Useful Python Odds and Ends

Online Data String Functions

Oct 27, 2015
HW Schedule

• Today (HW 2-7 out) : Build concordance
  – *Hardest* so far!
• This Thursday: Extra concordance functions
  – Much more manageable
• This Friday: initial (draft) project proposal due
• Next Tuesday: *Last HW on Python*
Review: More Python Dictionaries

• Open today’s activity, ACT 2-6
• In Task 1, download and save ACT2-6.py

• You have a dictionary called `passwordDictionary`
  — Print it:
    — `>>> printDictionary(passwordDictionary)`

• Let’s now modify it to make it sorted
Review: More Python Dictionaries

- REMEMBER: Keys Are Unique
- REMEMBER: Key/Value pairs are Unordered

<table>
<thead>
<tr>
<th>Function/Syntax</th>
<th>Input</th>
<th>Output</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>keys()</td>
<td>None</td>
<td>List of keys</td>
<td>&gt;&gt;&gt; freq.keys() ['the', 'cat']</td>
</tr>
<tr>
<td>values()</td>
<td>None</td>
<td>List of values</td>
<td>&gt;&gt;&gt; freq2.values() [3, 2]</td>
</tr>
<tr>
<td>&lt;key&gt; in &lt;dict&gt;</td>
<td>None</td>
<td>Boolean</td>
<td>&gt;&gt;&gt; 'the' in freq2 True</td>
</tr>
<tr>
<td>del(&lt;dict&gt;[&lt;key&gt;])</td>
<td>Dict. Entry</td>
<td>None</td>
<td>&gt;&gt;&gt; del(freq2['cat'])</td>
</tr>
</tbody>
</table>
def printDictionary(dictionary):
    '''Prints dictionary key-value pairs'''
    keyList = sorted(dictionary.keys())
    for key in keyList:
        print(key, ' -> ', dictionary[key])
    return
Making Things Interactive

• Programs often need to obtain input from the user
  – `input(prompt)` Python function

• Run our example in the activity:
  – `echo()` function

```python
def echo():
    myInput = input('write something... ')
    print('You wrote:', myInput)
    return
```
Making Things Interactive

• Change the `addPassword` function:

```python
def addPassword(dictionary, key, value):
    print('Changing Password."
    dictionary[key] = value
    return dictionary
```

To test the function:

```python
addPassword(passwordDictionary, 'Me', '123456')
```
Making Things Interactive

First step: \textit{warning} on change

```
def addPassword(dictionary, key, value):
    print('Changing Password. ')
    if key in dictionary:
        print('Warning: overwriting data! ')
    dictionary[key] = value
    return dictionary
```
Making Things Interactive

Second step: **confirmation** on change

```python
def addPassword(dictionary, key, value):
    print('Changing Password. ')
    if key in dictionary:
        print('Warning: overwriting data! ')
        option = input("Change password? ")
        if option != 'y' and option != 'yes':
            print("Returning original database")
            return dictionary
        dictionary[key] = value
    return dictionary
```
Making Things Interactive

Third step: validation on change

```python
def addPassword(dictionary, key, value):
    print('Changing Password.')
    if key in dictionary:
        print('Warning: overwriting data!')
        option = input("Change password? ")
        if option != 'y' and option != 'yes':
            print("Returning original database")
            return dictionary
oldPass = input("Give me your pass: ")
if oldPass != dictionary[key]:
    print("Returning original database")
    return dictionary
dictionary[key] = value
return dictionary
```
Generating Files

“Open” a file for writing, and use the write() function on the file object.

```
myNum = 1
myFile = open('output.txt', 'w')
myFile.write('This is an output file
')
myFile.write(str(myNum))
myFile.write('
')
myFile.close()
```

Convert to string before writing!
Getting Data from Online Sources

It works just like files!

• import urllib.request loads a "module" that defines:
  – A urllib.request.urlopen() function (read-only!)
  – A file-like type in which you can perform:
    • read() (needs to be decoded)
    • close()
Getting Data from Online Sources

Using the `urllib.request` Python module

```python
import urllib.request

url = "http://cs.brown.edu/courses/csci0931/2015-fall/2-text_analysis/ACT2-6/ACT2-6.html"

remoteFile = urllib.request.urlopen(url)
contents = remoteFile.read().decode('utf-8')
remoteFile.close()
```
Task 1: Python Dictionary Review

Download and save <code><a href="ACT2-6.py">ACT2-6.py</a></code>. Open it in IDLE and press <kbd class="key">F5</kbd>.

Look at the dictionary called <code>passwordDictionary</code>. This is a list of the 25 easiest passwords to break in 2011...
import urllib.request
import re

url = "http://cs.brown.edu/courses/csci0931/2015-fall/2-text_analysis/ACT2-6/ACT2-6.html"

remoteFile = urllib.request.urlopen(url)
contents = remoteFile.read().decode('utf-8')
remoteFile.close()

# Cleans up HTML tags (very roughly)

contents = re.split('<body[^>]*>', contents)[1]
contents = re.split('</body[^>]*>', contents)[0]
contents = re.sub('<[^>]+>', '', contents)

Next Unit in our course!!!
Working with Strings

Check out the documentation...

- The `str` type has lots of great member functions:
  - `find()`
  - `replace()`
  - `strip()`, `lstrip()`, `rstrip()`
  - `join()` — the opposite of `split()`
Working with Strings

Check out the documentation...

- The `str` type has lots of great member functions:
  - `find()`
  - `replace()`
  - `strip()`, `lstrip()`, `rstrip()`
  - `join()` — **the opposite of `split()`**
find()

• Finds the first position of a word in a text

• Can start looking at some position (inclusive), stop at another position (exclusive)
  – *Optional* arguments!

```python
>>> mobyString.find('me')
5
>>> mobyString.find('me', 7)
20
>>> mobyString.find('me', 22, len(mobyString))
139
```
replace()

• Replaces all occurrences of one string by another
• Can specify the maximum number of substitutions to be made
  – *Optional* argument!

Try these two:

```python
>>> mobyString.replace('I', 'YOUR-LOYAL-CS931-TEACHER')
>>> mobyString.replace('I', 'YOUR-LOYAL-CS931-TEACHER', 6)
```
strip(), lstrip(), rstrip()

• Removes whitespace at start and end of the string
  – lstrip() does that only for the start of the string
  – rstrip() does that only for the end of the string

• You can specify the string to be stripped as an optional argument (defaults to whitespace)
join()

• Joins a list of strings through the specified delimiter (on which the function is called)
  – If the list of words is a single string, the function treats that string as a list of characters (as usual)

Try these two:

```python
>>> delim = ':'
>>> delim.join(['a', 'b', 'c'])
'a:b:c'
>>> delim.join('word')
'w:o:r:d'
```