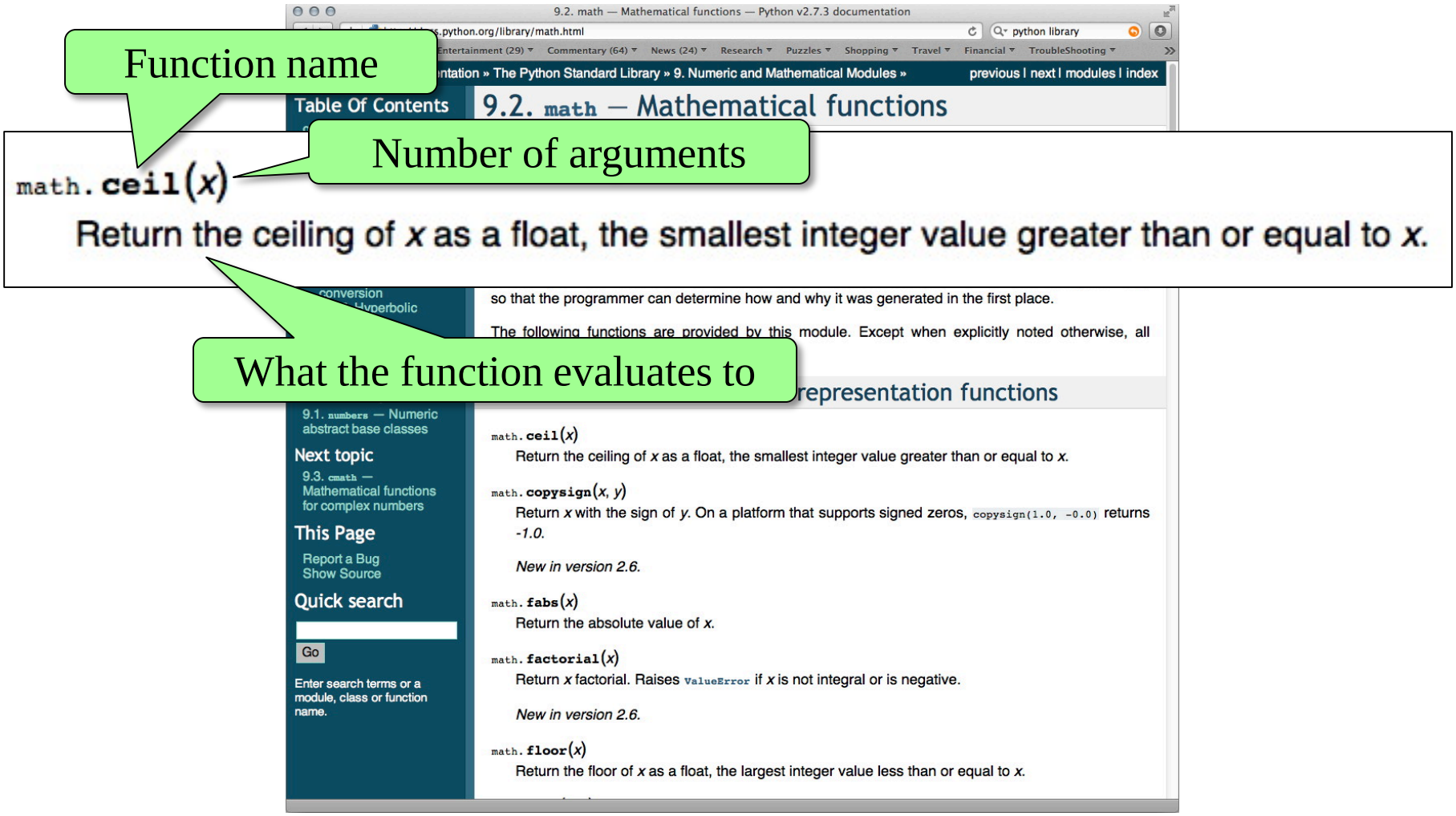
CS 1110:
Introduction to Computing Using Python

