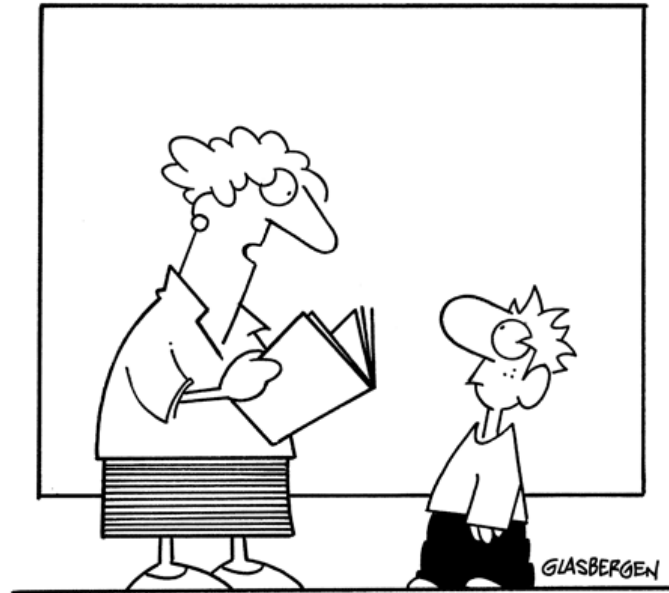


CSc 110, Autumn 2016

Lecture 25: Sets and Dictionaries

Adapted from slides by Marty Stepp and Stuart Reges

© Randy Glasbergen / glasbergen.com



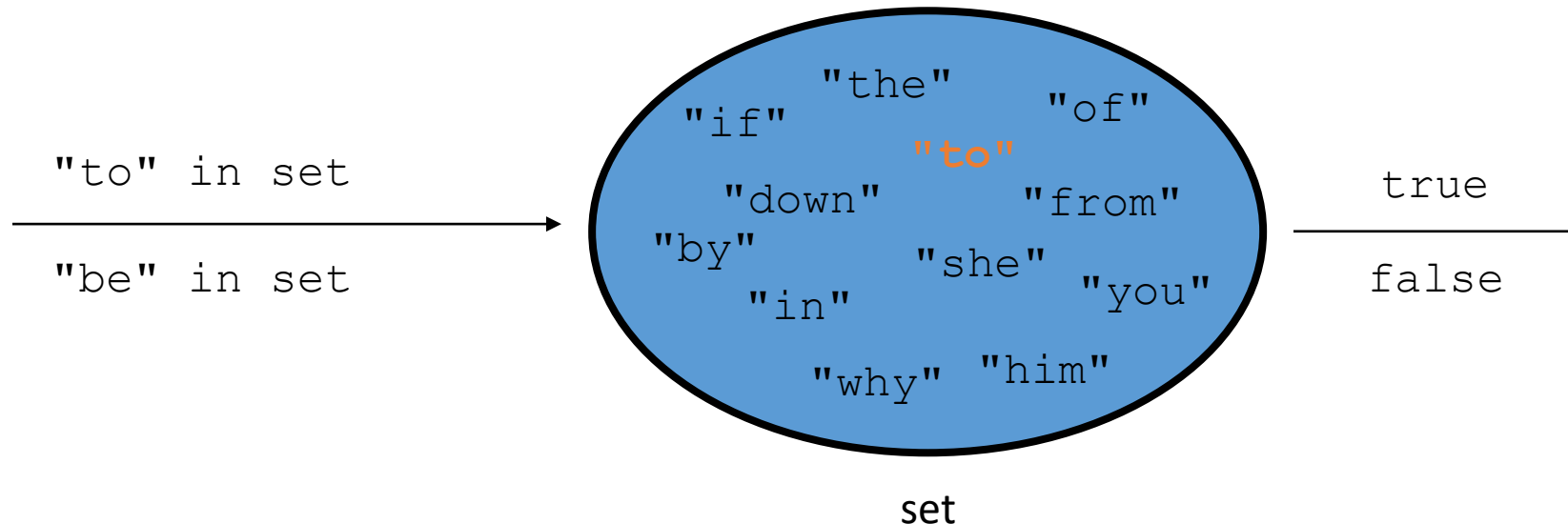
“Yes, some books come in high definition — dictionaries!”

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? Tuple?

