Announcements for Today

Reading
- Today: Chapter 18
- Online reading for Thursday

Assignments
- A4 graded by end of week
- Survey is still open
- A5 was posted Friday
- Shorter written assignment
- Due Thursday at Midnight
- A6 also posted Friday
- Due a week after prelim
- Designed to take two weeks
- Finish first part before exam

- Prelim, Nov 10th 7:30-9:00
  - Material up to Thursday
  - Review posted on Thursday
  - Recursion + Loops + Classes

- S/U Students are exempt

- Conflict with Prelim time?
  - Submit by Thursday

An Application

**Goal:** Presentation program (e.g. PowerPoint)

**Problem:** There are many types of content
- Examples: text box, rectangle, image, etc.
- Have to write code to display each one

**Solution:** Use object oriented features
- Define class for every type of content
- Make sure each has a draw method:

```python
for x in slide[i].contents:
    x.draw(window)
```

**Defining a Subclass**

```
class TextBox(SlideContent):
    """An object containing text."
    ""
    def __init__(self, x, y, text):
        ...  
    def draw(self):
        ...
```

```
class Image(SlideContent):
    """An image."
    ""
    def __init__(self, x, y, image_file):
        ...  
    def draw(self):
        ...
```

**Class Definition: Revisited**

```
class <name>(<superclass>):
    """Class specification"
    ""
    # getters and setters
    # initializer (__init__)
    # definition of operators
    # definition of methods
    # anything else
```

- Every class must extend something
- Previous classes all extended object

**Name Resolution Revisited**

1. Look first in instance (object folder)
2. Then look in the class (folder)
3. Subclasses add two more rules:
4. Repeat 3. until reach object

**Kivy Example**

```python
kivy.uix.widget.WidgetBase
kivy.uix.widget.Widget
kivy.uix.label.Label
```

Subclassing creates a hierarchy of classes
- Each class has its own super class or parent
- Until object at the "top"
- object has many features
  - Special built-in fields: __class__, __dict__
  - Default operators: __str__, __repr__

Abbreviate: SC to right

Object and the Subclass Hierarchy

```python
object
kivy.uix.widget.WidgetBase
kivy.uix.widget.Widget
kivy.uix.label.Label
```
A Simpler Example

class Employee(object):
    """An Employee with a bonus."""
    INSTANCE ATTRIBUTES:
    _bonus: annual bonus [float]"

class Executive(Employee):
    """An Employee with a bonus."""
    INSTANCE ATTRIBUTES:
    _bonus: annual bonus [float]"

Accessing the “Previous” Method

- What if you want to use the original version method?
  - New method = original + more
  - Do not want to repeat code from the original version
- Call old method explicitly
  - Use method as a function
  - Pass object as first argument
- Example:
  Employee.__str__(self)
  Cannot do with properties

Primary Application: Initializers

- Which __str__ do we use?
  - Start at bottom class folder
  - Find first method with name
  - Use that definition
- New method definitions override those of parent
- Also applies to
  - Initializers
  - Operators
  - Properties

Instance Attributes are (Often) Inherited

- Class Attribute: Assigned outside of any method definition

Also Works With Class Attributes

- Class Attribute: Assigned outside of any method definition