Testing

CSCI 39549
October 24, 2018

Testing

Agenda

Guest Lecturer

Mike Hernandez

Eendraght Maeckt Maght

SEAL OF THE CITY OF BROOKLYN

SIG · COLL · HUNTER · NOV · EBOR ·

MIHI CURA FUTURI

MDCCCLXX

DEVRY

Etsy

Course Grades

Grading (from the Syllabus)

Same overall percentages, but more granular breakdown:

- 1% Homework Assignments (2 total)
- 26% for each version of the team coding assignment (3 total)
- 5% for each demo (3 total)
- 2.5% for each retrospective (2 total)

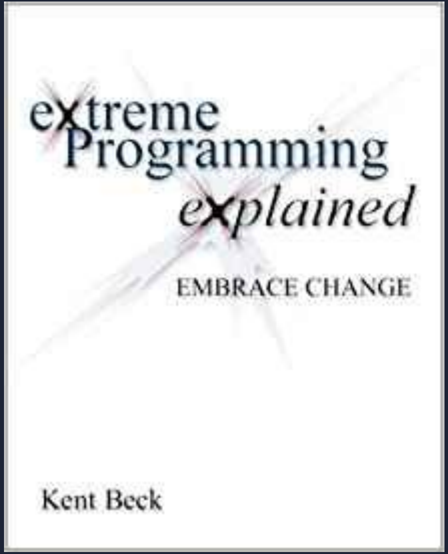
Grading (from the Syllabus)

Same overall percentages, but more granular breakdown:

- 1% Homework Assignments (2 total)
- 26% for each version of the team coding assignment (3 total)
- 5% for each demo (3 total)
- 2.5% for each retrospective (2 total)

If anything on Blackboard seems surprising, **talk to us**! We want you to be successful in this class.

Reading Review



extreme Programming explained

EMBRACE CHANGE

Kent Beck

What are defects aka “bugs”?

“Defects destroy the trust required for effective software development. The customers need to be able to trust the software. The managers need to be able to trust reports of progress. The programmers need to be able to trust each other. Defects destroy this trust.”

Extreme Programming Explained

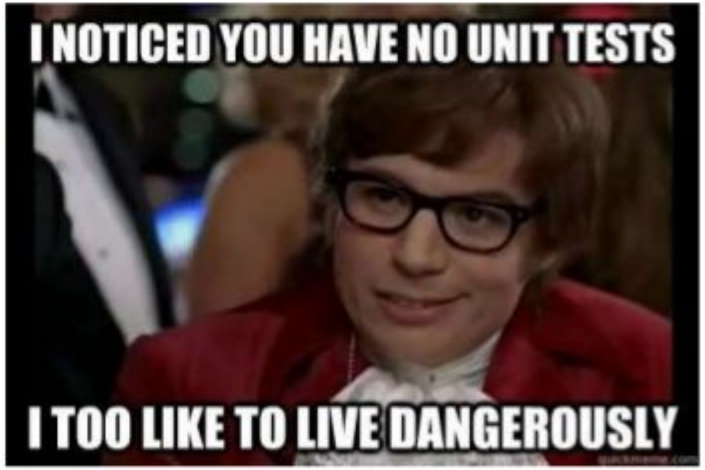
The Impact of Defects in Software Development

- Time consuming to fix
- Loss to the business
- Damaged customer relationships

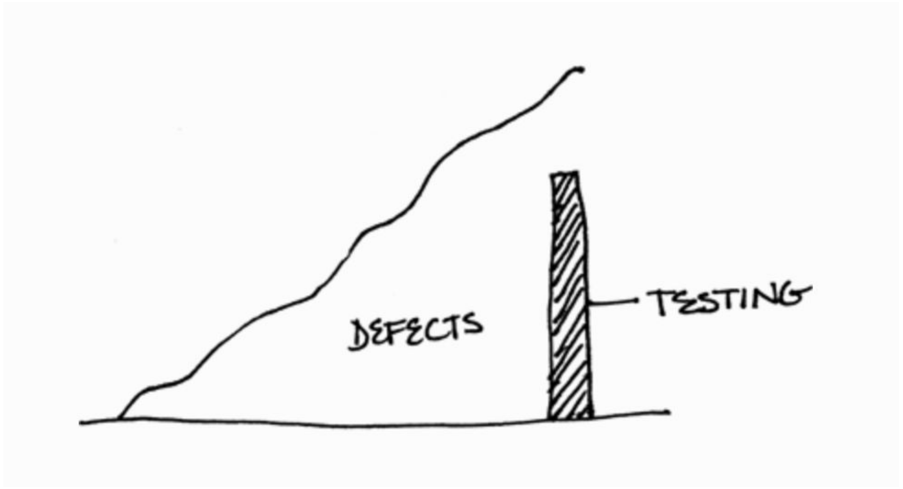
Why test our code?

