CS 1110 Fall 2019

- **Outcomes:**
  - **Fluency** in (Python) procedural programming
    - Usage of assignments, conditionals, and loops
    - Ability read and test programs from specifications
  - **Competency** in object-oriented programming
    - Ability to recognize and use objects and classes
  - **Knowledge** of searching and sorting algorithms
    - Knowledge of basics of vector computation
- **Website:**
  - www.cs.cornell.edu/courses/cs1110/2019fa/

Class Structure

- **Lectures.** Every Tuesday/Thursday
  - Not just slides; interactive demos almost every lecture
  - Because of enrollment, please stay with your section
  - **Semi-Mandatory**. 1% Participation grade from iClickers
- **Section/labs.** ACCEL Lab or Phillips 318
  - Guided exercises with TAs and consultants helping out
    - Tuesday: 12:20, 1:25, 2:30, 3:35
    - Wednesday: 10:10, 11:15, 12:20, 1:25, 2:30, 3:35, 7:20
  - Contact Lacy (lsl92@cornell.edu) for section conflicts
  - **Mandatory**. Missing more than 2 lowers your final grade

Class Materials

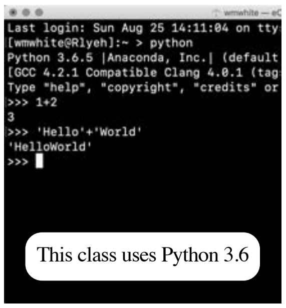
- **Textbook.** *Think Python, 2nd Ed.* by A. Downey
  - *Optional* text; only used as a reference
  - Available for free as PDF or eBook
  - Hardbound copies only available online
- **iClicker.** Acquire by **next Tuesday**
  - Credit for answering – even if wrong
  - iClicker App for smartphone **is not** acceptable
- **Python.** Necessary to use your own computer
  - See course website for how to install the software

Things to Do Before Next Class

- Visit the course website:
  - www.cs.cornell.edu/courses/cs1110/2019fa/
  - This IS the course syllabus, updated regularly
- Read **Get Started**
  - Obtain and *register* your iClicker
  - Enroll in Piazza
  - Sign into CMS and complete **Survey 0**
  - Install Python and complete **Lab 0**
  - Take the academic integrity quiz

Getting Started with Python

- Will use the "command line"
  - OS X/Linux: **Terminal**
  - Windows: **PowerShell**
  - Purpose of the first lab
- Once installed type "python"
  - Starts an *interactive shell*
  - Type commands at >>>
  - Responds to commands
- Use it like a calculator
  - Use to evaluate *expressions*



```
Last login: Sun Aug 25 14:11:04 on tty
[wmwhite@Rlyeh]:~ > python
Python 3.6.5 |Anaconda, Inc.| (default
[GCC 4.2.1 Compatible Clang 4.0.1 (tag
Type "help", "copyright", "credits" or
>>> 1+2
3
>>> 'Hello'+'World'
'HelloWorld'
>>> 
```

This class uses Python 3.6

Expressions and Values

- An **expression** represents something
  - Python *evaluates it,* turning it into a **value**
  - Similar to what a calculator does
- Examples:

`>>> 2.2`	Expression	(Literal)
`2.2`	Value	
`>>> (3 * 7 + 1) * 0.1`	Expression	(Complex)
`2.2`	Value	

What Are Types?

- Think about + in Python:

```
>>> 1+2
3
>>> "Hello"+"World"
"HelloWorld"
```

$\left.\begin{array}{l} \text{>>> 1+2} \\ \text{3} \end{array}\right\}$ adds numerically

$\left.\begin{array}{l} \text{>>> "Hello"+"World"} \\ \text{"HelloWorld"} \end{array}\right\}$ glues together

- A **type** is both
  - a set of *values*, and
  - the *operations* on them

Example: int

- **Values:** integers
  - …, –1, 0, 1, …
  - Literals are just digits: 1, 45, 43028030
  - No commas or periods
- **Operations:** math!
  - +, – (add, subtract)
  - *, // (mult, divide)
  - ** (power-of)
- **Important Rule:**
  - **int** ops make **ints**
  - (if making numbers)
- What about division?
  - 1 // 2 rounds to 0
  - / is **not** an **int** op
- Companion op: %
  - Gives the remainder
  - 7 % 3 evaluates to 1

Example: float

- **Values:** real numbers
  - 2.51, -0.56, 3.14159
  - Must have decimal
  - 2 is **int**, 2.0 is **float**
- **Operations:** math!
  - +, – (add, subtract)
  - *, / (mult, divide)
  - ** (power-of)
- Ops similar to **int**
- **Division** is different
  - Notice /, not //
  - 1.0/2.0 evals to 0.5
- But includes //, %
  - 5.4//2.2 evals to 2.0
  - 5.4 % 2.2 evals to 1.0
- Superset of **int**?

