http://www.cs.cornell.edu/courses/cs1110/2019sp

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 2019: Announcements

http://www.cs.cornell.edu/courses/cs1110/2019sp

Sections

- Please go only to the Section you are enrolled in

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/cs1110/2019sp/alternatives.html

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/cs1110/2019sp/aew.html

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.

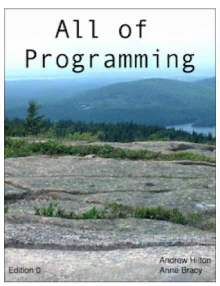
Oh the places you'll go! (with 1110)

Why are you taking this class?

- everyone's doing it!
- parents told you to
- fulfilling a requirement
- friend told you it's cool
- curious about programming
- want to learn python
- packing your resume
- CS or IS is your intended major

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, 2110, 3410, 4410/4411
  - ACSU Faculty of the Year, 2016
  - Engineering Teaching Award, 2017

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

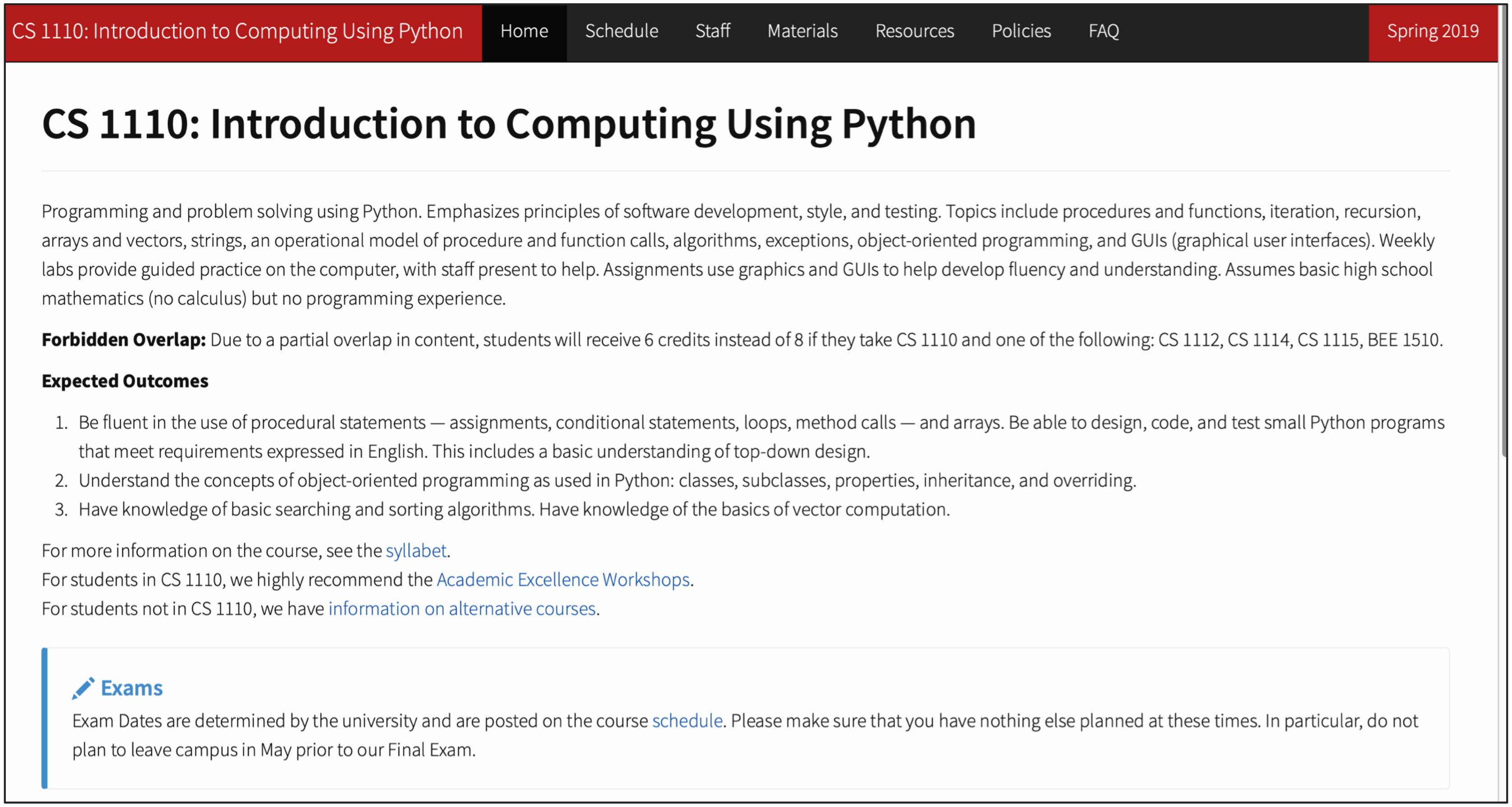
- NumPy and SciPy heavily used by scientists

