Orange on slides = significant changes from the handout (which goes to print early)

Announcements

All instructor/TA/consultant office hours now posted; see Staff page.

Can already program, just want to pick up Python? Take CS1133, Transition to Python, instead. 5 weeks S/U; doesn't fulfill college requirements. Contact Craig Frey, ccf27@cornell.edu, for more info.

AEWs start this week. Can still enroll on Student Center. Contact Anshul Sacheti (AEW Lead), as885@cornell.edu, and/or follow the link from the CS1110 webpage for more info.

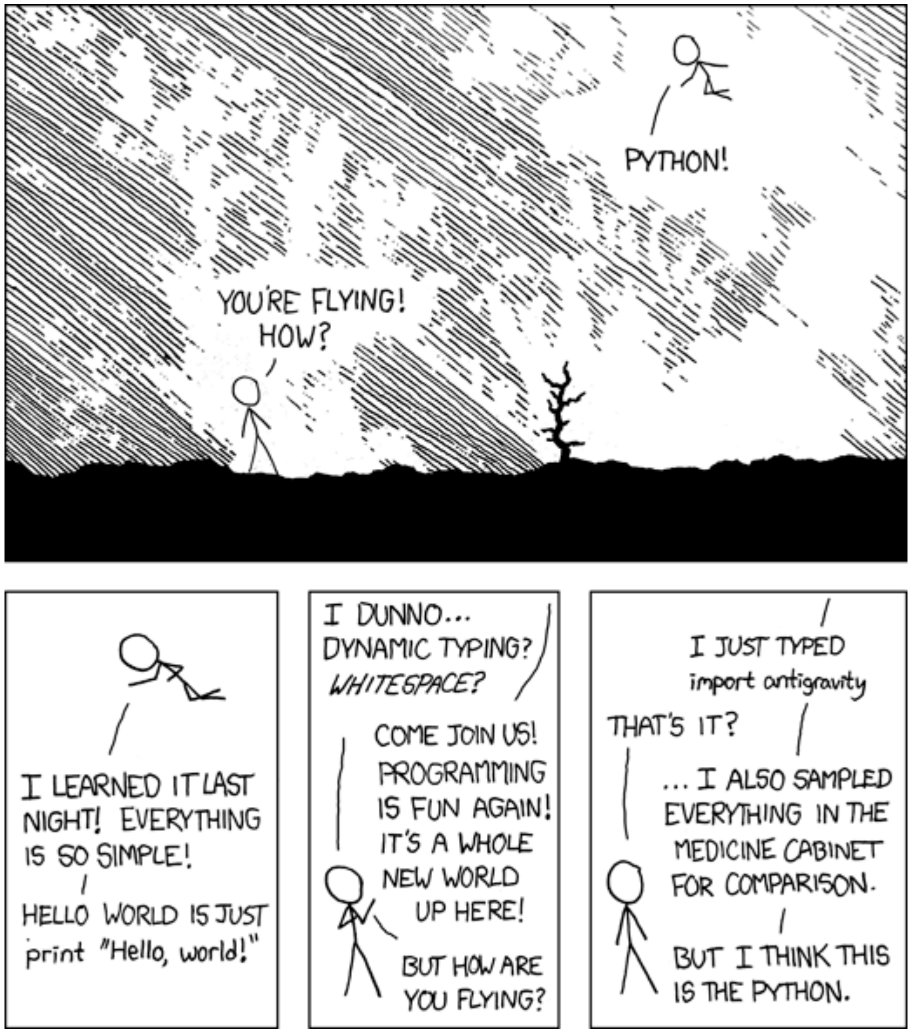
Added late, missed lab 1? Do it on your own ASAP, bring to your section TA to check it in at *the beginning of this week's lab*. Everything, including lecture video and handouts, at http://www.cs.cornell.edu/courses/cs1110/2013sp

Lab 1 grades posted on CS1110 CMS. Yours missing? Contact your lab TA (Who? see Times & Places page). Not in CS1110 CMS? See FAQ page.

Moved to a later lab as requested to make room for other students? *THANK YOU!!*

Readings for this week

- 2.6, python files (referred to there as "scripts"); 2.9, comments; 4.9, docstring
- 3.3, for `import` and dot notation
- 8.1, 8.2, 8.4, 8.5, 8.8, about string operations, including the dot notation. Don't worry about the "method" terminology (yet).
- 3.1-3.6, about functions


PYTHON!
YOU'RE FLYING! HOW?
I LEARNED IT LAST NIGHT! EVERYTHING IS SO SIMPLE!
HELLO WORLD IS JUST print "Hello, world!"
I DUNNO... DYNAMIC TYPING? WHITESPACE?
COME JOIN US! PROGRAMMING IS FUN AGAIN! IT'S A WHOLE NEW WORLD UP HERE!
BUT HOW ARE YOU FLYING?
I JUST TYPED import antigravity
THAT'S IT?
... I ALSO SAMPLED EVERYTHING IN THE MEDICINE CABINET FOR COMPARISON.
BUT I THINK THIS IS THE PYTHON.

