\HP\Desktop\AI>
```

Task 2 – Test-Driven Development for String Case Converter
• Ask AI to generate test cases for two functions:
• to_uppercase(text)
• to_lowercase(text)
Requirements:
• Handle empty strings
• Handle mixed-case input
• Handle invalid inputs such as numbers or None
Example Test Scenarios:
to_uppercase("ai coding") → "AI CODING"
to_lowercase("TEST") → "test"
to_uppercase("") → ""

to_lowercase(None) → Error or safe handling

Expected Output -2

- Two string conversion functions that pass all AI-generated test cases with safe input handling.


AI ASS-8.2(1918).py > ...

```python
#Example Test Scenarios:
#to_uppercase("ai coding") → "AI CODING"
#to_lowercase("TEST") → "test"
#to_uppercase("") → ""
#to_lowercase(None) → Error or safe handling
def to_uppercase(text):
    """
    Function to convert a string to uppercase.
    Input: text (string)
    Output: Uppercase version of the input string
    """
    if text is None:
        return "Error: Input cannot be None"
    return text.upper()
def to_lowercase(text):
    """
Function to convert a string to lowercase.
Input: text (string)

Output: Lowercase version of the input string
    """
    if text is None:
        return "Error: Input cannot be None"
    return text.lower()
# Test cases
print(to_uppercase("ai coding"))  # Expected: "AI CODING"
print(to_lowercase("TEST"))       # Expected: "test"
print(to_uppercase(""))           # Expected: ""
print(to_lowercase(None))        # Expected: "Error: Input cannot be None"
```

```
AI CODING
test

Error: Input cannot be None
PS C:\Users\HP\Desktop\AI> 
```

Task 3 – Test-Driven Development for List Sum Calculator

- Use AI to generate test cases for a function sum_list(numbers) that calculates the sum of list elements.

Requirements:

- Handle empty lists
- Handle negative numbers
- Ignore or safely handle non-numeric values

Example Test Scenarios:

sum_list([1, 2, 3]) → 6

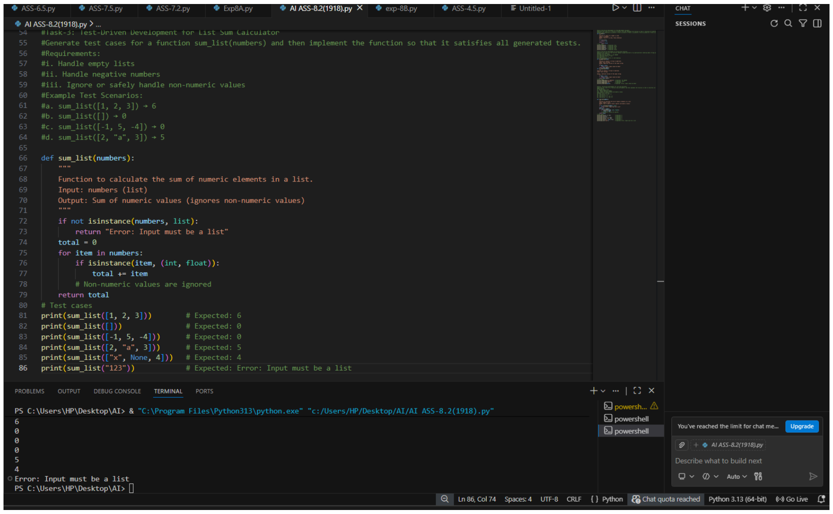
sum_list([]) → 0

sum_list([-1, 5, -4]) → 0

sum_list([2, "a", 3]) → 5

Expected Output 3

- A robust list-sum function validated using AI-generated test cases.



AI ASS-8.2(1918).py > ...

```python
#Task-3: Test-Driven Development for List Sum Calculator
#Generate test cases for a function sum_list(numbers) and then implement the function so that it satisfies all generated tests.
#Requirements:
#i. Handle empty lists
#ii. Handle negative numbers
#iii. Ignore or safely handle non-numeric values
#Example Test Scenarios:
#a. sum_list([1, 2, 3]) → 6
#b. sum_list([]) → 0
#c. sum_list([-1, 5, -4]) → 0
#d. sum_list([2, "a", 3]) → 5

def sum_list(numbers):
    """
    Function to calculate the sum of numeric elements in a list.
    Input: numbers (list)
    Output: Sum of numeric values (ignores non-numeric values)
    """
    if not isinstance(numbers, list):
        return "Error: Input must be a list"
    total = 0
    for item in numbers:
        if isinstance(item, (int, float)):
            total += item
        # Non-numeric values are ignored
    return total
# Test cases
print(sum_list([1, 2, 3]))        # Expected: 6
print(sum_list([]))               # Expected: 0
print(sum_list([-1, 5, -4]))      # Expected: 0
print(sum_list([2, "a", 3]))      # Expected: 5
print(sum_list(["x", None, 4]))   # Expected: 4
print(sum_list("123"))            # Expected: Error: Input must be a list
```

