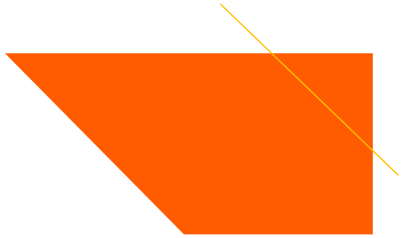
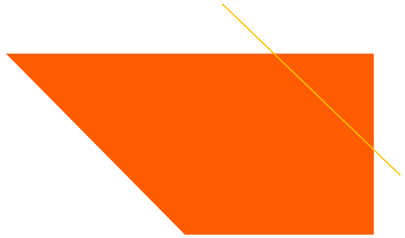
Introduction to Problem Solving in Python

COSI 10A

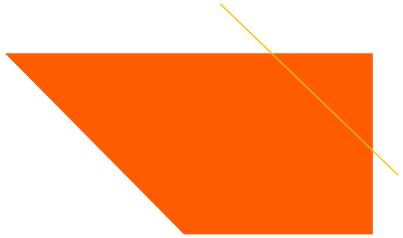


Class objectives

- Dictionaries (Section 8.1)
- Sets (Section 8.3)



Dictionaries & Sets



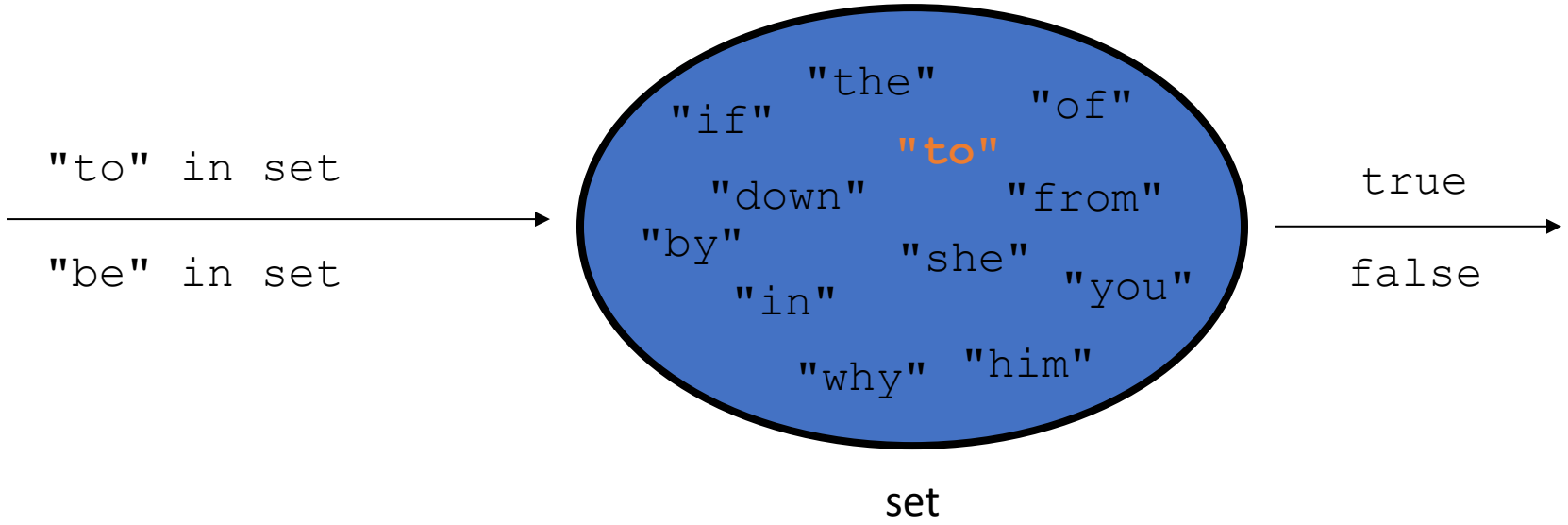
Exercise

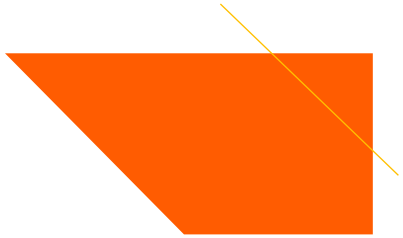
- Write a program that counts the number of unique words in a large text file (say, *Moby Dick* or the King James Bible).
  - Store the words in a structure and report the # of unique words.
  - Once you've created this structure, allow the user to search it to see whether various words appear in the text file.

- What structure is appropriate for this problem? List?

