### 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 12th 7:30-9:00
  - Material up to Thursday
  - Review posted on Thursday
  - recursion + loops + classes

- S/U Students are exempt
- Conflict with Prelim time?
  - Prelim 2 Conflict on CMS
  - Submit by Thursday

### An Application
- **Goal**: Presentation program (e.g. PowerPoint)
- **Problem**: There are many types of content
  - Examples: text box, rectangle, image, etc.
- **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

#### Kivy Example
```python
object
kivy.uix.widget.Widget
kivy.uix.widget.Button
kivy.uix.label.Label

# Special built-in fields:
__class__, __dict__

# Default operators:
__str__, __repr__
```

### Class Definition: Revisited

#### Summary
- Every class must extend **something**
- Previous classes all extended `object`

### Name Resolution Revisited

- To look up attribute/method name
  1. Look first in instance (object folder)
  2. Then look in the class (folder)
- Subclasses add two more rules:
  3. Look in the superclass
  4. Repeat 3. until reach `object`

---

**Abbreviate as SC to right:**

```python
class SlideContent(object):
    """Any object on a slide."""
    def __init__(self, x, y, w, h):
        def draw_frame(self):
            def select(self):

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):
```

---

**Abbreviate as SC to right:**

```python
class SlideContent(object):
    """Any object on a slide."""
    def __init__(self, x, y, w, h):
        def draw_frame(self):
            def select(self):

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):
```

---

**Abbreviate as SC to right:**

```python
class SlideContent(object):
    """Any object on a slide."""
    def __init__(self, x, y, w, h):
        def draw_frame(self):
            def select(self):

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):
```
Instance Attributes are (Often) Inherited

```
class Employee(object):

    __init__(self, n, d, s):
        self._name = n
        self._start = d
        self._salary = s

    __str__(self):
        return ('name: ' + self._name +
                ', year: ' + str(self._start) +
                ', salary: ' + str(self._salary))

dem Executive(Employee):

    __init__(self, n, d, b):
        self._name = n
        self._start = d
        self.bonus = b
```

Also Works With Class Attributes

```
Class Attribute: Assigned outside of any method definition

class Employee(object):

    """Instance is a salaried worker""
    # Class Attribute
    STD_SALARY = 50000.0

class Executive(Employee):

    """An Employee with a bonus""
    # Class Attribute
    STD_BONUS = 10000.0
```