Lecture 6

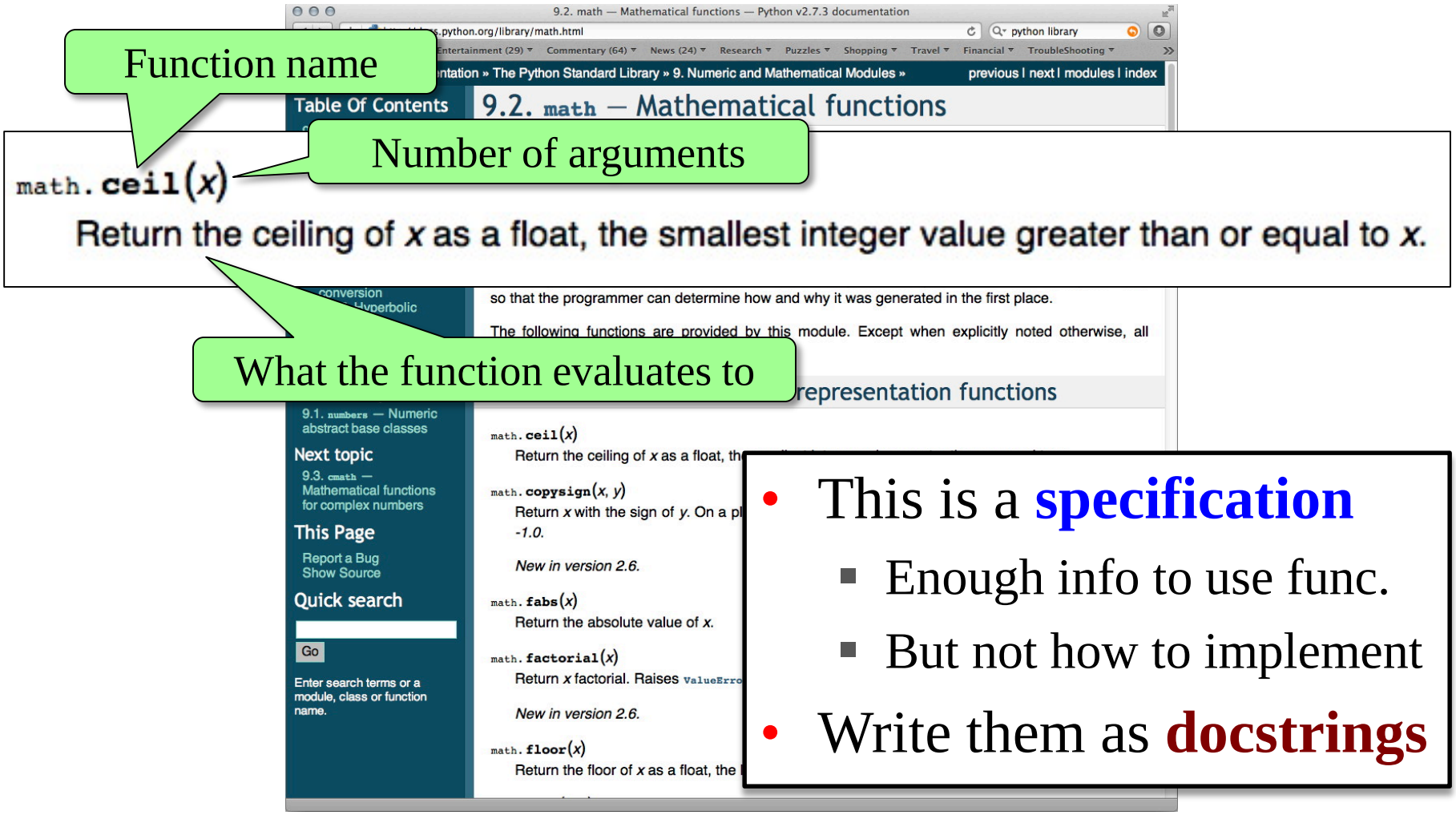
Specifications & Testing

[Andersen, Gries, Lee, Marschner, Van Loan, White]

Recall: The Python API



Recall: The Python API



- This is a **specification**
  - Enough info to use func.
  - But not how to implement
- Write them as **docstrings**