Using Big float Numbers

- **Exponent notation** is 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

- Python *prefers* this in some cases

```
>>> 0.00000000001
1e-11
```

Remember: values look like **literals**

Example: bool

- **Values:** True, False
  - That is it.
  - Must be capitalized!
- **Three Operations**
  - b and c (True if **both** True)
  - b or c (True if **at least one** is)
  - not b (True if b is **not**)
- Made by **comparisons**
  - **int**, **float** operations
  - But produce a **bool**
- Order comparisons:
  - i < j, i <= j
  - i >= j, i > j
- Equality, inequality:
  - i == j (**not** =)
  - i != j

Example: str

- **Values:** text, or *sequence of characters*
  - String literals must be in quotes
  - Double quotes: "Hello World!", " abcex3$g<&"
  - Single quotes: 'Hello World!', ' abcex3$g<&'
- **Operation:** + (catenation, or concatenation)
  - 'ab' + 'cd' evaluates to 'abcd'
  - concatenation can only apply to strings
  - 'ab' + 2 produces an **error**

Instructor: Walker White

Comparison of CS1110 and CS1112: Both introduce computing concepts. The courses emphasize techniques of problem analysis and the development of algorithms and programs.

CS 1110 (White): Computing using Python

CS 1112 (Fan): Computing using Matlab

CS 1110 and 1112 do not require previous programming experience. CS 1112 requires 1 semester of calculus. For more information, see http://www.cs.cornell.edu/undergrad/firstcscourse.

Course webpage (see top of page). Look at it several times a week: It is a major communication medium for the course. If you miss a handout, download it from the website.

Piazza www.piazza.com/cornell/fall2019/cs1110. Piazza is highly catered to getting you help fast and efficiently from classmates, the TAs, and instructors. We encourage you to post your questions on Piazza rather than use email to instructors.

CMS http://cmsx.cs.cornell.edu/. This is our course management system for handling assignments, grades, surveys, etc.

Course material (see course web page for more info)

(1) *Think Python: How to Think Like a Computer Scientist to Programming Using Java, 2nd Edition* by Allen Downey. Available as a free PDF from

http://greenteapress.com/wp/think-python-2e/

(2) Anaconda, a free Python environment.

(3) iClicker. We will be using iClickers, and every student is expected to bring their iClicker to every class. You may buy iClicker at the bookstore or online.

Homework will consist of 5 computer projects, which you can do with one partner, and some written assignments. Computer projects will be submitted electronically using our CMS (see above).

We do our best to make the programming assignments interesting! Graphics, manipulating files, games, string manipulation — we will show you what you can do with Python.

Tests: Two prelims and a final. To find out when they are, visit the course home page, scroll to the bottom of the page, and click the link. We give make-ups only in special circumstances.

Surveys: We are constantly improving this course based on student feedback. We will post surveys periodically on CMS and announce them to class. You are expected to complete them as part of your participation grade for the course.

Recitations-Sections-labs: Sections/labs are either in the ACCEL laboratory in Carpenter library, or Phillips 318, depending on your section. Each lab will ask you to do something on the computer, either to reinforce what is being taught in lecture or to introduce new topics to you. At the end, show what you did to the lab instructor to get credit (if you can't finish in time, show it to the instructor the next week).

Attendance will be taken. Miss three of them without valid excuses (given to us ahead of time) and your letter grade *will* decrease half a letter grade (e.g. B to B–).

Syllabus: A lecture-by-lecture topic list is on the course website. See "Lectures".

Academic integrity. This course is not a case of student against faculty. It is not about grades. It is about all of us working together to teach you as much about programming as possible in as efficient a manner as possible. The staff knows that you have other courses and strives to make your workload in this course reasonable. We are ready to help you in any way we can. On your side, we expect you to be honest. We expect you to come to us early if problems arise, so that we can solve them together. Do not wait four or five weeks, because then you may be too far behind.

Read the academic integrity statement on the course website and complete the Integrity Quiz on CMS.

Fix your PCs. To reduce chances of errors later, fix your PCs so that extensions (e.g. .py and .doc) always appear. To do this: Open an explorer window. Click menu item *Tools / Folder Options*. Click the view tab. Uncheck the box "Hide extensions for known file types". You may have to do something different depending on what Windows OS you use.

Practice, practice, practice. Learning to program well takes practice. The more time you spend on the computer, trying things out, getting acquainted with programming features and techniques, the better you will do in this course and later.

It is better to practice every day or every other day for ½ hour than it is to do nothing for a week or two and then spend 4 hours. Steady progress is best. In many ways, it is like learning a spoken language; if you stop, you will start to forget things.