Lecture 1: Introduction, Types & Expressions (Chapter 1)

CS 1110
Introduction to Computing Using Python

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

Why learn to program? (subtly distinct from, although a core part of, CS / IS)

“Teach computing, not Word”, the Economist

About Professor Lee
Research lifetime achievement awards:
- Association for Computing Machinery (ACM), 2018
- Assoc. for the Advancement of Artificial Intelligence (AAAI), 2013
- Assoc. for Computational Linguistics, 2017

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

Carpenter Memorial Advising Award: 2009

A.B. Cornell ’93, Ph.D. Harvard ’97
Lowest grade ever…?
Interest in optimization—what is the "best" way to operate a system given constraints and uncertainties?

Other courses:
- Intro to computing using Matlab
- Optimization with metaheuristics

Author: Insight Through Computing: A Matlab Introduction to Computational Science and Engineering with C. F. Van Loan

Honors:
- Carpenter Memorial Advising Award (2016)
- Engineering teaching awards (2011, 2019)

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)

<table>
<thead>
<tr>
<th>CS 1110: Python</th>
<th>CS 1112: MATLAB</th>
</tr>
</thead>
<tbody>
<tr>
<td>No programming experience necessary</td>
<td>No programming experience necessary</td>
</tr>
<tr>
<td>No calculus</td>
<td>1 semester of calculus</td>
</tr>
<tr>
<td>Non-numerical problems</td>
<td>Engineering-type problems</td>
</tr>
<tr>
<td>More about software design</td>
<td>More about computational science &amp; engineering</td>
</tr>
</tbody>
</table>

Both serve as a pre-requisite to CS 2110

Intro Programming Classes Compared (2)

<table>
<thead>
<tr>
<th>CS 1133: Python Short Course</th>
<th>CS 1380: Data Science For All</th>
</tr>
</thead>
<tbody>
<tr>
<td>No programming experience necessary</td>
<td>No programming experience necessary</td>
</tr>
<tr>
<td>No calculus</td>
<td>No calculus</td>
</tr>
<tr>
<td>Very basics of programming</td>
<td>Less programming than 1110, but also:</td>
</tr>
<tr>
<td>2 credits (7 weeks)</td>
<td>data visualization, prediction, machine learning</td>
</tr>
</tbody>
</table>

Course Website

http://www.cs.cornell.edu/courses/cs1110/2021sp/
Lectures
- Tuesday & Thursday 9:05am
- Not just talking! Demos, clicker questions, etc.
- **Watch pre-lecture videos ("lessons") or read from supplemental textbook before class!** Posted on course website the day before class. Lecture assumes that you have done the pre-lecture viewing/reading
- Lecture slides, code examples, and lecture recording available on website later, within 24 hours
- Watch the lessons and attend (or watch recording of) lecture regularly—don’t get behind

Lab (aka Sections)
- Guided exercises with TAs & consultants
- Start today: Tuesday, Feb 9
- **Attend the lab section in which you are enrolled.** We can’t maintain workable staff/student ratios otherwise.
- Need a different Section? Change (swap) section on Student Center
- Each lab has 2 parts, released on Tuesday: Part A due on Fri; Part B due the next Tues
- **Mandatory.** Missing > 4 parts (equivalent to 2 full labs) can lower your final grade.

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

### 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

### Storing and computing data

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

<table>
<thead>
<tr>
<th>True</th>
<th>42</th>
</tr>
</thead>
<tbody>
<tr>
<td>False</td>
<td>3.0 * 10^8</td>
</tr>
<tr>
<td></td>
<td>&quot;apple&quot;</td>
</tr>
<tr>
<td></td>
<td>Tower Road</td>
</tr>
<tr>
<td></td>
<td>0.00001</td>
</tr>
<tr>
<td></td>
<td>True</td>
</tr>
<tr>
<td></td>
<td>False</td>
</tr>
<tr>
<td></td>
<td>14850</td>
</tr>
<tr>
<td></td>
<td>&quot;awb93&quot;</td>
</tr>
</tbody>
</table>

This class uses Python 3

### Types

A type is a set of values and the 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:

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

Floating Point Errors

Python cannot store most real numbers exactly

- Similar to problem of writing 1/3 with decimals

Approximation results in representation error

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

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

Type: **float** (floating point number)

**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 −

**Note:** operator meaning can change from type to type

**Exponent notation** useful for large (or small) values

- $-22.51e6$ is $-22.51 \times 10^6$ or $-2251000$
- $22.51e-6$ is $22.51 \times 10^{-6}$ or $0.00002251$

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

Type: **int** (integers)

**Values:** ..., −3, −2, −1, 0, 1, 2, 3, 4, 5, ...

More Examples: 1, 45, 43028030

(no commas or periods)

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

```
>>> 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: k < j k <= j k == j k > j
- Equality, inequality: k == j k != j

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

Class Materials

**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!)

**Python.** Necessary if using your own computer

- See course website for how to install
Things to do before next class

1. Read textbook
   - Ch 1, Sections 2.1-2.3, 2.5, 2.6

2. Watch lesson videos

3. (If using your own computer) Install Python following instructions on the website

4. Attend lab on Tues/Wedn!

Lots of information on the website!
- Class announcements
- Consultant calendar
- Reading/Lessons schedule
- Lecture slides
- Exam dates
- Installation instructions

Read it thoroughly:
www.cs.cornell.edu/courses/cs1110/2021sp/

Communication

**cs1110-prof@cornell.edu**
- Includes: both professors & head TA
- **For sensitive correspondence.** Don’t email one prof, or both separately.

**cs1110-staff@cornell.edu**
- Includes: both profs, admin assistant, graduate TAs, head consultants
- **For time sensitive correspondence** (i.e., emergencies). E.g., Nobody at office hours.

**Ed Discussion:** online forum (start from link on course website)

Email from us: please check your spam filters for mail from kdf4, Lil2, cs1110-prof, or with [CS1110] in the subject line