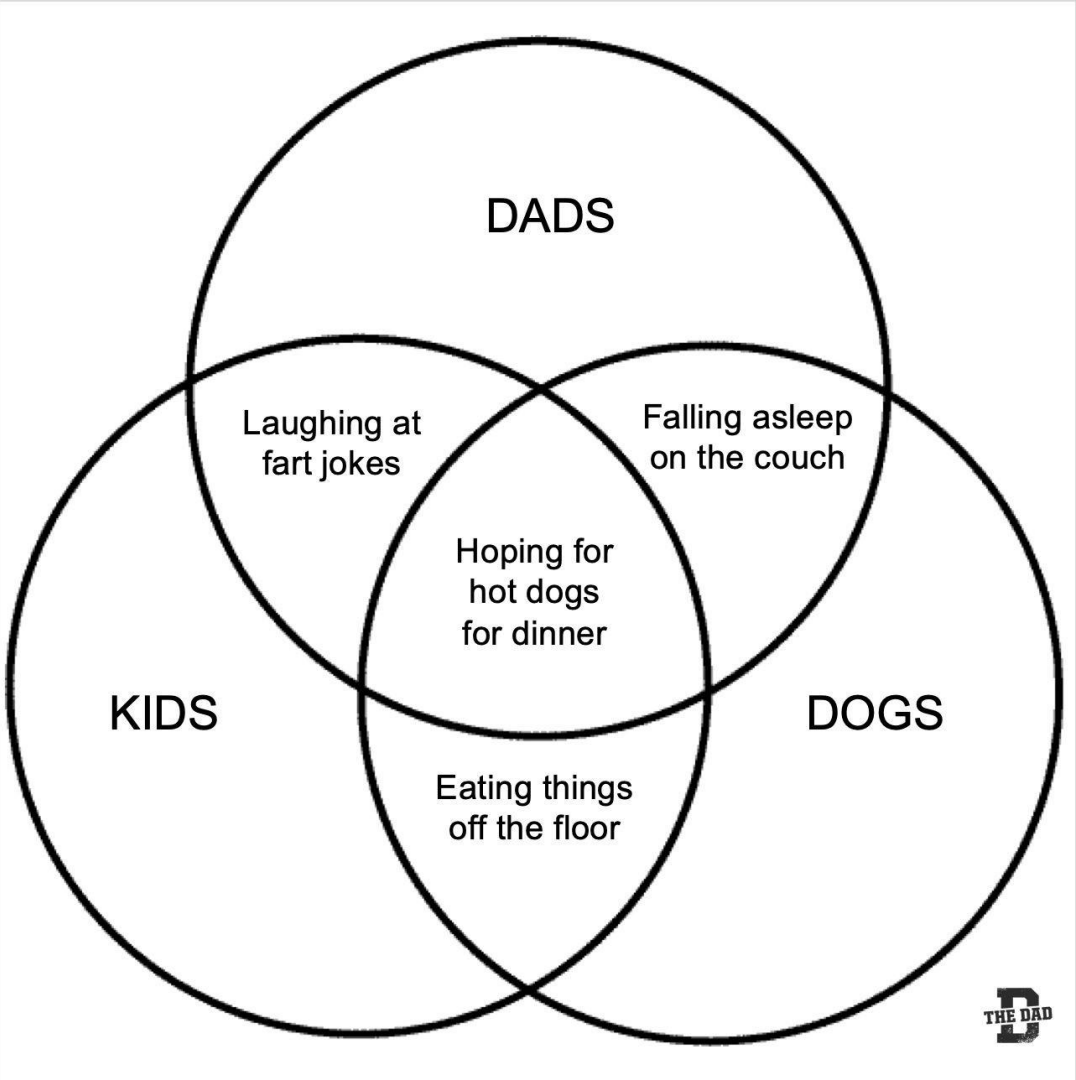
CSc 110 Sets

Benjamin Dicken



Set

What will print when this code is executed?

```
numbers = {5, 7, 10, 5, 3, 5, 9, 8, 20, 5}
print(numbers)
```

Set

What will print when this code is executed?

```
numbers = {5, 7, 10, 5, 3, 5, 9, 8, 20, 5}
print(numbers)
```

```
{3, 5, 7, 8, 9, 10, 20}
```

Notice that the duplicate numbers are automatically removed!

