Nonsense

In 1996, Alan Sokal - a professor of physics at the University of New York - conducted the following somewhat unethical experiment. He sent a hoax, peppered with nonsenseical reasoning and pseudoscientific jargon to the American journal Social Text that is published by Duke University. Through this experiment Sokal wanted to know whether a well made, but completely nonsenseical article could be published in a postmodern journal, if it sounded good and hit the editors with ideological but hollow concepts. The article was indeed published and caused a lot of outrage in the international academic world.

The Sokal affair raises the question whether it is possible to generate random nonsenseical language with a computer, which seems legitimate at first glance. Natural language is highly structured and not randomly built at all: not every set of letters represents a word, and not every set of words constitutes a meaningful and grammatically correct sentence. Still, certain aspects of randomly generated phrases sound familiar, if they are sufficiently structured. In addition, each author has his own pattern in selecting the order of words. By preserving the order of adjacent words in a text, but constructing the rest of the text at random, it is possible to generate a surprisingly good appearing text that largely preserves the style (but not the meaning) of the original text.

Assignment

1. Write a function `wordlist` with which a list of words can be read from a text file. The location of the text file has to be passed to the function as an argument. The function must return a list of words in the order in which they appear in the text file as a result. A word is created in the text file by the longest possible sequence of characters in which no whitespace characters (spaces, tabs, *carriage returns*, and *newlines*) occur. The words in the text file are separated from each other by one or more consecutive white space characters. Punctuation marks which might stick to a word, are part of that word.
2. Write a function `sequelwords` with two parameters: a parameter `words` to which a list of words must be passed, and a parameter `k` to which an integer ($k \geq 1$) must be passed. The function must return a dictionary, of which the keys are formed by all tuples of k words that occur consecutively in the given word list, and after which there is at least one more word. The dictionary displays each of these k-tuples on the list of words that occur in the given word list after the succession of k words, and this in the sequence in which these words occur in the given word list. If, for example, a key `('aa', 'bb', 'cc')` occurs in the dictionary with the matching value `['dd', 'ee', 'dd', 'ff']`, then this means that the given word list contains the words `aa`, `bb` and `cc` four times in a row, and that these occurrences are respectively followed by the words `dd`, `ee`, `dd` and `ff`. Note that the same word can follow the same k-tuple multiple times, causing it to occur multiple times in the list on which this k-tuple is displayed by the dictionary.
3. Write a function `nonsense` that returns a randomly generated text. This function has the following three parameters: a `start` parameter to which a tuple of words must be passed, a `sequel` parameter to which a dictionary must be passed which is constructed like the dictionaries that are returned by the `sequelwords` function, and a parameter `minimumlength` to which an integer must be passed. The random text is to be generated by the function in the following way:
    a. The words of the tuple `start` form the first words of the text, and are then separated from

each other by a single space.

b. Use the dictionary sequel to determine the next word of the text, by choosing any word from the list that is displayed by the dictionary on the start tuple. This new word is added at the end of the text, preceded by a single space.
c. Calculate a new value for the tuple start by leaving the first word out from the old value of the tuple start, and adding the the new word to back of the tuple that was found in step b.
d. Keep repeating this procedure from step b until
   - tuple start is not a key of the dictionary sequel, or
   - the generated text consists of at least minimumlength words and the last word ends with a period (.), a question mark (?) or an exclamation mark (!), or
   - the generated text consists of at least twice as many words as indicated by the minimumlength parameter.