A modern language

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

Course Website

http://www.cs.cornell.edu/courses/cs1110/2019sp/



CS 1110: Introduction to Computing Using Python | Home | Schedule | Staff | Materials | Resources | Policies | FAQ | Spring 2019

CS 1110: Introduction to Computing Using Python

Programming and problem solving using Python. Emphasizes principles of software development, style, and testing. Topics include procedures and functions, iteration, recursion, arrays and vectors, strings, an operational model of procedure and function calls, algorithms, exceptions, object-oriented programming, and GUIs (graphical user interfaces). Weekly labs provide guided practice on the computer, with staff present to help. Assignments use graphics and GUIs to help develop fluency and understanding. Assumes basic high school mathematics (no calculus) but no programming experience.

Forbidden Overlap: Due to a partial overlap in content, students will receive 6 credits instead of 8 if they take CS 1110 and one of the following: CS 1112, CS 1114, CS 1115, BEE 1510.

Expected Outcomes

1. Be fluent in the use of procedural statements — assignments, conditional statements, loops, method calls — and arrays. Be able to design, code, and test small Python programs that meet requirements expressed in English. This includes a basic understanding of top-down design.
2. Understand the concepts of object-oriented programming as used in Python: classes, subclasses, properties, inheritance, and overriding.
3. Have knowledge of basic searching and sorting algorithms. Have knowledge of the basics of vector computation.

For more information on the course, see the syllabet.
For students in CS 1110, we highly recommend the Academic Excellence Workshops.
For students not in CS 1110, we have information on alternative courses.

Exams

Exam Dates are determined by the university and are posted on the course schedule. Please make sure that you have nothing else planned at these times. In particular, do not plan to leave campus in May prior to our Final Exam.

If the website doesn't look like this,
you're looking at the wrong semester.

Communication

cs1110-prof@cornell.edu

- Includes: professor & head TA
- **For sensitive correspondence**

cs1110-staff@cornell.edu

- Includes: professor, admin assistant, graduate TAs, head consultants
- **For time sensitive correspondence (i.e., emergencies)** Nobody at office hours; Lab has no printouts, *etc.*

Piazza: not required, but fast

Canvas: official announcements posted here and emailed. (check your spam filters for mail from awb93 or with [CS1110] in subject line)

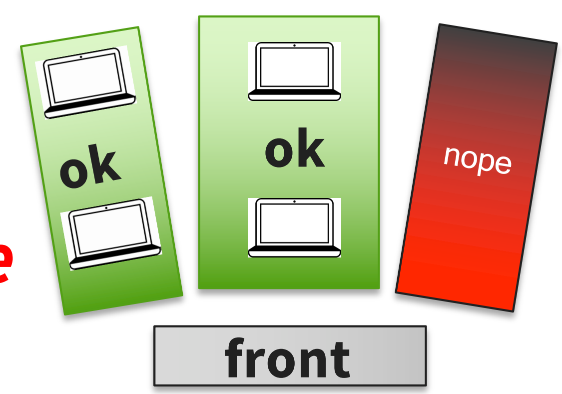
Lectures

Lectures:

- Tuesday/Thursday 9:05
  - Front doors close and lock at 9:05. After that, use the back doors. (See Policies page on our website.)
- Not just talking! Demos, clicker questions, *etc.*
- Slides posted to website afternoon before class

