Assignment-8

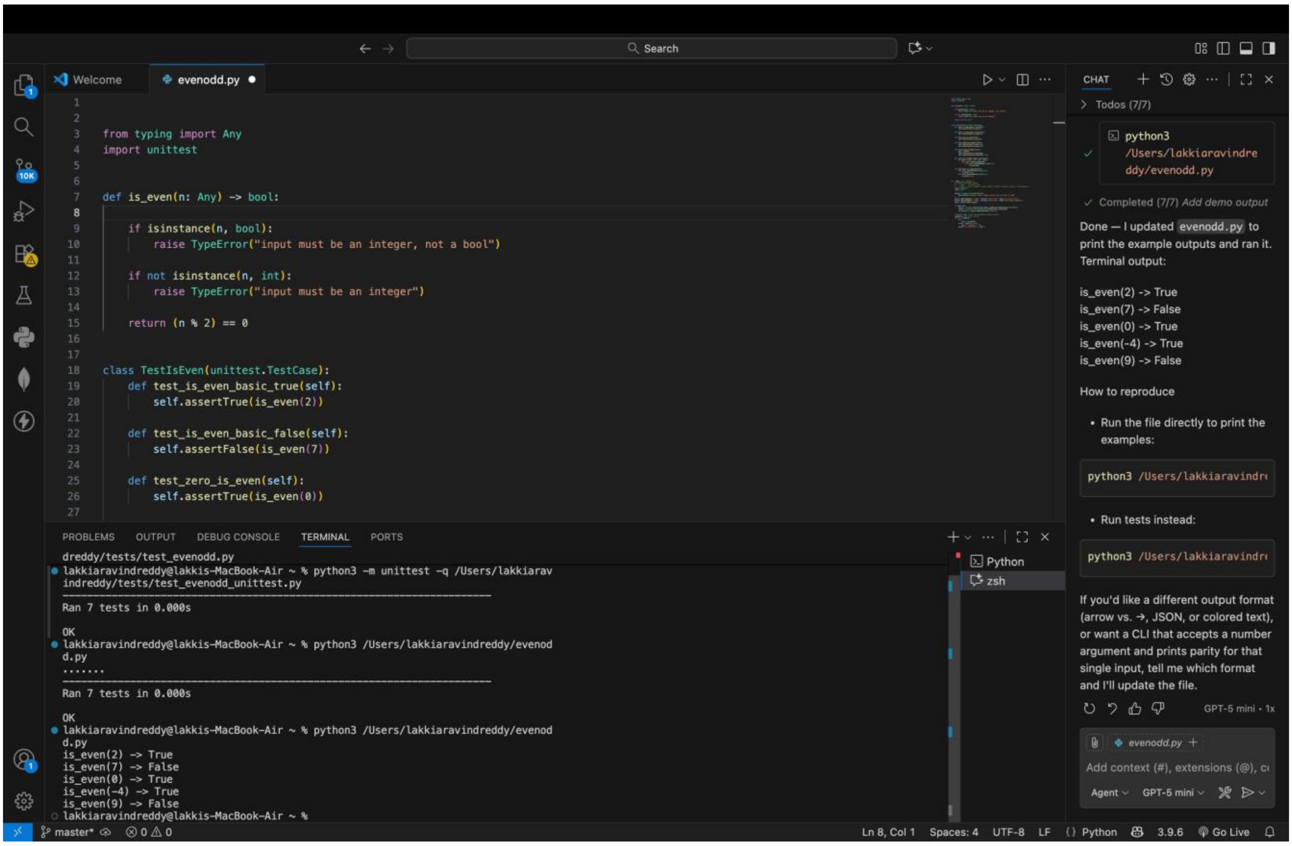
K.SAI ROHAN
2303A52025
Batch - 38

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**



```python


from typing import Any
import unittest


def is_even(n: Any) -> bool:

    if isinstance(n, bool):
        raise TypeError("input must be an integer, not a bool")

    if not isinstance(n, int):
        raise TypeError("input must be an integer")

    return (n % 2) == 0


class TestIsEven(unittest.TestCase):
    def test_is_even_basic_true(self):
        self.assertTrue(is_even(2))

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

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

