Lecture 1

Course Overview, Python Basics
We Are (Sort-Of) Full!

• Thank Biology for the new class size
  ▪ Plenty of room in 11:15 lecture
  ▪ However, 9am is at fire code capacity
• But labs/sections are all full
  ▪ While we have seats, we do not have staff
  ▪ Talk to me if I promised you a seat
• Enrollment limited to **ungraduate students**

8/23/18  Overview, Types & Expressions  2
About Your Instructor: Walker White

- **Director**: GDIAC
  - **Game Design Initiative** at Cornell
  - Teach game design
- (and CS 1110 in fall)
CS 1110 Fall 2018

• Outcomes:
  - **Fluency** in (Python) procedural programming
    • Usage of assignments, conditionals, and loops
    • Ability to create Python modules and programs
  - **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:
## Intro Programming Classes Compared

<table>
<thead>
<tr>
<th>CS 1110: Python</th>
<th>CS 1112: Matlab</th>
</tr>
</thead>
<tbody>
<tr>
<td>• No prior programming experience necessary</td>
<td>• No prior programming experience necessary</td>
</tr>
<tr>
<td>• <strong>No calculus</strong></td>
<td>• <strong>One semester of calculus</strong></td>
</tr>
<tr>
<td>• <em>Slight</em> focus on</td>
<td>• <em>Slight</em> focus on</td>
</tr>
<tr>
<td>• Software engineering</td>
<td>• Scientific computation</td>
</tr>
<tr>
<td>• Application design</td>
<td>• Engineering applications</td>
</tr>
</tbody>
</table>

But either course serves as a pre-requisite to CS 2110
CS 1133: Short Course in Python

• 2-credit course in how to use Python
  ▪ Material is roughly the first half of CS 1110
  ▪ Most of the Python of 1110, but not theory
  ▪ Two assignments; no exams
  ▪ No experience required

• This is the only S/U course this year!
  ▪ CS 1110 is no longer offered S/U
  ▪ Best for students that just want Python
Why Programming in Python?

- Python is easier for beginners
  - A lot less to learn before you start “doing”
  - Designed with “rapid prototyping” in mind
- Python is more relevant to non-CS majors
  - NumPy and SciPy heavily used by scientists
- Python is a more modern language
  - Popular for web applications (e.g. Facebook apps)
  - Also applicable to mobile app development
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 Jenna (jls478@cornell.edu) for section conflicts
  - **Mandatory.** Missing more than 2 lowers your final grade
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

  All Labs will be use the online system. But they are not intended to be “online”.

  **Mandatory.** Missing more than 2 lowers your final grade
Class Materials

• **Textbook.** *Think Python, 2nd Ed.* by Allen Downey
  - *Optional* text; only used as a reference
  - Book available for free as PDF or eBook
  - Hardbound copies only available online

• **iClicker.** Acquire one by **next Thursday**
  - Will periodically ask questions during lecture
  - Will get credit for answering – even if wrong
  - iClicker App for smartphone **is not** acceptable

• **Python.** Necessary if you want to use own computer
  - See course website for how to install the software
This Course is OS Agnostic

Windows 10

macOS 10.12 or higher
Do NOT Even THINK It!

macOS Mojave

Coming September/October
Things to Do Before Next Class

1. Register your iClicker
   - Does not count for grade if not registered
2. Enroll in Piazza
3. Sign into CMS
   - Complete the Quiz
   - Complete Survey 0
4. Complete Lab 0
   - Install (Anaconda) Python
   - Answer online questions

- Everything is on website!
  - Piazza instructions
  - Class announcements
  - Consultant calendar
  - Reading schedule
  - Lecture slides
  - Exam dates
- Check it regularly:
A Word About About Grades

• As Cornell students, we know that you care
• But this is not a weed-out course
  ▪ Students can do well regardless of experience
• But you may have to work hard!
  ▪ If no experience, budget 10+ hours of homework a week

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D/F</th>
</tr>
</thead>
<tbody>
<tr>
<td>All Students</td>
<td>33%</td>
<td>45%</td>
<td>20%</td>
<td>2%</td>
</tr>
<tr>
<td>AP Students</td>
<td>50%</td>
<td>40%</td>
<td>10%</td>
<td>0%</td>
</tr>
<tr>
<td>Some Experience</td>
<td>45%</td>
<td>35%</td>
<td>20%</td>
<td>0%</td>
</tr>
<tr>
<td>No Experience</td>
<td>25%</td>
<td>50%</td>
<td>22%</td>
<td>3%</td>
</tr>
</tbody>
</table>
A Word About About Grades

- As Cornell students, we know that you care
- But this is not a weed-out course
  - Students can do well regardless of experience
    - But you may have to work hard!
    - If no experience, budget 10+ hours of homework a week

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D/F</th>
</tr>
</thead>
<tbody>
<tr>
<td>All Students</td>
<td>33%</td>
<td>45%</td>
<td>20%</td>
<td>2%</td>
</tr>
<tr>
<td>AP Students</td>
<td>50%</td>
<td>40%</td>
<td>10%</td>
<td></td>
</tr>
<tr>
<td>Some Experience</td>
<td>45%</td>
<td>35%</td>
<td>20%</td>
<td></td>
</tr>
<tr>
<td>No Experience</td>
<td>25%</td>
<td>50%</td>
<td>22%</td>
<td>3%</td>
</tr>
</tbody>
</table>
This Course is Designed For

- Engineers who need a computing requirement
- Students that want to major/minor in CS

You Should Think Twice If

- You are uncomfortable with college-level math
- You are a junior/senior that has avoided STEM
- You only want to learn Python
Getting Started with Python

- Designed to be used from 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 `>>>`
  - Shell responds to commands
- Can use it like a calculator
  - Use to evaluate **expressions**

This class uses Python 3.6
The Basics

Overview, Types & Expressions

Values
- 42
- 12.345
- “Hello!”

