MORE OBJECTS IN PYTHON

Curt Clifton
Rose-Hulman Institute of Technology

Check out *PythonIterators* from SVN

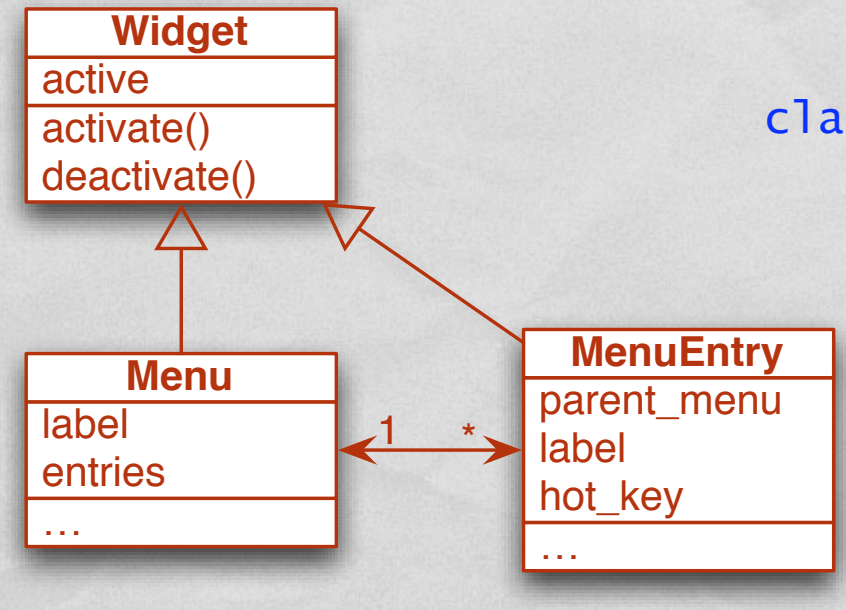
TODAY'S PLAN

- Inheritance, multiple inheritance, "super" calls
- Operator overloading and other "special" methods
- Iterators
- Teams

INHERITANCE IN PYTHON

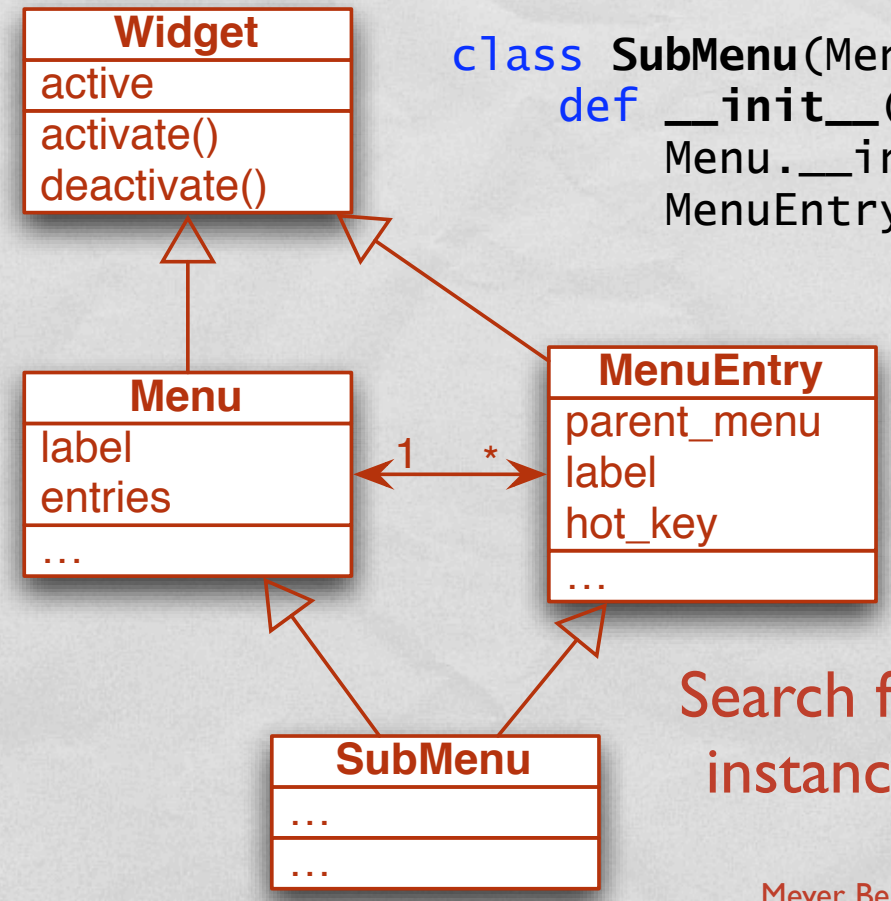
```
class Widget:
    def __init__(self):
        self.active = True
    def activate(self):
        self.active = True
    def deactivate(self):
        self.active = False
```

```
class Menu(Widget):
    def __init__(self, label):
        Widget.__init__(self)
        self.label = label
        self.entries = []
    def addEntry(self, entry):
        entry.setParent(self)
        self.entries.append(entry)
```




```
class MenuEntry(Widget):
    def __init__(self, label, hot_key):
        Widget.__init__(self)
        self.label = label
        self.hot_key = hot_key
        self.parent_menu = None
        self.active = False
    def setParent(self, parent):
        self.parent_menu = parent
```