```

```
dreddy/tests/test_evenodd.py
lakkiaravindreddy@lakkis-MacBook-Air ~ % python3 -m unittest -q /Users/lakkiarav
indreddy/tests/test_evenodd_unittest.py
----------------------------------------------------------------------
Ran 7 tests in 0.000s

OK
lakkiaravindreddy@lakkis-MacBook-Air ~ % python3 /Users/lakkiaravindreddy/evenod
d.py
.......
----------------------------------------------------------------------
Ran 7 tests in 0.000s

OK
lakkiaravindreddy@lakkis-MacBook-Air ~ % python3 /Users/lakkiaravindreddy/evenod
d.py
is_even(2) -> True
is_even(7) -> False
is_even(0) -> True
is_even(-4) -> True
is_even(9) -> False
lakkiaravindreddy@lakkis-MacBook-Air ~ %
```

```
python3
/Users/lakkiaravindre
ddy/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/lakkiaravindr
```

- Run tests instead:

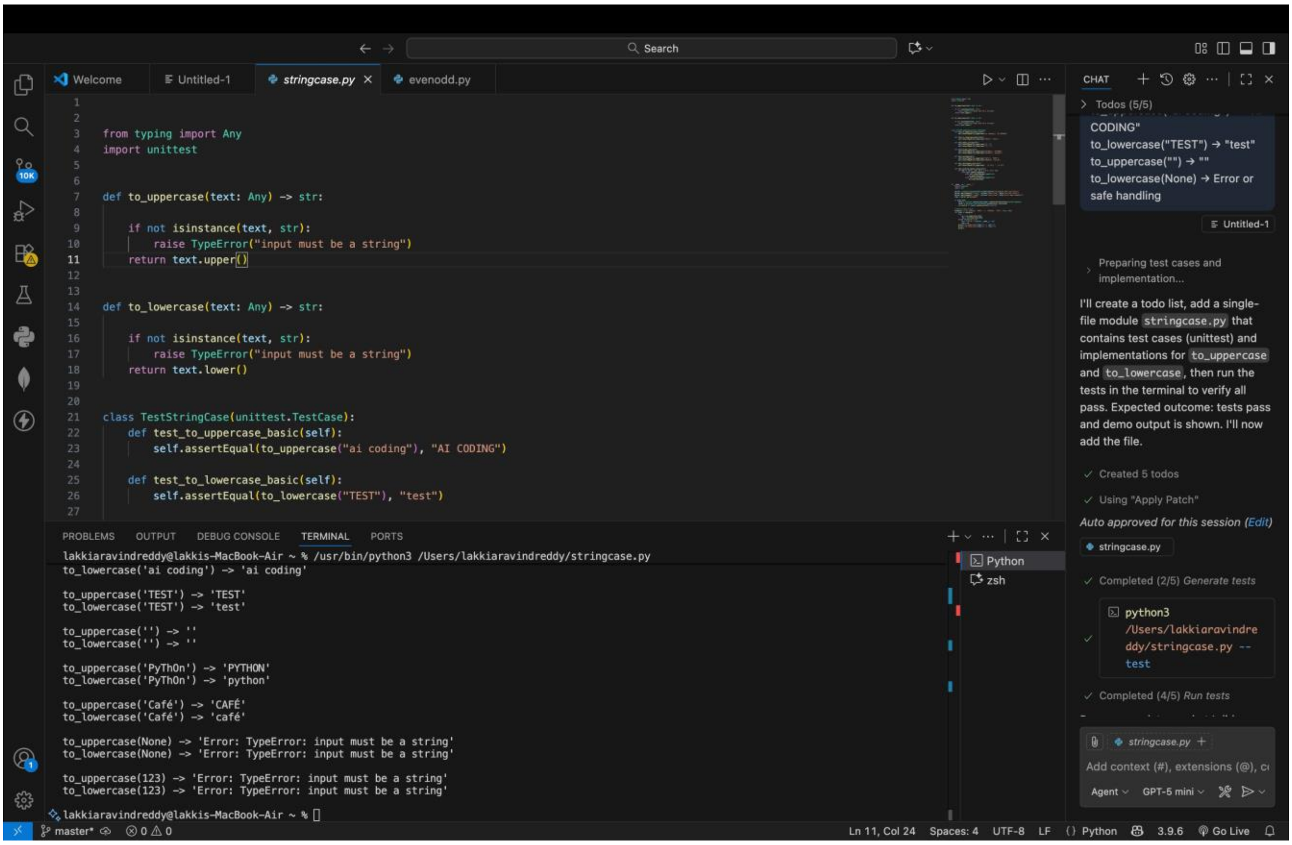
```
python3 /Users/lakkiaravindr
```

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.