Set

- A **set** is (another) data structure
- Helpful ways of thinking about it
  - A dictionary without the values
  - A “bag" of elements

Similarities

```
# Dictionary creation
ds = {'a':8, 'b':7, 'c':4}

# Dictionary loop
for key in ds:
    print(key)
```

```
# Set creation
ds = {'a', 'b', 'c'}

# Set loop
for element in ds:
    print(element)
```

Differences

```
ds = {'a':8, 'b':7, 'c':4}

# Remove from dictionary
del ds['c']

# Add to dictionary
ds['e'] = 20

# Create empty
ds_2 = {}
```

```
ds = {'a', 'b', 'c'}

# Set removal
ds.remove('c')

# Adding to set
ds.add('e')

# ???
ds_2 = {}
```

What will print?

```
numbers = {1, 2, 3, 4, 'word'}
numbers.add(5)
numbers.remove(5)
numbers.add(1)
numbers.remove(7)
numbers.add(5)
print(numbers)
```

What will print?

```
numbers = {1, 2, 3, 4, 'word'}
numbers.add(5)
numbers.remove(5)
numbers.discard(5)
numbers.add(1)
numbers.discard('words')
numbers.add(2)
print(numbers)
```

Looping through a set

- What would print?

```
names = {"Jones", "James", "Zac"}
for i in range(0, len(names)):
    print(names[i])
```

Looping through a set

- Why What would print?

```
names = {"Jones", "James", "Zac"}
for i in range(0, len(names)):
    print(names[i])
```

- Elements cannot be “looked up” by index (position) in the data structure
- You would end up with an error:

```
TypeError: 'set' object does not support indexing
```

Looping through a set

- Use this instead:

```
names = {"Ben", "James", "Zac"}
for name in names:
    print(name)
```

- Iterates through the *elements* of the set, not indexes

Differences from a Dictionary

```
ds = {'a':8, 'b':7, 'c':4}

# Get value from dictionary
value = ds['c']

# Change value in dictionary
ds['c'] = 23
```

```
ds = {'a', 'b', 'c'}

# ?

# ?
```

What would be in grades?

```
grades = set()
letters = ['C', 'B', 'E', 'C', 'A', 'B', 'B', 'A']
for l in letters:
    if l in grades:
        grades.remove(l)
    else:
        grades.add(l)
print(grades)
```

What will happen?

```
grades = {'A+', 'A', 'B', 'E', 'D', 'E', 'E-'}
grade_counts = {'A':5, 'B':10, 'C':7, 'D':4, 'E':2}
for element in grades:
    if element not in grade_counts:
        grades.discard(element)
    else:
        del grade_counts[element]
print(grades)
```

What will happen?

```
grades = {'A+', 'A', 'B', 'E', 'D', 'E', 'E-'}
grade_counts = {'A':5, 'B':10, 'C':7, 'D':4, 'E':2}
for element in grades:
    if element in grade_counts:
        del grade_counts[element]
print(grade_counts)
```

Activity

Exercise: Counting names

- Implement a program that . . .
  - Reads in a text file formatted like the example to the right named **names.txt**
  - Notice that some names repeat
  - The program should count how many unique names there are!
  - **Don't use a list or dictionary**

```
Lebron James
James Harden
Chris Paul
Chris Tucker
Kevin Durant
James Harden
Steve Tucker
Steve Smith
Eric Bledsoe
Steve Caroll
Chris Paul
Sally Smith
Kevin Durant
James Jones
Chris Paul
```

Exercise: Counting names

```
names = set()
names_file = open('names.txt', 'r')
for line in names_file:
    name = line.strip('\n')
    names.add(name)
print(len(names))
```

```
Lebron James
James Harden
Chris Paul
Chris Tucker
Kevin Durant
James Harden
Steve Tucker
Steve Smith
Eric Bledsoe
Steve Caroll
Chris Paul
Sally Smith
Kevin Durant
James Jones
Chris Paul
```

Activity

Exercise: Counting names

- Implement a program that . . .
  - Reads in a text file formatted like the example to the right named **names.txt**
  - Notice that some names repeat
  - The program should count how many unique names there are!
  - **Don't use a set or dictionary**

```
Lebron James
James Harden
Chris Paul
Chris Tucker
Kevin Durant
James Harden
Steve Tucker
Steve Smith
Eric Bledsoe
Steve Caroll
Chris Paul
Sally Smith
Kevin Durant
James Jones
Chris Paul
```

