CS 1026B – Assignment 2

Due: March 8, 2022 – 6:00 PM

Overview

Write a program using lists and functions to compare names using a phonetic algorithm.

Reminders

- Your code must be done individually.
- Your code will be graded in part by an automated system.
- Your code may be compared to other submissions using computer software.
- You can submit your code up to 48 hours late, with a deduction of 0.5% per hour (or part of an hour) that the assignment is late.

Background

People can have very similar sounding names that are spelled differently. When hearing a name without knowing the spelling, someone may spell the name in several different ways. We would like a tool that can roughly group similar sounding names together to allow us to search for names when an exact spelling may not be known.

Different sounding names can arise for a variety of reasons, including individual preference of parents in assigning first names and differing ways to transliterate names from other languages into the Latin alphabet, for example. There are obvious limitations and biases to approaches to group similar sounding names together: many were originally designed for names that were historically common in the United States, and likely perform better on common Western European names. They will likely not sufficiently capture differences or similarities in names from other backgrounds.

Phonetic algorithms are tools used to compare similar sounding names. In this assignment, we will implement a variety of the Soundex algorithm, originally developed and patented in 1918. In the version of Soundex algorithm that we are implementing, each name will be translated into a string of the form AXXX, where A is a lower-case letter and XXX is a three-digit numerical code. We call this string the Soundex encoding of the name. Two names will be judged to be similar sounding if their Soundex encodings are the same.

We are choosing the Soundex algorithm for simplicity, as it relies on a relatively small number of rules. However, as an older algorithm, Soundex will likely not perform as well as more modern algorithms on diverse groups of names.

Soundex Algorithm

The algorithm to replace a name with its code has several steps. The steps assume that the name is written in Latin letters with no accents or other modifiers. The steps of the algorithm are:

1. Convert the letters in the name to lower case.
2. Let F be the first letter of the name, which we save for later.

3. Replace all letters in the name with digits according to Table 1, which results in a string of digits, which we call D. (hint: construct seven different lists and use the 'in' keyword to test which character category a letter belongs to.)
4. Update D by replacing any multiple repeating digits (two or more) with a single copy of the digit.
5. Remove all occurrences of 0 in D.
6. Convert F to a digit using Table 1. If the digit for F is the same as the first digit of D, then remove the first digit of D and replace it with F. (hint: use a slice on the string, like we did with a list.) If the digit for F is not the same as the first digit of simply add F to the beginning of D. If D has length zero, then set D to be the digit for F.
7. Look at the length of the new string D (with the letter at the start).
    a. If the length is 4, do nothing.
    b. If the length is less than 4, add enough 0s to the end of D so that its length is 4. (hint: use the * operator for strings)
    c. If the length is greater than 4, remove characters from the end so that the length is 4 (hint: use a slice on the string).

Letters	**Replacement Digit**
a, e, i, o, u, y, h, w	0
b, f, p, v	1
c, g, j, k, q, s, x, z	2
d, t	3
l (the letter ell)	4
m, n	5
r	6

Table 1: Letter replacements.

Soundex Examples

Consider the name "Domaratzki".

1. We convert the string to lower case before starting the algorithm.
2. F = 'd'
3. After replacing all letters with digits, we get D = '3050603220'. (note that D starts with 3 as we keep the letter d at the start of the name when replacing letters with digits)
4. After replacing consecutive digits with a single copy of the digit, we get D = '305060320'.
5. After removing 0s, we get D = '35632'.
6. F = 'd' is converted to the digit '3', which is the same as the first digit of D, so we drop the first digit, and D = 'd5632'.
7. We drop digits from the end of D to get a string of length 4, so the Soundex encoding is 'd563'.

In the same way, the name 'Schmidt' would be converted to 's530':

schmidt –> 2205033 –> 20503 –> 253 –> s53 –> s530

Program

You should write a complete python program that asks for a list of names and then shows the pairs of names in the list that have the same Soundex encoding.

