Announcements

- Announcements will be made in lecture about:
  - Lab 3
  - Assignment 1
  - Extra help
- Details were not finalized when this handout went to press.

Anatomy of a Specification

def greet(n):
  """Prints a greeting to the name n
  Greeting has format 'Hello <n>!
  Followed by conversation starter.
  Parameter n: person to greet
  Precondition: n is a string"
  print 'Hello '+n+'!
  print 'How are you?'

One line description,
followed by blank line
More detail about the
function. It may be
many paragraphs.
Parameter description
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

Representative Tests

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:

```python
def last_name_first(n):
    """Returns: copy of <> but in the form <last-name>, <first-name>
    Precondition: <> 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:]
    return last+', '+first
```

- Representative Tests:
  - `last_name_first('Erik Andersen')` gives 'Andersen, Erik'
  - `last_name_first('Andersen      Erik')` gives 'Andersen, Erik'

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

```python
def assert_equals(expected,received):
    """Quit program if expected and received differ""
```

### Cornelltest Module

- 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](http://www.cs.cornell.edu/courses/cs1110/2017sp/lectures/02-14-17/modules/cornelltest.py)

Testing last_name_first(n)

```python
import name # The module we want to test
import cornelltest # Includes the test procedures

# First test case
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)

print 'Module name is working correctly'
```

Using Test Procedures

- 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
  - Also some print statements (to verify tests work)
- Turn tests on/off by calling the test procedure

Test Procedure

```python
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

```

No tests happen if you forget this

```python
print 'Module name is working correctly'
```