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**



```python


from typing import Any
import unittest


def to_uppercase(text: Any) -> str:

    if not isinstance(text, str):
        raise TypeError("input must be a string")
    return text.upper()


def to_lowercase(text: Any) -> str:

    if not isinstance(text, str):
        raise TypeError("input must be a string")
    return text.lower()


class TestStringCase(unittest.TestCase):
    def test_to_uppercase_basic(self):
        self.assertEqual(to_uppercase("ai coding"), "AI CODING")

    def test_to_lowercase_basic(self):
        self.assertEqual(to_lowercase("TEST"), "test")

```

```
lakkiaravindreddy@lakkis-MacBook-Air ~ % /usr/bin/python3 /Users/lakkiaravindreddy/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'
lakkiaravindreddy@lakkis-MacBook-Air ~ % 
```

CODING"
to_lowercase("TEST") → "test"
to_uppercase("") → ""
to_lowercase(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)

✓ Completed (2/5) *Generate tests*

```
python3 /Users/lakkiaravindreddy/stringcase.py --test
```

✓ Completed (4/5) *Run tests*

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

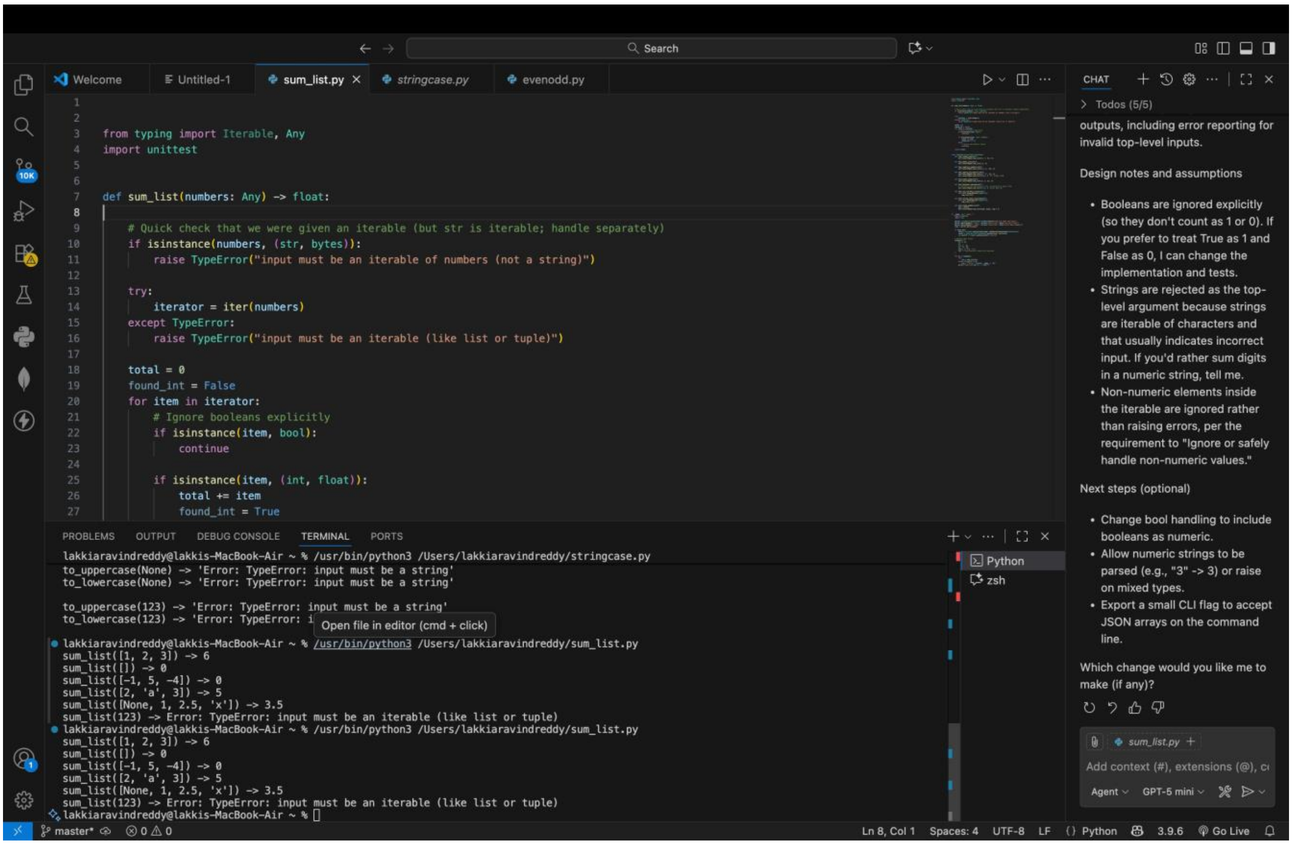
- **calculate_average()**