Why test our code?

- Frequent testing reduces costs and defects
- Validates the user experience (customer journey)
- Increases feedback while reducing stress


I NOTICED YOU HAVE NO UNIT TESTS
I TOO LIKE TO LIVE DANGEROUSLY

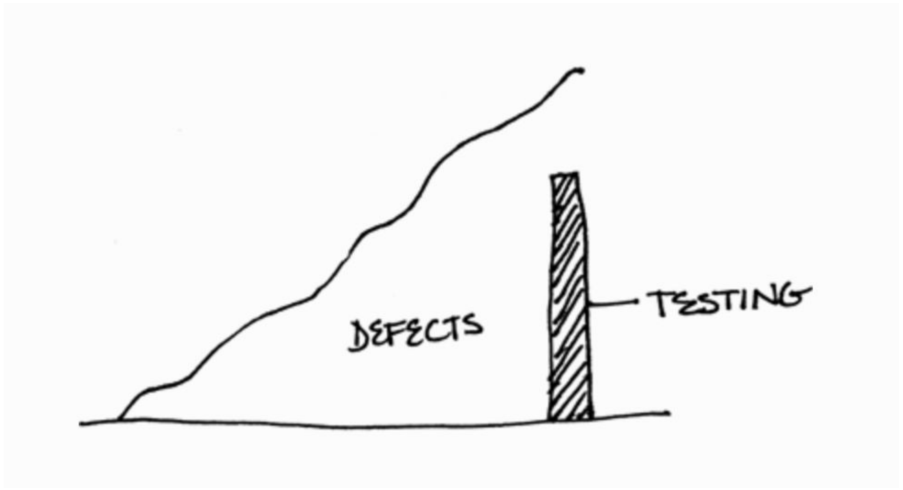
When to Test?



Late, expensive testing leaves many defects

Extreme Programming Explained

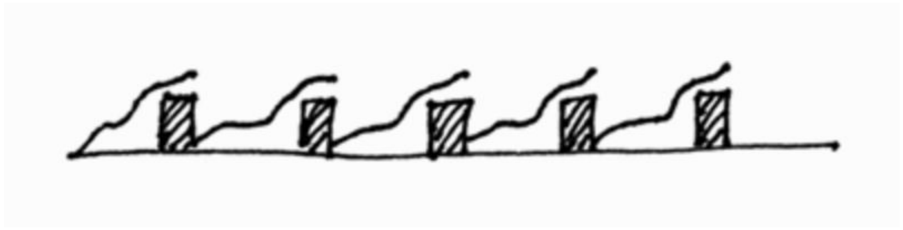
When to Test?



Late, expensive testing leaves many defects

Extreme Programming Explained

When to Test?



Frequent testing reduces costs and defects

Extreme Programming Explained

Automated or Manual Testing?

Manual testing is stressful and error-prone

Extreme Programming Explained

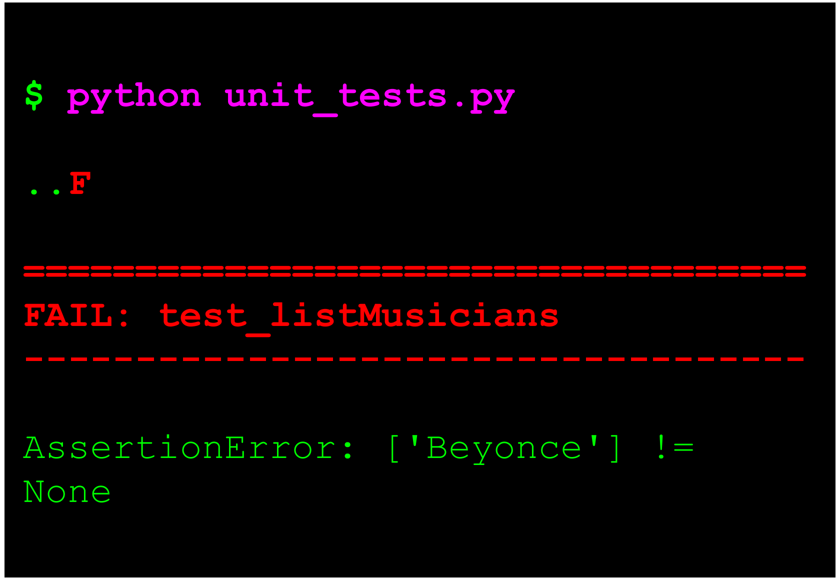
Automated or Manual Testing?