Types
- integer
- float (real number)
- string (of characters)

Expressions
- 34 * (23 + 14)
- 1.0 / 3.0
- "Hel" + "lo!"
Python and Expressions

• An expression **represents** something
  ▪ Python *evaluates it* (turns it into a value)
  ▪ Similar to what a calculator does

• Examples:
  ▪ **2.3**
    - Literal (evaluates to self)
  ▪ **(3 * 7 + 2) * 0.1**
    - An expression with four literals and some operators
Representing Values

- **Everything** on a computer reduces to numbers
  - Letters represented by numbers (ASCII codes)
  - Pixel colors are three numbers (red, blue, green)
  - So how can Python tell all these numbers apart?

- **Type:**
  - Memorize this definition!
  - A set of values and the operations on them.
    - Examples of operations: +, -, /, *
    - The meaning of these depends on the type
Example: Type int

- **Type int** represents integers
  - **values:** \(\ldots, -3, -2, -1, 0, 1, 2, 3, 4, 5, \ldots\)
    - Integer literals look like this: \(1, 45, 43028030\) (no commas or periods)
  - **operations:** \(+, -, *, //, **, \text{unary} -\)

- **Principle:** operations on **int** values must yield an **int**
  - **Example:** \(1 // 2\) rounds result down to 0
    - **Companion operation:** \(%\) (remainder)
    - \(7 \% 3\) evaluates to 1, remainder when dividing 7 by 3
  - **Operator / is not an int operation in Python 3**
Example: Type \textbf{float}

- **Type float** (floating point) represents \textbf{real numbers}
  - \textbf{values}: distinguished from integers by decimal points
    - In Python a number with a “.” is a \textit{float literal} (e.g. 2.0)
    - Without a decimal a number is an \textit{int literal} (e.g. 2)
  - \textbf{operations}: +, −, *, /, **, unary −
    - Notice that float has a different division operator
    - \textbf{Example}: 1.0/2.0 evaluates to 0.5
- **Exponent notation** is useful for large (or small) values
  - \(-22.51e6\) is \(-22.51 \times 10^6\) or \(-22510000\)
  - \(22.51e-6\) is \(22.51 \times 10^{-6}\) or \(0.00002251\)

\(8/23/18\)

Overview, Types & Expressions
Floats Have Finite Precision

- Python stores floats as **binary fractions**
  - Integer mantissa times a power of 2
  - Example: 1.25 is $5 \times 2^{-2}$

- Impossible to write most real numbers this way exactly
  - Similar to problem of writing 1/3 with decimals
  - Python chooses the closest binary fraction it can

- This approximation results in **representation error**
  - When combined in expressions, the error can get worse
  - **Example**: type $0.1 + 0.2$ at the prompt >>>

8/23/18  Overview, Types & Expressions  24
Example: Type `bool`

- **Type** `boolean` or `bool` represents logical statements
  - **values**: `True`, `False`
    - Boolean literals are just `True` and `False` (have to 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 <= j`  `i >= j`  `i > j`
  - Equality, inequality: `i == j`  `i != j`

"=" means something else!
Example: Type `str`

- **Type** `String` or `str` represents **text**
  - values: any sequence of characters
  - operation(s): + (catenation, or concatenation)

- **String literal**: sequence of characters in quotes
  - Double quotes: " abcex3$g<&" or "Hello World!"
  - Single quotes: 'Hello World!'

- Concatenation can only apply to strings.
  - 'ab' + 'cd' evaluates to 'abcd'
  - 'ab' + 2 produces an **error**
Example: Type str

- Type String or str represents text
  - values: any sequence of characters
  - operation(s): + (catenation, or concatenation)

- String literal: sequence of characters in quotes
  - Double quotes: " abcex3$g<&" or "Hello World!"
  - Single quotes: 'Hello World!'

- Concatenation can only apply to strings.
  - 'ab' + 'cd' evaluates to 'abcd'
  - 'ab' + 2 produces an error

The meaning of + depends on the type
Converting Values Between Types

• Basic form: \textit{type(value)}
  - \texttt{float}(2) converts value 2 to type \texttt{float} (value now 2.0)
  - \texttt{int}(2.6) converts value 2.6 to type \texttt{int} (value now 2)
  - Explicit conversion is also called “casting”

• Narrow to wide: \texttt{bool} ⇒ \texttt{int} ⇒ \texttt{float}
  - \textit{Widening}. Python does automatically if needed
    - \textbf{Example}: \(\frac{1}{2.0}\) evaluates to 0.5 (casts 1 to \texttt{float})
  - \textit{Narrowing}. Python \textit{never} does this automatically
    - Narrowing conversions cause information to be lost
    - \textbf{Example}: \texttt{float(int(2.6))} evaluates to 2.0