Osmuthalluthexium

You have set up a laboratory experiment where you make use of a particle accelerator for bombarding bismuth atoms with zinc. The atomic collisions generated by the experiment result in a large number of chemical elements. Since all of these elements remained unknown until today, this discovery will undoubtedly yield the Nobel Prize in Chemistry. In order to publish your results, you still need to come up with new names for each of the new elements.

Because you want to assign names that are consonant to the names of existing chemical elements, you decide to proceed in the following way. Chemical names are written with a capital letter, followed by two or more lowercase letters. So you stick to this usage of uppercase and lowercase letters. To recognize suffixes of element names, you (temporarily) append an underscore (_) to the names of the existing chemical elements. Then, you proceed as follows:

1. Randomly pick a name of an existing chemical element, and take the first three letters of that name as the initial letters of the new name you are going to construct.
2. Take the last two letters of the provisional new name, and search the names of the existing chemical elements for all possible characters (lowercase letters or underscores) that follow this bi-gram. Randomly pick a character from the list of candidates, and append it to the provisional new name.
3. If you have chosen an underscore during step 2, the new name may be considered to be complete. In that case, the underscore of course must be removed from the name. Keep on repeating step 2 until an underscore was chosen.

If the above procedure yields a name that was already assigned to an existing chemical element, you simply repeat the whole process until it produces a new name.

Assignment

Define a class NameGenerator that can be used to generate new names based on a sequence of example names, following the procedure outlined in the introduction. The objects of this class should at least have the following properties and methods:

- Each object of the class NameGenerator must be guaranteed to have the properties prefixes and triples. Upon creation of a new object these properties must respectively reference an empty set and an empty dictionary. The following two methods will be used to modify the content of these properties.
- A method addName that can be used to add a new example name to the generator. This example name must be passed as an argument to the generator. The method must add a string containing the first three letters of the example name to the set of prefixes (property prefixes), and must update the dictionary of triples (property triples). The latter is done by looking up each successive pair of lowercase letters as a key in the dictionary, and adding the letter following the pair in the example name (or an underscore for the final pair of letters) to the set that is mapped to the key by the dictionary. If the pair of letters did not yet occur as a key in the dictionary, a new key/value pair must be added to the dictionary, with the value being a set containing the letter following the pair of letters in the example name. The method must raise an AssertionError with the message invalid name if the example name that is passed as an argument does not exists of a capital letter followed by two or more lowercase letters.
- A method addNames that takes the location of a text file as an argument. This text file must contain a list of names, each on a separate line. The method must use each of these names to update the properties prefixes and triples, following the description given for the method addNames.
- A method name that takes no arguments. This method should return a new name that was generated following the procedure outlined in the introduction. Of course, step 1 of the procedure should make use of the property prefixes, and step 2 should make use of the property triples. The example names that were passed when calling the method addNames or that are contained in a text file that was passed when calling the method addNames, are not considered to be new names and thus are not allowed to be returned by this method.

In implementing these methods you should make sure to make optimal reuse of the methods that have already been implemented.

Example

In the following interactive session we assume that the file shortlist_elements.txt is located in the current directory.

```
>>> chemGen = NameGenerator()

>>> chemGen.addName('Osmium')
>>> chemGen.prefixes
{'Osm'}
>>> chemGen.triples
{'um': {'_'}, 'iu': {'m'}, 'mi': {'u'}, 'sm': {'i'}}

>>> chemGen.addName('bismut')
Traceback (most recent call last):
AssertionError: invalid name
>>> chemGen.addName('zINC')
Traceback (most recent call last):
AssertionError: invalid name
>>> chemGen.addName('pH')
Traceback (most recent call last):
AssertionError: invalid name

>>> chemGen.addName('Bismuth')
>>> chemGen.prefixes
{'Osm', 'Bis'}
>>> chemGen.triples
{'mu': {'t'}, 'mi': {'u'}, 'is': {'m'}, 'iu': {'m'}, 'um': {'_'}, 'sm': {'i', 'u'}, 'ut': {'_'}}

>>> chemGen.addNames('shortlist_elements.txt')
>>> chemGen.prefixes
{'Lan', 'Tel', 'Unu', 'Plu', 'Osm', 'Rut', 'Bis', 'Tha'}
>>> chemGen.triples
{'el': {'l'}, 'en': {'i'}, 'is': {'m'}, 'iu': {'m'}, 'al': {'l'}, 'an': {'i', 't'}, 'xi': {'u'}, 'ex': {'i'}, 'er': {'f'}, 'nh': {'e'}, 'ni': {'u'}, 'll': {'i', 'u'}, 'di': {'u'}, 'li': {'u'}, 'rd': {'i'}, 'to': {'n'}, 'rf': {'o'}, 'lu': {'u', 't'}, 'th': {'a', 'e'}, 'fo': {'r'}, 'nt': {'h'}, 'nu':

>>> chemGen.name()
'Osmuthalluthexium'
>>> chemGen.name()
'Ruthanthanium'
>>> chemGen.name()
'Lantherfordium'
>>> chemGen.name()
'Thanthenium'
```

References

- **Grant B (2013).** Researchers discover new element. *The Scientist.*

Je hebt een laboratoriumexperiment opgezet waarbij je bismutatomen aan de hand van een deeltjesversneller bombardeert met zink. Met dit experiment ben je erin geslaagd om een groot aantal nieuwe chemische elementen te maken. Aangezien deze elementen tot op vandaag nog onbekend bleven, zal de ontdekking je ongetwijfeld de Nobelprijs voor de Scheikunde opleveren. Om je resultaten te kunnen publiceren, moet je echter voor alle elementen nog een nieuwe naam zien te verzinnen.

