Dictionaries

Computer Science 111
Boston University

Vahid Azadeh-Ranjbar, Ph.D.

based in part on notes from the CS-for-All curriculum developed at Harvey Mudd College

Recall: Extracting Relevant Data from a File

- Assume that the results of a track meet are summarized in a comma-delimited text file (a *CSV file*) that looks like this:

```
Mike Mercury,BU,mile,4:50:00
Steve Slug,BC,mile,7:30:00
Len Lightning,BU,half-mile,2:15:00
Tom Turtle,UMass,half-mile,4:00:00
```

- We'd like to have a function that reads in such a results file and extracts just the results for a particular school.

Recall: Extracting Relevant Data from a File

```
def extract_results(filename, target_school):
    file = open(filename, 'r')

    for line in file:
        line = line[:-1]      # chop off newline at end

        fields = line.split(',')
        athlete = fields[0]
        school = fields[1]
        event = fields[2]
        result = fields[3]

        if school == target_school:
            print(athlete, event, result)

    file.close()
```

```
Mike Mercury,BU,mile,4:50:00
Steve Slug,BC,mile,7:30:00
Len Lightning,BU,half-mile,2:15:00
Tom Turtle,UMass,half-mile,4:00:00
```

Another Data-Processing Task

```
Mike Mercury,BU,mile,4:50:00
Steve Slug,BC,mile,7:30:00
Len Lightning,BU,half-mile,2:15:00
Tom Turtle,UMass,half-mile,4:00:00
```

- Now we'd like to count the number of results from each school, and report all of the counts:

```
>>> school_counts('results.txt')
There are 3 schools in all.
BU has 2 result(s).
BC has 1 result(s).
UMass has 1 result(s).
```

- Python makes this easy if we use a *dictionary*.

What is a Dictionary?

- A dictionary is a set of key-value pairs.

```
>>> counts = {'BU': 2, 'UMass': 1, 'BC': 1}
```

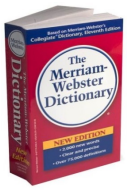
general syntax:

```
{key1: value1, key2: value2, key3: value3...}
```

- We can use the key like an index to lookup the associated value!

```
>>> counts['BU']
2
>>> counts['BC']
1
```

- It is similar to a "physical" dictionary:
  - keys = words
  - values = definitions
  - use the word to lookup its definition



Using a Dictionary

```
>>> counts = {}              # create an empty dictionary
>>> counts['BU'] = 2
                  key   value
>>> counts['BC'] = 1
>>> counts                   # a set of key: value pairs
{'BU': 2, 'BC': 1}
>>> counts['BU']             # use the key to get the value
2
>>> counts['BC']
1
>>> counts['UMass'] = 1
>>> counts
{'BU': 2, 'UMass': 1, 'BC': 1}    # order is not fixed
```

Other Dictionary Operations

```
>>> counts = {'BU': 2, 'UMass': 1, 'BC': 1}

>>> len(counts)
3

>>> 'BU' in counts       # is 'BU' one of the keys?
True

>>> 'Harvard' in counts
False

>>> 'Harvard' not in counts
True

>>> 2 in counts
False                    # 2 is not a key!
```

Processing All of the Items in a Dictionary

```
counts = {'BU': 2, 'UMass': 1, 'BC': 1}

for key in counts:      # get one key at a time
    print(key, counts[key])

# the above outputs:
BU 2
UMass 1
BC 1
```

- More generally:

```
for key in dictionary:
    # code to process key-value pair goes here
```

  - gets one key at a time and assigns it to `key`
  - continues looping until there are no keys left

Processing All of the Items in a Dictionary

```
counts = {'BU': 2, 'UMass': 1, 'BC': 1}

for key in counts:      # get one key at a time
    print(key, counts[key])
```

key	counts[key]	output

Processing All of the Items in a Dictionary

```
counts = {'BU': 2, 'UMass': 1, 'BC': 1}

for key in counts:      # get one key at a time
    print(key, counts[key])
```

key	counts[key]	output
'BU'	counts['BU'] → 2	BU 2
'UMass'	counts['Umass'] → 1	UMass 1
'BC'	counts['BC'] → 1	BC 1
none left		

