An Anagram of a Word

Final report prepared for the Python Code Challenges: Anagrams

July 05, 2018

Project Object

What is anagram? An anagram is word or phrase formed by rearranging the letters of a different word or phrase, typically using all the original letters exactly once. For example, the word boats can be rearranged into “boast”.

In this Python program, it will take a word as an input, randomly print out a valid anagram of this word, and show the total number of all possible valid anagrams of this word.

Keywords

Python, Anagram, Generators, Python yield

Possible Use Cases

- Crossword Puzzles
- Games such as Scrabble or Upwards
- Anagrams could unveil the words' hidden meanings. In the middle ages some scientists coded their findings in anagrams until they were ready to reveal them.
- Writers often use anagrams to add mystery or intrigue to a novel

Datasets

A word which contains only letters

Methodology

1. Algorithm

In this program, we mainly used enumerate method, yield keyword and DFS concept (Depth First Search) to solve this problem.

2. The Code

We used the “random” package to randomly choose an anagram of the word, then print it out. (Figure 1)

```
In [1]: import random
```

Figure 1

When the Python program is running in Jupyter Notebook, the following code will execute as the main function. (Figure 2)

```
if __name__ == "__main__":
    word = input("please input your word: ")
    while(isAWord(word) is False):
        print(word, 'is not a word, please re-enter your word:')
        word = input()

    word_lc = word.lower()
    res = []
    for i in anagrams(word_lc):
        res.append(i)
    print("There are ", len(res), " of anagrams of word:", word)
    anagram = random.choice(res)
    print("One anagram of word", word, "is:")
    print(anagram)
```

Figure 2

The isAWord function is used to define the input is a valid word or not. (Figure 3)

```
def isAWord(word):
    return word.isalpha()
```

Figure 3

The anagrams function is used to generate all valid anagrams of the input word. (Figure 4)

```
def anagrams(word):
    """ Generate all of the anagrams of a word. """
    if len(word) < 2:
        yield word
    else:
        for i, letter in enumerate(word):
            #avoid duplicating earlier words
            if not letter in word[:i]:
                for j in anagrams(word[:i]+word[i+1:]):
                    yield j+letter
```

Figure 4

3. Workflow

As the Figure 2 shows, our program firstly ask user to input a word

```
please input your word:
```

After getting the word, it will determine whether the word is a valid word. If the word is not valid, it will print a “Not a Word” error message and ask for another input.

```
please input your word: swqw'd
swqw'd is not a word, please re-enter your word:
```

If the word is valid, print one anagram of the input word and the total number of the anagrams of this word.

```
swqw'd is not a word, please re-enter your word:
sollers
There are  1260  of anagrams of word: sollers
One anagram of word sollers is:
soellrs
```

4. Final Output

```
please input your word: swqw'd
swqw'd is not a word, please re-enter your word:
sollers
There are  1260  of anagrams of word: sollers
One anagram of word sollers is:
sllsoer
```

Conclusion

Whether anagram is for a game or just to have fun, you can use anagrams for a variety of reasons. They are easy to make out of any name or phrase, or an interesting play on words and challenge us to be creative and witty.