

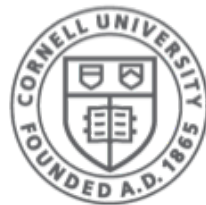
<http://www.cs.cornell.edu/courses/cs1110/2018sp>

Lecture 1: Introduction, Types & Expressions

(Chapter 1, Section 2.6)

CS 1110

Introduction to Computing Using Python



Cornell CIS
COMPUTING AND INFORMATION SCIENCE

[E. Andersen, A. Bracy, D. Gries, L. Lee, S. Marschner, and W. White]

CS 1110 Spring 2018: Announcements

<http://www.cs.cornell.edu/courses/cs1110/2018sp>

Sections

- Please go only to the Section you are enrolled in
- See our Section Swapping Station on Piazza:

<https://piazza.com/class/jckqwmqflaz6i?cid=10>

Enrollment

- There is a lot of turnover in the first week. Don't give up!
- Perhaps another class meets your needs?

<http://www.cs.cornell.edu/courses/cs1110/2018sp/resources/alternatives.php>

AEW Workshops (ENGRG 1010) Open to **all** students.

Additional (optional) discussion course. Small group, collaborative learning. Non-remedial. Highly recommended.

<http://www.cs.cornell.edu/courses/cs1110/2018sp/resources/aew.php>

Interlude: Why learn to program?

(subtly distinct from, although a core part of, CS / IS)

*Like philosophy, computing qua **computing is worth teaching** less for the subject matter itself and more **for the habits of mind that studying it encourages.***

The best way to encourage interest in computing in school is to ditch the vocational stuff ..., give the kids a simple programming language, and then get out of the way and let them experiment. For some, at least, it could be the start of a life-long love affair.

“Teach computing, not Word”, the Economist

http://www.economist.com/blogs/babbage/2010/08/computing_schools

Interlude (continued)

[T]he seductive intellectual core of... programming: here is a magic black box. [T]ell it to do whatever you want, within a certain set of rules, and it will do it; **within the confines of the box you are more or less God, your powers limited only by your imagination. But the price of that power is strict discipline: you have to *really know* what you want, and you have to be able to express it clearly in a formal, structured way** that leaves no room for the fuzzy thinking and ambiguity found everywhere else in life...

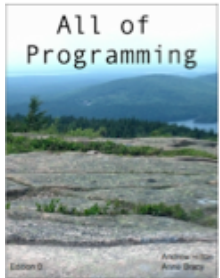
...The ability to make the machine dance to any tune you care to play is thrilling.

Oh the places you'll go! (with 1110)

Benjamin Van Doren, CALS

- bird lover since 3rd grade
- learned programming as a freshman in CS1110 Spring 2013
- helped create dataset for paper he co-authored: "Approximate Bayesian Inference for Reconstructing Velocities of Migrating Birds from Weather Radar"
- won Best Paper Award at AAAI 2013

About Professor Bracy



- BA, German Studies; BS, Symbolic Systems
- MS, Computer Science
- PhD, Computer Science
- Research Scientist, Intel Labs
- Principal Lecturer, WUSTL
- Co-Author of “[All of Programming](#)”
 - Google Play Book, Coursera Course!
- Senior Lecturer, Cornell University
 - CS 1110, 3410, 4410/4411
 - ACSU Faculty of the Year, 2016
 - Engineering Teaching Award, 2017

About Professor Lee

Lifetime achievement awards:

- Association for the Advancement of Artificial Intelligence, 2013
- Association for Computational Linguistics, 2017

In the press: New York Times, All Things Considered, Washington Post, etc.

Engineering teaching awards 1999, 2002, 2014;

Carpenter Memorial Advising Award 2009

- A.B. Cornell '93, Ph.D. Harvard '97

Lowest grade ever...?

Who does what?

What you see:

What you don't see:



Why should you take CS 1110?

Outcomes:

- **Fluency:** (Python) procedural programming
 - Use assignments, conditionals, & loops
 - Create Python modules & programs
- **Competency:** object-oriented programming
 - Recognize and use objects and classes
- **Knowledge:** searching & sorting algorithms

Intro Programming Classes Compared (1)

CS 1110: Python

- No programming experience necessary
- No calculus
- Non-numerical problems
- More about software design

CS 1112: MATLAB

- No programming experience necessary
- 1 semester of calculus
- Engineering-type problems
- Less about software design

Both serve as a pre-requisite to CS 2110

Intro Programming Classes Compared (2)

CS 1133: Python Short Course

- No programming experience necessary
- No calculus
- Very basics of programming
- Already full! ☹️

CS 1380: Data Science For All

- No programming experience necessary
- No calculus
- Less programming than 1110, but also: data visualization, prediction, machine learning

Why Python?

