Python - 4

September 30, 2020

1 Python - 4

1.0.1 Question 1 : Generate 50 random sentences, store them in a list and print the list.

[193]:
```
import random

#storing nouns, verbs and setence completers to make random sentences
nouns_1 = ["Dog", "Office", "Car", "New York", "Author", "Coffee shop", "My␣
 ↪house", "Sunscreen", "This guy", "Friends"]
nouns_2 = ["University", "Blue hoodie", "Cookies", "Park", "Doctor",␣
 ↪"Building", "Card", "Volunteers", "Flowers", "High-school children"]
verb_1 = ["pushes", "cries", "making", "bakes", "makes", "visits", "driving",␣
 ↪"eating", "playing", "curses", "pushing", "drives", "paints", "saw",␣
 ↪"waited", "builds", "creates", "shows", "fakes", "laughed"]
infinitive = ["to see California.", "for her birthday.", "because he hates her.
 ↪", "for all right reasons.", "because he is crazy.",
              "for work.", "due to illness.", "because it was missing.", "to␣
 ↪make lunch.", "for partying."]

#Creating function to make random sentences
def sentence_random():
    return(random.choice(nouns_1) + " " + random.choice(verb_1) + " " +  random.
 ↪choice(nouns_2).lower()) + " " + random.choice(infinitive)
#Random.choice will return the place of the word in random order from a␣
 ↪non-zero sequence

random_list = [] #creating an empty list
#loop to generate and append 50 random sentences
for x in range(50):
    random_list.append(sentence_random())
print("50 random sentences are: ", random_list)
```

```
50 random sentences are:  ['Car cries doctor to make lunch.', 'New York builds
cookies due to illness.', 'Car driving building because he is crazy.', 'Coffee
shop fakes doctor because he is crazy.', 'This guy pushes university for her
birthday.', 'Sunscreen eating park for partying.', 'My house shows doctor
because he is crazy.', 'My house builds blue hoodie because he hates her.',
'Sunscreen playing park because he hates her.', 'Office cries cookies due to
illness.', 'Dog laughed cookies because it was missing.', 'Car makes flowers due
to illness.', 'Office cries volunteers due to illness.', 'This guy drives doctor
to see California.', 'My house builds university due to illness.', 'Sunscreen
makes high-school children because it was missing.', 'This guy eating cookies
for partying.', 'Dog laughed university because he hates her.', 'Office fakes
high-school children because he hates her.', 'Coffee shop builds doctor because
he is crazy.', 'Car pushing blue hoodie because he is crazy.', 'Car laughed
high-school children to see California.', 'Sunscreen drives cookies to see
California.', 'Friends curses cookies for all right reasons.', 'Sunscreen
laughed blue hoodie because he hates her.', 'Car playing flowers due to
illness.', 'Friends fakes flowers because he hates her.', 'Car driving flowers
to make lunch.', 'Dog fakes flowers because he hates her.', 'Coffee shop curses
blue hoodie because he hates her.', 'Friends pushing cookies for her birthday.',
'Dog visits blue hoodie because he hates her.', 'Office saw high-school children
because he is crazy.', 'Car bakes high-school children to make lunch.', 'My
house drives card for partying.', 'Author eating flowers because he is crazy.',
'My house saw doctor because he is crazy.', 'My house visits cookies to make
lunch.', 'New York paints doctor because he hates her.', 'New York pushes
flowers because he is crazy.', 'Author curses cookies to make lunch.', 'Office
waited card for work.', 'Office laughed card because it was missing.', 'Office
playing blue hoodie because he hates her.', 'Sunscreen curses high-school
children for all right reasons.', 'My house builds high-school children for all
right reasons.', 'Office cries blue hoodie for work.', 'Coffee shop creates
university for work.', 'Car eating building to see California.', 'My house
pushing doctor due to illness.']
```

1.0.2 Question 2 : Use the lambda function to split each sentence into its constituent words. Use the space delimeter to split the sentence. The words in each sentence are stored in a separate list i.e. all the words in the first sentence are in one list, the words in the second sentence are in another list and so on. All the lists are stored as elements of one big list. Print this one big list that will have 50 sub-lists, one for each sentence.

