CSc 110
Sets

Adriana Picoral (she/her/hers)
adrianaps@arizona.edu

Activity

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

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

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

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

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

Differences

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

# Remove from dictionary
del ds['c']

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

# Create empty
ds_2 = {}
```

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

# Set removal
ds.remove('c')

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

# ???
ds_2 = {}
```

Where's the bug?

```
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?

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

Looping through a set

- Does this work?

```
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:

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

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

Differences from a Dictionary

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

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

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

```python
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?

```python
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)
```

RuntimeError: changed size during iteration

```python
grades = {'A+', 'A', 'B', 'E', 'D', 'E', 'E-'}
grade_counts = {'A':5, 'B':10, 'C':7, 'D':4, 'E':2}
```

```python
for element in grades:
   grades.discard(element)
```
❌

```python
for element in grade_counts:
 del grade_counts[element]
```
❌

```python
for element in grades:
 if element in grade_counts:
   del grade_counts[element]
```
✔

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

```python
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
```

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

```python
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
```

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

```python
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
```

CSc 110
Sets

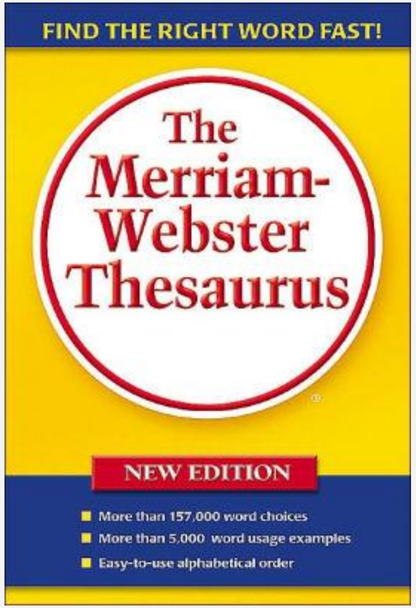
Adriana Picoral (she/her/hers)
adrianaps@arizona.edu

Exam 3

- No class Friday (Veteran’s Day) – no office hours on Friday
- Individual Exam November 16th
- Group Exam November 18th
    - No group members? Email me (adrianaps@arizona.edu)
- Review Session November 15th 5-7pm
- Study guide posted to website
- Infographic PA due on November 22
- TA evaluation form

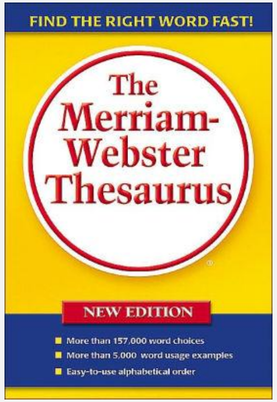
Representing a thesaurus

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


FIND THE RIGHT WORD FAST!
The Merriam-Webster Thesaurus
NEW EDITION
More than 157,000 word choices
More than 5,000 word usage examples
Easy-to-use alphabetical order

https://www.thesaurus.com

Representing a thesaurus

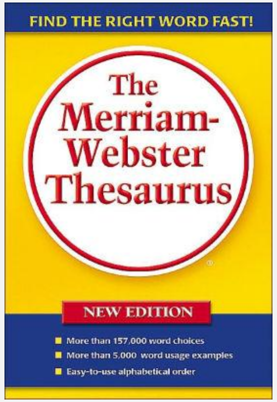

FIND THE RIGHT WORD FAST!
The Merriam-Webster Thesaurus
NEW EDITION
More than 157,000 word choices
More than 5,000 word usage examples
Easy-to-use alphabetical order

```
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


FIND THE RIGHT WORD FAST!
The Merriam-Webster Thesaurus
NEW EDITION
More than 157,000 word choices
More than 5,000 word usage examples
Easy-to-use alphabetical order

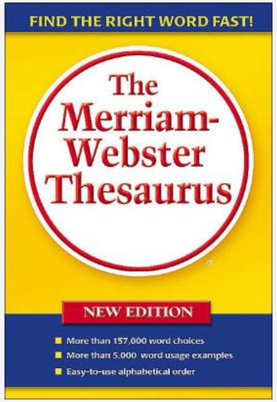
```
thesaurus = {'fast' : {'quick', 'agile', 'speedy'},
             'old' : {'aged', 'antique'},
             'slow' : {'sluggish'},
             'difficult' : {'hard', 'challenging', 'arduous'}}
```

```
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'}}
```

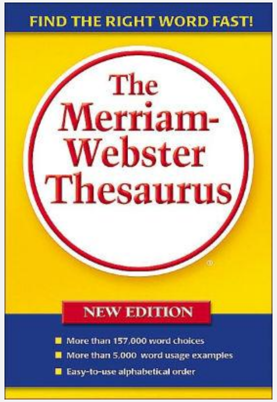
Add an additional similar word for 'slow'


FIND THE RIGHT WORD FAST!
The Merriam-Webster Thesaurus
NEW EDITION
More than 157,000 word choices
More than 5,000 word usage examples
Easy-to-use alphabetical order

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

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

Add an additional similar word for 'slow'


FIND THE RIGHT WORD FAST!
The Merriam-Webster Thesaurus
NEW EDITION
More than 157,000 word choices
More than 5,000 word usage examples
Easy-to-use alphabetical order

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

What words are in a text?