Exercise: Counting names

```
names = []
names_file = open('names.txt', 'r')
for line in names_file:
    name = line.strip('\n')
    if name not in names:
        names.append(name)
print(len(names))
```

```
Lebron James
James Harden
Chris Paul
Chris Tucker
Kevin Durant
James Harden
Steve Tucker
Steve Smith
Eric Bledsoe
Steve Caroll
Chris Paul
Sally Smith
Kevin Durant
James Jones
Chris Paul
```

Activity

Exercise: Counting names

- Implement a program that . . .
  - Reads in a text file formatted like the example to the right named **names.txt**
  - Notice that some names repeat
  - The program should count how many unique names there are!
  - **Don't use a set or list**

```
Lebron James
James Harden
Chris Paul
Chris Tucker
Kevin Durant
James Harden
Steve Tucker
Steve Smith
Eric Bledsoe
Steve Caroll
Chris Paul
Sally Smith
Kevin Durant
James Jones
Chris Paul
```

Exercise: Counting names

```
names = {}
names_file = open('names.txt', 'r')
for line in names_file:
    name = line.strip('\n')
    names[name] = ''
print(len(names))
```

```
Lebron James
James Harden
Chris Paul
Chris Tucker
Kevin Durant
James Harden
Steve Tucker
Steve Smith
Eric Bledsoe
Steve Caroll
Chris Paul
Sally Smith
Kevin Durant
James Jones
Chris Paul
```

Announcements

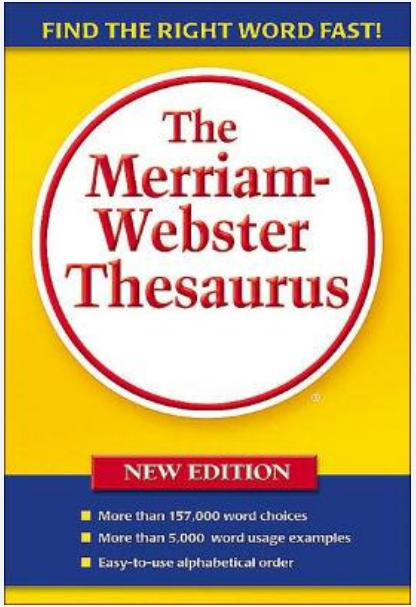
- Exam 3
  - Individual Exam November 16th
  - Group Exam November 18th
  - Review Session November 15th 5-7pm
  - Study Guide
- Other
  - Infographic PA
  - Veterans Day

Survey

bit.ly/csc110ta

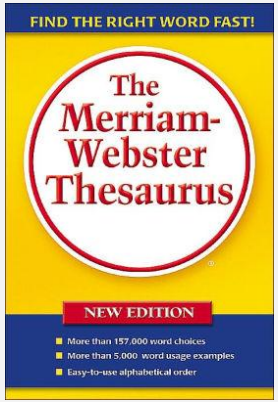
Representing a thesaurus

```
thesaurus = {'fast' : 'speedy',
             'old' : 'aged',
             'slow' : 'sluggish',
             'difficult' : 'challenging'}
```



https://www.thesaurus.com

Representing a thesaurus

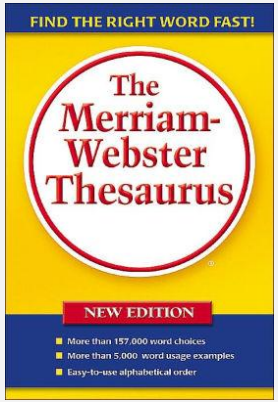



```
thesaurus = {'fast' : {'quick', 'agile', 'speedy'},
             'old' : {'aged', 'antique'},
             'slow' : {'sluggish'},
             'difficult' : {'hard', 'challenging', 'arduous'}}
```

Add 'strong' with two similar words

```
thesaurus = {'fast' : {'quick', 'agile', 'speedy'},
             'old' : {'aged', 'antique'},
             'slow' : {'sluggish'},
             'difficult' : {'hard', 'challenging', 'arduous'}}
```

