CS 1110 Fall 2017

Reading for today and next week: Ch. 1-2

• Outcomes:
  • Fluency in (Python) procedural programming
    • Usage of assignments, conditionals, and loops
    • Ability to design Python modules and programs
  • Competency in object-oriented programming
    • Ability to write programs using objects and classes.
  • Knowledge of searching and sorting algorithms
    • Knowledge of basics of vector computation
• Website: www.cs.cornell.edu/courses/cs1110/2017sp/

Communication

• cs-1110profs-L@cornell.edu
  • Includes: two profs, admin assistant
  • Main correspondence. Don’t email only one prof, or both separately
• cs-1110mgmt-L@cornell.edu
  • Includes: both profs, admin assistant, graduate TAs, head consultants
  • “Emergency contact number.” nobody at office hours; lab has no printouts
  • Email from us: please check your spam filters for mail from ELA63@cornell.edu, LJL2@cornell.edu, or with [CS1110] in the subject line.

Class Structure

• Lectures. Every Tuesday/Thursday
  • Not just slides; interactive demos almost every lecture
• Discussion Sections = “Labs”.
  • Guided exercises with TAs and consultants helping out
  • Handouts posted to the website the Monday before
  • Don’t panic if you are not registered yet.
  • Go to the lab section you are registered for.
  • If not enrolled in a lab section: do the lab on your own. If a lab section opens up, check it in then.
  • Mandatory. Missing more than 2 can lower your final grade.

Class Materials

• Textbook. Think Python by Allen Downey
  • Supplemental text; does not replace lecture
  • Book available for free as PDF or eBook
  • (no hard copy anymore; out of print)
• iClicker. Optional but useful.
  • Will periodically ask questions during lecture
  • Not part of the grade at all
• Python. Necessary if you want to use own computer
  • See course website for how to install the software

Things to Do Before Next Class

1. Read the textbook
   • Chapter 1 (browse)
   • Chapter 2 (in detail)
2. Install Python following our instructions: https://www.cs.cornell.edu/courses/cs1110/2017sp/python.php
3. Look at first lab handout
4. (optional) Piazza: a question-answering forum

Getting Started with Python

• Designed to be used from the “command line”
  • OS X/Linux: Terminal
  • Windows: Command Prompt
• 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 2.7.x
Python and Expressions

• An expression represents something
  • Python evaluates it (turns it into a value)
  • Similar to what a calculator does
• Examples:
  * 2.3
  * (3 * 7 + 2) * 0.1

Type: Set of values and the operations on them

• Type int represents integers
  * values: …, –3, –2, –1, 0, 1, 2, 3, 4, 5, …
  • Integer literals look like this: 1, 45, 43028030 (no commas or periods)
  * operations: +, –, *, /, **, 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 (use //= instead)

• Type float (floating point) represents real numbers
  * values: distinguished from integers by decimal points
    • In Python a number with a “.” is a float literal (e.g. 2.0)
    • Without a decimal a number is an int literal (e.g. 2)
  * operations: +, –, *, /, **
    • The meaning for floats differs from that for ints
    • Example: 1.0/2.0 evaluates to 0.5
  • 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

• 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
• Converting Values Between Types
  • Basic form: type(value)
    * float(2) converts value 2 to type float (value now 2.0)
    * int(2.6) converts value 2.6 to type int (value now 2)
    * Explicit conversion is also called “casting”
  • Narrow to wide: bool ⇒ int ⇒ float
    • Widening. Python does automatically if needed
      • Example: 1/2.0 evaluates to 0.5 (casts 1 to float)
    • Narrowing. Python never does this automatically
      • Narrowing conversions cause information to be lost
      • Example: float(int(2.6)) evaluates to 2.0

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<8" or "Hello World!"
  * Single quotes: 'Hello World!'
• Concatenation can only apply to strings.
  * 'ab' + 'cd' evaluates to 'abcd'
  * 'ab' + 2 produces an error