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 in Carpenter Hall Atrium between lectures

Labs vs. Assignments

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

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

- **Type int:**
  - Values: integers
  - Ops: +, -, *, /, %, **
- **Type float:**
  - Values: real numbers
  - Ops: +, -, *, /, **
- **Type bool:**
  - Values: True and False
  - Ops: not, and, or
- **Type str:**
  - Values: string literals
  - Double quotes: "abc"
  - Single quotes: 'abc'
  - Ops: + (concatenation)

Converting Values Between Types

- Basic form: `type(value)`
  - `float(2)` converts value 2 to type `float` (value now 2.0)
  - `int(2.8)` 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

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
- 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></td>
</tr>
<tr>
<td>Python evaluates it</td>
<td></td>
</tr>
<tr>
<td>End result is a value</td>
<td></td>
</tr>
<tr>
<td>Examples:</td>
<td></td>
</tr>
<tr>
<td>2.8</td>
<td></td>
</tr>
<tr>
<td>(3+5)/4</td>
<td></td>
</tr>
<tr>
<td>Complex Expression</td>
<td></td>
</tr>
<tr>
<td>Value</td>
<td></td>
</tr>
</tbody>
</table>

• Does something
• Python executes it
• Need not result in a value
• Examples:
  * print('Hello')
  * import sys

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:

- Variable x, with value 5 (of type int)
- Variable area, with value 20.1 (of type float)

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 value
  * 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

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

- The following is acceptable in Python:
  >>> x = 1
  >>> x = x / 2.0

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

Dynamic Typing

- 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(x) 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