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"}
```

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

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

Looping over a set?

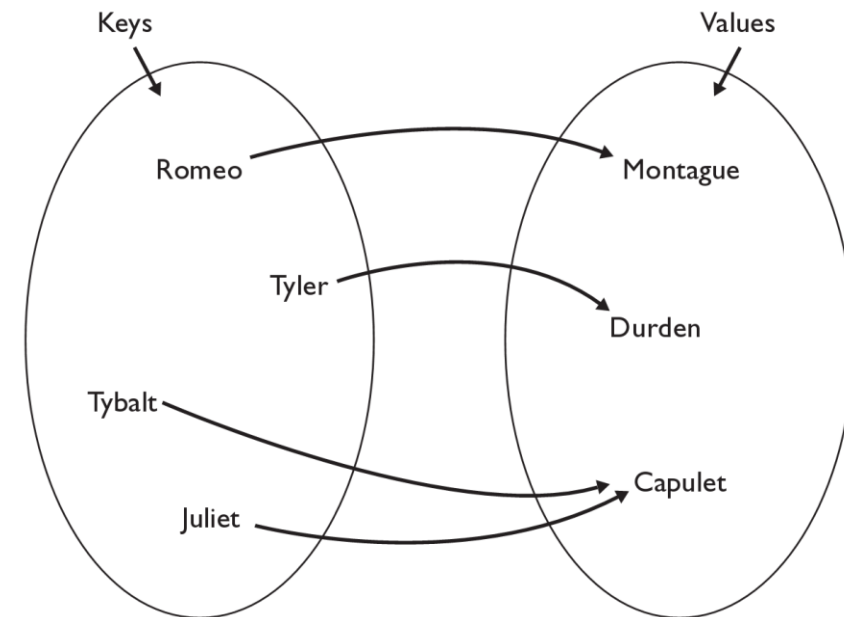
- You must use a for element in structure loop
 - needed because sets have no indexes; can't get element i

Exercise

- Write a program 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"

Dictionary functions

<code>my_dict[key] = value)</code>	adds a mapping from the given key to the given value; if the key already exists, replaces its value with the given one
<code>my_dict[key]</code>	returns the value mapped to the given key (error if key not found)
<code>items(key)</code>	return a new view of the dictionary's items ((key, value) pairs)
<code>pop(key)</code>	removes any existing mapping for the given key and returns it (error if key not found)
<code>popitem()</code>	removes and returns an arbitrary (key, value) pair (error if empty)
<code>keys()</code>	returns the dictionary's keys
<code>values()</code>	returns the dictionary's values

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

Maps and tallying

- a map can be thought of as generalization of a tallying list
 - the "index" (key) doesn't have to be an `int`

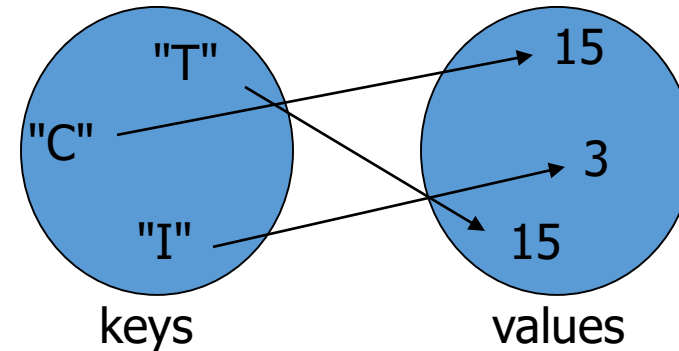
• count digits: 22092310907 →

index	0	1	2	3	4	5	6	7	8	9
value	3	1	3	0	0	0	0	1	0	2

(T)rump, (C)linton, (I)ndependent

- count votes: "TCCCCCCTTTTTCCCCCCTCTTITCTTITCCTIC"

key	"T"	"C"	"I"
value	15	15	3



items, keys and values

- `items` function returns tuples of each key-value pair
 - can loop over the keys in a for loop

```
ages = {}
ages["Merlin"] = 4
ages["Chester"] = 2
ages["Percival"] = 12
for cat, age in ages.items():
    print(name + " -> " + str(age))
```

- `values` function returns all values in the dictionary
 - no easy way to get from a value to its associated key(s)
- `keys` function returns all keys in the dictionary