Helping You Succeed in this Class

- **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 professor!
  - Available outside Call Auditorium between lectures

Labs vs. Assignments

<table>
<thead>
<tr>
<th>Labs</th>
<th>Assignments</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Held every week</td>
<td>• Every two weeks</td>
</tr>
<tr>
<td>• Graded on completeness</td>
<td>• First one due Sep. 18</td>
</tr>
<tr>
<td>• Always S/U</td>
<td>• Graded on correctness</td>
</tr>
<tr>
<td>• Try again if not finished</td>
<td>• Assign points out of 100</td>
</tr>
<tr>
<td>• Indirect affect on grade</td>
<td>• But first one is for mastery</td>
</tr>
<tr>
<td>• Can miss up to 2 labs</td>
<td>• Resubmit until perfect grade</td>
</tr>
<tr>
<td>• After that, grade reduced</td>
<td>• 40% of your final grade</td>
</tr>
<tr>
<td>• Similar to language drills</td>
<td>• Designed to be more fun</td>
</tr>
<tr>
<td>• Simple, but take time</td>
<td>• Graphics, game design</td>
</tr>
</tbody>
</table>

iClickers

- Have you registered your iclicker?
  - If not, visit [http://atsupport.cit.cornell.edu/pollsrv/](http://atsupport.cit.cornell.edu/pollsrv/)
- Instructions on iClickers can be found here:
  - Find these links on the course webpage
  - Click “Texts/iClickers”
  - Look under “iClickers”

Type: Set of values and the operations on them

<table>
<thead>
<tr>
<th>Type</th>
<th>Values</th>
<th>Ops</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>int</strong></td>
<td>integers</td>
<td>+, –, //, %, **</td>
</tr>
<tr>
<td><strong>float</strong></td>
<td>real numbers</td>
<td>+, –, *, /, **</td>
</tr>
<tr>
<td><strong>bool</strong></td>
<td>True and False</td>
<td>not, and, or</td>
</tr>
<tr>
<td><strong>str</strong></td>
<td>string literals</td>
<td>+ (concatenation)</td>
</tr>
</tbody>
</table>

Converting Values Between Types

- Basic form: \texttt{type(value)}
  - \texttt{float(2)} converts value 2 to type \texttt{float} (value now 2.0)
  - \texttt{int(2.8)} converts value 2.6 to type \texttt{int} (value now 2)
  - Explicit conversion is also called “casting”
- Narrow to wide: \texttt{bool ⇒ int ⇒ float}

  - **Widening.** Python does automatically if needed
    - Example: 1/2.0 evaluates to 0.5 (casts 1 to \texttt{float})
  - **Narrowing.** Python never does this automatically
    - Narrowing conversions cause information to be lost
    - Example: \texttt{float(int(2.6))} evaluates to 2.0

Operator Precedence

- What is the difference between the following?
  - 2*(1+3) \textbf{add, then multiply}
  - 2*1 + 3 \textbf{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

Type: Set of values and the operations on them
**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
  - Example: `1/2*3` is `(1/2)*3`

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

---

**Expressions vs Statements**

<table>
<thead>
<tr>
<th>Expression</th>
<th>Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Represents something</td>
<td>Does something</td>
</tr>
<tr>
<td>Python evaluates it</td>
<td>Python executes it</td>
</tr>
<tr>
<td>End result is a value</td>
<td>Need not result in a value</td>
</tr>
</tbody>
</table>

- Examples:
  - `2.3` Value
  - `(3+5)/4` Complex Expression

- Will see later this is not a clear cut separation

---

**Variables (Section 2.1)**

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

- Examples:
  - `x = 5`
  - `area = 20.1`

- `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
  - 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

- 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

---

**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

- 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

- Often want to track the type in a variable
  - What is the result of evaluating `x / y`?
  - Depends on whether `x`, `y` are `int` or `float` values

- Use expression `type(<expression>)` to get type
  - `type(2)` evaluates to `<type 'int'>`
  - `type(x)` evaluates to type of contents of `x`

- Can use in a boolean expression to test type
  - `type('abc') == str` evaluates to `True`