Low overhead

- Little to learn before you start “doing”
- Easier for beginners
- Designed with “rapid prototyping” in mind

Highly relevant to non-CS majors

- NumPy and SciPy heavily used by scientists

A modern language

- Popular for web applications (e.g. Facebook apps)
- Applicable to mobile app development

Course Website

<http://www.cs.cornell.edu/courses/cs1110/2018sp>

LOOK FOR THE SPRING 2018 PEGASUS!



No Pegasus? → *wrong semester*

Notice: link to CMS, not Blackboard

Communication

cs1110-prof@cornell.edu

- Includes: two profs, head TAs
- **Main correspondence.** Don't email only one prof, or both separately

cs1110-staff@cornell.edu

- Includes: both profs, admin assistant, graduate TAs, head consultants
- **“Emergency contact number.”** Nobody at office hours; Lab has no printouts, *etc.*

Piazza: not required, but fast (link on class website)

Email from us: please check your spam filters for mail from **AWB93, LJL2, cs1110-prof**, or with [CS1110] in the subject line.

Lectures

Lectures:

- Not just talking! Demos, clicker questions, etc.
- Every Tuesday/Thursday (9:05 or 11:15)
- Attend *either*, 11:15 is recorded by **VideoNote**
- Handouts (*including this one!*) posted to website afternoon before class
- Slides and code posted to the website after class