Example

In the following example we assume that the file shelovesyou.txt with the text from the song *She Loves You* by the Beatles is in the current folder. Click here to view the lyrics of this song. Note that the output of the example was partially omitted to save space, and that the output of the function nonsense was written over multiple lines as not to make the text unnecessarily wide.

```
>>> words = wordlist('shelovesyou.txt')
>>> words
['She', 'loves', 'you,', 'yeh,', 'yeh,', ..., 'yeh;', 'yeh,', 'yeh,', 'yeeeh!']

>>> k = 3
>>> start = tuple(words[:k])
>>> start
('She', 'loves', 'you,')
>>> continuation = nextwords(words, k)
>>> continuation
{
  ('yeh,', 'yeh.', 'She'): ['loves', 'loves'],
  ('should', 'be', 'glad.'): ['Ooh!', 'Ooh!', 'And', 'Ooh!', 'And', 'And'],
  ('you', 'shouuuld', 'be'): ['glad.'],
  ("It's", 'you', "she's"): ['thinking'],
  ...,
  ('like', 'that,', 'you'): ['know', 'know', 'know', 'know']
}

>>> nonsense(start, continuation, 25)
She loves you, yeh, yeh, yeh, yeeeh! You think you lost your
love, when I saw her yesterday. It's you she's thinking of,
and she told me what to say.

>>> nonsense(start, continuation, 25)
She loves you, yeh, yeh, yeh. She loves you, yeh, yeh, yeh.
She loves you, yeh, yeh, yeh. She loves you, yeh, yeh, yeh!
She loves you, yeh, yeh, yeh.

>>> nonsense(start, continuation, 25)
She loves you, yeh, yeh, yeh. She loves you, yeh, yeh, yeh!
And with a love like that, you know you should be glad. And
now it's up to you, I think it's only fair, if I should hurt
you too, apologize to her, because she loves you, and you
```

```
>>> nonsense(start, continuation, 25)
She loves you, yeh, yeh, yeh! And with a love like that, you
know you shouuuld be glad. Yeh, yeh, yeh; yeh, yeh, yeh;
yeh, yeh, yeh; yeh, yeh, yeh; yeh, yeh, yeh; yeh, yeh, yeh;
yeh, yeh, yeh; yeh, yeh, yeh; yeh, yeh, yeh; yeh, yeh, yeh;
yeh, yeh,
```

The following text files can be used to further test your solution. The first of these is also used by us to test your solution automatically.

- United States Declaration of Independence
- The Wonderful Wizard of Oz (L. Frank Baum)
- The Book of Psalms
- Sherlock Holmes: A Scandal in Bohemia (Arthur Conan Doyle)

Scientific Background

The theme of this task is more of a playfulness to familiarize you with reading information from files and learning to work with dictionaries in Python. Still, it is interesting to note that similar concepts are used in major scientific problems which use Markov chains to model probabilistic processes. In the study of genome sequences, there is, for example, a lot of evidence that indicates that DNA sequences contain higher-order structures (Karlin et al., 1998). This means that it is not sufficient to describe the frequencies of the nucleotides A, C, G and T in a genome, but also the correlations between longer chains of nucleotides. For example, the use of codons (triplets of nucleotides) to encode the amino acids from which proteins are constructed, already indicates that there are important three-letter correlations to be found in a genomic sequence. As for natural language, this task will hopefully have already made it clear that three-word correlations can help to display the characteristic properties of the language.

Karlin S, Campbell AM, Mrázek J (1998). Comparative DNA analysis across diverse genomes. *Annu Rev Genet* **32**, 185-225.

In 1996 voerde Alan Sokal — hoogleraar in de natuurkunde aan de universiteit van New York — het volgende ietwat onethische experiment uit. Hij stuurde een nepartikel, doorspekt met onzinnige redeneringen en pseudowetenschappelijk jargon, naar het Amerikaanse tijdschrift *Social Text* dat wordt uitgegeven door Duke University. Met dit experiment wilde Sokal te weten komen of een goed gemaakt, maar compleet onzinnig artikel zou kunnen gepubliceerd worden in een postmodern tijdschrift, als het maar goed zou klinken en de redactieleden om de oren zou slaan met ideologische maar holle concepten. Het artikel werd inderdaad gepubliceerd en veroorzaakte heel wat verontwaardiging in de internationale academische wereld.

Deze Sokal-affaire doet de vraag rijzen of het mogelijk is om een computer willekeurige nonsensicale taal te laten genereren, die op het eerste gezicht toch welluidend is. Natuurlijke taal is immers zeer gestructureerd en totaal niet willekeurig opgebouwd: niet elke reeks letters vormt een woord, en niet elke reeks woorden vormt een betekenisvolle en grammaticaal correcte zin. Toch klinken bepaalde aspecten van willekeurig gegenereerde zinnen vertrouwd in de oren, als ze maar voldoende gestructureerd zijn opgebouwd. Daarbij komt nog dat elke auteur een eigen patroon heeft om woorden te kiezen. Door de volgorde van naburige woorden in een tekst te behouden maar voor de rest de tekst willekeurig op te bouwen, is het mogelijk om willekeurige tekst te genereren die er verrassend goed uitziet en die grotendeels de stijl (maar niet de

betekenis) van de originele tekst behoudt.