MULTIPLE INHERITANCE




```
class SubMenu(Menu, MenuEntry):
    def __init__(self, label, hot_key):
        Menu.__init__(self, label)
        MenuEntry.__init__(self, label, hot_key)
```

```
print "Label:", copyAs.label
print "Active:", copyAs.active
print "Entries:", copyAs.entries
print "Hot Key:", copyAs.hot_key
```

Search for attributes always starts at the instance type and proceeds depth-first

Meyer, Bertrand. "Harnessing Multiple Inheritance" *Journal of Object-Oriented Programming*. 1(4), pp. 48–51. Available from *http://archive.eiffel.com/doc/manuals/technology/bmarticles/joop/multiple.html*

Q2,3

OPERATOR OVERLOADING

- Definition:
  - Changing or specifying the meaning of operators or built-in functions in a language when they are applied to instances of custom data types
- Example:
  - Overriding the *__str__(self)* method changed the behavior of *str()* built-in function used by *print*

OPERATOR OVERLOADING EXAMPLE

```
class Student:
    """A small sample of operator overloading.

    >>> s = Student()
    >>> print s
    freshman
    >>> print s + 1
    sophomore
    >>> print s + 10
    super-sr
    """
    year_names = ['freshman', 'sophomore', 'jr', 'sr', 'super-sr']
    def __init__(self, year = 0):
        self.year = year
    def __str__(self):
        return Student.year_names[self.year]
    def __add__(self, num):
        new_year = min(self.year + num, len(Student.year_names) - 1)
        return Student(new_year)
```

SOME OTHER "SPECIAL" METHODS

- *__lt__(self, other)*, also *le*, *eq*, *ne*, *gt*, *ge*
- *__hash__(self)*
- *__add__(self, other)*, also *sub*, *mul*, *div*
- See §3.4 of the Python reference for a metric bunch more

ITERATORS

- Iterating over containers is very common in Python:
  - *for p in [2, 3, 5, 7, 11]:*
    print p, "is prime"
  - *for c in 'Rose':*
    print c
- Can make our own iterable classes by:
  - Adding *__iter__(self)*
  - Making it return an object with a *next()* method
  - *next()* raises *StopIteration* at end

