Python - 3

September 30, 2020

1 Python - 3

1.1 Question 1:

Create a list of 10 random words of your choice from the English language. Sort the words in alphabetical order and print them

[2]:
```
word_list = ['Apple', 'Strawberry', 'Dog', 'Information', 'Jacket',␣
 ↪'Chocolate', 'Pyhton', 'Muffin', 'Book', 'Water' ]
print("Initial word list:", word_list) #Printing initial word list in random␣
 ↪order
word_list.sort() #Sorting the list in alphabetical order
print("Sorted word list: ", word_list) #List in sorted aplhabetical order
```

```
Initial word list: ['Apple', 'Strawberry', 'Dog', 'Information', 'Jacket',
'Chocolate', 'Pyhton', 'Muffin', 'Book', 'Water']
Sorted word list:  ['Apple', 'Book', 'Chocolate', 'Dog', 'Information',
'Jacket', 'Muffin', 'Pyhton', 'Strawberry', 'Water']
```

1.2 Question 2:

Now create a similar list but randomly generate these words from the English language i.e. you are not specifying the words but they are being randomly generated.

In the spirit of what we have been discussing in class, please take this opportunity to research different ways in which you can randomly generate English language words using resources and functions available online.

I did a similar research and settled on using the random_word package in Python. If you choose to use this, you will need to first install the random-word package. Do this by running the following command in a Jupyter Notebook cell: %pip install random-word

The folllowing lines are useful to retrieve random words using this package. You can run and study these lines or use other methods you found to generate random words.

from random_word import RandomWords r = RandomWords()
r.get_random_word()
r.get_random_words()

[3]:
```
from random_word import RandomWords
r = RandomWords()
randomlist = (r.get_random_words()) #Function to generate random wordlist
```

```
print("Initial word list:", randomlist) #initial random list
randomlist.sort()
print("\nSorted word list: ", randomlist) #Alphabetically sorted list

#There are some other functions as well which can randomly generate words, but␣
 ↪more specifically
#For example: 1. hasDictionaryDef (string) - This function will only return␣
 ↪words with dictionary definitions
```

```
Initial word list: ['planing', 'transumpt', 'superviolent', 'prebound',
'airworthiness', 'quadrumanal', 'clithrophobia', 'violently', 'cupuliform',
'strenth', 'warhorses', 'marriage', 'chalybeates', 'homemaker', 'tentaculata',
'hylozoist', 'pedophobia', 'portly', 'Markus', 'malachites', 'preemphasis',
'blacklight', 'aminopyralid', 'Brithenig', 'prophetize', 'imperialize', 'joseph
goebbels', 'websurfers', 'trufle', 'sagaciously', 'finaincial', 'translocating',
'intuitional', 'serviceability', 'kingpost', 'zyzzyvas', 'chasuble',
'reifiable', 'urethan', 'luxon', 'seam-rent', 'talkathons', 'Warlpiri',
'anticolonialist', 'instructive', 'unlogged', 'Hassid', 'journeywoman',
'disambiguating', 'vomitoria']

Sorted word list:  ['Brithenig', 'Hassid', 'Markus', 'Warlpiri',
'airworthiness', 'aminopyralid', 'anticolonialist', 'blacklight', 'chalybeates',
'chasuble', 'clithrophobia', 'cupuliform', 'disambiguating', 'finaincial',
'homemaker', 'hylozoist', 'imperialize', 'instructive', 'intuitional', 'joseph
goebbels', 'journeywoman', 'kingpost', 'luxon', 'malachites', 'marriage',
'pedophobia', 'planing', 'portly', 'prebound', 'preemphasis', 'prophetize',
'quadrumanal', 'reifiable', 'sagaciously', 'seam-rent', 'serviceability',
'strenth', 'superviolent', 'talkathons', 'tentaculata', 'translocating',
'transumpt', 'trufle', 'unlogged', 'urethan', 'violently', 'vomitoria',
'warhorses', 'websurfers', 'zyzzyvas']
```

1.3 Question 3:

We are going to use these words as part of an auto-completion program.

An auto-completion program is used to auto-complete and provide suggestions as you run searchs on the internet.

For example, when you start typing "ra" in the search box, the search box might auto-fill "rainbow" or "rainfall" or "raining".

The auto-fill is based on a variety of criteria. We will mimic some simple examples here.

Retrieve a random list of words. Convert all the words to lowercase and store them as a list in the variable called words. Sort the list alphabetically and print it.

