

dictionaries (aka hash tables or hash maps)

Genome 559: Introduction to Statistical
and Computational Genomics

Prof. James H. Thomas

Review

- You should be very comfortable with loops by now
- Start paying attention to program robustness and speed.
- Consider very large or very small input files.
- Consider files with the wrong format.
- Consider command-line options that are missing or in the wrong format.

Dictionaries

- A dictionary organizes linked information
- Examples:
 - word and definition
 - name and phone number
 - name and DNA sequence
 - username and password
- If you know the first entry, you can immediately get the second one

Rules for dictionaries

- The first item is a "key"
- Each key can only appear once
- A key must be an immutable object: number, string, or tuple
- Lists cannot be keys (they are mutable)
- The key should be the item you'll use to do look-ups

Key examples

Phone book: we have a name, we want a number

Name is the key

Crank call prevention: we have a number, we want a name

Number is the key

Creating a dictionary

```
#create an empty dictionary
```

```
myDict = {}
```

```
#create a dictionary with three entries
```

```
myDict = {"Curly":4123, "Larry":2057, "Moe":1122}
```

```
#add another entry
```

```
myDict["Shemp"] = 2232
```

```
#change Moe's phone number
```

```
myDict["Moe"] = 4040
```

```
#delete Moe from dictionary
```

```
del myDict["Moe"]
```

Using a dictionary

```
>>> myDict = {"Curly":4123, "Larry":2057, "Moe":1122}
>>> myDict["Moe"]
1122
>>> myDict.keys()
['Larry', 'Moe', 'Curly']
>>> "Curly" in myDict
True
>>> "curly" in myDict
False
>>> myDict.values()
[2057, 1122, 4123]
>>> len(myDict)
3
```

unlike a list, the key:value pairs are not in any particular order

curly is not the same as Curly

Using a dictionary

```
birthdays = { "George":"June 12", "W":"July 6", "Barack":"Aug 4" }  
for person in birthdays.keys():  
    print "Send", person, "a card on", birthdays[person]
```

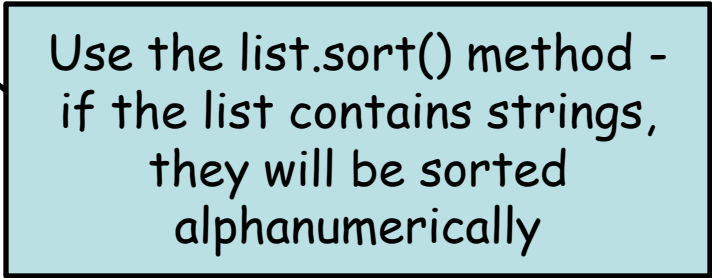
or possibly

```
for person in birthdays.keys():  
    if person == "Barack"  
        print "Send", person, "a card on", birthdays[person]  
    else  
        print "Send", person, "a bomb on", birthdays[person]
```

`dictionary.keys()` returns a list of the keys!

Sorting a dictionary

```
sortkeys = birthday.keys()
sortkeys.sort()
for person in sortkeys:
    print "Send", person, "a card on", birthdays[person]
```



Use the list.sort() method -
if the list contains strings,
they will be sorted
alphanumerically

Making a useful dictionary

Suppose we have a file that gives the alignment score for a large number of sequences:

```
seq1 <tab> 37  
seq2 <tab> 182  
etc.
```

```
import sys  
myFile = open(sys.argv[1], "r")  
scoreDict = {}  
for line in myFile:  
    fields = line.strip().split("\t")  
    scoreDict[fields[0]] = float(fields[1])  
myFile.close()
```

we now have a dictionary where we can look up a score for any name

Sample problem #1

The file "scores.txt" (linked from news on web site) contains blastn scores for a large number of sequences with a particular query. Write a program that reads them into a dictionary (this was given on the previous slide), sorts them by sequence name, and prints them.

```
>python sort_dict.py scores.txt  
seq00000      293  
seq00001      315  
seq00002      556  
seq00003      556  
seq00004      617  
seq00005      158  
etc.
```

Solution #1

```
import sys
myFile = open(sys.argv[1], "r")

# make an empty dictionary
scoreDict = {}
for line in myFile:
    fields = line.strip().split("\t")
    # record each value with name as key
    scoreDict[fields[0]] = float(fields[1])
myFile.close()

# get sorted key list
sort_keys = scoreDict.keys()
sort_keys.sort()

# print based on sorted keys
for key in sort_keys:
    print key + "\t" + scoreDict[key]
```

Sample problem #2

Suppose you have a list of sequence names whose scores you are interested in extracting from the large list of scores (in the same file scores.txt). Modify your previous program to read the list of sequence names from a file and print just those values. A sample seq_names.txt is also linked from news on web site.

```
>python get_scores.py scores.txt seq_names.txt
seq00036      784
seq57157      523
seq58039      517
seq67160      641
seq76732      44
seq83199      440
seq92309      446
```

Solution #2

```
import sys

# first get a list of the names of interest
seqNameFile = open(sys.argv[2], "r")
seqNameList = []
for line in seqNameFile:
    seqNameList.append(line.strip())
seqNameFile.close()

# now make a dictionary of the scores, keyed on name
dictFile = open(sys.argv[1], "r")
scoreDict = {}
for line in dictFile:
    fields = line.strip().split("\t")
    scoreDict[fields[0]] = int(fields[1])
dictFile.close()

# finally, use the dictionary
for seqName in seqNameList:
    print seqName + "\t" + scoreDict[seqName]
```

Challenge problem

Sort the list of scores in the same file (scores.txt) by score, with the highest scoring first. Print the sequence name and its score in that order. You can easily do this using a dictionary (don't worry about the fact that more than one sequence will have the same score, so some will get lost).

```
import sys
dictFile = open(sys.argv[1], "r")

scoreDict = {}
for line in dictFile:
    fields = line.strip().split("\t")
    scoreDict[int(fields[1])] = fields[0]
dictFile.close()

sortKeys = scoreDict.keys()
sortKeys.sort()
sortKeys.reverse() # sort makes ascending sort for numbers

for key in sortKeys:
    print scoreDict[key] + "\t" + key
```