Testing Testing
1 2 4

Testing Testing

1 2 4 3

How do you know your program works?

- Well I used it for a while....
- I proved it with math!
- I wrote a program to test it!

Programs test other programs

System tests make sure an entire program does the expected thing.

Unit tests take smaller pieces of code and make sure they follow their spec.

Programs test other programs

Black box tests you write only thinking about the spec

Clear box tests you write thinking about both the spec and the code

What kind of tests?

- A test that simulates a user playing an entire game of hangman and losing
- A test that calls
  `word_with_blanks("potato", "to")`
  and expects the output to be `"_ot_to"`
- A set of tests for `make_guess` constructed to ensure we take every branch of the if statement

Writing tests in python

```
import unittest

class TestThing(unittest.TestCase):
    def test_stuff(self):
        a = thing(23)
        self.assertEqual(a, "foo")
        #...

if __name__ == '__main__':
    unittest.main()
```

Choosing test cases

- What is the full domain of allowable inputs?
- Divide them into categories that are "similar" to each other
- Pick one input from each category, and a corresponding expected output.

What makes input "similar" or not?

- Look at the spec for hints
- Troglodyte counting:

  0

  1

  "many"

- Trigger each **specified** error
- Input is "dissimilar" if result has to be "dissimilar"
- Treat input on the boundary between two regions as similar to neither.

```python
def abs(x):
    """Return the distance of the
       number x from 0
    """
    # some code goes here
```

```python
def word_with_blanks(word, successes):
    """Return the string word, but
    with any letter not present in the
    string successes replaced with an
    underscore"""
    result = ""
    for character in word:
        if character in successes:
            result += character
        else:
            result += '_'
    return result
```

Beware overdetermined tests

```
def bigger_prime(n):
  """Return a prime number bigger
     than the number n
  """
  # Some code goes here
#...
def test_bigger_prime(self):
  self.assertEqual(bigger_prime(3), 5)
```

Beware overdetermined tests

```
def bigger_prime(n):
  """Return a prime number bigger
     than the number n
  """
  # Some code goes here
#...
def test_bigger_prime(self):
  ~~self.assertEqual(bigger_prime(3), 5)~~
```

Beware overdetermined tests

```
def bigger_prime(n):
  """Return a prime number bigger
     than the number n
  """
  # Some code goes here
#...
def test_bigger_prime(self):
  bp = bigger_prime(3)
  self.assertGreater(bp, 3)
  self.assertTrue(is_prime(bp))
```