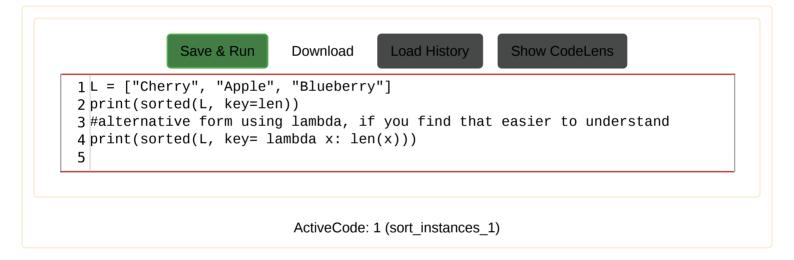
20.9. Sorting Lists of Instances

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().



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.



1 of 3 11/12/19, 8:24 pm

```
Save & Run
                           Download
                                        Load History
                                                        Show CodeLens
1 class Fruit():
     def __init__(self, name, price):
2
3
          self.name = name
          self.price = price
4
5L = [Fruit("Cherry", 10), Fruit("Apple", 5), Fruit("Blueberry", 20)]
6 for f in sorted(L, key=lambda x: x.price):
     print(f.name)
7
8
                           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
        1 class Fruit():
              def __init__(self, name, price):
        2
                  self.name = name
        3
        4
                  self.price = price
        5
              def sort_priority(self):
                  return self.price
        7L = [Fruit("Cherry", 10), Fruit("Apple", 5), Fruit("Blueberry", 20)]
        8 print "----sorted by price, referencing a class method----"
        9 for f in sorted(L, key=Fruit.sort_priority):
              print(f.name)
       10
       11 print "---- one more way to do the same thing----"
       12 for f in sorted(L, key=lambda x: x.sort_priority()):
              print(f.name)
       13
       14
                                     ActiveCode: 3 (sort instances 3)
:esasReturnValues.html)
(InstancesasReturnValues.html)
                                                                (ClassVariables Viesta Section ab 129.10 m Class \
                                          Mark as completed
```

2 of 3 11/12/19, 8:24 pm

© Copyright 2017 bradleymiller. Created using Runestone (http://runestoneinteractive.org/) 3.2.15.

 $\textbf{username: sanjaysheel1997} @ \textbf{gmail.com} \mid \textbf{Back to top} \\$

cesasReturnValues.html)
(InstancesasReturnValues.html)

(ClassVariablesNestaSectVioriab 26.10mClass)