20.9. Sorting Lists of Instances

You previously learned how to sort lists (../Sorting/intro-SortingwithSortandSorted.html#sort-chap). Sorting lists of instances of a class is not fundamentally different from sorting lists of objects of any other type. There is a way to define a default sort order for instances, right in the class definition, but it requires defining a bunch of methods or one complicated method, so we won't bother with that. Instead, you should just provide a key function as a parameter to sorted (or sort).

Previously, you have seen how to provide such a function when sorting lists of other kinds of objects. For example, given a list of strings, you can sort them in ascending order of their lengths by passing a key parameter. Note that if you refer to a function by name, you give the name of the function without parentheses after it, because you want the function object itself. The sorted function will take care of calling the function, passing the current item in the list. Thus, in the example below, we write `key=len` and not `key=len()`.

Save & Run Download Load History Show CodeLens

```
1 L = ["Cherry", "Apple", "Blueberry"]
2 print(sorted(L, key=len))
3 #alternative form using lambda, if you find that easier to understand
4 print(sorted(L, key= lambda x: len(x)))
5
```

ActiveCode: 1 (sort_instances_1)

When each of the items in a list is an instance of a class, you need to provide a function that takes one instance as an input, and returns a number. The instances will be sorted by their numbers.

(InstancesasReturnValues.html) (ClassVariables.html) **Next Section - 20.10. Class V**

Save & Run Download Load History Show CodeLens

```
class Fruit():
    def __init__(self, name, price):
        self.name = name
        self.price = price
L = [Fruit("Cherry", 10), Fruit("Apple", 5), Fruit("Blueberry", 20)]
for f in sorted(L, key=lambda x: x.price):
    print(f.name)

```

ActiveCode: 2 (sort_instances_2)

Sometimes you will find it convenient to define a method for the class that does some computation on the data in an instance. In this case, our class is too simple to really illustrate that. But to simulate it, I've defined a method `sort_priority` that just returns the price that's stored in the instance. Now, that method, sort_priority takes one instance as input and returns a number. So it is exactly the kind of function we need to provide as the key parameter for sorted. Here it can get a little confusing: to refer to that method, without actually invoking it, you can refer to `Fruit.sort_priority` . This is analogous to the code above that referred to `len` rather than invoking `len()` .

Save & Run Download Load History Show CodeLens

```
class Fruit():
    def __init__(self, name, price):
        self.name = name
        self.price = price
    def sort_priority(self):
        return self.price
L = [Fruit("Cherry", 10), Fruit("Apple", 5), Fruit("Blueberry", 20)]
print "-----sorted by price, referencing a class method-----"
for f in sorted(L, key=Fruit.sort_priority):
    print(f.name)
print "---- one more way to do the same thing-----"
for f in sorted(L, key=lambda x: x.sort_priority()):
    print(f.name)

```

ActiveCode: 3 (sort_instances_3)

(InstancesasReturnValues.html) (ClassVariables Next Section 20.10. Class V

Mark as completed

cesasReturnValues.html)
(InstancesasReturnValues.html)
(ClassVariables**Next Section 20.10. Class V**