Python and Puzzles

Rick Zucker

https://github.com/rnzucker

NPR Sunday Puzzle

- Variety of word and name puzzles - http://www.npr.org/series/4473090/sunday-puzzle
- ~25% lend themselves to programmatic solutions
- Examples:
  - Take the word EASY: Its first three letters — E, A and S — are the fifth, first, and nineteenth letters, respectively, in the alphabet. If you add 5 + 1 + 19, you get 25, which is the value of the alphabetical position of Y, the last letter of EASY.
    - Think of a common five-letter word that works in the opposite way — in which the value of the alphabetical positions of its last four letters add up to the value of the alphabetical position of its first letter?
  - Bail, Nail, and Mail are three four-letter words that differ only by their first letters. And those first letters (B, N, and M) happen to be adjacent on a computer keyboard. Can you think of five four-letter words that have the same property

Program Basics

- Take Unix *words* file
  - ~25,000 lines, in /usr/dict/words or /usr/share/dict/words or similar location
  - Limits: only includes, e.g, **walk**. Doesn’t have **walks**, **walked**, or **walking**
  - Found a 100K line file that includes conjugations and plurals
- Scan for words of right length and other properties
  - Number of vowels, consonants, etc.
- Create a new list to process
  - Often useful to add words in reverse order

Bail, Nail, Mail example

```
for i in range(num_words):
       num_chars = len(words[i])
       if (num_chars == 5): # Newline is included in the number of characters
           # Remove \n with rstrip()
           temp = words[i][0:4].rstrip()
           # Append reverse form of word to end
           if temp[0] in ROW1:
               four_chars[0].append(temp[::-1])
           elif temp[0] in ROW2:
               four_chars[1].append(temp[::-1])
           elif temp[0] in ROW3:
               four_chars[2].append(temp[::-1])
           else:
               print(temp, "starts with non-letter")
           num_four += 1
   print("\n", num_four, "four letter words\n")
   # Sort by reversed words, so easier to find matches
   four_chars[0].sort()
   four_chars[1].sort()
   four_chars[2].sort()
```

Other Plans

- Often geography problems
- Creating a class for cities/countries
  - State capitals, world capitals, major US cities, countries
- Return list in a category with:
  - Certain number of letters
  - Certain number of vowels
  - Certain number of consonants
- Example
  - Take the name of a country and a well-known city in the Middle East — 12 letters in all. Rearrange these letters to name another country and another well-known city in the Middle East. What places are these?

Backup

Bail, Nail, Mail example

```
# Check first three letters (in reversed form) of five consecutive words to see if they match.
    # We know that the fourth letter are all in the same row of the keyboard since we have a separate
    # sub-array for each ROW.
    for i in range(3):
        len_row = len(four_chars[i])
        for j in range(len_row-4):
            # Five consecutive words with same last three letters (words are reversed)
            if (four_chars[i][j][0:3] == four_chars[i][j+1][0:3]) and (four_chars[i][j][0:3] == four_chars[i][j+2][0:3]) and\
               (four_chars[i][j][0:3] == four_chars[i][j+3][0:3]) and (four_chars[i][j][0:3] == four_chars[i][j+4][0:3]):
                # Fill element array of words properly ordered. There can be more than five, so do it dynamically
                possible_words = []
                k = 0
                while (j+k < len_row) and (four_chars[i][j][0:3] == four_chars[i][j+k][0:3]):
                    possible_words.append(four_chars[i][j+k][::-1])
                    k += 1
                possible_words.sort(key=row_key)
                # We are only interested in the first five words, so we only check those
                if row_in_order(possible_words[0:5]):
                    print(possible_words[0:5])
```