TASK-3

- get_result()

Requirements:

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



```python


from typing import Iterable, Any
import unittest


def sum_list(numbers: Any) -> float:

    # Quick check that we were given an iterable (but str is iterable; handle separately)
    if isinstance(numbers, (str, bytes)):
        raise TypeError("input must be an iterable of numbers (not a string)")

    try:
        iterator = iter(numbers)
    except TypeError:
        raise TypeError("input must be an iterable (like list or tuple)")

    total = 0
    found_int = False
    for item in iterator:
        # Ignore booleans explicitly
        if isinstance(item, bool):
            continue

        if isinstance(item, (int, float)):
            total += item
            found_int = True
```

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

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 ~ % /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 ~ % ▯
```

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.

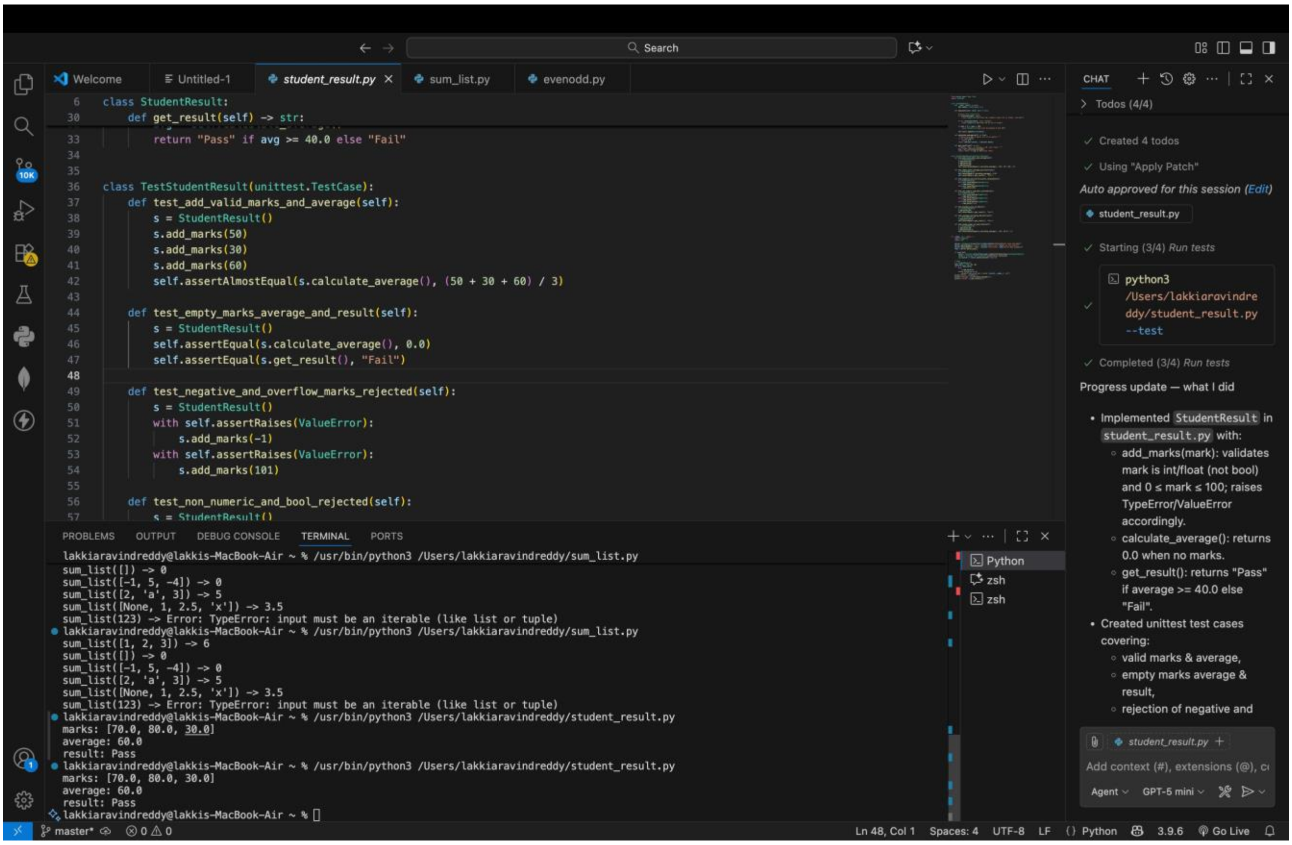
Which change would you like me to make (if any)?

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 ≥ 40 → Pass, otherwise Fail**



