CS 1133 Fall 2016: Walker White

• Outcomes:
  § Competency with basic Python programming
    • Ability to create Python modules and programs
    • Ability to use the most common built-in data types
  § Knowledge of object-oriented programming
    • Ability to recognize and use objects in Python.
    • Ability to understand classes written by others.
• Website:
  § www.cs.cornell.edu/courses/cs1133/2016fa/

Grading Policy

• There will be three assignments
  § Two smaller assignments, one larger
  § All will involve programming
• Must earn 85% to pass an assignment
  § Get two more attempts if you fail
  § But you must meet the posted deadlines!
• Must pass all three assignments
  § No exams; labs are not graded

Getting Started with Python

• Designed to be used from the “command line”
  § OS X/Linux: Terminal
  § Windows: Command Prompt
• Purpose of the first lab
  • Once installed type “python”
    § Starts an interactive shell
    § Type commands at `>>>`
    § Shell responds to commands
  • Can use it like a calculator
    § Use to evaluate expressions

This class uses Python 2.7.x
• Python 3 has many "issues"
  • May be incompatible

The Basics

<table>
<thead>
<tr>
<th>Types</th>
<th>Values</th>
<th>Expressions</th>
</tr>
</thead>
<tbody>
<tr>
<td>int (integer)</td>
<td>12,345</td>
<td>“Hello!” + “World”</td>
</tr>
<tr>
<td>float (real number)</td>
<td>22.51e6</td>
<td>34 “ (33 + 14)</td>
</tr>
<tr>
<td>string (of characters)</td>
<td>“Hello”</td>
<td>“Hello” * “5”</td>
</tr>
</tbody>
</table>

Type: Set of values and the operations on them

• Type int (integer):
  * values: …, 3, –2, –1, 0, 1, 2, 3, 4, 5, …
  * operations: +, −, *, /, **, unary −
  * Principal: operations on int values must yield an int
    * Example: 1 / 2 rounds result down to 0
      • Companion operation: % (remainder)
      • 7 % 3 evaluates to 1, remainder when dividing 7 by 3
      • Operator / is not an int operation in Python 3 (use // instead)

• Type float (floating point):
  * values: (approximations of) real numbers
    • In Python a number with a “.” is a float literal (e.g. 3.0)
    • Without a decimal a number is an int literal (e.g. 3)
  * operations: +, −, *, /, **, unary −
    • But meaning is different for floats
    • Example: 1.0/2.0 evaluates to 0.5
• Exponent notation is useful for large (or small) values
  • −22.61e6 is −22.51 * 10^6 or −22510000
  • 22.81e−6 is 22.51 * 10^-6 or 0.000002251

A second kind of float literal
**Type: Set of values and the operations on them**

- **Type boolean or bool:**
  - values: True, False
  - Boolean literals are just True and False (have to be capitalized)
  - operations: not, and, or
    - not b: True if b is false and False if b is true
    - b and c: True if both b and c are true; False otherwise
    - b || c: True if b is true or c is true; False otherwise
  - Often come from comparing int or float values
    - Order comparison: i < j i <= j i = j i != j i > j
    - Equality, inequality: i == j i != j

- **Type String or str:**
  - values: any sequence of characters
  - operation(s): + (catenation, or concatenation)
    - String literal: sequence of chars in quotes
      - Double quotes: "abc+x3$g<&" or "Hello World!"
      - Single quotes: 'Hello World!'
  - Concatenation can only apply to Strings.
    - "ab" + "cd" evaluates to "abcd"
    - "ab" + 2 produces an error

---

**Expressions vs Statements**

<table>
<thead>
<tr>
<th>Expression</th>
<th>Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Represents something</strong></td>
<td><strong>Does something</strong></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><strong>Examples:</strong></td>
<td><strong>Examples:</strong></td>
</tr>
<tr>
<td>* 2.3</td>
<td>* print &quot;Hello&quot;</td>
</tr>
<tr>
<td>* (3+6)/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),
  - a value (in the box)
- Examples
  - x = 5 Variable x, with value 5 (of type int)
  - area = 20.1 Variable area, w/ value 20.1 (of type float)
- Variable names must start with a letter
  - So 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 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

---

**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