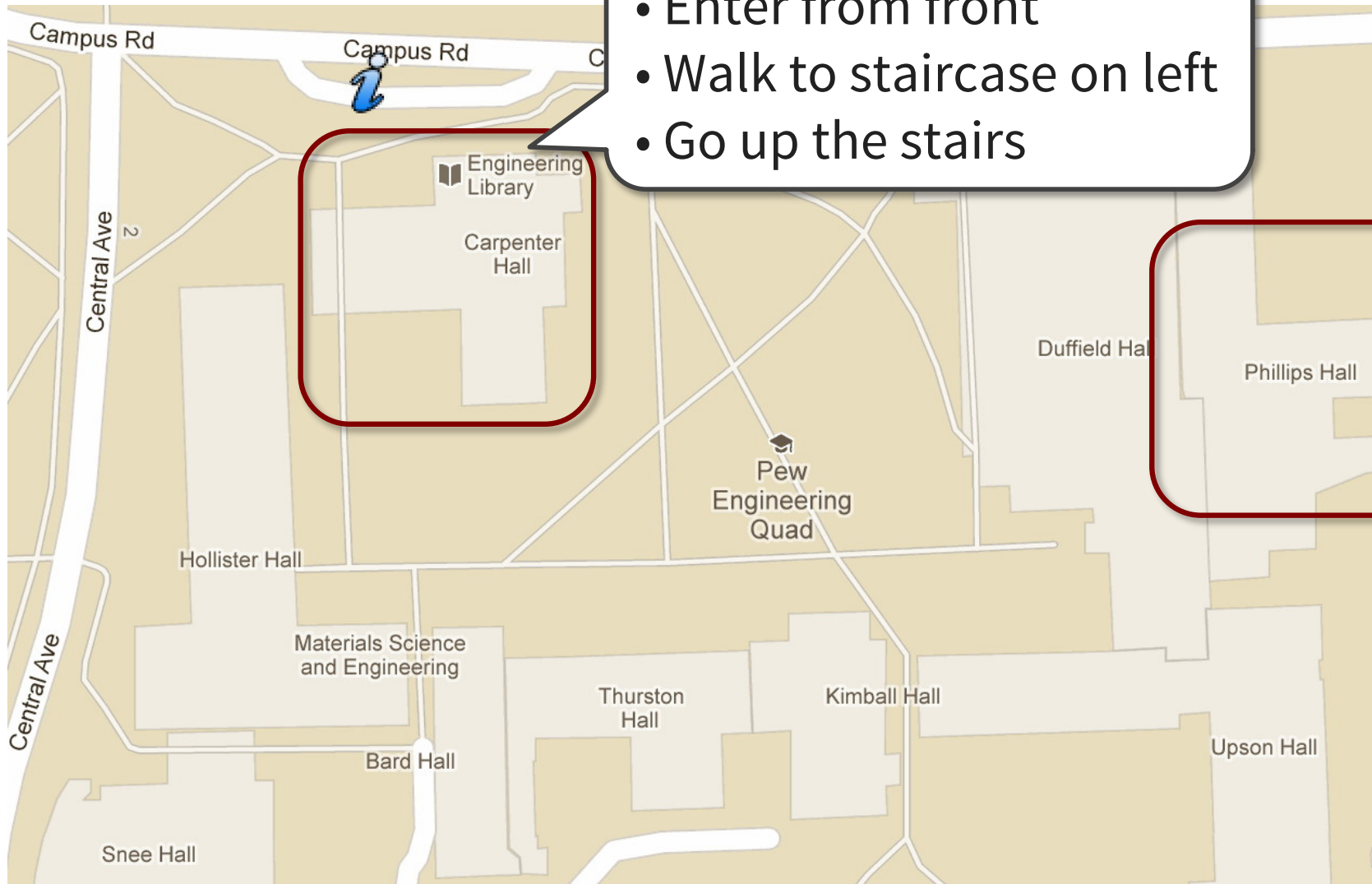


***Please, no cell phones during lecture
(except for during a Clicker question)***

Lab Sections (aka Sections)

- guided exercises with TAs & consultants
- Start Tuesday, January 30
- **Go to the lab section you are registered for.** We can't maintain workable staff/student ratios otherwise.
- Need a different Section? See our Section Swapping Station on Piazza: <https://piazza.com/class/jckqwmqflaz6i?cid=10>
- **Not enrolled in a lab section? *Don't panic.*** Do the lab on your own. If a lab section opens up, check it in then.
- Handouts posted to the website the Monday before
- **Mandatory.** Missing > 2 can lower your final grade.

ACCEL Labs



Computers available for you to use whenever labs are open (see website FAQ). Bring a USB stick to save your work b/c you can't save files on these machines.

Class Materials

sash means 2nd ed

Textbook. *Think Python, 2nd ed.* by Allen Downey

- *Supplemental*; does not replace lecture
- Available for free as PDF or eBook
- First edition is for the Python 2 (bad!)



iClicker. Optional but useful.

- Will periodically ask questions during lecture
- **Not** part of the grade → no registration
- We do support REEF Polling.

Python. Necessary if using your own computer

- See course website for how to install

Things to do before next class

1. Read textbook
 - Chapter 1
 - Sections 2.1-2.3, 2.5
2. (If using your own computer) Install Python **following our instructions:**
<http://www.cs.cornell.edu/courses/cs1110/2018sp/materials/python.php>
3. Look at first lab handout (available Monday)
4. (optional) Join Piazza, a Q&A forum

Everything is on website!

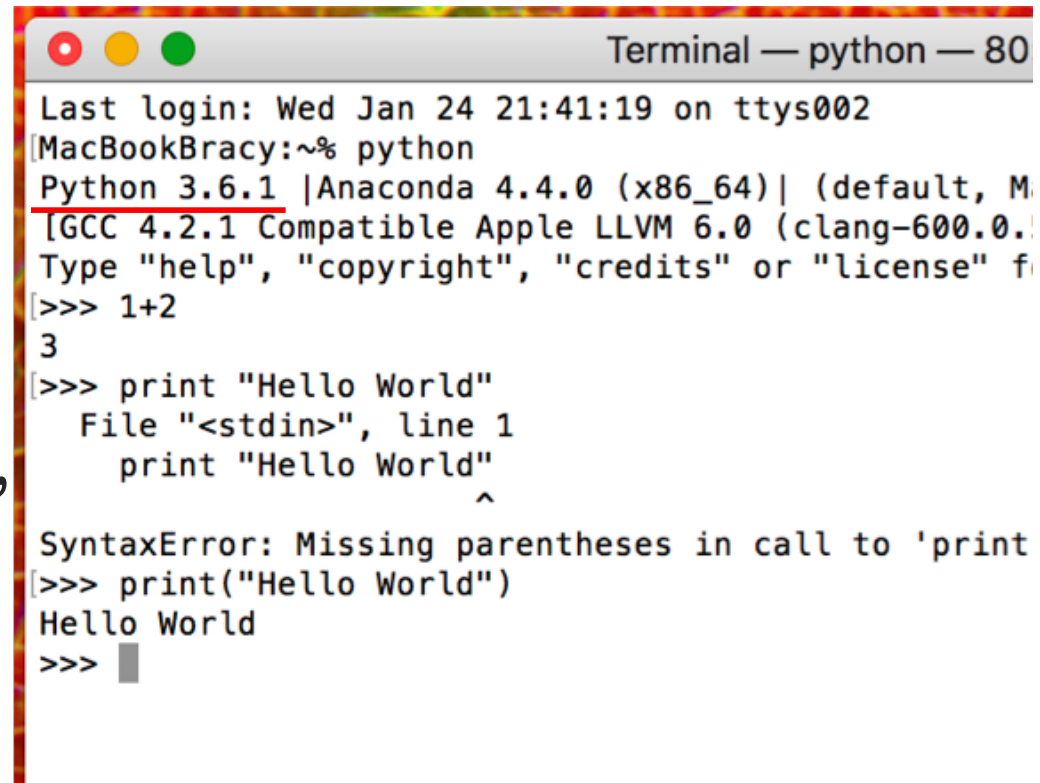
- Class announcements
- Consultant calendar
- Reading schedule
- Lecture slides
- Exam dates
- Piazza instructions