```python
class StudentResult:
    def get_result(self) -> str:
        return "Pass" if avg >= 40.0 else "Fail"


class TestStudentResult(unittest.TestCase):
    def test_add_valid_marks_and_average(self):
        s = StudentResult()
        s.add_marks(50)
        s.add_marks(30)
        s.add_marks(60)
        self.assertAlmostEqual(s.calculate_average(), (50 + 30 + 60) / 3)

    def test_empty_marks_average_and_result(self):
        s = StudentResult()
        self.assertEqual(s.calculate_average(), 0.0)
        self.assertEqual(s.get_result(), "Fail")

    def test_negative_and_overflow_marks_rejected(self):
        s = StudentResult()
        with self.assertRaises(ValueError):
            s.add_marks(-1)
        with self.assertRaises(ValueError):
            s.add_marks(101)

    def test_non_numeric_and_bool_rejected(self):
        s = StudentResult()
```

```
lakkiaravindreddy@lakkis-MacBook-Air ~ % /usr/bin/python3 /Users/lakkiaravindreddy/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)
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 ~ % /usr/bin/python3 /Users/lakkiaravindreddy/student_result.py
marks: [70.0, 80.0, 30.0]
average: 60.0
result: Pass
lakkiaravindreddy@lakkis-MacBook-Air ~ % /usr/bin/python3 /Users/lakkiaravindreddy/student_result.py
marks: [70.0, 80.0, 30.0]
average: 60.0
result: Pass
lakkiaravindreddy@lakkis-MacBook-Air ~ % 
```

Todos (4/4)

✓ Created 4 todos

✓ Using "Apply Patch"

Auto approved for this session (Edit)

student_result.py

✓ Starting (3/4) *Run tests*