Here is an output example of a sorted list of random words that I generated: ['alcoholic', 'babism', 'backfit', 'bemusing', 'boxsets', 'buoyant', 'carbonaro', 'chastising', 'chelerythrine', 'child-proof', 'cock-feather', 'comforted', 'counterterror', 'custrel', 'cuteness', 'defrostable', 'flesh-pots', 'frits', 'gypsey', 'heptathlon', 'lades', 'lazarets', 'lighter', 'livestock', 'melancholy', 'meso-scopic', 'miraculous', 'morello', 'opting', 'pickmaw', 'sabir', 'scamps', 'shaken', 'steppingstone', 'succinite', 'target-hunting', 'toffs', 'tonsilla', 'trader', 'tureens', 'twistings', 'unassimilated', 'un-parseable', 'unplowable', 'utopia', 'waterskiing', 'west germany', 'whomps', 'willet', 'wingding']

Note: Do not use these words in your assignment. You have to generate these words randomly.

[17]:
```
from random_word import RandomWords
r = RandomWords()
a = r.get_random_words(hasDictionaryDef="true") #Only getting words which have␣
↪dictionary meanings
print("Initial word list:", a)
words = [x.lower() for x in a] #Converting words to lower-case
words.sort() #Aplhabetically sorting the list
print("\nSorted List:", words) #printing list in lowercase
```

```
Initial word list: ['timberyards', 'vernalize', 'abrim', 'truck-driver',
'Landau', 'premiere', 'homeodomain', 'andesitic', 'undepleted', 'assieging',
'thriftlessly', 'predilections', 'snorkeler', 'sickle-billed', 'AZERTY',
'procambium', 'midnite', 'lobsterbacks', 'cerata', 'Vergil', 'radical', 'dare-
deviltry', 'altricial', 'tortion', 'break-dance', 'bubble-boy', 'voltammetry',
'subtlety', 'decoking', 'yandere', 'governmentally', 'thousands-of', 'Minoans',
'crystallites', 'squireship', 'expiation', 'whenceever', 'souke', 'fantail',
'mouse-mill', 'puromycin', 'telopeptides', 'angelican', 'picro', 'gunda', 'on-
camera', 'ninth', "Bluff's", 'reptantia', 'cellblock']

Sorted List: ['abrim', 'altricial', 'andesitic', 'angelican', 'assieging',
'azerty', "bluff's", 'break-dance', 'bubble-boy', 'cellblock', 'cerata',
'crystallites', 'dare-deviltry', 'decoking', 'expiation', 'fantail',
'governmentally', 'gunda', 'homeodomain', 'landau', 'lobsterbacks', 'midnite',
'minoans', 'mouse-mill', 'ninth', 'on-camera', 'picro', 'predilections',
'premiere', 'procambium', 'puromycin', 'radical', 'reptantia', 'sickle-billed',
'snorkeler', 'souke', 'squireship', 'subtlety', 'telopeptides', 'thousands-of',
'thriftlessly', 'timberyards', 'tortion', 'truck-driver', 'undepleted',
'vergil', 'vernalize', 'voltammetry', 'whenceever', 'yandere']
```

1.4 Question 4:

We now want to write a function where if we pass in an alphabet, the function will return (auto-complete) all words that start with that alphabet.

For this function, use a simple for loop. Call the function auto_complete_1. This function will take two inputs, the alphabet entered by the user and the list of words generated above. The function will return as output a list of words that start with that alphabet.

Think about how you will compare the alphabet passed into the function with the FIRST letter of each word in words. Each word in the list words is a string and you simply need to look at the first character of the string. Think about indices.

If the first letter of the word is the same as the passed in alphabet, add the word to the output list that will be returned once you have looped through all the words.

[18]:
```
def auto_complete_1(check, words):
    specific_list = [y for y in words if y[0].lower() == check.lower()] #using␣
↪list comprehension and inbuilt lower function
    print("\nThe list of words which match the input letter : " +␣
↪str(specific_list))
```

```
```

1.5 Question 5:

Call your function by passing in an alphabet of your choice and the list of words generated earlier. Print the list of auto-complete words. You can check your answer by quickly scanning the sorted list of original words you passed into the function. Try this for a few different alphabets and make sure your auto-complete function is working well.

Here is an example of the function output when I pass in the letter 'c' with the list of words above. ['carbonaro', 'chastising', 'cock-feather', 'cuteness', 'comforted', 'counterterror', 'custrel', 'child-proof', 'chelerythrine']

[20]:
```
check = input("\nEnter letter : ") #Entering letter
auto_complete_1(check, words)
```

```

Enter letter : r

The list of words which match the input letter : ['radical', 'reptantia']
```