Lecture 6

Objects
Announcements for this Lecture

Last Call

• Quiz: About the Course
• Take it by tomorrow
• Also remember survey

Assignment 1

• Assignment 1 is live
  ▪ Posted on web page
  ▪ Due Thur, Sep. 18th
  ▪ Due in place of Lab 4
• Lab 3 will help a lot
  ▪ Testing is a major part
  ▪ Try to finish it first
  ▪ But start this Saturday!
One-on-One Sessions

• Still ongoing: **1/2-hour one-on-one sessions**
  ▪ To help prepare you for the assignment
  ▪ **Primarily for students with little experience**

• There are still some spots available
  ▪ Sign up for a slot in CMS

• Will keep running after **September 18**
  ▪ Will open additional slots after the due date
  ▪ Will help students revise Assignment 1
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)

Are the the only types that exist?
Type: Set of values and the operations on them

- Want a point in 3D space
  - We need three variables
  - $x$, $y$, $z$ coordinates
- What if have a lot of points?
  - Vars $x_0$, $y_0$, $z_0$ for first point
  - Vars $x_1$, $y_1$, $z_1$ for next point
  - ...
  - This can get really messy
- How about a single variable that represents a point?
Type: Set of values and the operations on them

- Want a point in 3D space
  - We need three variables
  - $x, y, z$ coordinates
- What if have a lot of points?
  - Vars $x_0, y_0, z_0$ for first point
  - Vars $x_1, y_1, z_1$ for next point
  - ...
  - This can get really messy
- How about a single variable that represents a point?

- Can we stick them together in a “folder”? 
- Motivation for objects

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$x$</td>
<td>2.0</td>
<td></td>
</tr>
<tr>
<td>$y$</td>
<td>3.0</td>
<td></td>
</tr>
<tr>
<td>$z$</td>
<td>5.0</td>
<td></td>
</tr>
</tbody>
</table>
Objects: Organizing Data in Folders

- An object is like a **manila folder**
- It contains other variables
  - Variables are called **attributes**
  - These values can change
- It has an **ID** that identifies it
  - Unique number assigned by Python (just like a NetID for a Cornellian)
  - Cannot ever change
  - Has no meaning; only identifies

| id1 | x   | 2.0 |
|     | y   | 3.0 |
|     | z   | 5.0 |
Classes: Types for Objects

• Values must have a type
  ▪ An object is a value
  ▪ Object type is a class

• Modules provide classes
  ▪ Will show how later

• Example: tuple3d
  ▪ Part of CornellExtensions
  ▪ Just need to import it
  ▪ Classes: Point, Vector
Constructor: Function to make Objects

• How do we create objects?
  ▪ Other types have literals
  ▪ Example: 1, "abc", true
  ▪ No such thing for objects

• Constructor Function:
  ▪ Same name as the class
  ▪ Example: Point(0,0,0)
  ▪ Makes an object (manila folder)
  ▪ Returns folder ID as value

• Example: p = Point(0, 0, 0)
  ▪ Creates a Point object
  ▪ Stores object’s ID in p
>>> import tuple3d

Need to import module that has Point class.

>>> p = tuple3d.Point(0,0,0)

Constructor is function. Prefix w/ module name.

>>> id(p)

Shows the ID of p.

Actually a big number
Object Variables

- Variable stores object name
  - Reference to the object
  - Reason for folder analogy

- Assignment uses object name
  - Example: \( q = p \)
  - Takes name from \( p \)
  - Puts the name in \( q \)
  - Does not make new folder!

- This is the cause of many mistakes in this course
Objects and Attributes

• Attributes are variables that live inside of objects
  - Can **use** in expressions
  - Can **assign** values to them

• **Access:** `<variable>`.`<attr>`
  - **Example:** `p.x`
  - Look like module variables

• Putting it all together
  - `p = tuple3d.Point(1,2,3)`
  - `p.x = p.y + p.z`
Exercise: Attribute Assignment

- Recall, q gets name in p
  >>> p = tuple3d.Point(0,0,0)
  >>> q = p

- Execute the assignments:
  >>> p.x = 5.6
  >>> q.x = 7.4

- What is value of p.x?
  A: 5.6
  B: 7.4
  C: id4
  D: I don’t know
Exercise: Attribute Assignment

• Recall, q gets name in p
  >>> p = tuple3d.Point(0,0,0)
  >>> q = p

• Execute the assignments:
  >>> p.x = 5.6
  >>> q.x = 7.4

• What is value of p.x?

