

## 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](mailto: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 "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

### Expression

### Statement

- **Represents** something
  - Python *evaluates it*
  - End result is a value
- **Examples:**
  - $2,3$  Value
  - $(3+5)/4$  Complex Expression
- **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

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

- Examples:

Variable names must start with a letter (or \_).

x 5 Variable **x**, with value 5 (of type **int**)

area 20.1 Variable **area**, w/ value 20.1 (of type **float**)

The type belongs to the **value**, not to the **variable**.

## Variables and Assignment Statements

- Variables are created by **assignment statements**

“gets” Create a new variable name and give it a value

x = 5

the 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

the variable

Two steps to execute an assignment:  
1. evaluate the expression on the right  
2. store the result in the variable on the left

8/27/15

## 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.

## Execute the Statement: x = x + 2

- The variable x

x 5

- The command:

- Step 1: **Evaluate** the expression **x + 2**
- Step 2: **Store** its value in x

- This is how you execute an assignment statement

- Performing it is called **executing the command**
- Command requires both **evaluate** AND **store** to be correct
- Important *mental model* for understanding Python

## 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

type(x) == int  
x = float(x)  
type(x) == float

## 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**