Opgave

1. Schrijf een functie woordenlijst waarmee een lijst van woorden kan uitgelezen worden uit een tekstbestand. De locatie van het tekstbestand moet als argument aan de functie doorgegeven worden. De functie moet als resultaat een lijst van de woorden teruggeven, in de volgorde waarin ze in het tekstbestand voorkomen. Een woord wordt in het tekstbestand gevormd door de langst mogelijke reeks karakters waarin geen witruimtekarakters (spaties, tabs, *carriage returns* en *newlines*) voorkomen. De woorden worden in het tekstbestand dus van elkaar gescheiden door één of meer opeenvolgende witruimtekarakters. Leestekens die eventueel aan een woord zouden vasthangen, maken dus deel uit van dat woord.
2. Schrijf een functie vervolgwoorden met twee parameters: een parameter woorden waaraan een lijst van woorden moet doorgegeven worden, en een parameter k waaraan een natuurlijk getal ($k \geq 1$) moet doorgegeven worden. De functie moet als resultaat een dictionary teruggeven, waarvan de sleutels gevormd worden door alle tuples van k woorden die opeenvolgend in de opgegeven woordenlijst voorkomen, en waarna minstens nog één woord staat. De dictionary beeldt elk van deze k-tuples af op de lijst van woorden die in de gegeven woordenlijst voorkomen na deze opeenvolging van k woorden, en dit in de volgorde waarin deze woorden voorkomen in de gegeven woordenlijst. Als in de dictionary bijvoorbeeld een sleutel ('aa', 'bb', 'cc') voorkomt met bijhorende waarde ['dd', 'ee', 'dd', 'ff'], dan betekent dit dat in de gegeven woordenlijst de woorden aa, bb en cc vier keer na elkaar voorkomen, en dat deze voorkomens respectievelijk gevolgd worden door de woorden dd, ee, dd en ff. Merk dus op dat hetzelfde woord meerdere keren kan volgen op hetzelfde k-tuple, waardoor het ook meerdere keren voorkomt in de lijst waarop dit k-tuple wordt afgebeeld door de dictionary.
3. Schrijf een functie nonsens die een willekeurig gegenereerde tekst als resultaat teruggeeft. Deze functie heeft de volgende drie parameters: een parameter start waaraan een tuple van woorden moet doorgegeven worden, een parameter vervolg waaraan een dictionary moet doorgegeven worden die opgebouwd is zoals de dictionaries die door de functie vervolgwoorden teruggegeven worden, en een parameter minimumlengte waaraan een natuurlijk getal moet doorgegeven worden. De willekeurige tekst moet door de functie op de volgende manier gegenereerd worden:
    a. De woorden van het tuple start vormen de eerste woorden van de tekst, en worden daarbij telkens van elkaar gescheiden door één enkele spatie.
    b. Gebruik de dictionary vervolg om het volgende woord van de tekst te bepalen, door een willekeurig woord te kiezen uit de lijst die door de dictionary wordt afgebeeld op het tuple start. Dit nieuwe woord wordt achteraan de tekst toegevoegd, voorafgegaan door één enkele spatie.
    c. Bereken een nieuwe waarde voor het tuple start, door het eerste woord weg te laten uit de oude waarde van het tuple start, en achteraan het tuple het nieuwe woord toe te voegen dat werd gevonden in stap b.
    d. Blijf deze procedure herhalen vanaf stap b, totdat
        - tuple start geen sleutel is van de dictionary vervolg, of
        - de gegeneerde tekst bestaat uit minstens minimumlengte woorden en het laatste woord eindigt op een punt (.), een vraagteken (?) of een uitroepteken (!), of
        - de gegenereerde tekst bestaat uit minstens twee keer zoveel woorden als

aangegeven door de parameter minimumlengte.