```
PS C:\Users\HP\Desktop\AI> & "C:\Program Files\Python313\python.exe" "c:/Users/HP/Desktop/AI/AI ASS-8.2(1918).py"
6
0
0
0
5
4
Error: Input must be a list
PS C:\Users\HP\Desktop\AI> 
```

Task 4 – Test Cases for Student Result Class

- Generate test cases for a StudentResult class with the following methods:
- add_marks(mark)
- calculate_average()
- get_result()

Requirements:

- Marks must be between 0 and 100
- Average ≥ 40 → Pass, otherwise Fail

Example Test Scenarios:

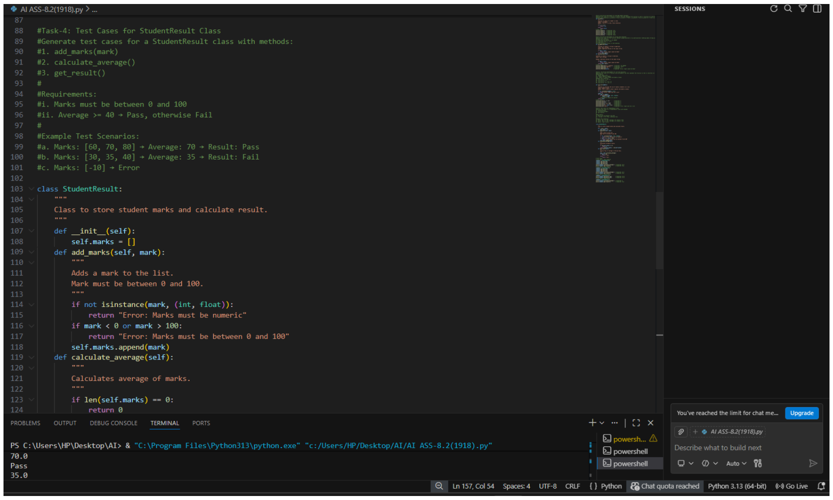
Marks: [60, 70, 80] → Average: 70 → Result: Pass

Marks: [30, 35, 40] → Average: 35 → Result: Fail

Marks: [-10] → Error

Expected Output -4

- A fully functional StudentResult class that passes all AI-generated test



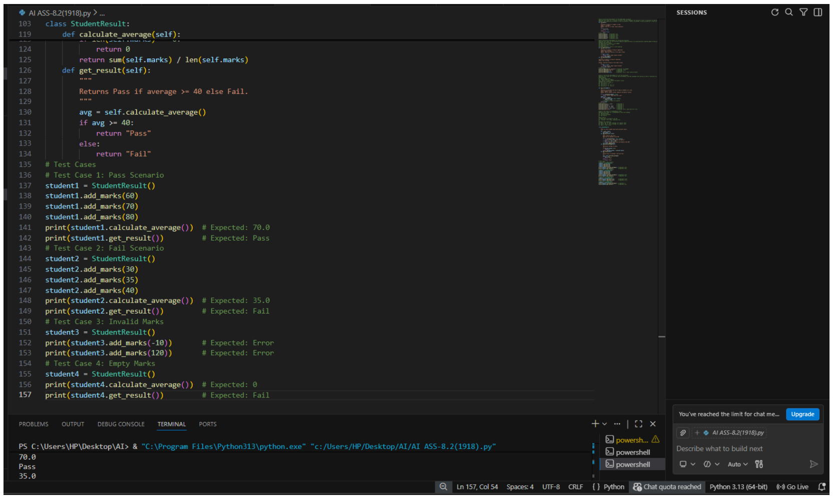
```python

#Task-4: Test Cases for StudentResult Class
#Generate test cases for a StudentResult class with methods:
#1. add_marks(mark)
#2. calculate_average()
#3. get_result()
#
#Requirements:
#i. Marks must be between 0 and 100
#ii. Average >= 40 → Pass, otherwise Fail
#
#Example Test Scenarios:
#a. Marks: [60, 70, 80] → Average: 70 → Result: Pass
#b. Marks: [30, 35, 40] → Average: 35 → Result: Fail
#c. Marks: [-10] → Error

class StudentResult:
    """
    Class to store student marks and calculate result.
    """
    def __init__(self):
        self.marks = []
    def add_marks(self, mark):
        """
        Adds a mark to the list.
        Mark must be between 0 and 100.
        """
        if not isinstance(mark, (int, float)):
            return "Error: Marks must be numeric"
        if mark < 0 or mark > 100:
            return "Error: Marks must be between 0 and 100"
        self.marks.append(mark)
    def calculate_average(self):
        """
        Calculates average of marks.
        """
        if len(self.marks) == 0:
            return 0
```