Check it regularly:

www.cs.cornell.edu/courses/cs1110/2018sp/

Getting Started with Python

- Designed to be used from the “command line”
 - OS X/Linux: **Terminal**
 - Windows: **Command Prompt**
 - Purpose of the first lab
- Install, then type “python”
 - Starts the *interactive mode*
 - Type commands at >>>
- First experiments:
 - evaluate *expressions*



```
Terminal — python — 80
Last login: Wed Jan 24 21:41:19 on ttys002
[MacBookBracy:~% python
Python 3.6.1 |Anaconda 4.4.0 (x86_64)| (default, M
[GCC 4.2.1 Compatible Apple LLVM 6.0 (clang-600.0.
Type "help", "copyright", "credits" or "license" f
[>>> 1+2
3
[>>> print "Hello World"
File "<stdin>", line 1
    print "Hello World"
          ^
SyntaxError: Missing parentheses in call to 'print'
[>>> print("Hello World")
Hello World
>>> █
```

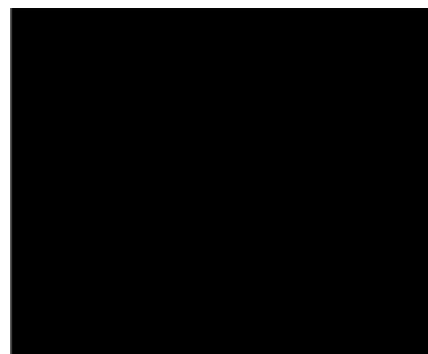
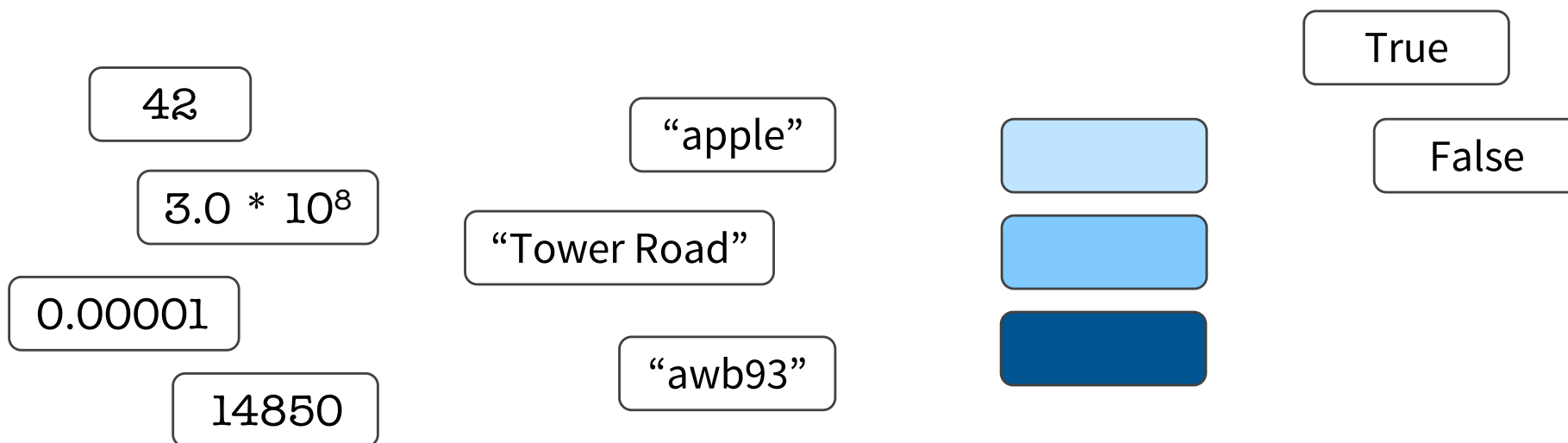
This class uses **Python 3**

