

# Assignment-8

L.Aravind  
2303A52028  
Batch - 38

## TASK-1

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

```
1
2
3 from typing import Any
4 import unittest
5
6
7 def is_even(n: Any) -> bool:
8
9     if isinstance(n, bool):
10         raise TypeError("input must be an integer, not a bool")
11
12     if not isinstance(n, int):
13         raise TypeError("input must be an integer")
14
15     return (n % 2) == 0
16
17
18 class TestIsEven(unittest.TestCase):
19     def test_is_even_basic_true(self):
20         self.assertTrue(is_even(2))
21
22     def test_is_even_basic_false(self):
23         self.assertFalse(is_even(7))
24
25     def test_zero_is_even(self):
26         self.assertTrue(is_even(0))
27
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
dreddy/tests/test_evenodd.py
lakkiaravindreddy@Lakkis-MacBook-Air ~ % python3 -m unittest -q /Users/lakkiaravindreddy/tests/test_evenodd_unittest.py
Ran 7 tests in 0.000s
OK
lakkiaravindreddy@Lakkis-MacBook-Air ~ % python3 /Users/lakkiaravindreddy/evenodd.py
.....
Ran 7 tests in 0.000s
OK
lakkiaravindreddy@Lakkis-MacBook-Air ~ % python3 /Users/lakkiaravindreddy/evenodd.py
is_even(2) -> True
is_even(7) -> False
is_even(0) -> True
is_even(-4) -> True
is_even(9) -> False
```

CHAT

Todos (7/7)

python3 /Users/lakkiaravindreddy/evenodd.py

Completed (7/7) Add demo output

Done — I updated evenodd.py to print the example outputs and ran it. Terminal output:

is\_even(2) -> True  
is\_even(7) -> False  
is\_even(0) -> True  
is\_even(-4) -> True  
is\_even(9) -> False

How to reproduce

- Run the file directly to print the examples:

python3 /Users/lakkiaravindreddy/evenodd.py

- Run tests instead:

python3 /Users/lakkiaravindreddy/evenodd\_unittest.py

If you'd like a different output format (arrow vs. →, JSON, or colored text), or want a CLI that accepts a number argument and prints parity for that single input, tell me which format and I'll update the file.

evenodd.py +

Add context (#), extensions (@), ci

Agent GPT-5 mini 1x

Ln 8, Col 1 Spaces: 4 UTF-8 LF Python 3.9.6 Go Live

# TASK-2

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

```
1
2
3 from typing import Any
4 import unittest
5
6
7 def to_uppercase(text: Any) -> str:
8
9     if not isinstance(text, str):
10         raise TypeError("input must be a string")
11     return text.upper()
12
13
14 def to_lowercase(text: Any) -> str:
15
16     if not isinstance(text, str):
17         raise TypeError("input must be a string")
18     return text.lower()
19
20
21 class TestStringCase(unittest.TestCase):
22     def test_to_uppercase_basic(self):
23         self.assertEqual(to_uppercase("ai coding"), "AI CODING")
24
25     def test_to_lowercase_basic(self):
26         self.assertEqual(to_lowercase("TEST"), "test")
27
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
lakkiravindreddy@lakkis-MacBook-Air ~ % /usr/bin/python3 /Users/lakkiravindreddy/stringcase.py
to_lowercase('ai coding') -> 'ai coding'

to_uppercase('TEST') -> 'TEST'
to_lowercase('TEST') -> 'test'

to_uppercase('') -> ''
to_lowercase('') -> ''

to_uppercase('PyThOn') -> 'PYTHON'
to_lowercase('PyThOn') -> 'python'

to_uppercase('Café') -> 'CAFÉ'
to_lowercase('Café') -> 'café'

to_uppercase(None) -> 'Error: TypeError: input must be a string'
to_lowercase(None) -> 'Error: TypeError: input must be a string'

to_uppercase(123) -> 'Error: TypeError: input must be a string'
to_lowercase(123) -> 'Error: TypeError: input must be a string'
```

CHAT

Todos (5/5)

CODING\*

```
to_uppercase("TEST") -> "test"
to_uppercase("") -> ""
to_uppercase(None) -> Error or safe handling
```

Preparing test cases and implementation...

I'll create a todo list, add a single-file module `stringcase.py` that contains test cases (unittest) and implementations for `to_uppercase` and `to_lowercase`, then run the tests in the terminal to verify all pass. Expected outcome: tests pass and demo output is shown. I'll now add the file.