Please, no cell phones during lecture

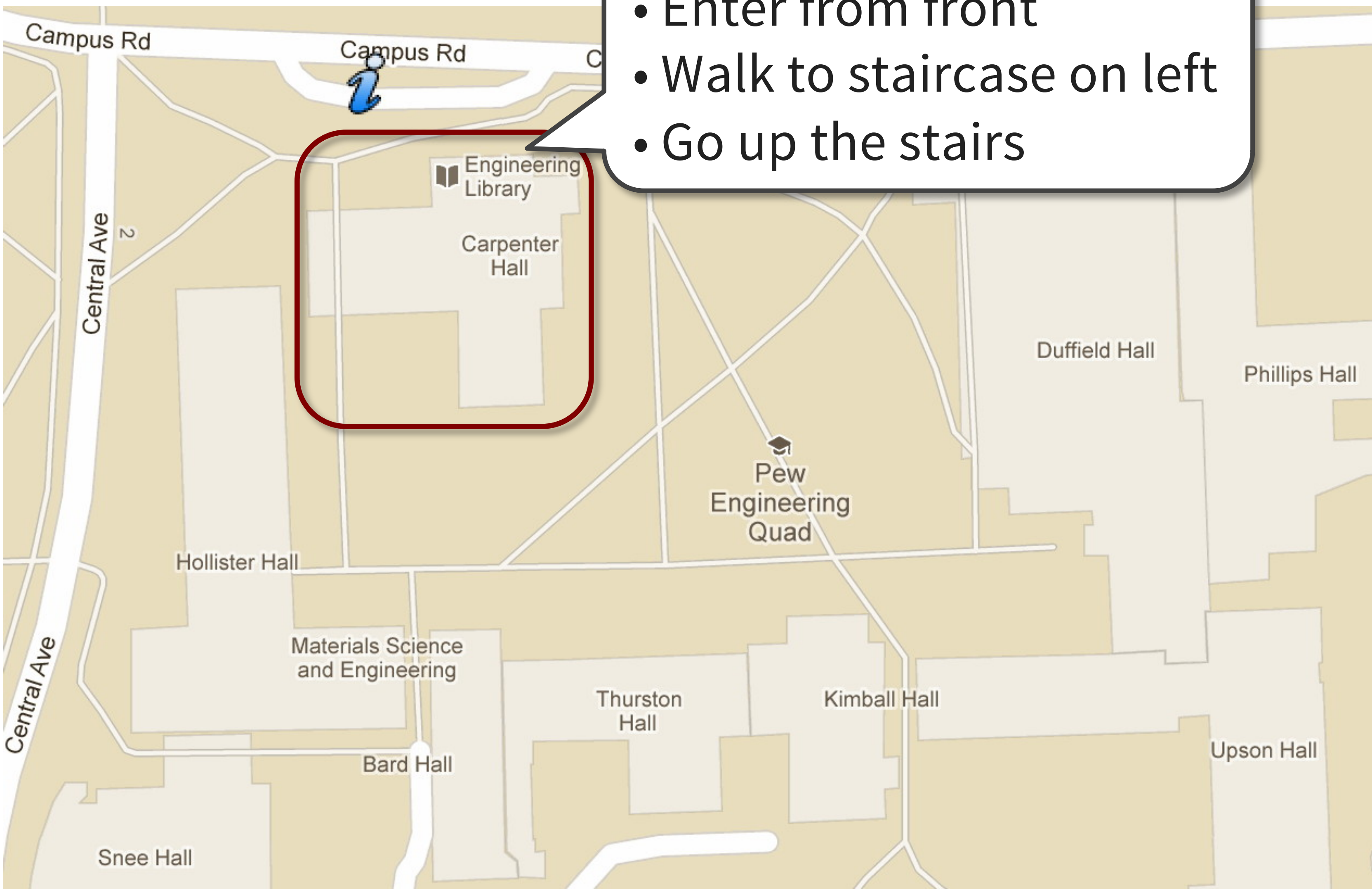
No laptop zone on the right, please do not use your laptop there



Lab Sections (aka Sections)

- guided exercises with TAs & consultants
- Start today: Tuesday, January 22
- **Go to the lab section you are registered for.** We can’t maintain workable staff/student ratios otherwise.
- Handouts posted to the website the Monday before
- **Mandatory**. Missing > 2 can lower your final grade.

ACCEL Labs



- Enter from front
- Walk to staircase on left
- Go up the stairs

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. Required. Begins Thursday.

- Will periodically ask questions during lecture
- Register on Canvas to get Participation points.
- We do not 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
   - Ch 1, Sections 2.1-2.3, 2.5
2. (If using your own computer) Install Python **following instructions on the website under Materials**
3. Go to Lab!
4. (optional) Join Piazza, a Q&A forum
5. Go to Canvas, register your Clicker.

