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
- E-mail conflicts to Jessica jd648@cornell.edu
- Will review by next week
- Have one week to complete
  - Fill out questions on handout
  - Show to TA before next lab
  - Show in consulting hours

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
- Type str:
  - Values: string literals
    - Double quotes: "abc"
    - Single quotes: 'abc'
  - Ops: + (concatenation)

Will see more types in a few weeks

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 “left to right”
- Example: $1/2*3$ is $(1/2)*3$ 

Section 2.7 in your text
- See website for more info
- 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>
<tr>
<td>Examples:</td>
<td>Examples:</td>
</tr>
</tbody>
</table>
| $2.3$ | print "Hello"
| $(3+6)/4$ | 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:

  \[ 5 \]
  Variable \( x \), with value 5 (of type \( \text{int} \))

  \[ 20.1 \]
  Variable \( area \), with value 20.1 (of type \( \text{float} \))

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

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

Variables and Assignment Statements

- Variables are created by assignment statements

  \[ x = 5 \]
  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 \( \texttt{>>=} \) 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 value of the variable \( x \)

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 \( \text{type}(x) \) to find out the type of the value in \( x \)
  * Use names of types for conversion, comparison

  \[ \text{type}(x) == \text{int} \quad x \quad \text{contains an int value} \]
  \[ \text{type}(x) == \text{float} \quad x \quad \text{now contains a float value} \]

- 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 \( \text{int} \) or \( \text{float} \) values

- Use expression \( \text{type}(<\text{expression}>) \) to get type

  \[ \text{type}(2) \text{ evaluates to } \text{<type 'int'>} \]
  \[ \text{type}(x) \text{ evaluates to type of contents of } x \]

- Can use in a boolean expression to test type

  \[ \text{str} \text{ evaluates to } \text{True} \]