Assignments

- Major portion (40%) of your final grade
  - Larger projects due every two weeks
- First assignment requires **mastery**
  - Submit, get feedback, resubmit, … until correct
  - Everyone eventually scores 10/10
- Later assignments are designed to be fun
  - Examples: graphics, image manipulation
  - Final project is a Breakout game project
- Submitted via **Course Management System** (CMS)
  - Visit cms.csuglab.cornell.edu/ to check you are enrolled

Participation: 2% of Final Grade

- **iClickers.** In lecture questions
  - Essentially a form of "stealth attendance"
  - Must answer 75% of questions for credit
  - But actual answers are not graded
- **Surveys.** What do you think of the class?
  - This is the first year teaching Python
  - Want data on who you are/why taking course?
  - What do you like/dislike about assignments?
  - Must answer 75% of surveys for full credit

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
   - Quiz: About the Course
   - Complete Survey 0
4. Read the textbook
   - Chapter 1 (browse)
   - Chapter 2 (in detail)

- Everything is on website!
  - Piazza instructions
  - Class announcements
  - Consultant calendar
  - Reading schedule
  - Lecture slides
  - Exam dates
- Check it regularly:
  - www.cs.cornell.edu/courses/cs1110/2012fa/

Helping You Succeed: Other Resources

- **Consultants.** ACCEL Lab Green Room
  - Daily office hours (see website) with consultants
  - Very useful when working on assignments
- **AEW Workshops**. Additional discussion course
  - Runs parallel to this class – completely optional
  - See website; talk to advisors in Olin 167.
- **Piazza.** Online forum to ask and answer questions
  - Go here first **before** sending question in e-mail
- **Office Hours.** Talk to the professors!
  - Available in Thurston 202 between lectures

Operator Precedence

- What is the difference between the following?
  - 2*(1+3) **add, then multiply**
  - 2*1 + 3 **multiply, then add**
- Operations are performed in a set order
  - Parentheses make the order explicit
  - What happens when there are no parentheses?
- **Operator Precedence**: The *fixed* order Python processes operators in *absence* of parentheses

Precedence of Python Operators

- **Exponentiation**: `**`
- **Unary operators**: `+ -`
- **Binary arithmetic**: `* / %`
- **Binary arithmetic**: `+ -`
- **Comparisons**: `< > <= >=`
- **Equality relations**: `== !=`
- **Logical not**
- **Logical and**
- **Logical or**

- Precedence goes downwards
  - Parentheses highest
  - Logical ops lowest
- Same line = same precedence
  - Read "ties" left to right (for all but `**`)
  - Example: `1/2*3` is `(1/2)*3`

- Section 2.7 in your text
- See website for more info
- Major portion of Lab 1

Variables (Section 2.1)

- A **variable**
  - is a **named** memory location (**box**)
  - contains a **value** (in the box)
  - can be used in expressions

The value in the box is then used in evaluating the expression.

- Examples

Variable names must start with a letter (or _).

The type belongs to the *value*, not to the *variable*.

x | 5 | Variable **x**, with value 5 (of type **int**)

area | 20.1 | Variable **area**, w/ value 20.1 (of type **float**)

Variables and Assignment Statements

- Variables are created by **assignment statements**
  - Create a new variable name and give it a value

`x = 5` — the value; the variable

x | 5

- This is a **statement**, not an **expression**
  - Tells the computer to DO something (not give a value)
  - Typing it into >>> gets no response (but it is working)
- Assignment statements can have expressions in them
  - These expressions can even have variables in them

`x = x + 2` — the expression; the variable

Two steps to execute an assignment:
1. evaluate the expression on the right
2. store the result in the variable on the left

Dynamic Typing

- Python is a **dynamically typed language**
  - Variables can hold values of any type
  - Variables can hold different types at different times
  - Use `type(x)` to find out the type of the value in x
  - Use names of types for conversion, comparison

```
type(x) == int
x = float(x)
type(x) == float
```

- The following is acceptable in Python:

```
>>> x = 1        ← x contains an int value
>>> x = x / 2.0  ← x now contains a float value
```

- Alternative is a **statically typed language** (e.g. Java)
  - Each variable restricted to values of just one type

String: Text as a Value

- String are quoted characters
  - `'abc d'` (Python prefers)
  - `"abc d"` (most languages)

Type: `str`

- How to write quotes in quotes?
  - Delineate with "other quote"
  - **Example**: `" ' "` or `' " '`
  - What if need both `"` and `'`?
- **Solution**: escape characters
  - Format: \ + letter
  - Special or invisible chars

Char	Meaning
\'	single quote
\"	double quote
\n	new line
\t	tab
\\	backslash

String are Indexed

- `s = 'abc d'`

0	1	2	3	4
a	b	c		d

- `s = 'Hello all'`

0	1	2	3	4	5	6	7	8
H	e	l	l	o		a	l	l

- Access characters with []
  - s[0] is 'a'
  - s[4] is 'd'
  - s[5] causes an error
  - s[0:2] is 'ab' (excludes c)
  - s[2:] is 'c d'
- Called "string slicing"
- What is s[3:6]?

A: 'lo a'
B: 'lo'
C: 'lo '
D: 'o '
E: I do not know

Strings have many other powers

```
s = 'abracadabra'
'a' in s == True
'cad' in s == True
'foo' in s == True
s.index('a') == 0
s.index('rac') == 2
s.count('a') == 5
len(s) == 11
s.strip('a') == 'bracadabr'
'  cs1110  '.strip() == 'cs1110'
```

s_1 in s_2 asks whether s_1 is a substring of s_2. Result is type **bool**.

s_1.index(s_2) returns the index of the first occurrence of s_2 in s_1.

s_1.count(s_2) returns the number of occurrences of s_2 in s_1.

len(s) returns the number of characters in s.

s_1.strip(s_2) returns a copy of s_1 with characters in s_2 removed from the ends.

Just s_1.strip() defaults to removing white space from the ends.

More (too much!) information in Python documentation on **www.python.org** (see Library Reference, built-in types)