Slides by D. Gries, L. Lee, S. Marschner, W. White

[xkcd.com]

Operations for Extracting Data from Strings (from last lecture's handout)

- s = 'abc d'

0	1	2	3	4
a	b	c		d

- Access portions with []
  - s[0] is 'a'
  - s[4] is 'd'
  - s[5] causes an error
  - s[0:2] is 'ab' (excludes c)
  - s[2:] is 'c d'
- Called "string slicing"

Better/more compact style

s = 'abracadabra'

A '#' marks a *comment* for the reader *(including the code's author)*. Python ignores the rest of the line.

# the following all evaluate to True

'a' in s ~~== True~~

'cad' in s ~~== True~~

not('foo' in s) ~~== False~~

s.index('a') == 0

s.index('rac') == 2

s.count('a') == 5

len(s) == 11

s.strip('a') == 'bracadabr'

' cs1110 '.strip() == 'cs1110'

A String Puzzle (Extraction Practice)

Given: variable data contains a string with at least two 'L's.

Example: data='PROF. LILLIAN LEE'

Goal: give an expression for the part of the string starting with the **2nd** 'L'. *(How can we use the index operation?)*

(1) Store in variable i the index of the first 'L'.

```
i = data.index('L')
```

(2) Store in variable tail the part of data starting *after* i

```
tail = data[i+1:]
```

(3) Give an expression for the part of tail starting with 'L'

```
tail[tail.index('L'):]
```

Given: `info` contains a comma-separated string with last name, difficulty, execution, and penalty.

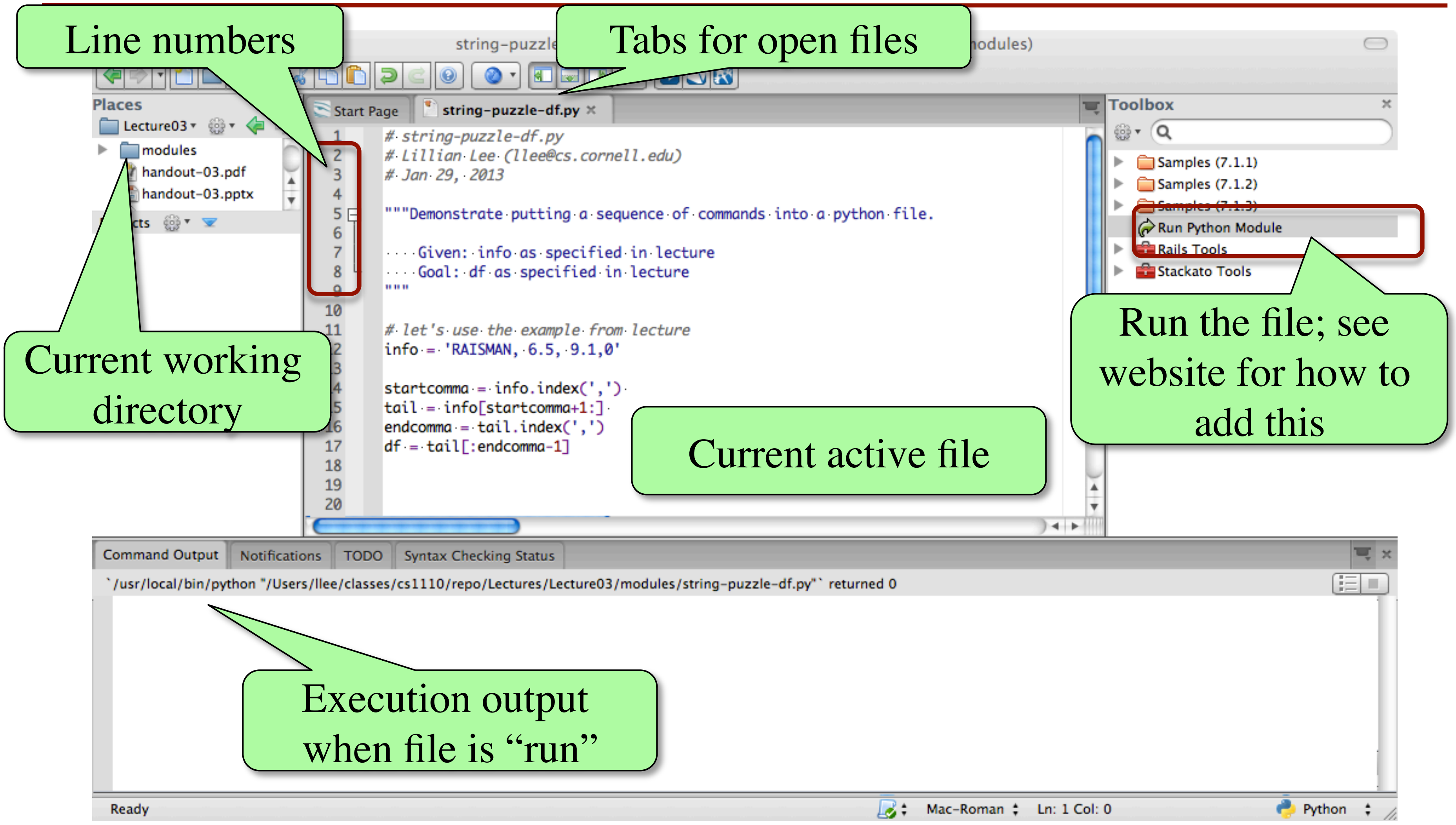
- *Example:* `info = 'RAISMAN, 6.7, 9.1,0'`

