# UNIT TESTS

An Introduction to Computer Science

# Unit Tests

Make it easier to…

1. Find problems earlier
2. Change things later
3. Glue together code

# Example Unit Test

```
from cisc108 import assert_equal

def feet_to_inches(distance:int)->int:
    return distance * 12

assert_equal(feet_to_inches(2), 24)
```

Test module

Function

Unit Test

# Representative Cases

| Input | → | Output | |
|---|---|---|---|
| 1 | → | 12 | Positive Integer |
| .5 | → | 6.0 | Positive Float |
| 0 | → | 0 | Zero |
| -4 | → | -48 | Negative Integer |
| 10000 | → | 120000 | Large Positive Integer |

# Code Coverage

```
from cisc108 import assert_equal

def feet_to_inches(distance:int)->int:
    return distance * 12

assert_equal(feet_to_inches(2), 24)
assert_equal(feet_to_inches(.5), 6.0)
assert_equal(feet_to_inches(0), 0)
assert_equal(feet_to_inches(-4), -48)
assert_equal(feet_to_inches(10000), 120000)
```


# Failing Unit Tests

Function: `inches_to_feet()`

Given input: `1`

Expected output: `12`

Actual output: `-5`

`12 != 5`, you have an error in your calculation!

# Passing Unit Tests

**Feedback:** Incorrect Answer

Trace Variables

**Instructor Feedback**

Your `make_polite` function did not produce the correct output for the values `''`.

Expected: `', please'`

Actual: `'No'`

# Judging Unit Tests

```
from cisc108 import assert_equal

def add(left: int, right: int) -> int:
    return left + 4

# Valid tests shows the correct behavior
assert_equal(add(1, 4), 5)
assert_equal(add(3, 4), 7)

# Thorough tests expose the incorrect version's failure
assert_equal(add(4, 3), 7)
assert_equal(add(5, 5), 10)
```