Lab 9: Dictionaries

CSE/IT 107

Introduction

The purpose of this lab is to give you some experience with dictionaries, a very nice data structure.

Reading

Chapter 11 in Think Python: How to Think Like a Computer Scientist (Think)

Coding Conventions

Follow PEP-8 recommendations for your code.

Problems

Make sure your source code files are appropriately named. Make sure your code has a main function; use `boilerplate.py` you created in Lab 1.

Think Python is available from `http://www.greenteapress.com/thinkpython/`. Page numbers and exercise numbers refer to those found in the PDF file available at the website.

1. Exercise 11.1 *Think Python*, page 102. Name your source code `think_11-1.py`. The file `words.txt` is available at `http://www.greenteapress.com/thinkpython/code/words.txt`.
2. Exercise 11.2 *Think Python*, page 103. Name your source code `think_11-2.py`.
3. Exercise 11.3 *Think Python*, page 104. Name your source code `think_11-3.py`.
4. Exercise 11.4 *Think Python*, page 105. Name your source code `think_11-4.py`.

5. Exercise 11.6 *Think Python*, page 106. Name your source code `think_11-6.py`. Run the memo version of the Fibonacci sequence and print out the first 500 values of the Fibonacci sequence to a file name `fib500.txt`, one value per line.

6. Exercise 11.9 *Think Python*, page 111.

7. Rewrite the program Rock-paper-scissors-lizard-Spock from lab 4 using dictionaries. Name your source code file `rock-spock-dict.py`.

8. This problem is from Google's Python Class (`https://developers.google.com/edu/python/`):

   "Read in the file specified on the command line. Do a simple split() on whitespace to obtain all the words in the file. Rather than read the file line by line, it's easier to read it into one giant string and split it once.

   Build a "mimic" dict that maps each word that appears in the file to a list of all the words that immediately follow that word in the file. The list of words can be be in any order and should include duplicates. So for example the key "and" might have the list ["then", "best", "then", "after", ...] listing all the words which came after "and" in the text. We'll say that the empty string is what comes before the first word in the file.

   With the mimic dict, it's fairly easy to emit random text that mimics the original. Print a word, then look up what words might come next and pick one at random as the next work.

   Use the empty string as the first word to prime things. If we ever get stuck with a word that is not in the dict, go back to the empty string to keep things moving.

   Note: the standard python module 'random' includes a random.choice(list) method which picks a random element from a non-empty list."

```
import random
import sys


def mimic_dict(filename):
  """Returns mimic dict mapping each word to list of words which follow it."""
  # +++your code here+++
  return


def print_mimic(mimic_dict, word):
  """Given mimic dict and start word, prints 200 random words."""
  # +++your code here+++
  return
```

```
# Provided main(), calls mimic_dict() and mimic()
def main():
  if len(sys.argv) != 2:
    print 'usage: ./mimic.py file-to-read'
    sys.exit(1)

  dict = mimic_dict(sys.argv[1])
  print_mimic(dict, '')


if __name__ == '__main__':
  main()
```

Use the file `mimic.py` and read in the files `alice.txt` (Alice in Wonderland) and `mphg.txt` (Monty Python and the Holy Grail), which are all available on Moodle.

You are building a dictionary that looks like this:

```
{'a': ['hat', 'tree', 'swallow'], 'hat': ['flew', 'went']}
```

To figure out how to create the dictionary try the following in an interpreter:

```
>>> d = {}
>>> d['a'] = [1]
>>> print d
>>> d.get('a').append(2)
>>> print d
```

Once you get the program working for 200 words, modify so it will create a mimic of the whole text.

For more information about this problem, read `http://en.wikipedia.org/wiki/Markov_chain` and `http://joshmillard.com/garkov/`

Submission

Create a tarball of your *.py files and fib1000.txt

```
tar czvf cse107_firstname_lastname_lab9.tar.gz *.py fib500.txt
```

To check the contents of your tarball, run the following command:

```
tar tf cse107_firstname_lastname_lab9.tar.gz
```

You should see a list of your Python source code files.

Upload your tarball in Moodle before the start of you next lab.