Manual testing is stressful and error-prone

Before Testing
After Testing
There's no such thing as a perfect app.
But there are countless under-tested apps.

Extreme Programming Explained

Automated or Manual Testing?

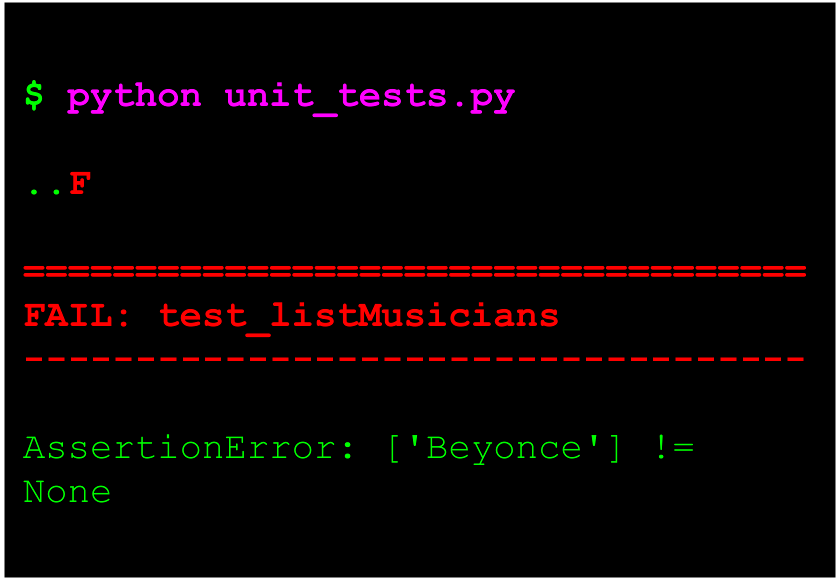


```
$ python unit_tests.py

..F

======================================
FAIL: test_listMusicians
--------------------------------------

AssertionError: ['Beyonce'] !=
None
```

Automating your tests means that a computer can run them the same way every time

Extreme Programming
Explained

Automated or Manual Testing?