```
PS C:\Users\HP\Desktop\AI> & "C:\Program Files\Python313\python.exe" "c:/Users/HP/Desktop/AI/AI ASS-8.2(1918).py"
70.0
Pass
35.0
```

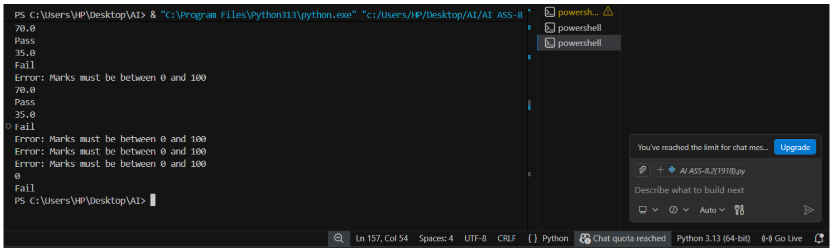
You've reached the limit for chat me...



```python
class StudentResult:
    def calculate_average(self):
            return 0
        return sum(self.marks) / len(self.marks)
    def get_result(self):
        """
        Returns Pass if average >= 40 else Fail.
        """
        avg = self.calculate_average()
        if avg >= 40:
            return "Pass"
        else:
            return "Fail"
# Test Cases
# Test Case 1: Pass Scenario
student1 = StudentResult()
student1.add_marks(60)
student1.add_marks(70)
student1.add_marks(80)
print(student1.calculate_average())  # Expected: 70.0
print(student1.get_result())         # Expected: Pass
# Test Case 2: Fail Scenario
student2 = StudentResult()
student2.add_marks(30)
student2.add_marks(35)
student2.add_marks(40)
print(student2.calculate_average())  # Expected: 35.0
print(student2.get_result())         # Expected: Fail
# Test Case 3: Invalid Marks
student3 = StudentResult()
print(student3.add_marks(-10))       # Expected: Error
print(student3.add_marks(120))       # Expected: Error
# Test Case 4: Empty Marks
student4 = StudentResult()
print(student4.calculate_average())  # Expected: 0
print(student4.get_result())         # Expected: Fail
```

```
PS C:\Users\HP\Desktop\AI> & "C:\Program Files\Python313\python.exe" "c:/Users/HP/Desktop/AI/AI ASS-8.2(1918).py"
70.0
Pass
35.0
```

You've reached the limit for chat me...

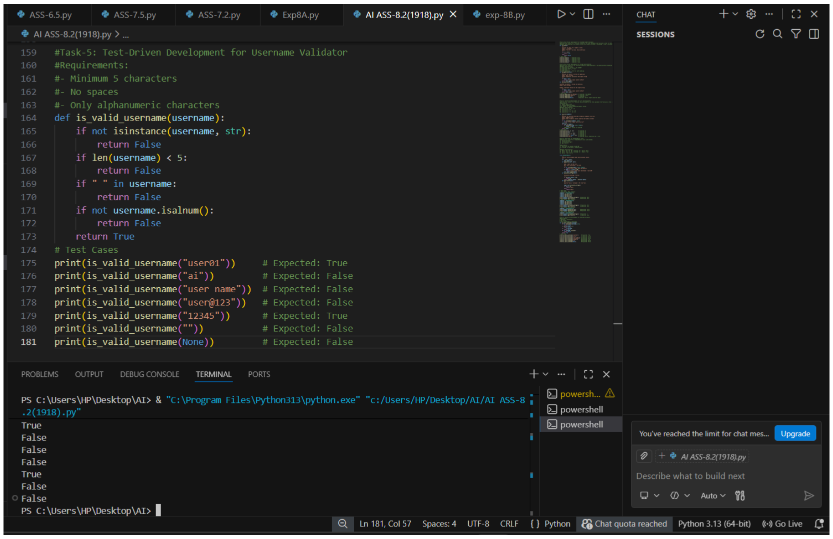


```
PS C:\Users\HP\Desktop\AI> & "C:\Program Files\Python313\python.exe" "c:/Users/HP/Desktop/AI/AI ASS-8
70.0
Pass
35.0
Fail
Error: Marks must be between 0 and 100
70.0
Pass
35.0
Fail
Error: Marks must be between 0 and 100
Error: Marks must be between 0 and 100
Error: Marks must be between 0 and 100
0
Fail
PS C:\Users\HP\Desktop\AI>
```

You've reached the limit for chat mes...

Task 5 – Test-Driven Development for Username Validator
Requirements:
• Minimum length: 5 characters
• No spaces allowed
• Only alphanumeric characters

Example Test Scenarios:
is_valid_username("user01") → True
is_valid_username("ai") → False
is_valid_username("user name") → False
is_valid_username("user@123") → False
Expected Output 5
A username validation function that passes all AI-generated test cases.