✓ Created 5 todos

✓ Using "Apply Patch"

Auto approved for this session (Edit)

stringcase.py

Completed (2/5) Generate tests

python3 /Users/lakkiravindreddy/stringcase.py --test

Completed (4/5) Run tests

stringcase.py +

Add context (#), extensions (@), or

Agent GPT-5 mini

# TASK-3

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  $\geq 40 \rightarrow$  Pass, otherwise Fail

```
1
2
3 from typing import Iterable, Any
4 import unittest
5
6
7 def sum_list(numbers: Iterable[Any]) -> float:
8
9     # Quick check that we were given an iterable (but str is iterable; handle separately)
10    if isinstance(numbers, (str, bytes)):
11        raise TypeError("input must be an iterable of numbers (not a string)")
12
13    try:
14        iterator = iter(numbers)
15    except TypeError:
16        raise TypeError("input must be an iterable (like list or tuple)")
17
18    total = 0
19    found_int = False
20    for item in iterator:
21        # Ignore booleans explicitly
22        if isinstance(item, bool):
23            continue
24
25        if isinstance(item, (int, float)):
26            total += item
27            found_int = True
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

lakkiaravindreddy@lakkis-MacBook-Air ~ % /usr/bin/python3 /Users/lakkiaravindreddy/stringcase.py

to\_uppercase(None) -> 'Error: TypeError: input must be a string'

to\_lowercase(123) -> 'Error: TypeError: input must be a string'

to\_uppercase(123) -> 'Error: TypeError: input must be a string'

to\_lowercase(123) -> 'Error: TypeError: 1'

Open file in editor (cmd + click)

• lakkiaravindreddy@lakkis-MacBook-Air ~ % /usr/bin/python3 /Users/lakkiaravindreddy/sum\_list.py

sum\_list([1, 2, 3]) -> 6

sum\_list([]) -> 0

sum\_list([-1, 5, -4]) -> 0

sum\_list([2, 'a', 3]) -> 5

sum\_list([2, 'a', 3]) -> 5

sum\_list([None, 1, 2.5, 'x']) -> 3.5

sum\_list(123) -> Error: TypeError: input must be an iterable (like list or tuple)

• lakkiaravindreddy@lakkis-MacBook-Air ~ % /usr/bin/python3 /Users/lakkiaravindreddy/sum\_list.py

sum\_list([1, 2, 3]) -> 6

sum\_list([]) -> 0

sum\_list([-1, 5, -4]) -> 0

sum\_list([2, 'a', 3]) -> 5

sum\_list([None, 1, 2.5, 'x']) -> 3.5

sum\_list(123) -> Error: TypeError: input must be an iterable (like list or tuple)

lakkiaravindreddy@lakkis-MacBook-Air ~ %

CHAT + + + + +

> Todos (5/5)

outputs, including error reporting for invalid top-level inputs.

Design notes and assumptions

- Booleans are ignored explicitly (so they don't count as 1 or 0). If you prefer to treat True as 1 and False as 0, I can change the implementation and tests.
- Strings are rejected as the top-level argument because strings are iterable of characters and that usually indicates incorrect input. If you'd rather sum digits in a numeric string, tell me.
- Non-numeric elements inside the iterable are ignored rather than raising errors, per the requirement to "ignore or safely handle non-numeric values."

Next steps (optional)

- Change bool handling to include booleans as numeric.
- Allow numeric strings to be parsed (e.g., "3" -> 3) or raise on mixed types.
- Export a small CLI flag to accept JSON arrays on the command line.

Which change would you like me to make (if any)?

🔄 ↶ ↷ 📄

sum\_list.py +

Add context (#), extensions (@), or

Agent GPT-5 mini

Ln 8, Col 1 Spaces: 4 UTF-8 LF Python 3.9.6 Go Live

# TASK-4

• 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  $\geq 40 \rightarrow$  Pass, otherwise Fail