[194]:
```python
#Making using of lambda function to split the words in each list and putting␣
↪them into 50 sub-lists
new_words = list(map(lambda n:n.split(), random_list))

print("Lists with individual words are: ", new_words)
```

```
Lists with individual words are:  [['Car', 'cries', 'doctor', 'to', 'make',
'lunch.'], ['New', 'York', 'builds', 'cookies', 'due', 'to', 'illness.'],
['Car', 'driving', 'building', 'because', 'he', 'is', 'crazy.'], ['Coffee',
'shop', 'fakes', 'doctor', 'because', 'he', 'is', 'crazy.'], ['This', 'guy',
'pushes', 'university', 'for', 'her', 'birthday.'], ['Sunscreen', 'eating',
'park', 'for', 'partying.'], ['My', 'house', 'shows', 'doctor', 'because', 'he',
'is', 'crazy.'], ['My', 'house', 'builds', 'blue', 'hoodie', 'because', 'he',
'hates', 'her.'], ['Sunscreen', 'playing', 'park', 'because', 'he', 'hates',
'her.'], ['Office', 'cries', 'cookies', 'due', 'to', 'illness.'], ['Dog',
'laughed', 'cookies', 'because', 'it', 'was', 'missing.'], ['Car', 'makes',
'flowers', 'due', 'to', 'illness.'], ['Office', 'cries', 'volunteers', 'due',
'to', 'illness.'], ['This', 'guy', 'drives', 'doctor', 'to', 'see',
'California.'], ['My', 'house', 'builds', 'university', 'due', 'to',
'illness.'], ['Sunscreen', 'makes', 'high-school', 'children', 'because', 'it',
'was', 'missing.'], ['This', 'guy', 'eating', 'cookies', 'for', 'partying.'],
['Dog', 'laughed', 'university', 'because', 'he', 'hates', 'her.'], ['Office',
'fakes', 'high-school', 'children', 'because', 'he', 'hates', 'her.'],
['Coffee', 'shop', 'builds', 'doctor', 'because', 'he', 'is', 'crazy.'], ['Car',
'pushing', 'blue', 'hoodie', 'because', 'he', 'is', 'crazy.'], ['Car',
'laughed', 'high-school', 'children', 'to', 'see', 'California.'], ['Sunscreen',
'drives', 'cookies', 'to', 'see', 'California.'], ['Friends', 'curses',
'cookies', 'for', 'all', 'right', 'reasons.'], ['Sunscreen', 'laughed', 'blue',
'hoodie', 'because', 'he', 'hates', 'her.'], ['Car', 'playing', 'flowers',
'due', 'to', 'illness.'], ['Friends', 'fakes', 'flowers', 'because', 'he',
'hates', 'her.'], ['Car', 'driving', 'flowers', 'to', 'make', 'lunch.'], ['Dog',
'fakes', 'flowers', 'because', 'he', 'hates', 'her.'], ['Coffee', 'shop',
'curses', 'blue', 'hoodie', 'because', 'he', 'hates', 'her.'], ['Friends',
'pushing', 'cookies', 'for', 'her', 'birthday.'], ['Dog', 'visits', 'blue',
'hoodie', 'because', 'he', 'hates', 'her.'], ['Office', 'saw', 'high-school',
'children', 'because', 'he', 'is', 'crazy.'], ['Car', 'bakes', 'high-school',
'children', 'to', 'make', 'lunch.'], ['My', 'house', 'drives', 'card', 'for',
'partying.'], ['Author', 'eating', 'flowers', 'because', 'he', 'is', 'crazy.'],
['My', 'house', 'saw', 'doctor', 'because', 'he', 'is', 'crazy.'], ['My',
'house', 'visits', 'cookies', 'to', 'make', 'lunch.'], ['New', 'York', 'paints',
'doctor', 'because', 'he', 'hates', 'her.'], ['New', 'York', 'pushes',
'flowers', 'because', 'he', 'is', 'crazy.'], ['Author', 'curses', 'cookies',
'to', 'make', 'lunch.'], ['Office', 'waited', 'card', 'for', 'work.'],
['Office', 'laughed', 'card', 'because', 'it', 'was', 'missing.'], ['Office',
'playing', 'blue', 'hoodie', 'because', 'he', 'hates', 'her.'], ['Sunscreen',
'curses', 'high-school', 'children', 'for', 'all', 'right', 'reasons.'], ['My',
'house', 'builds', 'high-school', 'children', 'for', 'all', 'right',
'reasons.'], ['Office', 'cries', 'blue', 'hoodie', 'for', 'work.'], ['Coffee',
'shop', 'creates', 'university', 'for', 'work.'], ['Car', 'eating', 'building',
'to', 'see', 'California.'], ['My', 'house', 'pushing', 'doctor', 'due', 'to',
'illness.']]
```

