CSc 110, Autumn 2016

Lecture 27: Sets and Dictionaries

Adapted from slides by Marty Stepp and Stuart Reges
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

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

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

• count votes: "TCCCCCCTTTTTTCCCCCCTCTTITCTTITCCTIC"
  
  # (T)rump, (C)linton, (I)ndependent

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

Diagram:
- Keys: "T" (15), "C" (15), "I" (3)
- Values: 15, 3
items, keys and values

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

```python
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