Assignments

• Major portion (40%) of your final grade
  § Larger projects due every two weeks
• First assignment requires mastery
  § Submit, get feedback, resubmit… until correct
  § Everyone eventually scores 10/10
• Later assignments are designed to be fun
  § Examples: graphics, image manipulation
  § Final project is a Breakout game project
• Submitted via Course Management System (CMS)
  § Visit cms.csuglab.cornell.edu to check you are enrolled

Helping You Succeed: Other Resources

• 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 professors!
  § Available in Thurston 102 between lectures

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

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 (for all but **) 
  § Example: 1/2*8 is (1/2)*8

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 names must start with a letter (or _).
  The value in the box is then used in evaluating the expression.

  Variable x, with value 5 (of type int)
  Variable area, w/ value 20.1 (of type float)
**Variables and Assignment Statements**

- Variables are created by assignment statements
  - `sets` Create a new variable name and give it a value
    - `the 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
      - `the expression` `x = x + 2`

**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

**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

**String: Text as a Value**

- String are quoted characters
  - `"abc d"` (Python prefers)
  - `"abc d"` (most languages)
  - How to write quotes in quotes?
    - Delineate with "other quote"
    - Example: `"'" or "'`
    - What if need both ‘ and '?
    - Solution: escape characters
      - Format: \\ + letter
      - Special or invisible chars

**String are Indexed**

- `s = 'abc d'
  - `s[0]` is 'a'
  - `s[4]` is 'd'
  - `s[5]` causes an error
  - `s[0:2]` is 'ab' (excludes 0)
  - `s[2:]` is 'c d'
  - Called "string slicing"

- `s = 'Hello all'
  - What is `s[3:6]`?
  - What is `s[4]`?

**Strings have many other powers**

- More (too much?) information in Python documentation on [www.python.org](http://www.python.org) (see Library Reference, built-in types)