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

Anatomy of a Specification

```
def to_centigrade(x):
    """Returns: x converted to centigrade

    Value returned has type float.

    Parameter x: temp in Fahrenheit
    Precondition: x is a float"""
    return 5*(x-32)/9.0
```

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

Anatomy of a Specification

```
def to_centigrade(x):
    """Returns: x converted to centigrade

    Value returned has type float.

    Parameter x: temp in Fahrenheit
    Precondition: x is a float"""
    return 5*(x-32)/9.0
```

"Returns" indicates a fruitful function

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

```
>>> to_centigrade(32.0)
0.0
>>> 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

NASA Mars Climate Orbiter

Source: Mars Climate Orbiter Mishap Investigation Board Phase I Report

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!
```

Test Cases: Finding Errors

- **Bug**: Error in a program. (Always
- **Debugging**: Process of finding bug
- **Testing**: Process of analyzing, runn
- **Test case**: A set of input values, to

Get in the habit of writing test case
function's specification – even *bef*

Some Test Cases

- number_vowels('Bob')
  Answer should be 1
- number_vowels('Aeiuo')
  Answer should be 5
- number_vowels('Grrr')
  Answer should be 0

```
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!
```

Test Cases: Finding Errors

Some Test Cases

- number_vowels('y')
  Answer should be 0? 1?
- number_vowels('Bobo')
  Answer should be 1? 2?

Some Test Cases

- number_vowels('Bob')
  Answer should be 1
- number_vowels('Aeiuo')
  Answer should be 5
- number_vowels('Grrr')
  Answer should be 0

```
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

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:]
    return last+', '+first
```

Look at precondition when choosing tests

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

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

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 that each include:**
    - 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('Erik Andersen')
cornelltest.assert_equals('Andersen, Erik', result)

# Second test case
result = name.last_name_first('Erik          Andersen')
cornelltest.assert_equals('Andersen, Erik', result)

print 'Module name is working correctly'
```

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('Erik Andersen')
cornelltest.assert_equals('Andersen, Erik', result)

# Second test case
result = name.last_name_first('Erik          Andersen')
cornelltest.assert_equals('Andersen, Erik', result)

print 'Module name is working correctly'
```

Actual Output

Input

Expected Output

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('Erik Andersen')
cornelltest.assert_equals('Andersen, Erik', result)

# Second test case
result = name.last_name_first('Erik          Andersen')
cornelltest.assert_equals('Andersen, Erik', result)

print 'Module name is working correctly'
```

Quits Python if not equal

Message will print out only if no errors.

Using Test Procedures

- In the real world, we have a lot of test cases
  - 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('Erik Andersen')
    cornelltest.assert_equals('Andersen, Erik', result)
    result = name.last_name_first(Erik              Andersen')
    cornelltest.assert_equals('Andersen, Erik', result)

# Execution of the testing code
test_last_name_first()
print 'Module name is working correctly'
```

No tests happen if you forget this

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:]
    return last+', '+first
```

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

Debugging

```
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"""
    #get index of space after first name
1   space_index = n.find(' ')
    #get first name
2   first = n[:space_index]
    #get last name
3   last  = n[space_index+1:]
    #return "<last-name>, <first-name>"
4   return last+', '+first
```

Which line is "wrong"?

A: Line 1

B: Line 2

C: Line 3 **CORRECT**

D: Line 4

E: I do not know

- last_name_first('Erik Andersen') gives 'Andersen, Erik'
- last_name_first('Erik Andersen') gives ' Andersen, Erik'

Using print statements to debug

```
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"""“
    #get index of space
    space_index = n.find(' ')
    #get first name
    first = n[:space_index]
    #get last name
    last  = n[space_index+1:]
    #return “<last-name>, <first-name>”
    return last+', '+first
```

What happens when I run this?