ITERATOR EXAMPLES

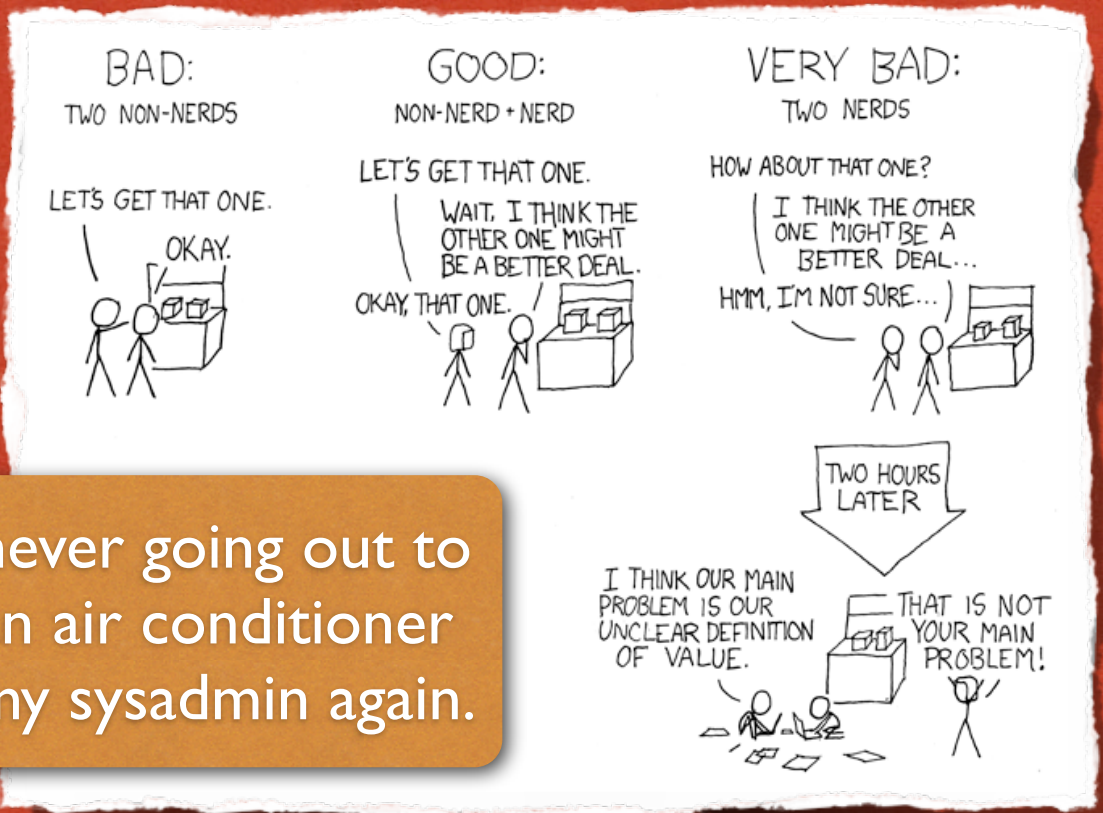
- Random-order iterator over a list without duplicating it
- Pre-order tree walk

```
class ShuffleIterator:
    def __init__(self, data):
        self.data = data
        self.order = range(len(data))
        random.shuffle(self.order)
        self.index = len(data)
    def __iter__(self):
        return self
    def next(self):
        if self.index == 0:
            raise StopIteration
        self.index -= 1
        itemIndex = self.order[self.index]
        return self.data[itemIndex]

s = 'Ni!'
for c in ShuffleIterator(s):
    print c
```

MILESTONE 2
PROJECT IDENTIFICATION

- **Team** milestone
- Due Friday at midnight
- Tasks—see milestone description for **specifics**
  - Email me your team name
  - Email me your preferred language, first-come basis!
  - Prepare a document identifying possible projects



I am never going out to buy an air conditioner with my sysadmin again.

SHOPPING TEAMS

CARTOON OF THE DAY

PROJECT TEAMS

- Boland, Crockett, Stark (Clojure?)
- Carlson, Jackson, Loughry (Objective J?)
- Fishman, Packard, Skaggs, Vanderford (Prolog? PHP?)
- Gahn, WIlliamson, Zynda (Objective C?)
- Jenne, Nowicki, Pick (Groovy?)
- Kent, Rhodes, Sickler (Smalltalk?)
- Knight, McGinnis, Pridal-LoPiccolo (Hadoop + Pig???)

TEAM MEETINGS

- Plan when to meet to work on milestone 2
- Share language preferences
- Start thinking about team names