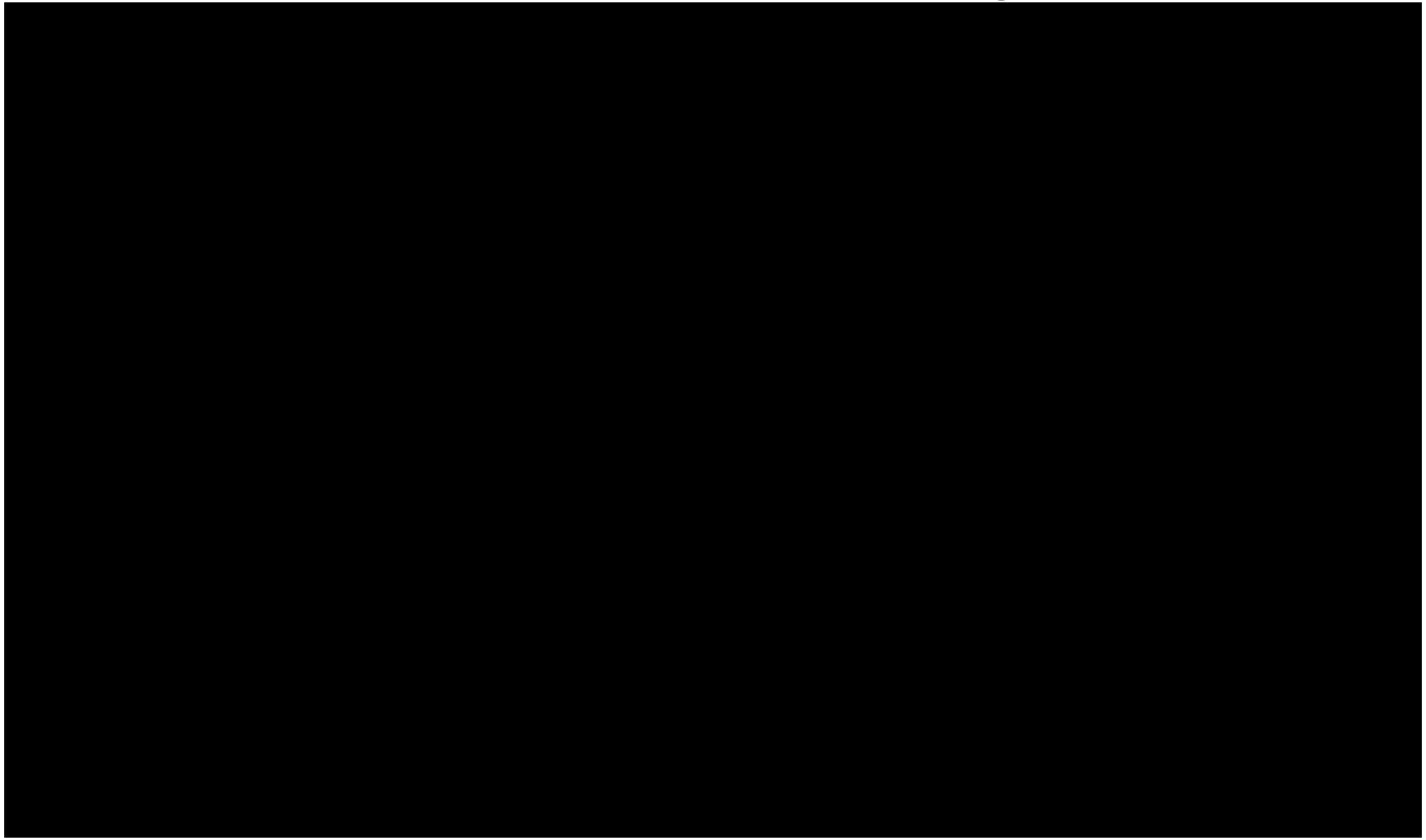


```
$ python unit_tests.py
..F
======================================
FAIL: test_listMusicians
--------------------------------------

AssertionError: ['Beyonce'] !=
None
```

Automating your tests means that a computer can run them the same way every time

Extreme Programming Explained

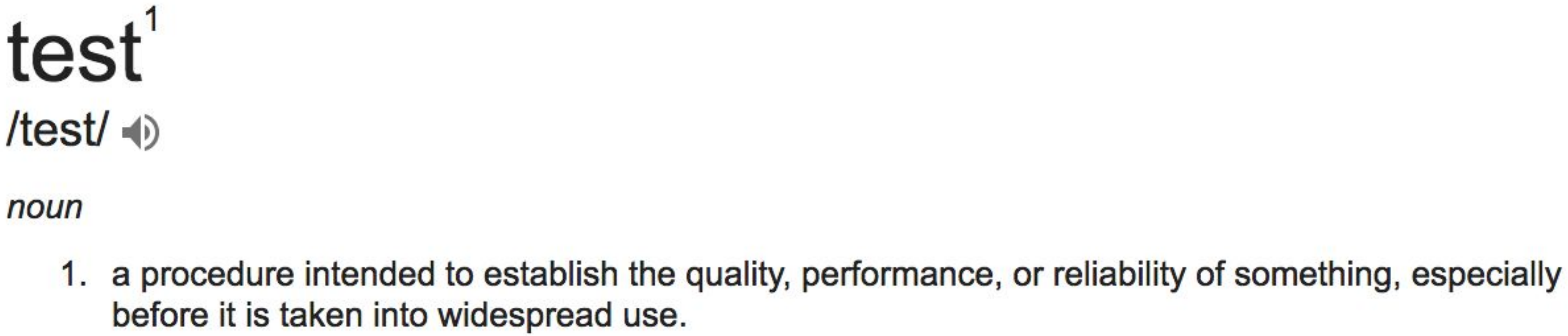
How it Works at Etsy



Extreme Programming
Explained

Test-Driven Development

Test-Driven?

