Unit Testing in Python

What is a Unit Test?

- A unit test is a small program that:
  - Calls a function with specific inputs
  - Checks if the output matches what you expect
- In Python, we often use the `unittest` module for this.

Why Write Unit Tests?

- Help clarify what you want the code to do.
    - Especially useful when working with an AI assistant!
    - Write tests before you write the corresponding code (Test-Driven Development).
- Check that our code works as expected.
- Catch bugs early.
- Make it safer to change code.
    - You can rerun all your past tests to make sure they still pass.
- Provide documentation for how the code is supposed to work.

Example: Testing a Function

```
def longer_string(a, b):
    if len(a) > len(b):
        return a
    else:
        return b
```

```
import unittest

class TestLongerString(unittest.TestCase):
    def test_longer(self):
        self.assertEqual(longer_string("cat", "giraffe"), "giraffe")
```

How to Run Unit Tests

- Save your tests in a file (e.g., `test_example.py`)
- Run in the terminal:

```
python -m unittest test_example.py
```

- Passing tests show a dot (.)
- Failing tests show an F and an error message

How to Write Unit Tests

- ask your AI assistant!

```
Please add a test for longer_string that checks that it returns the first string when both strings are the same length.
```

- Discussion question: why might we care that it returns the first string when they're the same length?

Unit Tests for ADVENT

- See session03/unit-tests.md
- Exercise: what do each of the tests check for?
    - Not sure? Ask your AI assistant!
- Run the tests

```
    python -m unittest discover
```

- discover: run all tests in the current directory and subdirectories

Discussion Question

- What is the role of the setUp function?
- Are the tests checking the behavior of *just* the functions? What else are they checking?

Let's Add a Test

Let's specify what should happen if the user tries to move in a direction where there's no exit

- The game should inform the player that they can't go that way.
- And leave them where they are.

```
Please add another test, for moving in a direction with no exit.
In the initial state, the player is in the cave, and it only has a north exit.
Let's test that if the player tries to move east, they get a message saying they can't go that way.
```

```
def test_move_invalid_direction(self) -> None:
    # In the initial state, the player is in the cave, which only has a north exit.
    result: str = self.game.move("east")
    # The correct behavior would be to get a message saying you can't go that way and stay in the cave.
    self.assertIn("can't go that way", result.lower())
    self.assertEqual(self.game.current_room, "cave")
```

Debugging

What happens when we run the test?

```
presnick@m-fvfhf249q6lx advent % python -m unittest
.E.
======================================================================
ERROR: test_move_invalid_direction (test_advent.TestAdventGame.test_move_invalid_direction)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/Users/presnick/Documents/Documents/code/umsi211-f25-course-resources/weeks2-3/advent/test_advent.py", line 7, in test_move_invalid_direction
    result = self.game.move("east")
             ^^^^^^^^^^^^^^^^^^^^^^
  File "/Users/presnick/Documents/Documents/code/umsi211-f25-course-resources/weeks2-3/advent/advent.py", line 32, in move
    return f"You move {direction}.\n" + self.describe_current_room()
                                        ^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/Users/presnick/Documents/Documents/code/umsi211-f25-course-resources/weeks2-3/advent/advent.py", line 15, in describe_current_room
    room = self.rooms[self.current_room]
           ~~~~~~~~~~^^^^^^^^^^^^^^^^^^^
KeyError: 'east'

----------------------------------------------------------------------
Ran 3 tests in 0.001s

FAILED (errors=1)
```

Cut and paste the error into your chat with the AI assistant to get help diagnosing and fixing. Rerun the tests to check whether its fix worked.

Practice: Create More Tests

- What else should we be testing for?
  - Think about things that *should* happen, desired behaviors.
  - Also what *shouldn't* happen.
    - What could cause an error?
    - What would be player cheats, letting them do things they shouldn't be able to do?
- This is a design exercise. It requires creativity.
  - It's worth thinking this through yourself, but the AI assistant can help brainstorm.
  - Use your own judgment about which ideas to pursue.