What Is the Output?

```
d = {4: 10, 11: 2, 12: 3}

count = 0
for x in d:
    if x > 5:
        count += 1

print(count)
```

A. 0

B. 1

C. 2

D. 3

E. none of these

What Is the Output?

```
d = {4: 10, 11: 2, 12: 3}

count = 0
for x in d:          # x gets one key at a time!
    if x > 5:
        count += 1

print(count)
```

A. 0

B. 1

C. 2

D. 3

E. none of these

Using a Dictionary to Compute Counts

```
Mike Mercury,BU,mile,4:50:00
Steve Slug,BC,mile,7:30:00
Len Lightning,BU,half-mile,2:15:00
Tom Turtle,UMass,half-mile,4:00:00
```

```
def school_counts(filename):
    file = open(filename, 'r')

    counts = {}

    for line in file:
        fields = line.split(',')

        school = fields[1]
        if school not in counts:
            counts[school] = 1    # new key-value pair
        else:
            counts[school] += 1   # existing k-v pair

    file.close()

    print('There are', len(counts), 'schools in all.')
    for school in counts:
        print(school, 'has', counts[school], 'result(s).')
```

Using a Dictionary to Compute Counts

```
Mike Mercury,BU,mile,4:50:00
Steve Slug,BC,mile,7:30:00
Len Lightning,BU,half-mile,2:15:00
Tom Turtle,UMass,half-mile,4:00:00
```

```
def school_counts(filename):
    file = open(filename, 'r')

    counts = {}

    for line in file:
        fields = line.split(',')

        school = fields[1]
        if school not in counts:
            counts[school] = 1
        else:
            counts[school] += 1

    file.close()

    print('There are', len(counts), 'schools in all.')
    for school in counts:
        print(school, 'has', counts[school], 'result(s).')
```

Using a Dictionary to Compute Counts

```
Mike Mercury,BU,mile,4:50:00
Steve Slug,BC,mile,7:30:00
Len Lightning,BU,half-mile,2:15:00
Tom Turtle,UMass,half-mile,4:00:00
```

```
def school_counts(filename):
    file = open(filename, 'r')

    counts = {}

    for line in file:
        fields = line.split(',')

        school = fields[1]
        if school not in counts:
            counts[school] = 1
        else:
            counts[school] += 1

    file.close()

    print('There are', len(counts), 'schools in all.')
    for school in counts:
        print(school, 'has', counts[school], 'result(s).')
```

Another Example of Counting

```
def word_frequencies(filename):
    file = open(filename, 'r')
    text = file.read()      # read it all in at once!
    file.close()

    words = text.split()

    d = {}

    for word in words:
        if word not in d:
            d[word] = 1
        else:
            d[word] += 1

    return d     # so we can use it later!
```

Shakespeare, Anyone?

```
>>> freqs = word_frequencies('romeo.txt')

>>> freqs['Romeo']
1

>>> freqs['ROMEO:']     # case and punctuation matter
47

>>> freqs['love']
12

>>> len(freqs)
2469
```

Act I of *Romeo & Juliet.*
See PS 8, Problem 4!

- In his plays, Shakespeare used ***31,534 distinct words!***

http://www-math.cudenver.edu/~wbriggs/qr/shakespeare.html

- He also coined a number of words:

gust	besmirch	unreal
swagger	watchdog	superscript

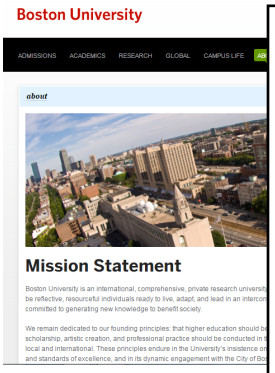
http://www.pathguy.com/shakeswo.htm
http://www.shakespeare-online.com/biography/wordsinvented.html

ps8pr4: *Generate* Text Based on Shakespeare!

```
>>> d = create_dictionary('romeo.txt')

>>> generate_text(d, 50)
ROMEO: Out of mine own word: If you merry! BENVOLIO:
Come, go to. She hath here comes one of the year, Come
hither, nurse. ROMEO: Well, in spite, To be gone.
BENVOLIO: For men depart.[Exeunt all Christian souls!-
Were of wine. ROMEO: Bid a sea nourish'd with their
breaths with
```

ps8pr4: *Generate* Text Based on Shakespeare ...Or Anyone Else!



