Objectives
• Dictionaries

How Does `in` Work for Lists?
• Example: `guess in prevGuesses`, where `prevGuesses` is a list object
  ➢ For each element in list, checks if element equals (==) `guess`
• In the worst case, how many elements does `in` have to check?
  ➢ How could we improve the search?

Faster Lookups
• If I wanted to know the Registrar’s phone number, …
  ➢ Would I search through an alphabetized list of phone numbers?
  ➢ No, I would look up the Registrar and find the phone number associated with the Registrar
• This type of data structure is known as a dictionary in Python
  ➢ Maps a key to a value
  ➢ Phone book’s key: “Registrar”, value: phone number

Examples of Dictionaries
• Any other things we’ve done/used in class?

Examples of Dictionaries
• Real-world:
  ➢ Dictionary
  ➢ Textbook’s index
  ➢ Cookbook
  ➢ URL (Uniform Resource Locator)
• Examples from class
  ➢ Function name --> function definition
  ➢ Variable name --> value
  ➢ ASCII value --> character

Example: Textbook’s Index

<table>
<thead>
<tr>
<th>Keys</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;integer&quot;</td>
<td>20</td>
</tr>
<tr>
<td>&quot;list&quot;</td>
<td>60</td>
</tr>
<tr>
<td>&quot;string&quot;</td>
<td>45</td>
</tr>
<tr>
<td>&quot;float&quot;</td>
<td>25</td>
</tr>
</tbody>
</table>

Lots of empty space to add new values

Keys are not in any order

Keys are not in any order
Dictionaries in Python

• Map keys to values
  ➢ Keys are probably not alphabetized
  ➢ Mappings are from one key to one or more values
    • Keys are unique, Values are not necessarily unique
      ➢ Example: student id -> last name
    • Keys must be immutable (numbers, strings)
• Similar to Hashtables/Hashmaps in other languages

How would we handle if there is more than one value for a key?

Why Dictionaries?

• Another way to store data
• Allow fast lookup of data
  ➢ Requires keys, unique keys
    • Data may not have a natural mapping

<table>
<thead>
<tr>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fast lookup (much faster than lists if a lot of elements)</td>
<td>Requires a lot of space, unique keys</td>
</tr>
</tbody>
</table>

Creating Dictionaries in Python

Syntax:

```
{<key>:<value>, ..., <key>:<value>}
```

```
empty = {}
ascii = { 'a':97, 'b':98, 'c':99, ..., 'z':122 }
```

Dictionary Operations

<table>
<thead>
<tr>
<th>Method Name</th>
<th>Functionality</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;dict&gt;.clear()</td>
<td>Remove all items from dictionary</td>
</tr>
<tr>
<td>&lt;dict&gt;.keys()</td>
<td>Returns a copy of dictionary’s list of keys</td>
</tr>
<tr>
<td>&lt;dict&gt;.values()</td>
<td>Returns a copy of dictionary’s list of values</td>
</tr>
<tr>
<td>&lt;dict&gt;.get(x[, default])</td>
<td>Returns &lt;dict&gt;[x] if x is a key; Otherwise, returns None (or default value)</td>
</tr>
</tbody>
</table>

Unlike strings and lists, doesn’t make sense to do slicing, concatenation, repetition for dictionaries

Dictionary Methods

<table>
<thead>
<tr>
<th>Method Name</th>
<th>Functionality</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax:</td>
<td></td>
</tr>
<tr>
<td>&lt;dictionary&gt;[&lt;key&gt;]</td>
<td></td>
</tr>
</tbody>
</table>

Examples:

```
ascii['z']
directory['registrar']
```

**KeyError** if key is not in dictionary
  ➢ Runtime error; exits program
Alternatively, Using get method

- `<dict>.get(x[, default])`
  - Returns `<dict>[x]` if x is a key; Otherwise, returns None (or default value)
  
  ```
  ascii.get('z')
  directory.get('registrar')
  ```

- If no mapping, get None back instead of KeyError

Accessing Values Using Keys

- Typically, you will check if dictionary has a key before trying to access the key

  ```
  if 'z' in ascii:
      value = ascii['z']
  Know mapping exists before trying to access
  ```

- Or handle if get default back

  ```
  val = ascii.get('z')
  if val is None:
      # do something …
  ```

Special Value: None

- Special value we can use
  - E.g., Return value from function when there is an error
  - Similar to null in Java

- If you execute

  ```
  list = list.sort()
  print list
  ```
  - Prints None because list.sort() does not return anything

Example Using None

```python
# returns the lowercase letter translated by the key.
# If letter is not a lowercase letter, returns None
def translate_letter(letter, key):
    if letter < 'a' or letter > 'z':
        return None
    # As usual …

# example use
enc_letter = translate_letter(char, key)
if enc_letter is None:
    print "Error in message: ", char
```

Inserting Key-Value Pairs

- Syntax:

  ```
  <dictionary>[<key>] = <value>
  ```

- `ascii['a'] = 97`
  - Creates new mapping of 'a' --> 97

Textbook's Index

```text
<table>
<thead>
<tr>
<th>_keys</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;integer&quot;</td>
<td>20</td>
</tr>
<tr>
<td>&quot;list&quot;</td>
<td>60</td>
</tr>
<tr>
<td>&quot;string&quot;</td>
<td>45</td>
</tr>
<tr>
<td>&quot;float&quot;</td>
<td>25</td>
</tr>
</tbody>
</table>
```
Textbook’s Index

- **bookindex["dictionary"]** = 58

<table>
<thead>
<tr>
<th>Keys</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;integer&quot;</td>
<td>20</td>
</tr>
<tr>
<td>&quot;list&quot;</td>
<td>60</td>
</tr>
<tr>
<td>&quot;string&quot;</td>
<td>45</td>
</tr>
<tr>
<td>&quot;dictionary&quot;</td>
<td>58</td>
</tr>
<tr>
<td>&quot;float&quot;</td>
<td>25</td>
</tr>
</tbody>
</table>

Adding/Modifying Key-Value Pairs

- Syntax:
  
  `<dictionary>[<key>] = <value>`

- `directory[‘registrar’]` = 8455
  
  Modifies old entry (if it existed) and changes mapping for `‘registrar’` to 8455

Problem

- Given a file of the form
  
  `<lastname> <year>`

- Create a mapping between the last names and years
  
  - How do we want to model the data?
  - What is the key? What is the value?

Why Data File Problems Ad Nauseam?

- "Parsing" data files for different purposes is very common

Simplified web application access log:

128.4.131.54 [09/Aug/2005:14:01:35] GET /dspace/simple-search
128.4.133.139 [09/Aug/2005:14:32:45] GET /dspace/adv-search
...

I write scripts to

- create user sessions (use as test cases)
- analyze user sessions (avg. length, patterns)
- emulate user sessions

Problem

- Given a file of the form
  
  `<lastname> <year>`

- Create and display a mapping between the last names and years
  
  - How to display the mapping in a pretty way?
  - What order is the data printed in?

- Modify the previous program to keep track of the number of students of each year
  
  - How do we want to model the data?
  - What is the key? What is the value?
  
  - Could we solve this using a list?
Analyzing years_dictionary2.py

- Anything useful/general that we could put in a function?

This Week

- Lab 8 due Friday
- Broader Issue: Digital Humanities