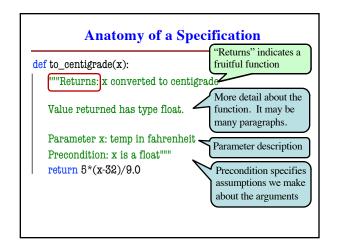
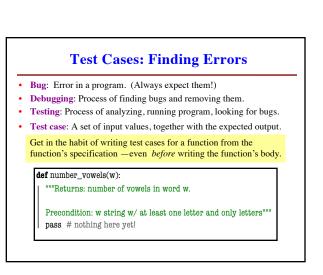


• Usage: s = u.read() # s is a string



Preconditions • Precondition is a promise >>> to_centigrade(32.0) If precondition is true, 0.0 the function works >>> to_centigrade(212) If precondition is false, 100.0 no guarantees at all >>> to_centigrade('32') • Get software bugs when Traceback (most recent call last): Function precondition is File "<stdin>", line 1, in <module> not documented properly File "temperature.py", line 19 ... Function is used in ways ${\tt TypeError: unsupported operand\ type}(s)$ that violates precondition for -: 'str' and 'int' Precondition violated



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

Running Example The following function has a bug: def last_name_first(n): """Returns: copy of <n> but in the form <last-name>, <first-name> Precondition: <n> is in the form <first-name> <last-name> with one or more blanks between the two names"" end_first = n.find(' ') first = n[:end_first] last = n[end_first+1:] Look at precondition return last+', '+first when choosing tests • Representative Tests: last_name_first('Walker White') give 'White, Walker' last_name_first('Walker White') gives 'White, 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 cornelltest module (for testing)
 - It defines one or more test cases
 - · A representative input
 - · The expected output
- · The test cases use the cornelltest function

def assert_equals(expected,received):

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

Testing last_name_first(n) import name # The module we want to test import cornelltest # Includes the test procedures # First test case result = name.last_name_first('Walker White') Quits Python ${\tt cornelltest.assert_equals('White, Walker', result)}^{\bullet}$ if not equal # Second test case result = name.last name first('Walker White') cornelltest.assert_equals('White, Walker', result) Message will print out only if no errors. print 'Module name is working correctly