Write a **function** that creates a set of words in a string. The function takes in a string as an argument and returns a set of strings.

```
def main():
   my_text = "This is just a test to test the function to get words"
   print(get_words(my_text))

main()
```

Get Words Function

```
def get_words(input_text):
    words = input_text.strip("\n").split(" ")
    words_set = set()
    for w in words:
        words_set.add(w.lower())
    return set(words_set)

def main():
    my_text = "This is just a test to test the function to get words"
    print(get_words(my_text))

main()
```

What are stop words?

In text mining, stop words are extremely common words in a language.

Stop words in English:

- a
- the
- another
- is
- are
- in

Remove stop words from set

Write a **function** to remove stop words from a set of words.
The function should take in two sets, one of words and another of stop words.
Each word in the stop word set should be removed from the set of words.

Remove Stop Words Function

```
def remove_stopwords(words_set, stopwords):
    for w in stopwords:
        if w in words_set:
            words_set.remove(w)
```

Reusing your functions

Reuse your **get_words()** function to create a set of words in stranger_things.txt

Important Words in Stranger Things Dialogue

```
def create_wordset(filename):
   file_lines = open(filename, "r").readlines()
   all_words = set()
   for line in file_lines:
       this_set = get_words(line.strip("\n"))
       for w in this_set:
           all_words.add(w.lower())
   return all_words

def main():
   words_set = create_wordset("stranger_things.txt")
   print(len(words_set))

main()
```

Reusing your functions

Reuse your **remove_stopwords()** function to remove stop words from the count set (you can use stopwords.txt to create a set of stop words)

Important Words in Stranger Things Dialogue

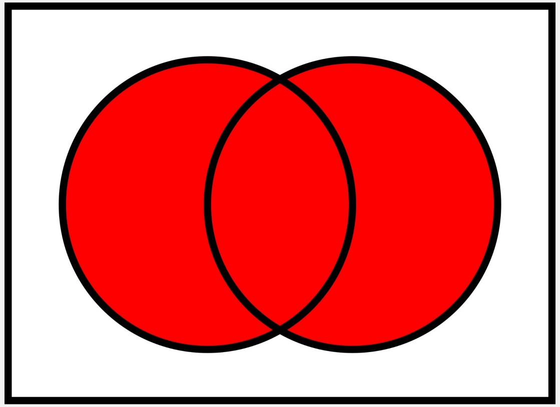
```
def create_wordset(filename):
   file_lines = open(filename, "r").readlines()
   all_words = set()
   for line in file_lines:
       this_set = get_words(line.strip("\n"))
       for w in this_set:
           all_words.add(w.lower())
   return all_words

def main():
   words_set = create_wordset("stranger_things.txt")
   print(len(words_set))
   stopwords = create_wordset("stopwords.txt")
   remove_stopwords(words_set, stopwords)
   print(len(words_set))
```

Union and Intersection

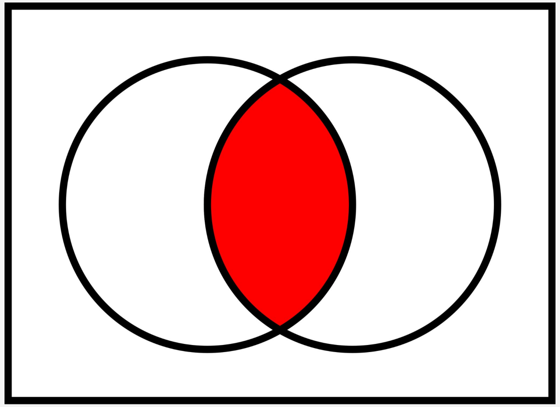
- Union Syntax: `set1.union(set2)`

```
s1 = {1, 2, 3}
s2 = {3, 4, 5}
print(s1, s2, s1.union(s2))
{1, 2, 3} {3, 4, 5} {1, 2, 3, 4, 5}
```



- Intersection Syntax: `set1.intersection(set2)`

```
s1 = {1, 2, 3}
s2 = {3, 4, 5}
print(s1, s2, s1.intersection(s2))
{1, 2, 3} {3, 4, 5} {3}
```



What do you think this will print?

```
my_list = {"apple", "banana", "pineapple", "pear", "strawberry", "orange", "blueberry"}

berries = set()
for fruit in my_list:
   if "berry" in fruit:
       berries.add(fruit)

fruits = my_list.difference(berries)
print(fruits)
```

Rewrite the previous code to remove stop words from a word set to use a set operation