Voorbeeld

Onderstaand voorbeeld veronderstelt dat het tekstbestand shelovesyou.txt met de tekst van het nummer *She Loves You* van de Beatles zich in de huidige map bevindt. Klik hier om de tekst van dit nummer te bekijken. Merk op dat de uitvoer van het voorbeeld gedeeltelijk werd weggelaten om plaats te besparen, en dat de uitvoer van de functie nonsens over verschillende regels werd uitgeschreven om de tekst niet onnodig breed te maken.

```
>>> woorden = woordenlijst('shelovesyou.txt')
>>> woorden
['She', 'loves', 'you,', 'yeh,', 'yeh,', ..., 'yeh;', 'yeh,', 'yeh,', 'yeeeh!']

>>> k = 3
>>> start = tuple(woorden[:k])
>>> start
('She', 'loves', 'you,')
>>> vervolg = vervolgwoorden(woorden, k)
>>> vervolg
{
  ('yeh,', 'yeh.', 'She'): ['loves', 'loves'],
  ('should', 'be', 'glad.'): ['Ooh!', 'Ooh!', 'And', 'Ooh!', 'And', 'And'],
  ('you', 'shouuuld', 'be'): ['glad.'],
  ("It's", 'you', "she's"): ['thinking'],
  ...,
  ('like', 'that,', 'you'): ['know', 'know', 'know', 'know']
}

>>> nonsens(start, vervolg, 25)
She loves you, yeh, yeh, yeh, yeeeh! You think you lost your
love, when I saw her yesterday. It's you she's thinking of,
and she told me what to say.

>>> nonsens(start, vervolg, 25)
She loves you, yeh, yeh, yeh. She loves you, yeh, yeh, yeh.
She loves you, yeh, yeh, yeh. She loves you, yeh, yeh, yeh!
She loves you, yeh, yeh, yeh.

>>> nonsens(start, vervolg, 25)
She loves you, yeh, yeh, yeh. She loves you, yeh, yeh, yeh!
And with a love like that, you know you should be glad. And
now it's up to you, I think it's only fair, if I should hurt
you too, apologize to her, because she loves you, and you

>>> nonsens(start, vervolg, 25)
She loves you, yeh, yeh, yeh! And with a love like that, you
know you shouuuld be glad. Yeh, yeh, yeh; yeh, yeh, yeh;
yeh, yeh, yeh; yeh, yeh, yeh; yeh, yeh, yeh; yeh, yeh, yeh;
yeh, yeh, yeh; yeh, yeh, yeh; yeh, yeh, yeh; yeh, yeh, yeh;
yeh, yeh,
```

De volgende tekstbestanden kunnen gebruikt worden om je oplossing verder uit te testen. De eerste daarvan wordt ook door ons gebruikt om je oplossing automatisch te testen.

- United States Declaration of Independence
- The Wonderful Wizard of Oz (L. Frank Baum)

- The Book of Psalms
- Sherlock Holmes: A Scandal in Bohemia (Arthur Conan Doyle)

Wetenschappelijke achtergrond

Het thema van deze opgave is eerder een spielerei om je vertrouwd te maken met het inlezen van informatie uit bestanden en het leren werken met dictionaries in Python. Toch is het interessant om weten dat gelijkaardige concepten gebruikt worden bij belangrijke wetenschappelijke problemen die gebruik maken van Markov ketens om probabilistische processen te modelleren. Bij de studie van genoomsequenties is er bijvoorbeeld heel wat bewijsmateriaal dat aangeeft dat DNA sequenties hogere-orde structuren bevatten (Karlin *et al.*, 1998). Dit betekent dat het niet voldoende is om de frequenties van de nucleotiden `A`, `C`, `G` en `T` te beschrijven in een genoom, maar ook de correlaties tussen langere reeksen nucleotiden. Het gebruik van codons (tripletten van nucleotiden) om de aminozuren waaruit eiwitten zijn opgebouwd te coderen, geeft bijvoorbeeld reeds aan dat er belangrijke drie-letter correlaties te vinden zijn in een genoomsequentie. Voor wat natuurlijke taal betreft, zal deze opgave je hopelijk reeds intuitief hebben doen aanvoelen dat drie-woord correlaties kunnen helpen om karakteristieke eigenschappen van de taal weer te geven.

Karlin S, Campbell AM, Mrázek J (1998). Comparative DNA analysis across diverse genomes. *Annu Rev Genet* **32**, 185-225.