1.0.3 Question 3 : Write a mapper function that will create a list of 50 dictionaries. This function will take the one big list you created above as input. Each dictionary represents one sub-list and contains key:value pairs where the key is a word and the value is the number of times the word occured in the sub-list.

[196]:
```
#Built a function which takes input the list we created in the above question␣
↪where each word is written separately into a sub-list
#Function is suppiosed to create a list of dictionary from the above sub list
```

```
#key value pair is being created in the loop which maps the word with the times␣
 ↪it appears in the list
#it iterates through the loop and keeps on adding the number of times a␣
 ↪particular string appears in the sub-list, else returns 1 as the vlaue.
def mapper(new_words):
    dictionary = dict()
    for y in range(len(new_words)):
        if (new_words[y] in dictionary.keys()):
            dictionary[new_words[y]] = dictionary[new_words[y]] + 1
        else:
                dictionary[new_words[y]] = 1
    return dictionary
```

1.0.4 Question 4 : Call the mapper function you wrote above using a map command and pass as input the big list you created in Question 2. Cast the output to a list and print the output. This will be a list of 50 dictionaries as explained in Question 3.

[197]:
```
#Using in-built map function to call the function mapper created above which␣
 ↪makes 50 dictionaries with key-value pairs in each of them
#Creating a list to store the output of 50 dictionaries
individual_count = list(map(mapper,new_words))
print("50 dictionaries with individual word count are: ", individual_count)
```

```
50 dictionaries with individual word count are:  [{'Car': 1, 'cries': 1,
'doctor': 1, 'to': 1, 'make': 1, 'lunch.': 1}, {'New': 1, 'York': 1, 'builds':
1, 'cookies': 1, 'due': 1, 'to': 1, 'illness.': 1}, {'Car': 1, 'driving': 1,
'building': 1, 'because': 1, 'he': 1, 'is': 1, 'crazy.': 1}, {'Coffee': 1,
'shop': 1, 'fakes': 1, 'doctor': 1, 'because': 1, 'he': 1, 'is': 1, 'crazy.':
1}, {'This': 1, 'guy': 1, 'pushes': 1, 'university': 1, 'for': 1, 'her': 1,
'birthday.': 1}, {'Sunscreen': 1, 'eating': 1, 'park': 1, 'for': 1, 'partying.':
1}, {'My': 1, 'house': 1, 'shows': 1, 'doctor': 1, 'because': 1, 'he': 1, 'is':
1, 'crazy.': 1}, {'My': 1, 'house': 1, 'builds': 1, 'blue': 1, 'hoodie': 1,
'because': 1, 'he': 1, 'hates': 1, 'her.': 1}, {'Sunscreen': 1, 'playing': 1,
'park': 1, 'because': 1, 'he': 1, 'hates': 1, 'her.': 1}, {'Office': 1, 'cries':
1, 'cookies': 1, 'due': 1, 'to': 1, 'illness.': 1}, {'Dog': 1, 'laughed': 1,
'cookies': 1, 'because': 1, 'it': 1, 'was': 1, 'missing.': 1}, {'Car': 1,
'makes': 1, 'flowers': 1, 'due': 1, 'to': 1, 'illness.': 1}, {'Office': 1,
'cries': 1, 'volunteers': 1, 'due': 1, 'to': 1, 'illness.': 1}, {'This': 1,
'guy': 1, 'drives': 1, 'doctor': 1, 'to': 1, 'see': 1, 'California.': 1}, {'My':
1, 'house': 1, 'builds': 1, 'university': 1, 'due': 1, 'to': 1, 'illness.': 1},
{'Sunscreen': 1, 'makes': 1, 'high-school': 1, 'children': 1, 'because': 1,
