Random Writer

Overview

Imagine taking a book (say, *Tom Sawyer*) and determining the probability that each character occurs. You would probably find that spaces are the most common, that the character “e” is fairly common, and that the character “q” is rather uncommon. After completing this **level 0** analysis, you would be able to produce random *Tom Sawyer* text based on character probabilities. It wouldn't have much in common with the real thing, but at least the characters would tend to occur in the proper proportion. In fact, here's an example of what you might produce (of 125 characters in length):

Level 0

rla bsht eS ststofo hhfosdsdewno oe wee h .mr ae irii ela iad o r te u t mnyto onmalysnce, ifu en c fDwn oee iteod eehu if uh

Now imagine doing a slightly more sophisticated **level 1** analysis by determining the probability that each character follows every other character. You would probably discover that “h” follows “t” more frequently than “x” does, and you would probably discover that a space follows “.” more frequently than “,” does. You could now produce some randomly generated *Tom Sawyer* text by picking a character to begin with, and then always choosing the next character based on the previous one and the probabilities revealed by the analysis. Here's an example (again, of 125 characters in length):

Level 1

redunisom t romed bututonobutcushowerooledredery sldomeyo, itere l thed cepe hy s. arycror "I n-Ohim ope wny iscad the amitin

Now imagine doing a **level *k*** analysis by determining the probability that each character follows every possible sequence of characters of length *k*. A **level 5** analysis of *Tom Sawyer*, for example, would reveal that “r” follows “Sawye” more frequently than any other character. After a level *k* analysis, you would be able to produce random *Tom Sawyer* text by always choosing the next character based on the previous *k* characters (which we will call the **seed**) and the probabilities revealed by the analysis.

At only a moderate level of analysis (say, levels 5 to 7), the randomly generated text begins to take on many of the characteristics of the source text. It probably won't make complete sense, but you will be able to tell that it was derived from *Tom Sawyer* as opposed to, say, *The Sound and the Fury*. Here are some more examples:

Level 2

ee twit ot hathim--yarded, wit don of th aftlegaididoget, gete-throce put awake ow awaslof uslances!' so the se mok malonow t

Level 4

ry more was out the was he did no while her face, and a moment the never would him questive it. They're about a shows if she

Level 6

topped along twelve miles belongs to the foliage for what he did it would have it up. Lie down on a woman is of the child, yo

Level 8

this rock, but nothing. But she never would hang round in the distance in his book still live, and saw it grow smooth and nat

Level 10

that would hurt. So he thought filled him with bodings. Within five minutes longer. He entered the kitchen presently--the ve

Level 25

he had finished and still stood alive and whole, their wavering impulse to break their oath and save the poor betrayed prison

Specification

Write a Python program that takes the following command line arguments:

- A non-negative integer *k*;
- A non-negative integer *length*; and
- The name of an input file that contains the source text.

You may assume that the arguments provided will be valid (e.g., the input file exists, it contains more than *k* characters, etc).

Your program should then pick *k* consecutive characters at random from the source text and use them as the initial **seed** in the random writing process outlined above. Your program should then write *length* characters as output. Each of these additional characters should be chosen based on the current seed. Note that, each time a character, *c*, is outputted, the seed is updated by removing its first character and appending *c* to the end.

For example, suppose that *k* = 2, and the source file contains the following text:

```
the three pirates charted that course the other day
```

Here is how the first three characters might be chosen:

(1) A two-character seed is chosen at random to become the initial seed. Let's suppose that “th” is chosen.

(2) The first character must be chosen based on the probability that it follows the seed (currently “th”) in the source text. The source text contains five occurrences of “th”. Three times it is followed by “e”, once it is followed by “r”, and once it is followed by “a”. Thus, the next character must be chosen so that there is a 3/5 (60%) chance that an “e” will be chosen, a 1/5 (20%) chance that an “r” will be chosen, and a 1/5 (20%) chance that an “a” will be chosen. Let's suppose that we choose an “e” this time.

(3) The next character must be chosen based on the probability that it follows the seed (currently “he”) in the source text. The source text contains three occurrences of “he”. Twice it is followed by a space, and once it is followed by “r”. Thus, the next character must be chosen so that there is a 2/3 (67%) chance that a space will be chosen, and a 1/3 (33%) chance that an “r” will be chosen. Let's suppose that we choose an “r” this time.

(4) The next character must be chosen based on the probability that it follows the seed (currently “er”) in the source text. The source text contains only one occurrence of “er”, and it is followed by a space. Thus, the next character must be a space.

(5) If your program ever gets into a situation in which there are no characters to choose from (which can happen if the only occurrence of the current seed is at the exact end of the source), your program should pick a new random seed and continue.

Your program should continue this process until *length* characters have been outputted.

Constraints and notes

Note the following constraints and notes (see the rubric below for more detail):

- You must use good coding style (which includes things such as including an informative header at the top of your program, commenting your source code appropriately, using meaningful variable and (if applicable) constant identifiers, and so on);
- Use functions wisely, especially for tasks that your program does repeatedly;
- You should use Python’s **random** library;
- If you wish, you may obtain the source text from *stdin* (instead of from a file); and
- Display your output to `stdout` (i.e., simply using the `print()` function).

Deliverable

Submit a Python 3 source file that can be executed through an IDE (such as Thonny) or via the command line (e.g., `python3 RandomWriter.py k length infile` **or** `python3 RandomWriter.py k length < infile`).

Here's an example with actual parameters that uses the first method:

```
python3 RandomWriter 10 1500 time-machine.txt
```

Here's an example with actual parameters that uses the second (preferred) method:

```
python3 RandomWriter 10 1500 < time-machine.txt
```

Sample output

Here is sample output of the following command:

```
python3 RandomWriter.py 25 500 the-adventures-of-tom-sawyer.txt
```

```
 death, is that my fault? I'll not cry, if
she does. My friend, you'll help me in this thing--for _my_
sake--that's
why you're here--I mightn't be able alone. If you flinch, I'll
kill you.
Do you understand that? And if I have to kill you, I'll kill
her--and
then I reckon nobody'll ever know much about who done this
business."

"Well, if it's got to be done, let's get at it. The quicker the
better--I'm all in a shiver."

"Do it _now_? And company there? Look here--I'll get suspicious
of you,
fir
```

Hints

Here are some hints to help get you started:

- Of course, feel free to Google things and/or discuss high-level ideas with your classmates. As always, do not Google outright solutions of core components and make sure to **cite sources** (including your classmates).
- There is a simple way to attack the problem of randomly selecting the next character that should follow the current seed. First, store the entire source text as a string. For this, check out Python's `read()` function. To choose the next character, find each occurrence of the seed in the string and store the character that follows it into a Python list. When you have found all of the occurrences, choose a character at random from the list!
- Use array slices to recalculate the seed after having randomly selected the next character (i.e., remove the first character from the current seed and append the next character to the end of the current seed – generating the next seed).
- Lastly, Project Gutenberg[1] maintains a huge library of public domain books that you can use as source texts. It contains over 60,000 free eBooks. Note that you will want to use the **text** versions for your program.

Rubric

Please note the following rubric for this programming assignment:

Random Writer		
#	**Item**	**Points**
1	Good coding style	5
2	Functions used wisely	3
3	Random library used	1
4	Input properly obtained	2
5	Output properly generated	3
6	Output length is correct	1
7	Level analysis properly performed	10
TOTAL		**25**

1 See https://www.gutenberg.org/.