

Dictionaries

GENOME 559: Introduction to Statistical and Computational Genomics

Prof. William Stafford Noble

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 quickly and easily get the second one.
- Accessing a dict entry is fast, but not quite as fast as indexing in a list or string.

Rules for dictionaries

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

Key examples

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

Name is the key, number is the value

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

Number is the key, name is the value

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
```

get all the keys as a list

the keys are not in any particular order!

curly is not the same as Curly

get all the values as a list

the number of key:value pairs

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  
openFile = open(sys.argv[1], "r")  
scoreDict = {}  
for line in openFile:  
    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

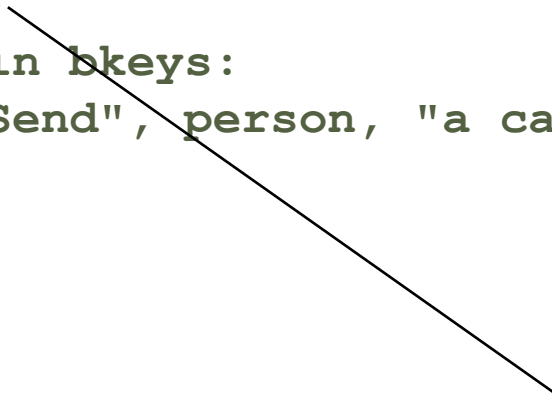
Traversing a dictionary by key

```
# birthdays is a dictionary with names as keys  
# and birth dates as values  
  
for person in birthdays.keys():  
    print "Send", person, "a card on", birthdays[person]
```


Sorting a dictionary by key

```
# birthdays is a dictionary with names as keys
# and birth dates as values

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



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

dictionary basics

```
D = {'dna': 'T', 'rna': 'U'} # dictionary literal assignment
D = {}                       # make an empty dictionary
D.keys()                     # get the keys as a list
D.values()                   # get the values as a list
D['dna']                      # get a value based on key
D['dna'] = 'T'               # set a key:value pair
del D['dna']                  # delete a key:value pair
'dna' in D                    # True if key 'dna' is found in D, else False
```

The keys must be immutable objects (e.g. string, int, tuple).

The values can be anything (including a list or another dictionary).

The order of elements in the list returned by `D.keys()` or `D.values()` is arbitrary (effectively random).

Each key can be stored only once in the dictionary, so if you set the value for a key for a second time it **OVERWRITES** the old value!

Sample problem #1

File format:

```
seq1 <tab> score  
seq2 <tab> score  
etc.
```

- The file 'small-scores.txt' contains scores assigned to many different sequences.
- Write a program find-best-score.py that prints each sequence next to the square of its associated value.

```
$ python print-squares.py  
small-scores.txt
```

```
AGGSIIR    0.000376361
```

```
AGGSIIR    0.0662605
```

```
IALKPK     1.64762
```

```
IALKPK     0.000906147
```

```
AGGSIIR    0.00154437
```

```
NPIDVK     0.34584
```

```
IALKPK     0.224348
```

```
NPIDVK     0.0187335
```

```
IALKPK     0.00124857
```

Solution #1

```
import sys

# Open the file for reading.
in_file_name = sys.argv[1]
in_file = open(in_file_name, "r")

# Read each line and print the square.
for line in in_file:
    words = line.rstrip().split("\t")
    score = float(words[0])
    sequence = words[1]
    print "%s\t%g" % (sequence, score * score)
in_file.close()
```

Sample problem #2

- Write a program `find-best-score.py` that prints the maximal score assigned to each sequence. You can also run it on `small-scores.txt` and `large-scores.txt`.

```
$ python find-best-score.py small-scores.txt
```

```
IALKPK    1.2836
```

```
AGGSIIR   0.257411
```

```
NPIDVK    0.588082
```

```
import sys

# Open the file for reading.
in_file_name = sys.argv[1]
in_file = open(in_file_name, "r")

# Set up a dictionary.
best_score = {} # Key = sequence, value = score

for line in in_file:
    words = line.rstrip().split("\t")
    score = float(words[0])
    sequence = words[1]

    if ( (not sequence in best_score) or
        (score > best_score[sequence]) ):
        best_score[sequence] = score
in_file.close()

for sequence in best_score.keys():
    print "%s\t%g" % (sequence, best_score[sequence])
```

Solution #2

Sample problem #3

The file "unique-scores.txt" contains blastn scores for a large number of sequences with a particular query. Write a program that reads them into a dictionary, 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 #3

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

# make an empty dictionary and populate it
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 key list and sort it
keys = scoreDict.keys()
keys.sort()

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


Sample problem #4

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 "unique-scores.txt"). Modify your previous program to read a list of sequence names from a second file and print the scores for just those sequences. A sample "seq-names.txt" is linked from the 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 #4

```
import sys
```

```
# 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()
```

```
# 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")
```

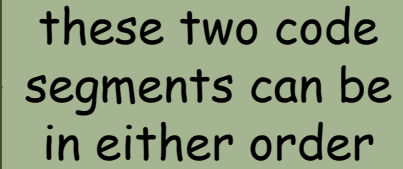
```
    scoreDict[fields[0]] = int(fields[1])
```

```
dictFile.close()
```

```
# finally, use the dictionary
```

```
for seqName in seqNameList:
```

```
    print seqName + "\t" + str(scoreDict[seqName])
```



these two code
segments can be
in either order

Challenge problems

1. Extend your program in sample problem 2 so that it gives useful user feedback when a sequence name is missing from the dictionary.
2. 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 do this using a dictionary (ignore the fact that more than one sequence may have the same score, so some may get lost).

Challenge 1 solution

```
import sys
# 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()
# 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:
    if not seqName in scoreDict:
        print seqName, "not found"
    else:
        print seqName + "\t" + scoreDict[seqName]
```

Challenge 2 solution

```
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
```