   A: 5.6
   B: 7.4  CORRECT
   C: id4
   D: I don’t know
Call Frames and Objects

• Mutable objects can be altered in a function call
  ▪ Object vars hold names!
  ▪ Folder accessed by both global var & parameter

• Example:

```python
def incr_x(q):
    q.x = q.x + 1

>>> p = Point(0,0,0)

>>> incr_x(p)
```

Global STUFF

Call Frame

9/2/12 Objects 15
Call Frames and Objects

- Mutable objects can be altered in a function call
  - Object vars hold names!
  - Folder accessed by both global var & parameter

- Example:

```python
def incr_x(q):
    q.x = q.x + 1

>>> p = Point()
>>> incr_x(p)
```

Global STUFF

Call Frame

Objects
• Mutable objects can be altered in a function call
  ▪ Object vars hold names!
  ▪ Folder accessed by both global var & parameter

• Example:

```python
def incr_x(q):
    q.x = q.x + 1
```

```python
>>> p = Point()
>>> incr_x(p)
```

Global STUFF

Call Frame

Objects
Methods: Functions Tied to Objects

- **Method**: function tied to object
  - Method call looks like a function call preceded by a variable name:
    \[(\text{variable}) \cdot (\text{method})(\text{arguments})\]
  - **Example**: `p.distanceTo(q)`
  - **Example**: `p.abs()` # makes x, y, z ≥ 0

- Just like we saw for strings
  - `s = 'abracadabra'`
  - `s.index('a')`

- **Are strings objects?**
Surprise: All Values are in Objects!

• Including basic values
  - int, float, bool, str

• Example:
  >>> x = 2.5
  >>> id(x)

• But they are immutable
  - Contents cannot change
  - Distinction between value and identity is immaterial
  - So we can ignore the folder
Surprise: All Values are in Objects!

• Including basic values
  ▪ int, float, bool, str

• Example:
  >>> x = 'foo'
  >>> id(x)

• But they are immutable
  ▪ No string method can alter the contents of a string
  ▪ x.replace('o','y') evaluates to 'fyy' but x is still 'foo'
  ▪ So we can ignore the folder
Class Objects

- Use name **class object** to distinguish from other values
  - Not int, float, bool, str
- Class objects are **mutable**
  - You can change them
  - Methods can have effects besides their return value
- **Example:**
  - \( p = \text{Point}(3,-3,0) \)
  - \( p.\text{clamp}(-1,1) \)

**Example:** Files

\[
f = \text{open('jabber.txt')}\\
s = f.\text{read()}\\
f.\text{close()}
\]

Opens a file on your disk; returns a **file object** you can read
# Base Types vs. Classes

<table>
<thead>
<tr>
<th>Base Types</th>
<th>Classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Built-into Python</td>
<td>• Provided by modules</td>
</tr>
<tr>
<td>• Refer to instances as <em>values</em></td>
<td>• Refer to instances as <em>objects</em></td>
</tr>
<tr>
<td>• Instantiate with <em>literals</em></td>
<td>• Instantiate w/ <em>constructors</em></td>
</tr>
<tr>
<td>• Are all immutable</td>
<td>• Can alter attributes</td>
</tr>
<tr>
<td>• Can ignore the folders</td>
<td>• Must represent with folders</td>
</tr>
</tbody>
</table>
Aside: Name Resolution

- `<object>.<name>` means
  - Go the folder for `object`
  - Look for attr/method `name`
  - If missing, check `class folder`
- Class folder is a **shared folder**
  - Only one for the whole class
  - Shared by all objects of class
  - Stores common features
  - Typically where methods are
- Do not worry about this yet

```
Point
__init__(x, y, z)
distanceTo(other)
abs()
```
Where To From Here?

- Right now, just try to understand **objects**
  - All Python programs use objects
  - Most small programs use objects of classes that are part of the Python Library
- OO Programming is about **creating classes**
  - Eventually you will make your own classes
  - Classes are the primary tool for organizing more complex Python programs
  - But we need to learn other basics first
A1: The Module urllib2

- Module urllib2 is used to read web pages
  - Function urlopen creates a url object
  - \( u = \text{urllib2}.\text{urlopen}('http://www.cornell.edu') \)

- url has a method called read()
  - Returns contents of web page
  - **Usage**: \( s = u.\text{read}() \) # \( s \) is a string