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)
```

Class Definition: Revisited

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

Name Resolution Revisited

1. Look first in instance (object folder)
2. Then look in the class (folder)
3. Look in the superclass
4. Repeat 3. until reach object

Defining a Subclass

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

class TextBox(SlideContent):
    """An object containing text."
    def __init__(self, x, y, text):
        SlideContent.__init__(self, x, y, 0, 0)
        self.text = text
    def draw(self):
        ...

class Image(SlideContent):
    """An image."
    def __init__(self, x, y, image_file):
        SlideContent.__init__(self, x, y, 0, 0)
        self.image_file = image_file
    def draw(self):
        ...
```

Object and the Subclass Hierarchy

- 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__`

Kivy Example

```python
import kivy

kivy.app.get_running_app()
```
Method Overriding

- 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
  - Access to old version is lost
  - New version used instead
- Example: \_\_init\_

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
- Use the function super()
  - “Converts” type to parent class
  - Now methods go to the class
- Example:

```python
super().__str__()
```

About super()

- super() is very limited
  - Can only go one level
  - BAD: super(), super()

Primary Application: Initializers

- Need arguments for more
  - super(class, self)

Instance Attributes are (Often) Inherited

- Created in Employee initializer
- Created in Executive initializer

Also Works With Class Attributes

- Class Attribute: Assigned outside of any method definition