Add 'strong' with two similar words

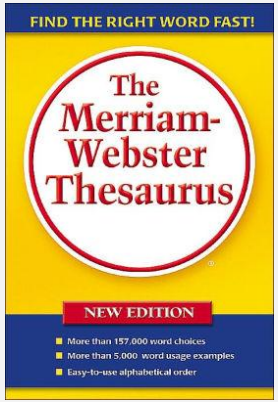



```
thesaurus = {'fast' : {'quick', 'agile', 'speedy'},
             'old' : {'aged', 'antique'},
             'slow' : {'sluggish'},
             'difficult' : {'hard', 'challenging', 'arduous'}}

thesaurus['strong'] = set()
thesaurus['strong'].add('durable')
thesaurus['strong'].add('robust')
```

Add an additional similar word for 'slow'

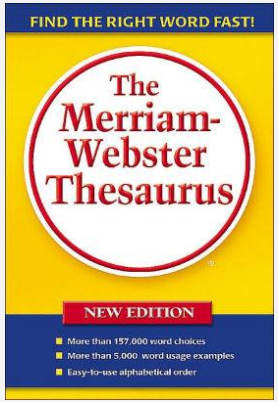
```
thesaurus = {'fast' : {'quick', 'agile', 'speedy'},
             'old' : {'aged', 'antique'},
             'slow' : {'sluggish'},
             'difficult' : {'hard', 'challenging', 'arduous'},
             'strong' : {'durable', 'robust'}}
```

Add an additional similar word for 'slow'



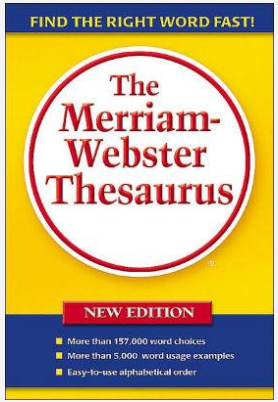

```
thesaurus = {'fast' : {'quick', 'agile', 'speedy'},
             'old' : {'aged', 'antique'},
             'slow' : {'sluggish'},
             'difficult' : {'hard', 'challenging', 'arduous'},
             'strong' : {'durable', 'robust'}}

thesaurus['slow'].add('gradual')
```

Add an additional similar word for 'slow'



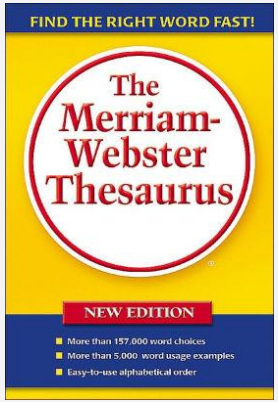

```
thesaurus = {'fast' : {'quick', 'agile', 'speedy'},
             'old' : {'aged', 'antique'},
             'slow' : {'sluggish', 'gradual'},
             'difficult' : {'hard', 'challenging', 'arduous'},
             'strong' : {'durable', 'robust'}}
```