1. Prompt for the user for a collection of names. Store the names in a list.
    a. You can assume that the user does not type in duplicate names.
    b. You can assume the user does not type in any extra characters that are not part of the name, such as spaces.
    c. You can assume that all names consist of letters from the English alphabet without any accents. In particular, there are no spaces, apostrophes or hyphens in the names, for simplicity. You can also assume that the user enters in a non-zero number of letters for each name.
    d. You should end processing when the user types in DONE. You can assume that DONE is not a name. Use the following prompt **exactly** (without quotation marks when displayed) once before accepting input, as shown in the example below:
    "Enter names, one on each line. Type DONE to quit entering names."

    Do not have any prompts on any of the next lines where the user types in names (i.e., for all the calls to input(), use an empty prompt.
2. Compute the Soundex encoding for each name and store a **tuple** (soundex,name) for each pair in a list.
3. Write a nested loop that compares each distinct pair of names in the list and determines which have the same Soundex encoding. (Here *distinct* means pairs of different names).
4. If a pair of names has the same Soundex encoding, construct a line of output for each pair that has the same of the form

    ```
    Name1 and Name2 have the same Soundex encoding.
    ```

    In this output, Name1 should come before Name2 in alphabetical order (hint: use min() and max() for strings). Note that every **different** names Name1 and Name2 that have the same Soundex encoding should appear in the output. You should **not** output lines that say "Name1 and Name1 have the same Soundex encoding." Do **not** print the output immediately: store all of these lines of output in a list.
5. After all the lines of output have been constructed, print all of the output in alphabetical order, sorted by Name1. That is, in the final output, the first line should start with the name that comes first in alphabetical order (hint: use sort(), then print.)

Functions

You are **required** to break your code into functions. The following functions are required:

1. A function to read the input from the user. The function should have no parameters and return a list of strings (the names).
2. A function to complete step 3 of the Soundex algorithm (replacing letters with digits). The function should take a string parameter and return a string (of digits). It should not return an int.
3. A function to complete step 4 of the Soundex algorithm (replacing multiple digits with a single copy of the digit). The function should take a string parameter (the string of digits with possible repeats) and return a string (of digits with no repeats).

4. A function to complete step 7 of the Soundex algorithm (making sure the length of the final encoding is 4). The function should take a string parameter (partial Soundex string created after step 6) and return a string (the final Soundex encoding).
5. A function to complete the overall Soundex algorithm (steps 1-7). This function should take a string as a parameter (the name) and return the final Soundex encoding as a string.
6. A main function.

You can give the functions (except main) any name that is suitable. You may add additional functions beyond this if you like.

Code Requirements

Your code must satisfy the following requirements in addition to producing correct output.

1. Your code must be documented appropriately. Do not document every line of code, but major portions of your code must be commented.
2. Your code must include a comment at the top of program that includes your name and describes the overall function of the program.
3. Your code should use appropriate structures, such as loops, lists and functions. Functions must be used based on the requirements above. You should also use a main() method.
4. You should give a docstring for each function you write (except for main). See the programming standards for more details.
5. You should review the programming standards document for information on commenting, variable naming, documentation of functions, and other issues.

Submission Details

- You must submit your code to gradescope using the instructions on ow;.
- You must name your code Assign2.py.
- You must follow the format of the output shown in the example below, including prompts.

Assignment Marking

- Your code will be marked by an automated tool. The testing program assumes that
    - The code is saved as a file called Assign2.py
    - You are using a python3 version (recommended python3.9 or higher). If you use python2, you will likely lose marks.

  If your code has to be marked manually due to failure to follow these instructions, you will receive a deduction from your final mark.
- Your code will be evaluated by the Teaching Assistants for the code requirements described above.

Example Execution

The following sample run of the program is provided for your reference only. It is not a complete example of all possible cases. You should design your own test cases to ensure your code is working. The red lines in the input below represent the user input.

```
Enter names, one on each line. Type DONE to quit entering names.
Liang
Smythe
Smith
Davies
Davis
Johnson
Xiang
Domaratzki
Johanssen
Leung
DONE
Davies and Davis have the same Soundex encoding.
Johanssen and Johnson have the same Soundex encoding.
Leung and Liang have the same Soundex encoding.
Smith and Smythe have the same Soundex encoding.

Process finished with exit code 0
```

Gradescope Test Cases

There are 13 gradescope tests. They are available on owl to review, along with the reason for each test case. If you are failing a test, you should review the test first to identify the problem that is being tested, to help debug your code. You can debug with the test case or a similar case that tests a similar input.