- Welcome to the cutting edge!
- Eyes open, please!

```
>>> terminal time >>>
```

Storing and Computing Data

What data might we want to work with?
(What's on your computer?)



Expressions

An expression **represents** something

- Python ***evaluates it*** (turns it into a value)
- Similar to a calculator

Examples:

- 2.3

Literal
(evaluates to self)

- $(3 * 7 + 2) * 0.1$

An expression with four
literals and some operators

Types

A set of values & operations on these values

- Examples of operations: $+$, $-$, $/$, $*$
- Meaning of operations depends on type

Memorize this definition!

How to tell the Type of a Value

Command: `type(<value>)`

Example:

```
>>> type(2)
<type 'int'>
```

```
>>> terminal time >>>
```

Type: **float** (floating point)

Values: (approximations of) real numbers

- With a “.”: a **float literal** (e.g., 2.0)
- Without a decimal: an **int literal** (e.g., 2)

Operations: +, −, *, /, **, unary −

Notice: operator meaning can change from type to type

Exponent notation useful for large (or small) values

- $-22.51e6$ is $-22.51 * 10^6$ or -22510000
- $22.51e-6$ is $22.51 * 10^{-6}$ or 0.00002251

A second kind
of **float** literal

Floating Point Errors

Python stores floats as **binary fractions**

- Integer mantissa times a power of 2
- Example: 1.25 is $5 * 2^{-2}$

mantissa

exponent

Can't write most real numbers this way exactly

- Similar to problem of writing $1/3$ with decimals
- Python chooses the closest binary fraction it can

Approximation results in **representation error**

- When combined in expressions, the error can get worse
- **Example:** $0.1 + 0.2$

```
>>> terminal time >>>
```

Type: **int** (integers)

Values: ..., -3, -2, -1, 0, 1, 2, 3, 4, 5, ...

More Examples:: 1, 45, 43028030

(no commas or periods)

division (technically a float operator)

integer division

Operations: +, -, *, **, /, //, %, unary -

multiply

to power of

```
>>> terminal time >>>
```

Type: **bool** (boolean)

Values: True, False

- Boolean literals True and False (must be capitalized)

Operations: not, and, or

- not b: **True** if b is false and **False** if b is true
- b and c: **True** if both b and c are true; **False** otherwise
- b or c: **True** if b is true or c is true; **False** otherwise

Often come from comparing **int** or **float** values

- Order comparison: $i < j$ $i \leq j$ $i \geq j$ $i > j$
- Equality, inequality: $i == j$ $i != j$



"==" means something else!

Boolean Misconceptions

Booleans expressions *sound like* English, but subtle differences cause problems:

- In English, “A = B and C” often means “A = B and A = C”

Example: “Ithaca is cold and snowy”

- Means: “Ithaca is cold” and “Ithaca is snowy”
- **Does not mean:** “Ithaca is cold” and.... “snowy”

Python requires *fully specified* Boolean expressions

- In English, “A or B” often means “A or B **but not both**”

Example: “I’ll take CS 1110 or CS 1112” (but not both)

In Python, “A or B” always means “A or B **or both**”

Type: **str** (string) for **text**

Values: any sequence of characters

Operation(s): + (catenation, or concatenation)

Again: operator + changes from type to type

String literal: sequence of characters in quotes

- Double quotes: " **abcex3\$g<&**" or "Hello World!"
- Single quotes: 'Hello World!'

Concatenation applies only to strings

- "ab" + "cd" evaluates to "abcd"
- "ab" + 2 produces an **error**

```
>>> terminal time >>>
```