```
python3 /Users/lakkiaravindreddy/student_result.py --test
```

✓ Completed (3/4) *Run tests*

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

student_result.py

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

Agent GPT-5 mini

Test-Driven Development for Username Validator

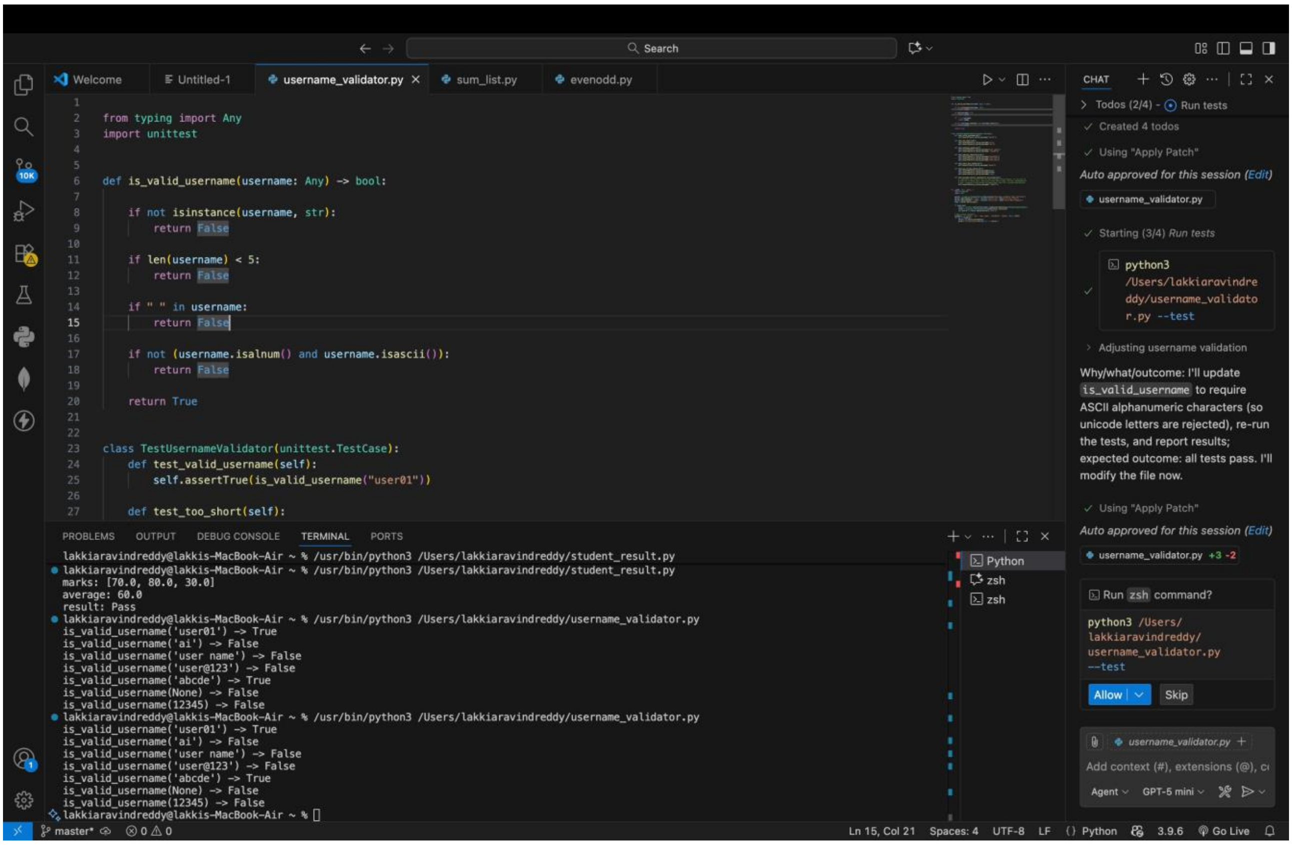
Requirements:

TASK-5

- 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

from typing import Any
import unittest


def is_valid_username(username: Any) -> bool:

    if not isinstance(username, str):
        return False

    if len(username) < 5:
        return False

    if " " in username:
        return False

    if not (username.isalnum() and username.isascii()):
        return False

    return True


class TestUsernameValidator(unittest.TestCase):
    def test_valid_username(self):
        self.assertTrue(is_valid_username("user01"))

    def test_too_short(self):
```

```
lakkiaravindreddy@lakkis-MacBook-Air ~ % /usr/bin/python3 /Users/lakkiaravindreddy/student_result.py
lakkiaravindreddy@lakkis-MacBook-Air ~ % /usr/bin/python3 /Users/lakkiaravindreddy/student_result.py
marks: [70.0, 80.0, 30.0]
average: 60.0
result: Pass
lakkiaravindreddy@lakkis-MacBook-Air ~ % /usr/bin/python3 /Users/lakkiaravindreddy/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
lakkiaravindreddy@lakkis-MacBook-Air ~ % /usr/bin/python3 /Users/lakkiaravindreddy/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
lakkiaravindreddy@lakkis-MacBook-Air ~ % 
```

Todos (2/4) - Run tests

✓ Created 4 todos

✓ Using "Apply Patch"

Auto approved for this session (Edit)

username_validator.py

✓ Starting (3/4) *Run tests*

```
python3 /Users/lakkiaravindreddy/username_validator.py --test
```

Adjusting username validation

Why/what/outcome: I'll update `is_valid_username` to require ASCII alphanumeric characters (so unicode letters are rejected), re-run the tests, and report results; expected outcome: all tests pass. I'll modify the file now.

✓ Using "Apply Patch"

Auto approved for this session (Edit)

username_validator.py +3 -2

Run `zsh` command?

```
python3 /Users/lakkiaravindreddy/username_validator.py --test
```

Allow | Skip

username_validator.py +

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

Agent · GPT-5 mini