```python
#Task-5: Test-Driven Development for Username Validator
#Requirements:
#- Minimum 5 characters
#- No spaces
#- Only alphanumeric characters
def is_valid_username(username):
    if not isinstance(username, str):
        return False
    if len(username) < 5:
        return False
    if " " in username:
        return False
    if not username.isalnum():
        return False
    return True
# Test Cases
print(is_valid_username("user01"))     # Expected: True
print(is_valid_username("ai"))         # Expected: False
print(is_valid_username("user name"))  # Expected: False
print(is_valid_username("user@123"))   # Expected: False
print(is_valid_username("12345"))      # Expected: True
print(is_valid_username(""))           # Expected: False
print(is_valid_username(None))         # Expected: False
```

```
PS C:\Users\HP\Desktop\AI> & "C:\Program Files\Python313\python.exe" "c:/Users/HP/Desktop/AI/AI ASS-8
.2(1918).py"
True
False
False
False
True
False
False
PS C:\Users\HP\Desktop\AI>
```

You've reached the limit for chat mes...

DOCTEST:
python -m doctest -v Ass_8_2.py
True
False
True
True
False
AI CODING
test
Error: Input cannot be None
6
0
0

```
5
Average: 70.0
Result: Pass
Average: 35.0
Result: Fail
True
False
False
False
All tests passed successfully.
Trying:
is_even(2)
Expecting:
True
ok
Trying:
is_even(7)
Expecting:
False
ok
Trying:
is_even(0)
Expecting:
True
ok
Trying:
is_even(-4)
Expecting:
True
ok
Trying:
is_even(9)
Expecting:
False
ok
10 items had no tests:
Ass_8_2

Ass_8_2.StudentResult
Ass_8_2.StudentResult.init
Ass_8_2.StudentResult.add_marks
Ass_8_2.StudentResult.calculate_average
Ass_8_2.StudentResult.get_result
Ass_8_2.is_valid_username
Ass_8_2.sum_list
Ass_8_2.to_lowercase
Ass_8_2.to_uppercase
1 items passed all tests:
5 tests in Ass_8_2.is_even
5 tests in 11 items.
5 passed and 0 failed.
Test passed.
```

```
import pytest
from Ass_8_2 import is_even, to_uppercase, to_lowercase, sum_list, StudentResult, is_valid_username
```

# Test cases for Task 1 - Even/Odd Number Validator

```
def test_is_even():
assert is_even(2) == True
assert is_even(7) == False
assert is_even(0) == True
assert is_even(-4) == True
assert is_even(9) == False
```

## Test cases for Task 2 - String Case Converter

```
def test_to_uppercase():
assert to_uppercase("ai coding") == "AI CODING"
assert to_uppercase("") == ""
assert to_uppercase(None) == "Error: Input cannot be None"
assert to_uppercase("Test") == "TEST"
def test_to_lowercase():
assert to_lowercase("TEST") == "test"
assert to_lowercase("") == ""
assert to_lowercase(None) == "Error: Input cannot be None"
assert to_lowercase("Test") == "test"
```

## Test cases for Task 3 - List Sum Calculator

```
def test_sum_list():
assert sum_list([1, 2, 3]) == 6
assert sum_list([]) == 0
assert sum_list([-1, 5, -4]) == 0
assert sum_list([2, "a", 3]) == 5
assert sum_list([2, 3, -3, "a", 4]) == 6
assert sum_list([100, -50, 20]) == 70
```

## Test cases for Task 4 - StudentResult Class

```
def test_student_result():
student = StudentResult()
student.add_marks(60)
student.add_marks(70)
student.add_marks(80)
assert student.calculate_average() == 70.0
assert student.get_result() == "Pass"
student = StudentResult()
student.add_marks(30)
student.add_marks(35)
student.add_marks(40)
assert student.calculate_average() == 35.0
assert student.get_result() == "Fail"
```

## Test cases for Task 5 - Username Validator

```
def test_is_valid_username():
assert is_valid_username("user01") == True
assert is_valid_username("ai") == False
assert is_valid_username("user name") == False
assert is_valid_username("user@123") == False
assert is_valid_username("validUser") == True
assert is_valid_username("us") == False
```

```
PS C:\Users\HP\Desktop\AI> & "C:\Program Files\Python313\python.exe"
"c:/Users/HP/Desktop/AI/AI ASS-8.2(1918).py”
============================================
test session starts
```

```
platform win32 -- Python 3.12.3, pytest-9.0.2, pluggy-1.6.0
rootdir: PS C:\Users\HP\Desktop\AI>
A.I.AC
rootdir: PS C:\Users\HP\Desktop\AI>
A.I.AC
collected 6 items
Ass_8_2.py ......
[100%]
============================================================
6 passed in 0.09s
```