'it': 1, 'was': 1, 'missing.': 1}, {'This': 1, 'guy': 1, 'eating': 1, 'cookies':
1, 'for': 1, 'partying.': 1}, {'Dog': 1, 'laughed': 1, 'university': 1,
'because': 1, 'he': 1, 'hates': 1, 'her.': 1}, {'Office': 1, 'fakes': 1, 'high-
school': 1, 'children': 1, 'because': 1, 'he': 1, 'hates': 1, 'her.': 1},
{'Coffee': 1, 'shop': 1, 'builds': 1, 'doctor': 1, 'because': 1, 'he': 1, 'is':
```

```
1, 'crazy.': 1}, {'Car': 1, 'pushing': 1, 'blue': 1, 'hoodie': 1, 'because': 1,
'he': 1, 'is': 1, 'crazy.': 1}, {'Car': 1, 'laughed': 1, 'high-school': 1,
'children': 1, 'to': 1, 'see': 1, 'California.': 1}, {'Sunscreen': 1, 'drives':
1, 'cookies': 1, 'to': 1, 'see': 1, 'California.': 1}, {'Friends': 1, 'curses':
1, 'cookies': 1, 'for': 1, 'all': 1, 'right': 1, 'reasons.': 1}, {'Sunscreen':
1, 'laughed': 1, 'blue': 1, 'hoodie': 1, 'because': 1, 'he': 1, 'hates': 1,
'her.': 1}, {'Car': 1, 'playing': 1, 'flowers': 1, 'due': 1, 'to': 1,
'illness.': 1}, {'Friends': 1, 'fakes': 1, 'flowers': 1, 'because': 1, 'he': 1,
'hates': 1, 'her.': 1}, {'Car': 1, 'driving': 1, 'flowers': 1, 'to': 1, 'make':
1, 'lunch.': 1}, {'Dog': 1, 'fakes': 1, 'flowers': 1, 'because': 1, 'he': 1,
'hates': 1, 'her.': 1}, {'Coffee': 1, 'shop': 1, 'curses': 1, 'blue': 1,
'hoodie': 1, 'because': 1, 'he': 1, 'hates': 1, 'her.': 1}, {'Friends': 1,
'pushing': 1, 'cookies': 1, 'for': 1, 'her': 1, 'birthday.': 1}, {'Dog': 1,
'visits': 1, 'blue': 1, 'hoodie': 1, 'because': 1, 'he': 1, 'hates': 1, 'her.':
1}, {'Office': 1, 'saw': 1, 'high-school': 1, 'children': 1, 'because': 1, 'he':
1, 'is': 1, 'crazy.': 1}, {'Car': 1, 'bakes': 1, 'high-school': 1, 'children':
1, 'to': 1, 'make': 1, 'lunch.': 1}, {'My': 1, 'house': 1, 'drives': 1, 'card':
1, 'for': 1, 'partying.': 1}, {'Author': 1, 'eating': 1, 'flowers': 1,
'because': 1, 'he': 1, 'is': 1, 'crazy.': 1}, {'My': 1, 'house': 1, 'saw': 1,
'doctor': 1, 'because': 1, 'he': 1, 'is': 1, 'crazy.': 1}, {'My': 1, 'house': 1,
'visits': 1, 'cookies': 1, 'to': 1, 'make': 1, 'lunch.': 1}, {'New': 1, 'York':
1, 'paints': 1, 'doctor': 1, 'because': 1, 'he': 1, 'hates': 1, 'her.': 1},
{'New': 1, 'York': 1, 'pushes': 1, 'flowers': 1, 'because': 1, 'he': 1, 'is': 1,
'crazy.': 1}, {'Author': 1, 'curses': 1, 'cookies': 1, 'to': 1, 'make': 1,
'lunch.': 1}, {'Office': 1, 'waited': 1, 'card': 1, 'for': 1, 'work.': 1},
{'Office': 1, 'laughed': 1, 'card': 1, 'because': 1, 'it': 1, 'was': 1,
'missing.': 1}, {'Office': 1, 'playing': 1, 'blue': 1, 'hoodie': 1, 'because':
1, 'he': 1, 'hates': 1, 'her.': 1}, {'Sunscreen': 1, 'curses': 1, 'high-school':
1, 'children': 1, 'for': 1, 'all': 1, 'right': 1, 'reasons.': 1}, {'My': 1,
'house': 1, 'builds': 1, 'high-school': 1, 'children': 1, 'for': 1, 'all': 1,
'right': 1, 'reasons.': 1}, {'Office': 1, 'cries': 1, 'blue': 1, 'hoodie': 1,
'for': 1, 'work.': 1}, {'Coffee': 1, 'shop': 1, 'creates': 1, 'university': 1,
'for': 1, 'work.': 1}, {'Car': 1, 'eating': 1, 'building': 1, 'to': 1, 'see': 1,
'California.': 1}, {'My': 1, 'house': 1, 'pushing': 1, 'doctor': 1, 'due': 1,
'to': 1, 'illness.': 1}]
```

