INTRO. TO OBJECT-ORIENTED PROGRAMMING IN PYTHON

Curt Clifton
Rose-Hulman Institute of Technology

TODAY'S PLAN

- Some notes on *scope*
- Brief introduction to syntax for objects in Python
- OO Exercise
- Remember:
  - Milestone 1 due tomorrow night
  - Project Friday tomorrow, no class

PREPARATION

- In Eclipse, check out the *PythonOOIntro* project from your individual repository for the course
- Open the file *scope.py*

http://icanhascheezburger.com/category/loldog/page/3/

SCOPE IN PYTHON

- See code and comments in *scope.py* to answer quiz questions 1 and 2

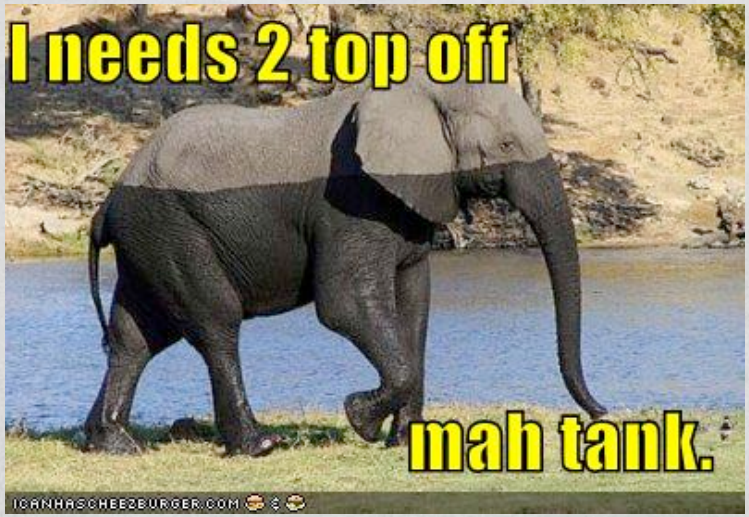
Q1,2

BUT I WANT TO ASSIGN TO THE TOP-LEVEL VARIABLE!

- You can prevent Python from creating a shadowing, local variable using *global*
- Example:

```
def fn3():
    global x
    print "x in fn3:", x
    x = 15
    print "x in fn3:", x
```

MUTATION != ASSIGNMENT



- Look at *fn4* and quiz question 4

IMPORT AND ALIASING

- See *scope_user.py*
- Quiz questions 5 and 6

Q5,6

BUILT-IN SCOPE

- Python doesn't keep you from assigning to built-in names
- Try this:
  - Add this code to *scope.py:*

```
print str(1)
def str(n):
    return 'boo'
print str(1)
```

  - Run *scope.py*
  - Add `print str(1)` to *scope_user.py* and run it
- Definition of *str* in *scope.py* shadows the built-in!

OBJECTS IN PYTHON

- Class definitions
- Class attributes
- Instantiation
- “Fields” and “methods”

- Code for coming examples is in *class_examples.py*

CLASS DEFINITIONS

```
class ClassName:
    """Doc string."""
    # 0 or more additional statements
```

CLASS ATTRIBUTES

```
class Attrib:
    """Example of class attributes."""
    x, y = 2, 13

print "Attrib:", Attrib.x, Attrib.y
```

FUNCTIONS AS CLASS ATTRIBUTES

```
class AttribWithFunc:
    """Example adding fn attribute."""
    def fact(n):
        result = 1
        for i in xrange(1, n+1):
            result *= i
        return result

print "fact:", AttribWithFunc.fact
print "Calling fact:", AttribWithFunc.fact(5)
```

Error!

Q7, **not** Q8

CLASS INSTANTIATION

```
class MakeMe:
    """Example for instantiation."""
    def __init__(self, x):
        self._x = x

one = MakeMe(1)
two = MakeMe(2)
print "One-two punch:", one._x, two._x
```

FIELDS AND METHODS

- Fields
  - Like local variables, they're created by assignment
- Methods
  - Functions that "belong to" objects
  - **All** class functions are methods!

```
class CountDown:
    def __init__(self, n):
        self._n = n
    def tick(self):
        self._n -= 1
        if self._n <= 0:
            print 'BOOM!'

counter = CountDown(5)
for i in xrange(8):
    counter.tick()
```

EXERCISE

- In the file *television.py*…
- Create a class that models a television, including:
  - On/off status
  - Current channel
  - Volume setting
  - Mute setting
- Include methods for adjusting all the settings
- Notes:
  - Volume should return to previous value when unmuting
  - TV should be unmuted when turned on

Commit your work to SVN when done!