```
6 class StudentResult:
30     def get_result(self) -> str:
33         return "Pass" if avg >= 40.0 else "Fail"
34
35
36 class TestStudentResult(unittest.TestCase):
37     def test_add_valid_marks_and_average(self):
38         s = StudentResult()
39         s.add_marks(50)
40         s.add_marks(30)
41         s.add_marks(60)
42         self.assertAlmostEqual(s.calculate_average(), (50 + 30 + 60) / 3)
43
44     def test_empty_marks_average_and_result(self):
45         s = StudentResult()
46         self.assertEqual(s.calculate_average(), 0.0)
47         self.assertEqual(s.get_result(), "Fail")
48
49     def test_negative_and_overflow_marks_rejected(self):
50         s = StudentResult()
51         with self.assertRaises(ValueError):
52             s.add_marks(-1)
53         with self.assertRaises(ValueError):
54             s.add_marks(101)
55
56     def test_non_numeric_and_bool_rejected(self):
57         s = StudentResult()
```

```
lakkiaaravindreddy@lakkis-MacBook-Air ~ % /usr/bin/python3 /Users/lakkiaaravindreddy/sum_list.py
sum_list([]) -> 0
sum_list([-1, 5, -4]) -> 0
sum_list([2, 'a', 3]) -> 5
sum_list([None, 1, 2.5, 'x']) -> 3.5
sum_list([123]) -> Error: TypeError: input must be an iterable (like list or tuple)
lakkiaaravindreddy@lakkis-MacBook-Air ~ % /usr/bin/python3 /Users/lakkiaaravindreddy/sum_list.py
sum_list([1, 2, 3]) -> 6
sum_list([1]) -> 0
sum_list([-1, 5, -4]) -> 0
sum_list([2, 'a', 3]) -> 5
sum_list([None, 1, 2.5, 'x']) -> 3.5
sum_list([123]) -> Error: TypeError: input must be an iterable (like list or tuple)
lakkiaaravindreddy@lakkis-MacBook-Air ~ % /usr/bin/python3 /Users/lakkiaaravindreddy/student_result.py
marks: [70.0, 80.0, 30.0]
average: 60.0
result: Pass
lakkiaaravindreddy@lakkis-MacBook-Air ~ % /usr/bin/python3 /Users/lakkiaaravindreddy/student_result.py
marks: [70.0, 80.0, 30.0]
average: 60.0
result: Pass
lakkiaaravindreddy@lakkis-MacBook-Air ~ %
```

Progress update — what I did

- Implemented StudentResult in student\_result.py with:
  - add\_marks(mark): validates mark is int/float (not bool) and 0 ≤ mark ≤ 100; raises TypeError/ValueError accordingly.
  - calculate\_average(): returns 0.0 when no marks.
  - get\_result(): returns "Pass" if average ≥ 40.0 else "Fail".
- Created unittest test cases covering:
  - valid marks & average,
  - empty marks average & result,
  - rejection of negative and

# 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.

The screenshot shows a code editor with a Python file named `username_validator.py`. The code defines a function `is_valid_username` and a test class `TestUsernameValidator`.

```
1 from typing import Any
2 import unittest
3
4
5
6 def is_valid_username(username: Any) -> bool:
7
8     if not isinstance(username, str):
9         return False
10
11     if len(username) < 5:
12         return False
13
14     if " " in username:
15         return False
16
17     if not (username.isalnum() and username.isascii()):
18         return False
19
20     return True
21
22
23 class TestUsernameValidator(unittest.TestCase):
24     def test_valid_username(self):
25         self.assertTrue(is_valid_username("user01"))
26
27     def test_too_short(self):
```

The terminal output shows the results of running the tests:

```
lakkiravindreddy@lakkis-MacBook-Air ~ % /usr/bin/python3 /Users/lakkiravindreddy/student_result.py
marks: [70.0, 80.0, 30.0]
average: 60.0
result: Pass
lakkiravindreddy@lakkis-MacBook-Air ~ % /usr/bin/python3 /Users/lakkiravindreddy/username_validator.py
is_valid_username("user01") -> True
is_valid_username("ai") -> False
is_valid_username("user name") -> False
is_valid_username("user@123") -> False
is_valid_username("abcde") -> True
is_valid_username(None) -> False
is_valid_username(12345) -> False
lakkiravindreddy@lakkis-MacBook-Air ~ % /usr/bin/python3 /Users/lakkiravindreddy/username_validator.py
is_valid_username("user01") -> True
is_valid_username("ai") -> False
is_valid_username("user name") -> False
is_valid_username("user@123") -> False
is_valid_username("abcde") -> True
is_valid_username(None) -> False
is_valid_username(12345) -> False
lakkiravindreddy@lakkis-MacBook-Air ~ %
```

The chat sidebar on the right shows a conversation with an AI agent, discussing the requirements and the test cases.