Lots of information on the website!

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

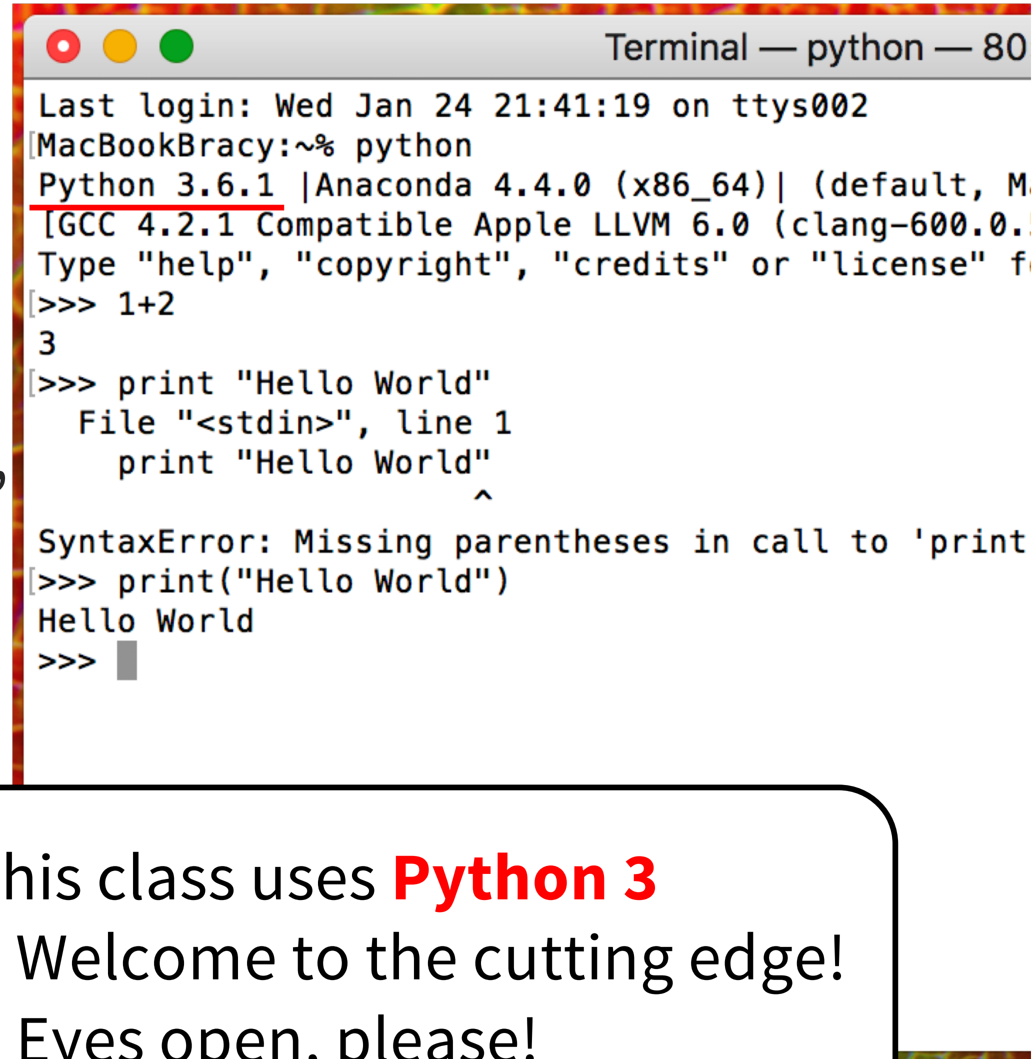
Read it thoroughly:

www.cs.cornell.edu/courses/cs1110/2019sp/

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*



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?)

42

$3.0 * 10^8$

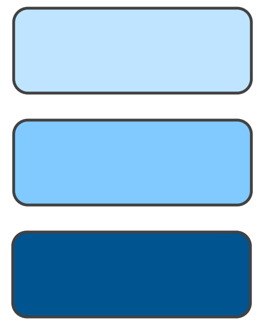
0.00001

14850

"apple"

"Tower Road"

"awb93"



True

False

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 $\underbrace{5}_{\text{mantissa}} * 2^{\underbrace{-2}_{\text{exponent}}}$

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 <= j` `i >= 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 >>>`