Boston University is an international, comprehensive, private research university, committed to educating students to be reflective, resourceful individuals ready to live, adapt, and lead in an interconnected world. Boston University is committed to generating new knowledge to benefit society.

We remain dedicated to our founding principles: that higher education should be accessible to all and that research, scholarship, artistic creation, and professional practice should be conducted in the service of the wider community–local and international. These principles endure in the University's insistence on the value of diversity, in its tradition and standards of excellence, and in its dynamic engagement with the City of Boston and the world.

Boston University comprises a remarkable range of undergraduate, graduate, and professional programs built on a strong foundation of the liberal arts and sciences. With the support and oversight of the Board of Trustees, the University, through our faculty, continually innovates in education and research to ensure that we meet the needs of students and an ever-changing world.

mission.txt

ps8pr4: *Generate* Text Based on Shakespeare ...Or Anyone Else!

```
Boston University is an international, comprehensive, private research
university, committed to educating students to be reflective, resourceful
individuals ready to live, adapt, and lead in an interconnected world. Boston
University is committed to generating new knowledge to benefit society.

We remain dedicated to our founding principles: that higher education should be
accessible to all and that research, scholarship, artistic creation, and
professional practice should be conducted in the service of the wider community–
local and international. These principles endure in the University's insistence
on the value of diversity, in its tradition and standards of excellence, and in
its dynamic engagement with the City of Boston and the world.

Boston University comprises a remarkable range of undergraduate, graduate, and
professional programs built on a strong foundation of the liberal arts and
sciences. With the support and oversight of the Board of Trustees, the
University, through our faculty, continually innovates in education and research
to ensure that we meet the needs of students and an ever-changing world.
```

mission.txt

```
>>> d2 = create_dictionary('mission.txt')

>>> generate_text(d2, 20)
We remain dedicated to benefit society. Boston
University is an ever-changing world. Boston University
comprises a strong foundation of diversity,
```

Markov Models

- Allow us to model *any* sequence of real-world data.
  - human speech
  - written text
  - sensor data
  - etc.
- Can use the model to *generate* new sequences that are based on existing ones.
- We'll use a *first-order* Markov model.
  - each term in the sequence depends only on the *one* term that immediately precedes it

A Markov Model in Dictionary Form

```
Boston University is a comprehensive university.
It is committed to educating students to be ready
to live and to lead in an interconnected world.
It is committed to generating new knowledge.
It is amazing!
```

edited_mission.txt

sentence-start symbol

key = a word w

value = a list of the words that follow w in the text

```
{'$': ['Boston', 'It', 'It', 'It'],
 'Boston': ['University'],
 'University': ['is'],
 'is': ['a', 'committed', 'committed', 'amazing!'],
 'to': ???,
 'committed': ???,
 ... }
```

A Markov Model in Dictionary Form

```
Boston University is a comprehensive university.
It is committed to educating students to be ready
to live and to lead in an interconnected world.
It is committed to generating new knowledge.
It is amazing!
```

edited_mission.txt

sentence-start symbol

key = a word w

value = a list of the words that follow w in the text

```
{'$': ['Boston', 'It', 'It', 'It'],
 'Boston': ['University'],
 'University': ['is'],
 'is': ['a', 'committed', 'committed', 'amazing!'],
 'to': ['educating', 'be', 'live', 'lead', 'generating'],
 'committed': ???,
 ... }
```