Goal: store the difficulty as a string, with no extra spaces or punctuation, in variable `df`

Where, in the following sequence of commands, does the first (conceptual) error occur?

A: `startcomma = info.index(',')`
B: `tail = info[startcomma+1:]`
C: `endcomma = tail.index(',')`
D: `df = tail[:endcomma-1]`
E: this sequence achieves the goal

Writing a Python File in Komodo



```
# string-puzzle-df.py
# Lillian Lee (llee@cs.cornell.edu)
# Jan 29, 2013

"""Demonstrate putting a sequence of commands into a python file.

    Given: info as specified in lecture
    Goal: df as specified in lecture
"""

# let's use the example from lecture
info = 'RAISMAN, 6.5, 9.1,0'

startcomma = info.index(',')
tail = info[startcomma+1:]
endcomma = tail.index(',')
df = tail[:endcomma-1]
```

```
`/usr/local/bin/python "/Users/llee/classes/cs1110/repo/Lectures/Lecture03/modules/string-puzzle-df.py"` returned 0
```

Req'd Format for CS1110 Python Files

```
# string-puzzle-df.py
# Lillian Lee (LJL2@cornell.edu)
# Tue Jan 29, 2013

""" Demonstrates putting a sequen
    of commands into a ...."""

startcomma = info.in
# more stuff ...
df
```

Header:
file name, authoring info

Docstring: *note the triple quotes*. Multi-line comment explaining the purpose & function of the file.

Note: Unlike with the command prompt, evaluating an expression produces nothing when a Python file (script, module) is run. The author probably wanted `print df` here.

Given: info contains a comma-separated string with last name, difficulty, execution, and penalty.

- *Example:* `info = 'RAISMAN, 6.7, 9.1,0'`

Goal: store the difficulty as a string, with no extra spaces or punctuation, in variable df

Where, in the following sequence of commands, does the first (conceptual) error occur? **(We'll post correct code)**

A: `startcomma = info.index(',')`

B: `tail = info[startcomma+1:]` **+2 instead, or use**

C: `endcomma = tail.index(',')`

D: `df = tail[:endcomma-1]` **tail[:endcomma].strip()**

E: this sequence achieves the goal

Function Calls

- Python supports expressions with math-like functions
- Function expressions have the form $\underbrace{\textbf{fun}}_{\textbf{function name}}(\underbrace{x}_{\textbf{argument}},y,\ldots)$

- Examples of *built-in* functions:
  - Numerical functions: `round(`*`number`*`)`, `pow(`*`base, exp`*`)`
  - Getting user input: `raw_input()`
  - Help function: `help()`

Using a Function From Another File (such files are called *modules*)

Example: what if we want 'Raisman', not 'RAISMAN'?

Lucky us: someone has written a module (file) named `string` that contains a function `capwords`.

```
~~name = info[:info.index(',')]      # name contains 'RAISMAN'~~
import string                        # Tell Python to access this module
print string.capwords(info)  # use the string module's capwords
```