```
"""Unit test for the module string example

Tests my function from class 2/9"""
import cornelltest      # cornelltest assert functions
import string_example           # function to be tested


def test_firstparens():
    """Test procedure for firstparens"""
    print 'Testing firstparens'
    # Test case 1
    result = string_example.firstparens('A (B) C (D)')
    cornelltest.assert_equals('B',result)

    # Test case 2
    result = string_example.firstparens('A B (C)')
    cornelltest.assert_equals('C',result)


# Script code
print 'Working correctly'
```

Did not call the function!!

```
def firstparens(text):
    """Returns: substring in ()
    Uses the first set of parens
    Param text: string
    Precondition: a string with at least one complete () pair before (
    """

    #first open parenthesis
    first_open_parenthesis = 2

    #first close parenthesis
    first_close_parenthesis = text.index(")")

    #string slice
    result = text[first_open_parenthesis+1:first_close_parenthesis]

    #return result
    return result
```

I intentionally broke it

CORRECT

A: First test case fails
B: Second test case fails
C: Prints ‘Working correctly’

Good news about Assignment A1 & Lab 3

[They're posted --- Happy Valentine's Day]

1. This week: lab 3 out, but you have <u>two</u> weeks to do it, and it helps you with A1. (Part of A1 ***is*** Lab 3)

2. Next week: no new lab to do.
   All Wed. Feb 22 labs are drop-in office hours open to all.
   (Nobody at the Tuesday labs – break).

3. This week through early March: optional one-on-one with a staff member to help *just you* with lab 3, etc.
   Sign up for a slot on CMS under the "Special" "assignment"

4. After the due date, you'll have multiple opportunities to revise to get a perfect. Last opportunity to submit is March 2nd.

CS 1110 Spring 2017, Assignment 1: Currency Conversion*

http://www.cs.cornell.edu/courses/cs1110/2017sp/assignments/hw1.pdf

February 14, 2017

Figure 1: BTC stands for the cryptocurrency Bitcoin, but we sorta wish it stood for BaTCoin.

Thinking about that trip overseas? If you can swing it, it is best to go when the exchange rate is in your favor, i.e., when your dollars buy more in the foreign currency. So, it would be nice to have a function that, given your current amount of cash in US dollars, tells you how much your money is worth in another currency.

However, there is no set mathematical formula to compute this conversion. The value of one currency with respect to another is constantly changing. In fact, in the time that it takes you to read this paragraph, the exchange rate between the dollar and the Euro has probably changed several times. How can we possibly write a program to handle something like that?

One solution is to make use of a *web service*. A web service is a program that, when you send it web requests, l tell us hods to

Important instructions: The Rules and how to get credit

Primary learning objectives. You will exercise the following: use of string operations and methods on a real-world problem; use of iterative development and testing for a larger-scale project than we have tackled before.

Navigating links in this pdf. Text in any shade of blue in this handout is a clickable link.

Contents

1 Rules **2**
- 1.1 How To Partner (You Only Get One) . . . 2
- 1.2 What Collaborations Are (Dis-)Allowed And How To Document Them . . . 2
- 1.3 Python You Are NOT Allowed To Use In This Assignment . . . 2

2 Getting Credit: You Have Multiple Tries to Get All The Points **3**
- 2.1 Start Early! You'll All Be Working With the Same Machine! . . . 3
- 2.2 Initial File Submission Before The Initial Deadline . . . 3

Partnering (See section 1.1)

- You may do this assignment with *at most one* other person
- If you choose to work with a partner, *before either of you submit any files*, the two of you must link your A1 files/fates on CMS.
- If your partnership dissolves, there are special “group divorce” procedures you must follow

Academic Integrity Rules Gloss (1.2)

- Never look at another else's code.
- Never show your code (except course staff).
- DO specifically acknowledge by name all help you received, whether or not it was "legal"

Submit early and often

- Your initial solutions must be submitted to CMS by Thursday, February 23rd at 11:59pm.
- But, we urge you to first submit whatever preliminary progress you have to CMS by 2pm.
  You can replace older submissions with improved ones up to the deadline.

  - This will give you practice with CMS and provide you a chance to alert us during business hours if any problems arise.
  - Since you've been warned to submit early, do not expect that we will accept work that doesn't make it onto CMS on time, for whatever reason,