ME 498 K; Spring 2023 — Homework 2

Aidan Hunt (ahunt94@uw.edu)

Due: April 14, 2023

In this homework, you will practice using lists, dictionaries, strings, and for loops. You will also practice reading documentation to learn how to use new data types and built-in methods that will help you accomplish this homework's tasks. If you have questions about this assignment, post them to the class discussion board for the fastest response. **Submit your response to each question as a separate file**.

1. Reading up on Dictionaries

An important skill to develop as a programmer is the ability to read documentation and learn how to use new functions and data structures. In class, we walked through various aspects of how to use the `list` data structure, including how to create lists, index lists, and traverse lists, as well as list reference semantics. In this problem, you will repeat this for dictionaries, another built-in data structure.

The questions below are examples of those you should ask when learning how to use a new data structure. Briefly respond (1-3 sentences) to each question below in your own words, and provide a code example that illustrates your answer. You should do this in a Jupyter Notebook, which will allow you to include "markdown cells" for your brief response and code cells for your coding examples. See the course notes on lists for guidance on formatting this type of mixed text/code document.

I recommend consulting the Python documentation or A Whirlwind Tour of Python for guidance on dictionaries, although you are welcome to use any other resources to strengthen your understanding (good programmers are good at Googling things, after all). However, the examples you present **must be your own original work**. Note: you will likely include your responses to this question in your Python guide, so consider your future self when writing your answers!

1. What are the components of a dictionary? Show the syntax for creating a dictionary by manually specifying the values of these components (e.g., like we manually created lists in lecture).
2. How can a dictionary be created from two existing lists? Show an example.
3. How do you access data that is stored in a dictionary? How can you add data to an existing dictionary? How can you remove data from an existing dictionary? Show an example of each.
4. Like lists, there are several different ways to traverse dictionaries using a for loop. Demonstrate at least 2 approaches and highlight why each approach might be useful.
5. Are dictionaries mutable? Illustrate your response with an example. Your response should consider both keys and values.
6. What kind of data is well represented by a dictionary, rather than a list? Why? **Note:** you do not need a code example for this question.

Submit your response to this question to Gradescope as an `.ipynb` file.

2. Word Search

TheCode.txt contains text from the National Society of Professional Engineers Code of Ethics, and meWiki.txt contains the text from the Wikipedia page on mechanical engineering. Write a script that, for each file, parses the text and creates two dictionaries based on the contents:

1. A dictionary whose keys are the letters of the alphabet. The value corresponding to each key should be a list of words in the text file that start with that letter. If a word occurs multiple times in the text file, you should include the word in the corresponding list each time it occurs. See wordDict_ethics.txt and wordDict_me.txt for target output (your script just needs to create dictionaries, not text files).
2. A dictionary whose keys are each **unique** word in the text file. The value corresponding to each key should be the number of times each word is found in that file. See countDict_ethics.txt and countDict_me.txt for target output.

You may assume that any sequence of letters is a valid "word". However, when determining the letter that a word starts with, as well as identifying unique words, you should not consider capitalization (i.e., 'Engineer' is the same as 'engineer'). Additionally, the words you identify should not include digits, and should not include punctuation. This will mean, for example, "engineer's" will be sorted/counted the same as "engineers", and "quasi-governmental" would be sorted/counted as "quasigovernmental". What characters count as punctuation? Take a look at the punctuation constant in the string module, which is described later in this specification.

To read TheCode.txt or meWiki.txt, use the following code snippet (the with keyword ensures that the Python closes the file after it is opened and read):

```
with open(textFile, 'r', encoding='utf-8') as file:
    textData = file.read() # Extract the data from the text file
```

This will open the text file whose name is represented by the variable textFile and store the contents as a single string named textData. However, since we are interested in individual words in the file, you will need to find a way to convert this string into a sequence of individual words. In doing so, you may assume that individual words are separated by whitespace and/or linebreaks.

To work with this text data, you'll want to become more familiar with the str data type. Like the list data type, the str data type has many helpful built-in methods you can use to perform operations on text. Additionally, the built-in string module defines several constants that you should use to help create your dictionaries. For example, rather than manually typing out all the letters in the alphabet, you can instead use:

```
import string # Import the string module

letters = string.ascii_lowercase # Get all lowercase letters
```

You may be tempted to use logical statements to sort words and filter punctuation. While this could be used to build a more sophisticated file parser, **you are not allowed to use if/else statements or logical operators to solve this problem**. Instead, you should consider the list, dict, and str methods that can help you solve the problem.

Your code will be graded on whether it correctly produces the requested dictionaries, as well on whether it is well-organized, well-documented, and follows the Style Guide. As always, **you should define functions that represent distinct tasks** in order to structure your code and reduce redundancy. Each function that you define should include a docstring that describes all parameters, returns, and behavior, and your script should include a program summary at the beginning that describes, in general, what the script does.

Submit your code to Gradescope as either WordSearch.py or WordSearch.ipynb.