Sets

- **set**: A collection of unique values (no duplicates allowed) that can perform the following operations efficiently:
  - add, remove, search (contains)
  - We don't think of a set as having indexes; we just add things to the set in general and don't worry about order





Creating a `Set`

- An empty set:

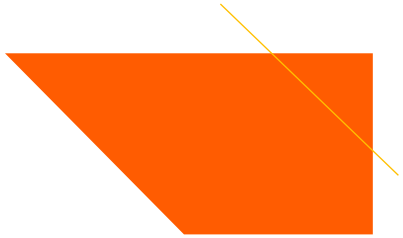
```
a = set()
```

- A set with elements in it:

```
b = {"the", "hello", "happy"}
```

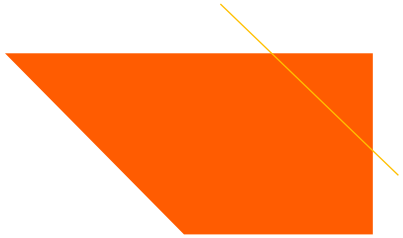
`a.add(`**val**`)`	adds element val to a
`a.discard(`**val**`)`	removes val from a if present
`a.pop()`	removes and returns a random element from a
`a - b`	returns a new set containing values in a but not in b
`a | b`	returns a new set containing values in either a or b
`a & b`	returns a new set containing values in both a and b
`a ^ b`	returns a new set containing values in a or b but not both

You can also use `in`, `len()`, etc.



Looping over a set?

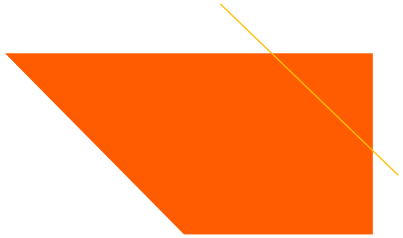
- You must use a for element in structure loop
  - needed because sets have no indexes; can't `get` element `i`



Exercise

- Write a program that counts the number of unique words in a large text file (say, *Moby Dick* or the King James Bible).
  - Store the words in a structure and report the # of unique words.
  - Once you've created this structure, allow the user to search it to see whether various words appear in the text file.

- What structure is appropriate for this problem? Set!



Exercise

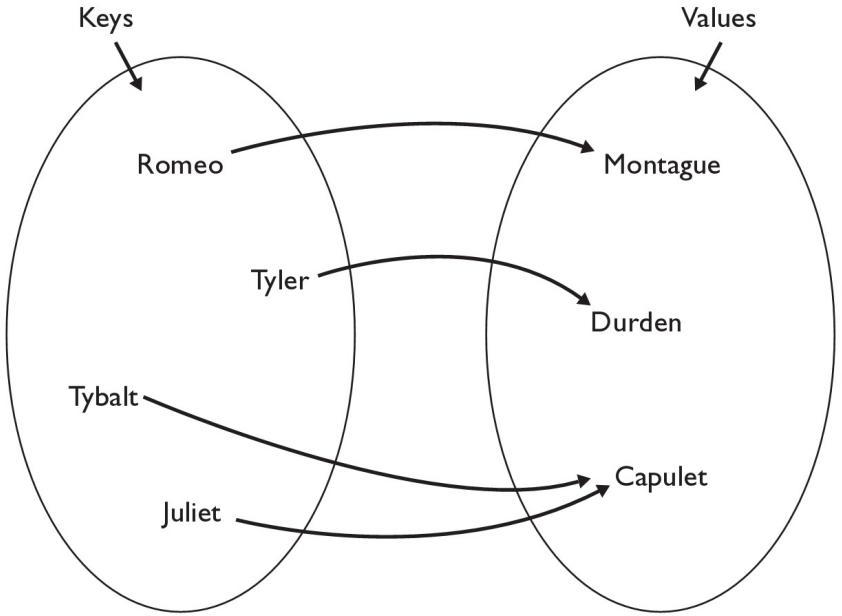
- Write a program to count the number of occurrences of each unique word in a large text file (e.g. *Moby Dick*).
  - Allow the user to type a word and report how many times that word appeared in the book.
  - Report all words that appeared in the book at least 500 times.

- What structure is appropriate for this problem?

Dictionaries

- **dictionary**: Holds a set of unique *keys* and a collection of *values*, where each key is associated with one value.
  - a.k.a. "map", "associative array", "hash"

- basic dictionary operations:
  - **put**(*key*, *value*): Adds a mapping from a key to a value.
  - **get**(*key*): Retrieves the value mapped to the key.
  - **remove**(*key*): Removes the given key and its mapped value.



`my_dict["Juliet"]` returns `"Capulet"`