Lect 7 – Dictionaries and Text Analysis 2

Rob Capra

INLS 490-172

List of words

- Word list from the Moby Lexicon Project
- http://en.wikipedia.org/wiki/Moby_Project
- http://www.greenteapress.com/thinkpython/code/words.txt

Car Talk Puzzler

Give me a word with three, consecutive double letters. I'll give you a couple of words that almost qualify, but don't. For example, the word committee, c-o-m-m-i-t-t-e-e. It would be great except for the i that sneaks in there. Or Mississippi – M-i-s-s-i-s-s-i-p-p-i. If you could take out those i's it would work. But there is a word that has three consecutive pairs of letters and to the best of my knowledge this may be the only word. Of course there are probably 500 more but I can only think of one. What is the word?

1. **Discuss algorithms in pairs**
2. **Think of at least two algorithms**

http://www.cartalk.com/content/seeing-double
(TPY Exercise 9.7)

Dictionaries

- So far we have seen *sequential* collections
  - Strings, lists, tuples
  - Have an order from left to right
  - Use integer indices to access values
- Dictionaries are a *mapping* type
  - Unordered, associative collection
  - Mapping from *keys* to *values*
    - Keys can be any immutable type
    - Values can be any Python data object (including other collections)
  - Dictionaries are mutable

Dictionary Example

```
e2s = {}
e2s['one'] = 'uno'
e2s['two'] = 'dos'
e2s['three'] = 'tres'
print e2s
print e2s['two']
```

Ordering is undefined

Output:

```
{'three': 'tres', 'two': 'dos', 'one': 'uno'}
dos
```

Dictionary Operations

```
inv = {'apples': 430, 'bananas':312,
       'oranges': 523, 'pears':217}
print inv
inv['pears'] = 0
inv['bananas'] += 200
del inv['oranges']
print inv
print len(inv)
```

Dictionaries are mutable

Output:

```
{'pears': 217, 'apples': 430, 'oranges': 523,
'bananas': 312}
{'pears': 0, 'apples': 430, 'bananas': 512}
3
```

Dictionary Methods

Method	Parameters	Description
keys	none	Returns a **view** of the **keys** in the dict
values	none	Returns a **view** of the **values** in the dict
items	none	Returns a **view** of the **key-value pairs** in the dict
get	key	Returns the **value** associated with the key; if the key does not exist, returns **None**
get	key,alt	Returns the **value** associated with the key; if the key does not exist, returns **alt**

Dictionary Operations

```
inv = {'apples': 430, 'bananas':312,
       'oranges': 523, 'pears':217}
for akey in inv.keys():
    print "The key", akey, "maps to value", inv[akey]
tmp = list(inv.keys())
print tmp
for akey in inv:
    print akey, inv[akey]
```

Dictionary Operations

```
inv = {'apples': 430, 'bananas':312,
       'oranges': 523, 'pears':217}
print(list(inv.values()))
# items() returns k-v pairs as tuples
print(list(inv.items()))
for (k,v) in inv.items():
    print k,v
for k in inv:
    print k, inv[k]
```

in and not in work on keys

```
inv = {'apples': 430, 'bananas':312,
       'oranges': 523, 'pears':217}

if 'bananas' in inv:
    print "We have ", inv['bananas'], 'bananas'
else:
    print "Yes sir! We have no bananas."
```

A VERY BIG ISSUE WITH DICTIONARIES

```
inv = {'apples': 430, 'bananas':312,
       'oranges': 523, 'pears':217}

print inv['apples']
#print inv['kiwi']      # error!
print inv.get('apples')
print inv.get('kiwi')
print inv.get('kiwi',0)
```

Text of Emma by Jane Austen

- Project Gutenberg
- Also available at:
- http://www.greenteapress.com/thinkpython/code/emma.txt

Histogram of words

```
import string

def process_file(filename):
    hist = dict()
    fp = open(filename)
    for line in fp:
        process_line(line,hist)
    return hist

def process_line(line, hist):
    line = line.replace('-',' ')
    for word in line.split():
        word = word.strip(string.punctuation +
string.whitespace)
        word = word.lower()
        hist[word] = hist.get(word,0) + 1
```

Using the histogram

```
def total_words(hist):
    return sum(hist.values())

def different_words(hist):
    return len(hist)

hist = process_file('emma.txt')

t = most_common(hist)
print "Total words = ", total_words(hist)
print "Different words = ", 
different_words(hist)
```

Histogram of words

```
def most_common(hist):
    t = []
    for key, value in hist.items():
        t.append((value, key))
    t.sort(reverse=True)
    return t

print "Most common:"
for freq, word in t[0:10]:
    print word, "\t", freq
```

Histogram of words

```
def subtract(d1, d2):
    result = dict()
    for key in d1:
        if key not in d2:
            result[key] = None
    return result
words = process_file('words.txt')
diff = subtract(hist, words)
print "In emma, but not in words.txt:"
for word in diff.keys():
    print word,
```

courses1.txt

```
760 Capra
509 Arguello
512 Haas
523 Capra
884 Kelly
```

courses2.txt

```
760 Capra
509 Arguello
512 Haas
523 Capra
884 Kelly
509 Kelly
523 Haas
523 Mostafa
509 Losee
```

Mini-Exercise #1 (not to turn in)

- Work in pairs (of 2!)
- Read the file courses1.txt
- Build a dictionary with the course number as the key and the instructor as the value.
  - Assume for now that each course is only taught by one instructor.
- Using the dictionary:
  - Print out the numbers of all courses in the dict
  - Print out all the instructors names
  - Print out the courses taught by instructor Capra

Mini-Exercise #2 (not to turn in)

- Read the file courses2.txt
- Build a dictionary of lists.
  - The dict keys should be the course numbers.
  - The dict values should be a list of all the instructors who have taught that course.
  - Hint: look back at the list methods such as append()
- Using the dictionary:
  - Print out the numbers of all courses in the dict
  - Print out all the instructors names
  - Print out all the instructors who have taught 523.
  - Print out the courses taught by instructor Capra

Dict of Dict of List

```
idx = {'a': { 'X': [1, 2], 'Y': [3, 4]},
       'b': { 'X': [5, 6], 'Z': [7, 8]}}

print idx
print "-------"
print idx['a']
print "-------"
print idx['a']['X']
print "-------"
if 'c' in idx:
    print idx['c']
print "-------"
for j in idx:
    print "  ", j
    for k in idx[j]:
        print "    ", k
        for m in idx[j][k]:
            print "      ", m
```

Dictionary Methods

Method	**Parameters**	**Description**
keys	none	Returns a **view** of the **keys** in the dict
values	none	Returns a **view** of the **values** in the dict
items	none	Returns a **view** of the **key-value pairs** in the dict
get	key	Returns the **value** associated with the key; if the key does not exist, returns **None**
get	key,alt	Returns the **value** associated with the key; if the key does not exist, returns **alt**
setdefault	key, default	Returns value if the key is in the dict; if not, inserts key with the value of default