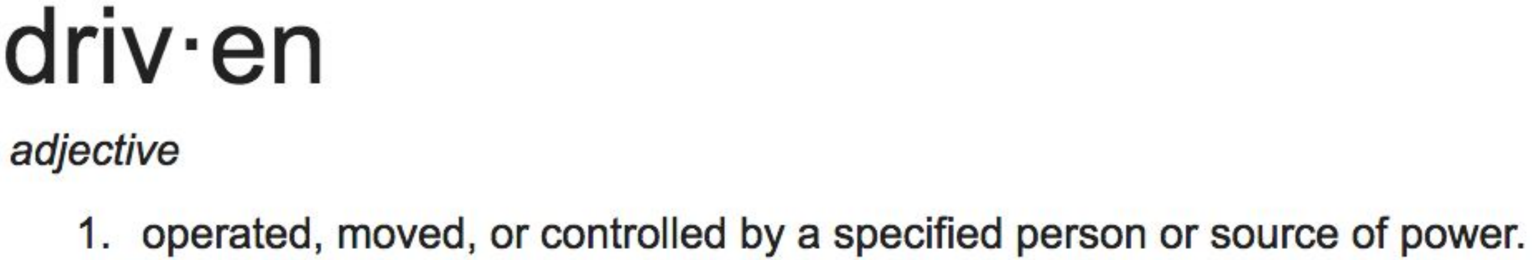


test^{1}

/test/

noun

1. a procedure intended to establish the quality, performance, or reliability of something, especially before it is taken into widespread use.



driv·en

adjective

1. operated, moved, or controlled by a specified person or source of power.

TDD = Red, Green, Refactor

Write a succinct failing test. Run it.
Make sure it fails.

Write just enough code to make the test pass.

Refactor to improve the code without changing behavior.

Workshop

Enter the Dojo: *The Tennis Scoring Kata*

Kata (型 or 形 literally: "form"), a Japanese word, are detailed choreographed patterns of movements practiced either solo or in pairs. The term form is used for the corresponding concept in non-Japanese martial arts in general.

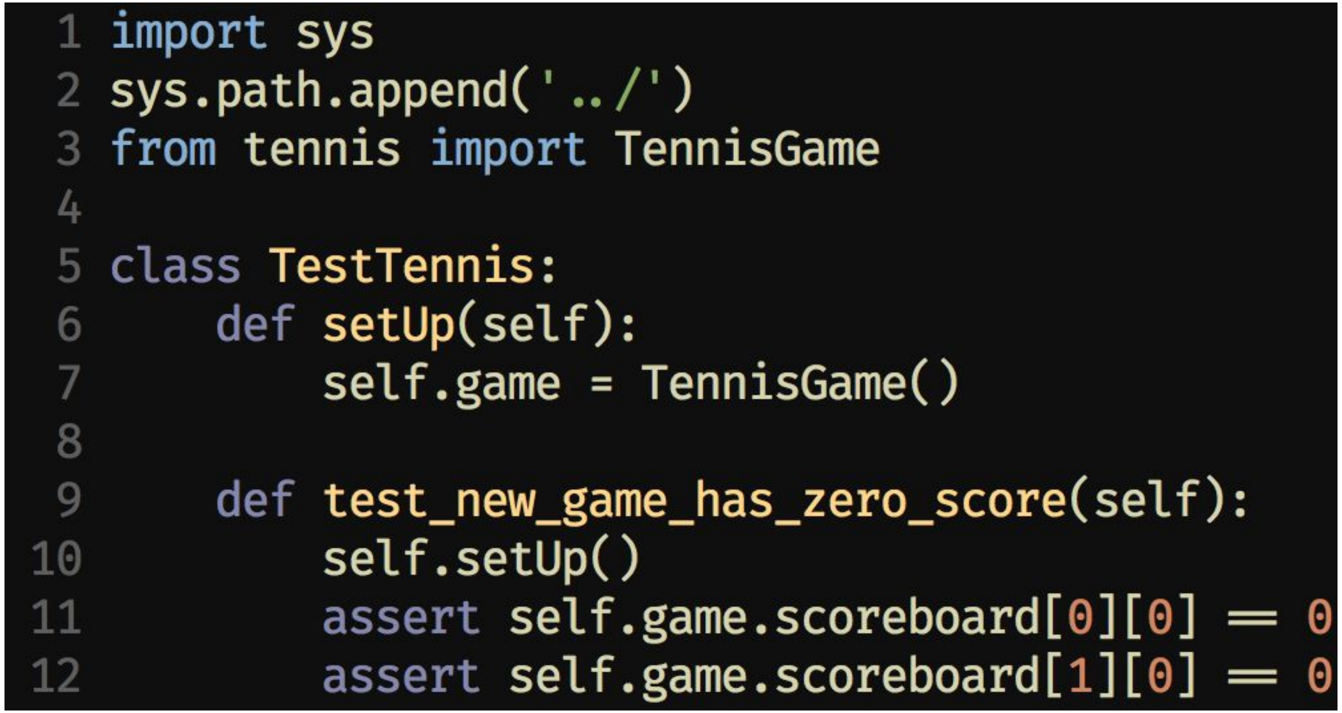
Today’s Dojo format:

