2303A51415
Batch 03
Assignment 8.2

2303a51214

Batch:-04

Task 1: Even/Odd Validator - AI Prompt

Generate unittest test cases for function is_even(n) that:

- Accepts only integers
- Handles zero, negative, large integers
- Raises TypeError for invalid input

Task 1: Test Case

```
class TestIsEven(unittest.TestCase):     def
  test_even(self):


        self.assertTrue(is_even(2))
def test_odd(self):

        self.assertFalse(is_even(7))
def test_zero(self):

        self.assertTrue(is_even(0))
def test_negative(self):

        self.assertTrue(is_even(-4))
def test_invalid(self):          with
```

```
self.assertRaises(TypeError):
is_even("2")
```

Task 1: Implementation

```
def is_even(n):     if not
isinstance(n, int):

        raise TypeError("Input must be an
integer")     return n % 2 == 0
```

Task 2: String Case Converter - AI Prompt

```
Generate test cases for:
to_uppercase(text) to_lowercase(text)
Handle empty strings and invalid
inputs.
```

Task 2: Implementation

```
def to_uppercase(text):     if
not isinstance(text, str):

        raise TypeError("Input must be a
string")     return text.upper()

def to_lowercase(text):     if
not isinstance(text, str):

        raise TypeError("Input must be a
string")     return text.lower()
```

Task 3: List Sum Calculator - AI Prompt

```
Generate test cases for sum_list(numbers):

 - Handle empty list
```

- Handle negatives

- Ignore non-numeric values

Task 3: Implementation

```
def sum_list(numbers):     if not
isinstance(numbers, list):

        raise TypeError("Input must be a
list")     total = 0     for item in
numbers:          if isinstance(item,
(int, float)):

            total += item
return total
```

Task 4: StudentResult Class - AI Prompt

```
Generate test cases for StudentResult class
with:

add_marks(mark)
calculate_average()
get_result()
```

Task 4: Implementation

```
class StudentResult:
def __init__(self):
self.marks = []

    def add_marks(self, mark):          if not
isinstance(mark, (int, float)):              raise
TypeError("Mark must be numeric")          if mark <
0 or mark > 100:              raise ValueError("Mark
must be between 0 and 100")
self.marks.append(mark)
```

```
    def calculate_average(self):
if not self.marks:              return 0
return sum(self.marks) / len(self.marks)

    def get_result(self):

        avg = self.calculate_average()
return "Pass" if avg >= 40 else "Fail"
```

Task 5: Username Validator - AI Prompt

```
Generate test cases for username validation:

 - Minimum length 5

 - No spaces

 - Only alphanumeric
```

Task 5: Implementation

```
def is_valid_username(username):
if not isinstance(username, str):

        raise TypeError("Username must be a
string")     if len(username) < 5:

        return False     if "
" in username:          return
False     if not
username.isalnum():

        return False
return True
```