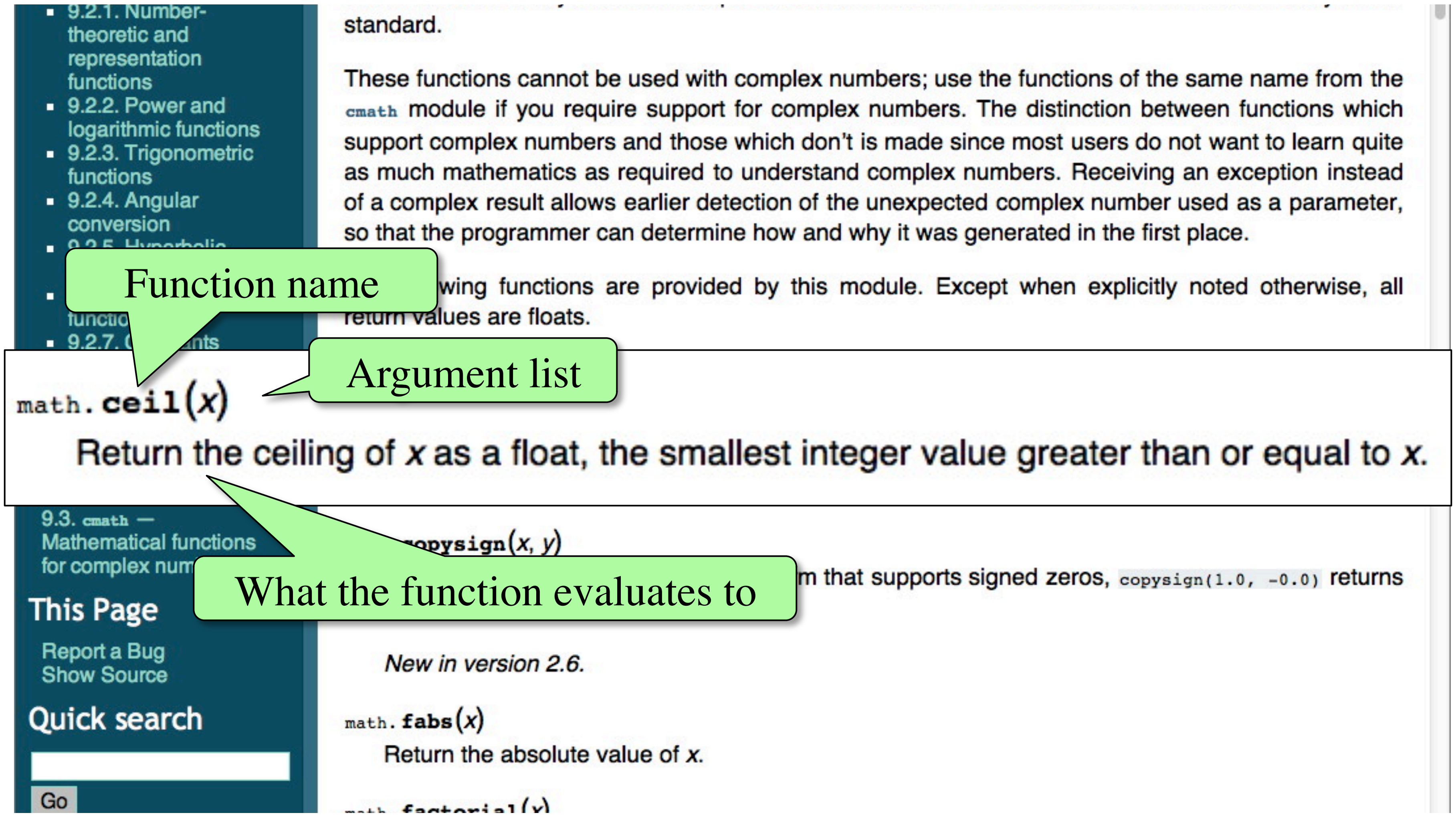
Grouping related functions and code into files is an important organizational principle.

Python Comes with Many Modules

- `io`
  - Read/write from files
- `math`
  - Mathematical functions
- `random`
  - Generate random numbers
  - Can pick any distribution
- `string`
  - Useful string functions
- `sys`
  - Information about your OS

- Complete list:
- http://docs.python.org/library
- **Library**: built-in modules
  - May change each release
  - Why version #s are an issue

Reading the Python Documentation



Function name
Argument list
What the function evaluates to
9.2.1. Number-theoretic and representation functions
9.2.2. Power and logarithmic functions
9.2.3. Trigonometric functions
9.2.4. Angular conversion
9.3. cmath — Mathematical functions for complex num
This Page
Report a Bug
Show Source
Quick search
Go

standard.

These functions cannot be used with complex numbers; use the functions of the same name from the `cmath` module if you require support for complex numbers. The distinction between functions which support complex numbers and those which don't is made since most users do not want to learn quite as much mathematics as required to understand complex numbers. Receiving an exception instead of a complex result allows earlier detection of the unexpected complex number used as a parameter, so that the programmer can determine how and why it was generated in the first place.

wing functions are provided by this module. Except when explicitly noted otherwise, all return values are floats.

`math.ceil(x)`

Return the ceiling of *x* as a float, the smallest integer value greater than or equal to *x*.

`opysign(x, y)`

m that supports signed zeros, `copysign(1.0, -0.0)` returns

New in version 2.6.

`math.fabs(x)`

Return the absolute value of *x*.

`math.factorial(x)`