1.0.5 Question 5 : Write a reducer function to combine all the dictionaries in Question 4 to a single dictionary of key:value pairs where the key is a word and the value is the number of times the word occurs across all the dictionaries.

[198]:
```python
from functools import reduce #importing reduce from the functools library␣
 ↪module
```

[199]:
```python
#defining a reducer function to add all dictionaries into one primary␣
 ↪dictionary
#The primary dictionary has the word as key and the number of times it appears␣
 ↪in all the sub-dictionaries as value
```

```
#The loop in the function iterates through the dictionary used above and keeps␣
 ↪on adding value if the same word repeat
#Or else it prints the value as 1
def reducer(old,new):
    all_dictionary = old
    for key, value in new.items():
        if key in all_dictionary.keys():
            all_dictionary[key] = all_dictionary[key] + new[key]
        else:
            all_dictionary[key] = 1
    return all_dictionary
```

1.0.6 Question 6 : Call the reducer function you wrote in Question 5 using the reduce command and pass as one of the inputs the list of 50 dictionaries you output in Question 4. Print the output. This will be a single dictionary of key:value pairs.

[200]:
```
#calling the reducer function above by using reduce command
#passing the individual_count list in the function to let the function compare␣
 ↪and count to provide one primary dictionary
#dictionary has key values pairs as the number of times a certain word appears␣
 ↪in all of the dictionaries
complete_count = reduce(reducer,individual_count, {})
print("Consolidated complete dictionary: ", complete_count)
```

```
Consolidated complete dictionary:  {'Car': 9, 'cries': 4, 'doctor': 8, 'to': 16,
'make': 5, 'lunch.': 5, 'New': 3, 'York': 3, 'builds': 5, 'cookies': 9, 'due':
7, 'illness.': 7, 'driving': 2, 'building': 2, 'because': 23, 'he': 20, 'is': 9,
'crazy.': 9, 'Coffee': 4, 'shop': 4, 'fakes': 4, 'This': 3, 'guy': 3, 'pushes':
2, 'university': 4, 'for': 11, 'her': 2, 'birthday.': 2, 'Sunscreen': 6,
'eating': 4, 'park': 2, 'partying.': 3, 'My': 8, 'house': 8, 'shows': 1, 'blue':
7, 'hoodie': 7, 'hates': 11, 'her.': 11, 'playing': 3, 'Office': 8, 'Dog': 4,
'laughed': 5, 'it': 3, 'was': 3, 'missing.': 3, 'makes': 2, 'flowers': 7,
'volunteers': 1, 'drives': 3, 'see': 4, 'California.': 4, 'high-school': 7,
'children': 7, 'pushing': 3, 'Friends': 3, 'curses': 4, 'all': 3, 'right': 3,
'reasons.': 3, 'visits': 2, 'saw': 2, 'bakes': 1, 'card': 3, 'Author': 2,
'paints': 1, 'waited': 1, 'work.': 3, 'creates': 1}
```

