Sorting and Comparators

Dan Fleck

Coming up: Sorting a list

Sorting a list

- Sorting a list seems simple:

```
myList = [1, 4, 3, 5]
myList.sort()
print myList
>>> [1, 3, 4, 5]
```

Backwards? Not so hard really

Coming up: Okay, how about backwards then!

Okay, how about backwards then!

- A little harder, but we can do it

```
myList = [1, 4, 3, 5]
myList.sort()
myList.reverse()
print myList
>>> [5, 4, 3, 1]
```

What about a heterogeneous list?

Okay, heterogeneous list?

- First, what is a heterogeneous list?
- *heterogeneous* - consisting of dissimilar or diverse ingredients or constituents
- *homogeneous* - of the same or a similar kind or nature

– mw.com

Okay, heterogeneous list?

- First, what is a heterogeneous list? In Python then:

- *heterogeneous list* – has elements of multiple data types (int, float, String, list)

- *homogeneous list* – all elements are the same type

So, lets sort one

```
myList = [1, ‘B’, 4, ‘A’, [1, 2, 3], 3, 5]
myList.sort()
print myList
>>> [1, 3, 4, 5, [1, 2, 3], 'A', 'B']
```

Uhhh… why?

The Python people had a problem… how does one compare an int to a list? Ideas?

Sorting heterogeneous types

They didn't have a good idea either, but in a computer you **MUST** provide an answer, even if it isn't a good one!

Python sorts alphabetically based on the name of the type.

ALL ints < ALL lists < ALL strings < ALL tuples

Note: Python does sort ints, floats, doubles together… so that will work, but all of them will be "less than" lists, strings, tuples)

Sorting classes

What about classes?

Oh… still alphabetic:

ALL classes < ALL ints < ALL lists < ALL strings < ALL tuples

What about just a homogeneous list of classes?

Sorting classes

What about just a homogeneous list of classes?

```
class MyClass:
    def __init__(self, x):
        self.x = x
c1 = MyClass(1)
c2 = MyClass(2)
myList = [c2, c1]
myList.sort()
print myList
```

```
[<__main__.MyClass instance at 0x70148>, <__main__.MyClass instance at 0x70120>]
[<__main__.MyClass instance at 0x70120>, <__main__.MyClass instance at 0x70148>]
```

Coming up: Making classes pretty

Making classes pretty

- classes… when you print a class, the default action is to print the memory location of the class. Not very helpful.
- You can override that behavior by providing a special method to be called:
  - def __repr__(self):

    “Returns a string representing this class”

Making classes pretty

```
class MyClass:
    def __init__(self, x):
        self.x = x

    def __repr__(self):
        “Returns a string representing this class”
        return ‘MyClass:’+str(self.x)
```

Back to sorting

```
c1 = MyClass(1)
c2 = MyClass(2)
myList = [c2, c1]
myList.sort()
print myList

>>> [MyClass:2, MyClass:1]
>>> [MyClass:1, MyClass:2]
```

It works! Or does it? How did it do the sorting? Can you think of how it KNOWS what to sort?

Neither could the Python authors!

Back to sorting

```
c2 = MyClass(2)
c1 = MyClass(1)
myList = [c2, c1]
myList.sort()
print myList
```

Change the order

```
>>> [MyClass:2, MyClass:1]
>>> [MyClass:2, MyClass:1]
```

Python doesn’t know how to sort your class unless *YOU* tell it how!

__cmp__ method

- Another special method in Python tells sort routines *HOW* do I compare two of these things?

-
  ```
def __cmp__(self, other):
 """Return -1 if self < otherInstance
    Return 0 if they are equal
    Return +1 if self > otherInstance """
 if self.x == other.x:
      return 0
 elif self.x < other.x:
      return -1
 else:
      return 1
```

What if I want descending order?

__cmp__ method

- Frequently you want to order by some instance variable that is part of the class. So, you can just use the built-in cmp method, which works for the basic types (string, int, float, etc…)

-
  ```
  def __cmp__(self, other):
   """Return -1 if self < otherInstance
      Return 0 if they are equal
      Return +1 if self > otherInstance """
   return cmp(self.x, other.x)
  ```

What if I want descending order?

Sorting

- In summary
  - Sorting in Python is done automatically for the built-in types.
  - Sorting heterogeneous types is done alphabetically by type name
  - Sorting a list of your own classes though requires a comparator function (__cmp__) where you tell Python how to compare two instances of your class
- And don't forget to define __repr__ just to make it easier for people to use your class.

Coming up: Lets Try it

Lets Try it

- Look at inclass_sort.py and make it sort by GPA and then names.

End of presentation