Thesaurus file format

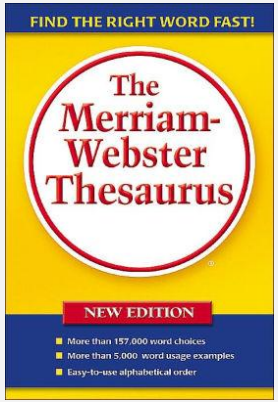



```
strong : durable robust
brave : fearless courageous
difficult : hard challenging arduous
slow : sluggish gradual
```

```
> brave
words similar to brave are:
   courageous fearless
> difficult
words similar to difficult are:
   hard arduous challenging
> instant
Sorry, I do not know that word
> exit
```



```
> brave
words similar to brave are:
   courageous fearless
> ADD
What word to add? brave
What is a similar word? adventurous
> brave
words similar to brave are:
   courageous adventurous fearless
> exit
```



```
def add_word(base_word, similar_word):
    # ???

def add_line(line):
    # ???

def save_thesaurus():
    # ???

def load_thesaurus():
    # ???

def main():
    # ???

main()
```

```
def add_word(thes, base_word, similar_word):
    ''' Add similar_word to the set of words that are similar
        to base_word. The steps:
      (1) if base_word is not already in the thesaurus, add it
          and map to an empty set
      (2) get the set associated with base_word
      (3) add the similar_word to the set
    '''
```

```
def add_word(thes, base_word, similar_word):
    if base_word not in thes:
        thes[base_word] = set()
    thes[base_word].add(similar_word)
```

```
def add_line(thes, line):
    ''' This function takes a single line from a thesaurus
        file. For example:

        strong : durable robust stable

      The function should process the line, and then add
      the word(s) to the thesaurus dictionary. Also
      remember to use the add_word function!
    '''
```

```
def add_line(thes, line):
    sp = line.split(' : ')
    key = sp[0]
    similar = sp[1].split(' ')
    for word in similar:
        add_word(thes, key, word)
```

```
def add_line(thes, line):
    sp = line.split(' : ')
    for word in sp[1].split(' '):
        add_word(thes, sp[0], word)
```

```
def add_word(thes, base_word, similar_word):
    if base_word not in thes:
        thes[base_word] = set()
    thes[base_word].add(similar_word)

def add_line(thes, line):
    sp = line.split(' : ')
    for word in sp[1].split(' '):
        add_word(thes, sp[0], word)
```

```
def save_thesaurus():
    # ???

def load_thesaurus():
    # ???

def main():
    # ???

main()
```

```
def load_thesaurus(thes):
    ''' Open the thesaurus file and store each
    line into the thesaurus dictionary. Remember,
    you can use the add_line function.
    '''
```

```
def load_thesaurus(thes):
    f = open('thesaurus.txt', 'r')
    lines = f.readlines()
    for line in lines:
        add_line(thes, line.strip('\n'))
```

```
def save_thesaurus(thes):
    ''' Save the contents of the thesaurus
    dictionary into thesaurus.txt. You can
    overwrite the old contents.
    '''
```

```
def save_thesaurus(thes):
    f = open('thesaurus.txt', 'w')
    for k, v in thes.items():
        l = list(v)
        f.write(k + ' : ' + ' '.join(l) + '\n')
    f.close()
```

```
def add_word(thes, base_word, similar_word):
    if base_word not in thes:
        thes[base_word] = set()
    thes[base_word].add(similar_word)

def add_line(thes, line):
    sp = line.split(' : ')
    for word in sp[1].split(' '):
        add_word(thes, sp[0], word)

def load_thesaurus(thes):
    f = open('thesaurus.txt', 'r')
    lines = f.readlines()
    for line in lines:
        add_line(thes, line.strip('\n'))
```

```
def save_thesaurus(thes):
    f = open('thesaurus.txt', 'w')
    for k, v in thes.items():
        l = list(v)
        f.write(k + ' : ' + ' '.join(l) + '\n')
    f.close()


def main():
    # ???

main()
```

Activity

```
def main():
    thesaurus = {}
    load_thesaurus(thesaurus)
    while True:
        text = input('> ')
        if text == 'exit':
            return
        if text.startswith('ADD'):
            ### (A) ask for a key-word, and a similar word, add
        else:
            if text in thesaurus:
                ### (B) show similar words to the word entered
            else:
                ### (C) ???
```

```
def main():
    thesaurus = {}
    load_thesaurus(thesaurus)
    while True:
        text = input('> ')
        if text == 'exit':
            break
        if text.startswith('ADD'):
            word = input('What word to add? ')
            similar_word = input('What is a similar word? ')
            add_word(thesaurus, word, similar_word)
        else:
            if text in thesaurus:
                similar = thesaurus[text]
                print('words similar to ' + text + ' are:')
                print('  ' + ' '.join(similar))
            else:
                print('Sorry, I do not know that word')
```

```
def add_word(thes, base_word, similar_word):
    if base_word not in thes:
        thes[base_word] = set()
    thes[base_word].add(similar_word)

def add_line(thes, line):
    sp = line.split(' : ')
    for word in sp[1].split(' '):
        add_word(thes, sp[0], word)

def load_thesaurus(thes):
    f = open('thesaurus.txt', 'r')
    lines = f.readlines()
    for line in lines:
        add_line(thes, line.strip('\n'))
```

```
def save_thesaurus(thes):
    f = open('thesaurus.txt', 'w')
    for k, v in thes.items():
        l = list(v)
        f.write(k + ' : ' + ' '.join(l) + '\n')
    f.close()

def main():
    thesaurus = {}
    load_thesaurus(thesaurus)
    while True:
        text = input('> ')
        if text == 'exit':
            break
        if text.startswith('ADD'):
            word = input('What word to add? ')
            similar_word = input('What is a similar word? ')
            add_word(thesaurus, word, similar_word)
        else:
            if text in thesaurus:
                similar = thesaurus[text]
                print('words similar to ' + text + ' are:')
                print('  ' + ' '.join(similar))
            else:
                print('Sorry, I do not know that word')
main()
```