Announcements

- Announcements will be made in lecture about:

 - Assignment 1
 - Extra help
- Details were not finalized when this handout went to press.
- BE SURE TO CHECK THE LECTURE SLIDES FOR TODAY, at:

http://www.cs.cornell.edu/courses/cs1110/2017sp/lectures/index.php

Anatomy of a Specification

def greet(n):

One line description, followed by blank line

"""Prints a greeting to the name n

Greeting has format 'Hello <11>1 Followed by conversation starte many paragraphs.

More detail about the function. It may be

Parameter n: person to greet-Precondition: n is a string"" print 'Hello '+n+'!' print 'How are you?'

Parameter description Precondition specifies

assumptions we make about the arguments

Anatomy of a Specification

def to_centigrade(x):

"Returns" indicates a fruitful function

Returns: x converted to centigrade

More detail about the Value returned has type float. function. It may be many paragraphs.

Parameter x: temp in Fahrenhei Precondition: x is a float""

Parameter description

return 5*(x-32)/9.0 Precondition specifies assumptions we make about the arguments

Preconditions

- Precondition is a promise
 - If precondition is true, the function works
 - If precondition is false, no guarantees at all
- Get software bugs when
 - Function precondition is not documented properly
 - Function is used in ways that violates precondition
- >>> to_centigrade(32)
- >>> to_centigrade(212) 100.0
- >>> to_centigrade('32')

Traceback (most recent call last): File "<stdin>", line 1, in <module> File "temperature.py", line 19 ...

TypeError: unsupported operand type(s) for -: 'str' and 'int'

Precondition violated

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):

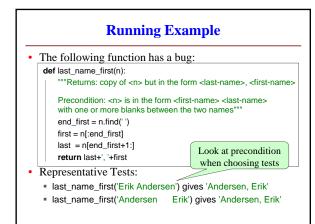
"""Returns: number of vowels in word w.

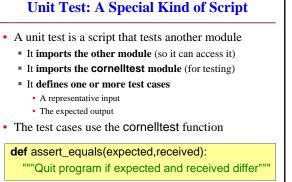
Precondition: w string w/ at least one letter and only letters""" pass # nothing here yet!

Representative Tests

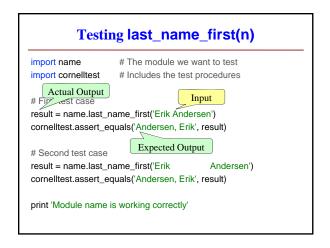
- Cannot test all inputs
- "Infinite" possibilities
- Limit ourselves to tests that are representative
 - · Each test is a significantly different input
 - Every possible input is similar to one chosen
- An art, not a science
 - If easy, never have bugs
 - Learn with much practice

- 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





Contains useful testing functions Need to download it and put in same folder as other files Available at: http://www.cs.cornell.edu/courses/cs1110/2017sp/lectures/02-14-17/modules/cornelltest.py



In the real world, we have a lot of test cases Must cleanly organize them Idea: Put test cases inside another procedure Each function tested gets its own procedure Procedure has test cases for that function

Using Test Procedures

Also some print statements (to verify tests work) Turn tests on/off by calling the 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('Erik Andersen') cornelltest.assert_equals('Andersen, Erik', result) result = name.last_name_first('Andersen Erik') cornelltest.assert_equals('Andersen, Erik', result) # Execution of the testing code test_last_name_first() _______ No tests happen if you forget this print 'Module name is working correctly'