Announcements for Today

If Not Done Already
• Enroll in Piazza
• Sign into CMS
  * Fill out the Survey
  * Complete AI Quiz
• Read the textbook
  * Chapter 1 (browse)
  * Chapter 2 (in detail)

Lab 1
• Please stay in your section
  * If you drop, you are stuck
• Fill out the Survey
• Complete AI Quiz
• Read the textbook
  * Chapter 1 (browse)
  * Chapter 2 (in detail)

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!
  * Will make an announcement next week

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

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

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>
<tr>
<td>Examples:</td>
<td>Examples:</td>
</tr>
<tr>
<td>2.3</td>
<td>print “Hello”</td>
</tr>
<tr>
<td>(2+5)/4</td>
<td>import sys</td>
</tr>
</tbody>
</table>

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:**
  - 5
    - variable `x`, with value 5 (of type `int`)
  - 20.1
    - variable `area`, w/ value 20.1 (of type `float`)

- Variable names must start with a letter (or `_`).
- The **type** belongs to the **value**, not to the **variable**.

### Variables and Assignment Statements

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

- **Two steps to execute an assignment:**
  1. Evaluate the expression on the right
  2. Store the result in the variable on the left

### Execute the Statement: \( x = x + 2 \)

- **Draw variable** \( x \) **on piece of paper**:
  - \( x = 5 \)

- **Step 1:** evaluate the expression \( x + 2 \)
  - **For** \( x \), use the value in variable \( x \)
  - **Write** the expression somewhere on your paper

- **Step 2:** Store the value of the expression in \( x \)
  - **Cross off** the old value in the box
  - **Write** the new value in the box for \( x \)

- Check to see whether you did the same thing as your neighbor, discuss it if you did something different.

### 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` \( \rightarrow \) \( x \) **contains** an **int** value
  - `x = 2.0` \( \rightarrow \) \( x \) 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

- 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`