Outcomes:
- Competency with basic Python programming
  - Ability to create Python modules and programs
  - Ability to use the most common built-in data types
- Knowledge of object-oriented programming
  - Ability to recognize and use objects in Python.
  - Ability to understand classes written by others.

Website:
- www.cs.cornell.edu/courses/cs1133/2016fa/

Grading Policy:
- There will be two assignments
- Both will involve programming
- Must earn 85% to pass an assignment
- Get two more attempts if you fail
- But you must meet the posted deadlines!
- Must pass both assignments
- No exams; labs are not graded

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

The Basics
- Values
  - Type `int` (integer):
    - values: ..., –3, –2, –1, 0, 1, 2, 3, 4, 5, ...
    - "Whole" numbers w/o decimals
    - operations: +, −, *, /, **, unary –
  - `Principal`: 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):
  - values: (approximations of) real numbers
    - In Python a number with a "." is a float literal (e.g. 3.0)
    - Without a decimal a number is an int literal (e.g. 3)
  - operations: +, −, *, /, **, unary –
    - But meaning is different for floats
    - Example: 1.0 / 2.0 evaluates to 0.5
  - Exponent notation is useful for large (or small) values
    - \(-22.51 \times 10^{-6}\) is \(-22.51 \times 10^{-6}\) or \(-22510000\)
    - \(22.51 \times 10^{-6}\) is \(22.51 \times 10^{-6}\) or \(0.00002251\)
Type: Set of values and the operations on them

- **Type boolean** or `bool`:
  - **values**: `True`, `False`
  - **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`

Expressions vs Statements

<table>
<thead>
<tr>
<th>Expression</th>
<th>Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Represents</strong> something</td>
<td><strong>Does</strong> something</td>
</tr>
<tr>
<td>- Python <em>evaluates it</em></td>
<td>- Python <em>executes it</em></td>
</tr>
<tr>
<td>- End result is a value</td>
<td>- Need not result in a value</td>
</tr>
<tr>
<td><strong>Examples</strong>:</td>
<td><strong>Examples</strong>:</td>
</tr>
<tr>
<td>- <code>2.3</code></td>
<td>- print &quot;Hello&quot;</td>
</tr>
<tr>
<td>- <code>(3+6)/4</code></td>
<td>- import sys</td>
</tr>
</tbody>
</table>

Variables (Section 2.1)

- A **variable** is
  - a named memory location (box),
  - a value (in the box)
- **Examples**
  - `x = 5` Variable `x`, with value 5 (of type `int`)
  - `area = 20.1` Variable `area`, with value 20.1 (of type `float`)
  - Variable names must start with a letter
    - So `1e2` is a `float`, but `e2` is a variable name

Variables and Assignment Statements

- Variables are created by **assignment statements**
  - Create a new variable name and give it a value
    - `x = 5` the variable
  - 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

Dynamic Typing

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