

# **Test Cases: Finding Errors**

- Bug: Error in a program. (Always expect them!)
- Debugging: Process of finding bugs and removing them.
- · Testing: Process of analyzing, running program, looking for bugs.
- Test case: A set of input values, together with the expected output.

Get in the habit of writing test cases for a function from the function's specification—even *before* writing the function's body.

#### def number\_vowels(w):

"""Returns: number of vowels in word w

Precondition: w string w/ at least one letter and only letters  $^{\tt uuu}$  pass # nothing here yet!

### **Representative Tests**

- · Cannot test all inputs
  - "Infinite" possibilities
- Limit ourselves to tests that are **representative** 
  - Each test is a significantly different input
  - Every possible input is similar to one chosen
- An art, not a science
- If easy, never have bugs
- Learn with much practice

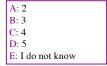
### Representative Tests for number\_vowels(w)

- Word with just one vowel
  - For each possible vowel!
- Word with multiple vowels
  - Of the same vowel
  - Of different vowels
- · Word with only vowels
- Word with no vowels

# **How Many "Different" Tests Are Here?**

number\_vowels(w)

INPUT	OUTPUT
'hat'	1
'charm'	1
'bet'	1
'beet'	2
'beetle'	3



#### 

White') gives 'White, Walker'

last\_name\_first('Walker

## **Unit Test: A Special Kind of Script**

- A unit test is a script that tests another module
  - It imports the other module (so it can access it)
  - It imports the cornell module (for testing)
  - It defines one or more test cases
    - · A representative input
    - · The expected output
- The test cases use the cornell function

def assert\_equals(expected,received):

"""Quit program if expected and received differ"""

### 

### **Using Test Procedures**

- In the real world, we have a lot of test cases
  - I wrote 1000+ test cases for a C++ game library
  - You need a way to cleanly organize them
- Idea: Put test cases inside another procedure
  - Each function tested gets its own procedure
  - Procedure has test cases for that function
  - Also some print statements (to verify tests work)
- Turn tests on/off by calling the test procedure

### **Test Procedure**

def test\_last\_name\_first():

"""Test procedure for last\_name\_first(n)"""
print("Testing function last\_name\_first')

result = name.last\_name\_first('Walker White')

$$\label{eq:cornell.assert_equals} \begin{split} & cornell.assert\_equals('White, Walker', result) \\ & result = name.last\_name\_first('Walker & White') \end{split}$$

cornell.assert\_equals('White, Walker', result)

# Execution of the testing code No tests happen test\_last\_name\_first() if you forget this

print('Module name is working correctly')