[201]:
```
all_words_count = complete_count.items()
print(all_words_count)
```

```
dict_items([('Car', 9), ('cries', 4), ('doctor', 8), ('to', 16), ('make', 5),
('lunch.', 5), ('New', 3), ('York', 3), ('builds', 5), ('cookies', 9), ('due',
7), ('illness.', 7), ('driving', 2), ('building', 2), ('because', 23), ('he',
20), ('is', 9), ('crazy.', 9), ('Coffee', 4), ('shop', 4), ('fakes', 4),
('This', 3), ('guy', 3), ('pushes', 2), ('university', 4), ('for', 11), ('her',
2), ('birthday.', 2), ('Sunscreen', 6), ('eating', 4), ('park', 2),
('partying.', 3), ('My', 8), ('house', 8), ('shows', 1), ('blue', 7), ('hoodie',
7), ('hates', 11), ('her.', 11), ('playing', 3), ('Office', 8), ('Dog', 4),
('laughed', 5), ('it', 3), ('was', 3), ('missing.', 3), ('makes', 2),
('flowers', 7), ('volunteers', 1), ('drives', 3), ('see', 4), ('California.',
4), ('high-school', 7), ('children', 7), ('pushing', 3), ('Friends', 3),
('curses', 4), ('all', 3), ('right', 3), ('reasons.', 3), ('visits', 2), ('saw',
2), ('bakes', 1), ('card', 3), ('Author', 2), ('paints', 1), ('waited', 1),
('work.', 3), ('creates', 1)])
```

1.0.7 Question 7 : Write another reducer function that will find the word that occurs the most number of times AND is in the list of nouns you used to randomly generate the sentences. Return the key-value pair for this word.

[202]:
```
#combining both noun lists into one
nouns_1 = ["Dog", "Office", "Car", "New York", "Author", "Coffee shop", "My␣
 ↪house", "Sunscreen", "This guy", "Friends"]
nouns_2 = ["University", "Blue hoodie", "Cookies", "Park", "Doctor",␣
 ↪"Building", "Card", "Volunteers", "Flowers", "High-school children"]
noun_list = []
noun_list.extend(nouns_1)
noun_list.extend(nouns_2)
print("Consolidated noun list: ", noun_list)

#generating Function to check the most occuring word from the above␣
 ↪consolidated noun_list for nouns
def reducer_noun(previous, new):
    if new[1] > previous[1] and new[0] in noun_list: #Condition to check if the␣
 ↪word is in noun list
        return new
    else:
        return previous
```

```
Consolidated noun list:  ['Dog', 'Office', 'Car', 'New York', 'Author', 'Coffee
shop', 'My house', 'Sunscreen', 'This guy', 'Friends', 'University', 'Blue
hoodie', 'Cookies', 'Park', 'Doctor', 'Building', 'Card', 'Volunteers',
'Flowers', 'High-school children']
```

1.0.8 Question 8 : Call the reducer function you wrote in Question 7 using the reduce command and pass as one of the inputs the key:value pairs from the output of Question 6. Remember to pass key:value pairs as shown in the preface. Print the output.

This will be the key:value pair for the word that occured the most number of times across all the sentences and is not one of the excluded words specified in Question 3.

[210]:
```
#
most_appearing_noun = reduce(reducer_noun,all_words_count)
print("Most appearing noun : ", most_appearing_noun)
```

```
Most appearing noun :  ('Car', 9)
```

1.0.9 Question 9 : Print ALL key:value pairs for ALL words that occured the most number of times i.e. not just the first word. For example, if there are 4 words that occured the highest number of times, print all these 4 words. You do not have to use the reduce function to do this.

[209]:
```
#I would prefer doing it with the reduce function only to get the key value␣
↪pair which appears max number of times

def reducer_new(previous,new):
    if new[1] > previous[1]: #Condiiton to check if the word has appeared␣
↪before or not
        return new
    else:
        return previous
```

[208]:
```
max_appearing_word = reduce(reducer_new,all_words_count)
print("Most appearing word: ", max_appearing_word)

#In my dictionary, only the word 'for' appears to be occuring most number of␣
↪times and no other word appears this (24) maximum number of times.
```

```
Most appearing word:  ('because', 23)
```