Work in pairs: L/R
Use Python + Pytest (pip install pytest if you haven’t)

1. L writes a failing test
2. R writes just enough code to make it pass
3. L refactors the code to be sensible (often step 2 is “cheat” code)
4. R causes a red condition: either adjusting the pre-existing assertion or adding a new failing test
5. L writes just enough code to make it pass
6. R refactors the code to be sensible
7. Repeat from step 1.

Iteration 1, Step 1 (L):

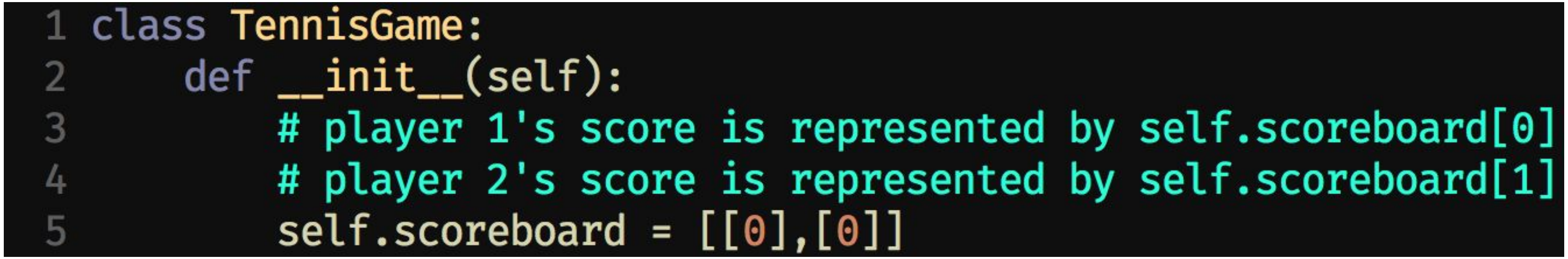
tennis_test.py:



```python
import sys
sys.path.append('../')
from tennis import TennisGame

class TestTennis:
    def setUp(self):
        self.game = TennisGame()

    def test_new_game_has_zero_score(self):
        self.setUp()
        assert self.game.scoreboard[0][0] == 0
        assert self.game.scoreboard[1][0] == 0
```

Iteration 1, Step 2 (R):

tennis.py:



```python
class TennisGame:
    def __init__(self):
        # player 1's score is represented by self.scoreboard[0]
        # player 2's score is represented by self.scoreboard[1]
        self.scoreboard = [[0],[0]]
```

Tennis Scoring Rules:

- 2 players
- 1 scoreboard
- Points are tallied as follows: 0, 15, 30, 40
- If a player has 40 points and scores again, they win!*

Unless... Deuce

- When both players have 40, the game enters a state called “Deuce.”
- When the game is in the Deuce state, a score by either player assigns the player “Advantage.”
- If a player has Advantage and scores again, they win!
- If the player *without* Advantage scores, the game state reverts back to Deuce. There is no limit to the amount of times a game can revert to Deuce!

Homework

Homework: add tests to your app

👉 **TODO: With your team, add tests to existing functionality for your team's app.**

👉 **TODO: Update your team's CircleCI config.yml to run your tests automatically whenever you push code to Git.**

→ Sample config.yml file is on the Syllabus on Github!