

Kaelyn Escuadra
December 2, 2021
Creative Coding
Prof. Zach Whalen
Tutorial

Chaotic Poem Generator

1. First, select a text and upload it into the Collab Notebook using the program's temporary file storage tool.

```
with open('/content/text.txt') as f:  
    text = f.read() #for my project, I used 'The Yellow Wallpaper' by  
Charlotte Gilman
```

and then clean it up!

```
text = text.replace("\"", "\\").replace("'", "\'")
```

2. Time to import TextBlob!

TextBlob helps to understand the language within the text to accurately divide and categorize different parts of the writing.

```
from textblob import TextBlob  
import nltk  
nltk.download('punkt')  
nltk.download('averaged_perceptron_tagger')  
nltk.download('brown')
```

Let's also import the random function! This will help us later.

```
import random  
from random import randint
```

3. Next, let's create variables and commands that will then divide and randomize our text.
(*Note: TextBlob won't always be perfect... some textual editing during this step may be required!)

```
blob = TextBlob(text) #this first step will call forth the entire text  
tagged_words = blob.tags #this will divide the text into individual  
words and specify their part of speech  
sentence_list = blob.sentences #this will divide the text into  
sentences
```

```
phrases_list = blob.noun_phrases #this will divide the text into
phrases
```

Let's keep going!

```
sentence = random.choice(blob.sentences) #from the list of sentences we
created, the random function allows a sentence to be arbitrarily
selected
words = sentence.split() #this command will then divide our randomly
selected sentence into individual words
random.shuffle(words) #finally, this will then shuffle all of our words
```

4. Now, with our mixed-up words, we have to put our scrambled sentence back together.

```
resentence = ''.join(words)
```


In my case, I rejoined my sentence by starting a new line after each word, using '\n'.

For a simply shuffled, yet chaotic poem the process could stop here...
To print your chaotic poetry results...

```
print(resentence)
```

5. But let's shuffle some of the words in the sentence as well-- for more chaos. *chef's kiss*.

```
for word in words:
    shuffleTest = randint(0,1)
    drow = ' '
    if (shuffleTest>0.1): #this is how the code knows what words to pick
and shuffle. If the test for a word returns an integer larger than
0.1, the word will be reprinted backwards.
        for i in word:
            drow = i + drow
    print(drow)
    else:
        word #otherwise, according to this line, the word will print
normally
```



6. Finally, we can print our chaotic poem!

```
print(word)
```

Other ways to make your poem more chaotic:

Add color!

```
from termcolor import colored
```

And then—in your print line-- just specify what should be colored and what color!

```
print(colored(word), 'yellow') #I used yellow to go with the theme of the  
'yellow' wallpaper
```

Switch up the punctuation

```
...  
else:  
    word += '!'
```

Different punctuation can be added after certain words to create an even greater sense of chaos!

COMPLETE CODE:

```
with open('/content/the yellow wallpaper.txt') as f:
    text = f.read()

text = text.replace("\"", "\\").replace("'", "\'")

from textblob import TextBlob
import nltk
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')
nltk.download('brown')

import random
from random import randint

blob = TextBlob(text)
tagged_words = blob.tags
sentence_list = blob.sentences
phrases_list = blob.noun_phrases

sentence = random.choice(blob.sentences)
random.shuffle(words)

resentence = '\n'.join(words)

for word in words:
    shuffleTest = randint(0,1)
    drow = ' '
    if (shuffleTest>0.1):
        for i in word:
            drow = i + drow
    print(drow)
    else:
        word
```

Examples with Charlotte Gilbert's *The Yellow Wallpaper*

[illegible]

esoht
those
unblinking
dna
and
dna
and
,drusba
absurd,
they
down
,lwarc
crawl,
and
seye
eyes
pU
Up
are
sideways
.erehwyreve
everywhere.