Lecture Handout #4
January 28, 2016

Python Pattern: Sorting a Dictionary

Dictionaries (by default) in Python are *un-ordered*, so we need to use an `OrderedDict` to store a sorted (ordered) dictionary. *This pattern gets a little complex, but always works.*

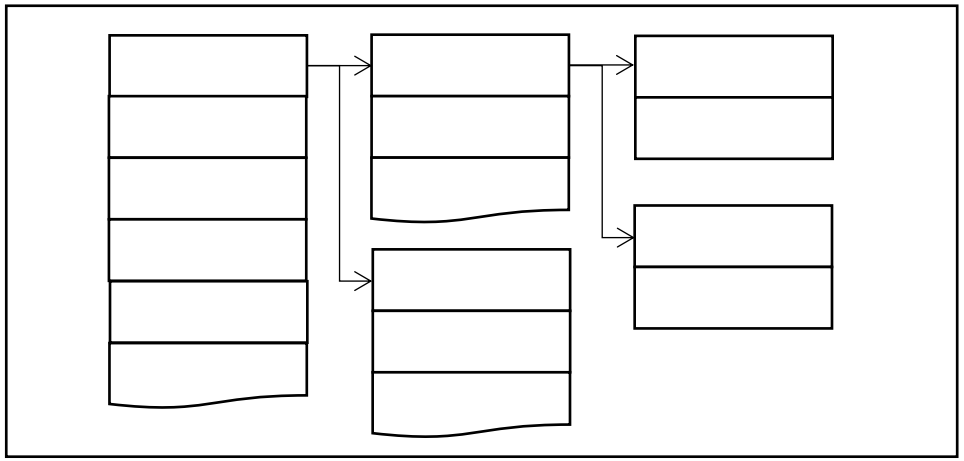
```
1: # Import the OrderedDict
2: from collections import OrderedDict
3: ...
4:
5: # Assume a myDictionary with data
6: # myDictionary = { ... }
7:
8: sortedDictionary = OrderedDict(
     sorted( myDictionary.items(),
             key = lambda: d: d[1]["sort field"]
           )
   )
```

By default, the keys are sorted in ascending order. You can reverse this by placing this argument into the `sorted` function (Line 8):

```
   reverse = True
```

Diversity within Majors

Inside of Computer Science, a lot has been written about the need and benefits of diversity. If we assume that the optimal population for every major is 50% women and 50% men, what majors are doing well?



Python Pattern: An Array of Dictionaries

Next week, we will start working with d3.js. The d3.js library is optimized to work with an *array of dictionaries*.

Right now, our data is a single dictionary. To convert a dictionary to an array of dictionaries (where each key becomes an entry in the array), the following Python pattern will help:

```
1: list = []
2: for key in myDictionary:
3:   list.append( {
4:     "key_name": key,
5:     "data": myDictionary[key]
6:   } )
```

An application of this pattern with our diversity data:

```

```

Python Pattern: Writing Python to a JSON File

Finally, for use in d3.js, we will write (or "serialize") our array of dictionaries to a JSON format using the last major Python pattern:

```
1: import json
2:
3: outdata = json.dumps(list, indent=2)
4: outfile = open("out.json", "w")
5: outfile.write(outdata)
6: outfile.close()
```