A Markov Model in Dictionary Form

```
Boston University is a comprehensive university.
It is committed to educating students to be ready
to live and to lead in an interconnected world.
It is committed to generating new knowledge.
It is amazing!
```

edited_mission.txt

sentence-start symbol

key = a word w

value = a list of the words that follow w in the text

```
{'$': ['Boston', 'It', 'It', 'It'],
 'Boston': ['University'],
 'University': ['is'],
 'is': ['a', 'committed', 'committed', 'amazing!'],
 'to': ['educating', 'be', 'live', 'lead', 'generating'],
 'committed': ['to', 'to'],
 ... }
```

A Markov Model in Dictionary Form

```
Boston University is a comprehensive university.
It is committed to educating students to be ready
to live and to lead in an interconnected world.
It is committed to generating new knowledge.
It is amazing!
```

edited_mission.txt

sentence-start symbol

key = a word w

value = a list of the words that follow w in the text

```
{'$': ['Boston', 'It', 'It', 'It'],
 'Boston': ['University'],
 'University': ['is'],
 'is': ['a', 'committed', 'committed', 'amazing!'],
 'to': ['educating', 'be', 'live', 'lead', 'generating'],
 'committed': ['to', 'to'],
 ... }
```

- *Sentence-ending words* should not be used as keys.
  - words that end with a `'.'`, `'?'`, or `'!'` (e.g., `'world.'`)

Model Creation Function

```
def create_dictionary(filename):
    # read in file and split it into a list of words

    d = {}
    current_word = '$'

    for next_word in words:
        if current_word not in d:
            d[current_word] = [next_word]
        else:
            d[current_word] += [next_word]

        # update current_word...

    return d
```

key = a word w

value = a list of the words that follow w in the text

Model Creation Example

```
words = ['Boston', 'University', 'is', 'a', 'comprehensive',
   'university.', 'It', 'is', 'committed', ...]
d = {}
current_word = '$'

for next_word in words:
    if current_word not in d:
        d[current_word] = [next_word]
    else:
        d[current_word] += [next_word]

    # update current_word to be either next_word or '$'...
```

current_word	next_word	action taken

Model Creation Example

```
words = ['Boston', 'University', 'is', 'a', 'comprehensive',
   'university.', 'It', 'is', 'committed', ...]
d = {}
current_word = '$'

for next_word in words:
    if current_word not in d:
        d[current_word] = [next_word]
    else:
        d[current_word] += [next_word]

    # update current_word to be either next_word or '$'...
```

current_word	next_word	action taken
'$'	'Boston'	d['$'] = ['Boston']
'Boston'	'University'	d['Boston'] = ['University']
'University'	'is'	d['University'] = ['is']
'is'	'a'	d['is'] = ['a']
'a'	'comprehensive'	d['a'] = ['comprehensive']
'comprehensive'	'university.'	d['comprehensive']=['university.']
'$'		

Model Creation Example

```
words = ['Boston', 'University', 'is', 'a', 'comprehensive',
   'university.', 'It', 'is', 'committed', ...]
d = {}
current_word = '$'

for next_word in words:
    if current_word not in d:
        d[current_word] = [next_word]
    else:
        d[current_word] += [next_word]

    # update current_word to be either next_word or '$'...
```

current_word	next_word	action taken
'$'	'Boston'	d['$'] = ['Boston']
'Boston'	'University'	d['Boston'] = ['University']
'University'	'is'	d['University'] = ['is']
'is'	'a'	d['is'] = ['a']
'a'	'comprehensive'	d['a'] = ['comprehensive']
'comprehensive'	'university.'	d['comprehensive']=['university.']
'$'	'It'	d['$'] → ['Boston', 'It']
'It'	'is'	d['It'] = ['is']
'is'	'committed'	d['is'] → ['a', 'committed']

generate_text(word_dict, num_words)

start with `current_word = '$'`

repeat `num_words` times:

- `next_word` = random choice from the words that can follow `current_word` (use the model!)
- print `next_word`, followed by a space (use `end=' '`)
- update `current_word` to be either `next_word` or `'$'`

`print()`