Omdat je namen wilt geven die analoog zijn aan de namen voor bestaande chemische elementen, besluit je om op de volgende manier te werk te gaan. Chemische namen worden geschreven met een hoofdletter, gevolgd door twee of meer kleine letters. Dus hou je ook vast aan dit gebruik van hoofdletters en kleine letters. Om de uitgangen van elementnamen te herkennen, voeg je achteraan de namen van alle bestaande chemische elementen (tijdelijk) een underscore (_) toe. Voor de rest ga je als volgt te werk:

1. Kies willekeurig een naam van een bestaand chemisch element, en neem de eerste drie letters daarvan als beginletters voor de nieuwe naam die je gaat construeren.
2. Neem de laatste twee letters van de voorlopige nieuwe naam, en zoek binnen de namen van de bestaande chemische elementen naar alle mogelijke karakters (kleine letters of een underscore) die volgen op dit letterpaar. Kies een willekeurige karakter uit deze lijst van mogelijkheden, en voeg het achteraan toe aan de nieuwe naam.
3. Als je in stap 2 een underscore hebt gekozen, dan beschouw je de nieuwe naam als afgewerkt. Je verwijdert uiteraard de underscore op het einde van de naam. Zolang je echter in stap 2 een letter kiest, blijf je stap 2 herhalen.

Als bovenstaande procedure een naam oplevert die reeds bestaat, dan herhaal je de procedure totdat ze een nieuwe naam oplevert.

Opgave

Definieer een klasse `NaamGenerator` die kan gebruikt worden om volgens de hierboven beschreven procedure nieuwe namen te genereren op basis van een reeks voorbeeldnamen. De objecten van deze klasse moeten minstens over de volgende eigenschappen en methoden beschikken:

- Elk object van de klasse `NaamGenerator` moet gegarandeerd de eigenschappen `prefixen` en `triples` hebben. Bij het aanmaken van een nieuw object moeten deze eigenschappen respectievelijk verwijzen naar een lege verzameling en een lege dictionary. De volgende twee methoden worden gebruikt om de inhoud van deze eigenschappen te wijzigen.
- Een methode `naamToevoegen` waarmee een nieuwe voorbeeldnaam aan de generator kan toegevoegd worden. Deze voorbeeldnaam moet als stringargument aan de methode doorgegeven worden. De methode moet ervoor zorgen dat een string met de eerste drie letters van de voorbeeldnaam aan de verzameling van prefixen (eigenschap `prefixen`) toegevoegd wordt, en dat ook de dictionary van triples (eigenschap `triples`) wordt bijgewerkt. Dit laatste gebeurt door elk paar opeenvolgende kleine letters van de voorbeeldnaam op te zoeken als sleutel in de dictionary, en de daaropvolgende letter van de voorbeeldnaam (of een underscore voor het laatste letterpaar) toe te voegen aan de verzameling die door de dictionary op die sleutel wordt afgebeeld. Indien het letterpaar nog niet als sleutel voorkwam in de dictionary, dan moet een nieuw sleutel/waarde paar aangemaakt worden, met als waarde een verzameling die de letter bevat die volgt op het letterpaar. De methode moet een `AssertionError` met de tekst `ongeldige naam` opwerpen indien de voorbeeldnaam die als argument wordt doorgegeven niet bestaat uit een hoofdletter, gevolgd door twee of meer kleine letters.
- Een methode `namenToevoegen` waaraan de locatie van een tekstbestand als argument moet doorgegeven worden. Dit tekstbestand moet een lijst van namen bevatten, elk op een afzonderlijke regel. De methode moet ervoor zorgen dat elk van deze namen gebruikt wordt om de eigenschappen `prefixen` en `triples` bij te werken, conform de beschrijving die we hebben gegeven bij de methode `naamToevoegen`.
- Een methode `naam` waaraan geen argumenten moeten doorgegeven worden. Deze methode moet een nieuwe naam teruggeven die werd gegenereerd op basis van de procedure die in de inleiding werd besproken. Uiteraard wordt hierbij in stap 1 gebruik gemaakt van de eigenschap `prefixen`, en in stap 2 van de eigenschap `triples`. De voorbeeldnamen die werden doorgegeven bij het aanroepen van de methode `naamToevoegen` of die in een tekstbestand voorkwamen dat werd opgegeven bij het aanroepen van de methode `namenToevoegen`, worden niet als nieuwe namen beschouwd en mogen dus ook niet door de methode teruggeven worden.

Zorg er bij de implementatie van al deze methoden voor dat je optimaal gebruik maakt van de methoden die je reeds eerder geïmplementeerd hebt.

Voorbeeld

Bij onderstaande voorbeeldsessie gaan we ervan uit dat het bestand shortlist_elementen.txt zich in de huidige directory bevindt.

```
>>> chemGen = NaamGenerator()

>>> chemGen.naamToevoegen('Osmium')
>>> chemGen.prefixen
{'Osm'}
>>> chemGen.triples
{'um': {'_'}, 'iu': {'m'}, 'mi': {'u'}, 'sm': {'i'}}

>>> chemGen.naamToevoegen('bismut')
Traceback (most recent call last):
AssertionError: ongeldige naam
>>> chemGen.naamToevoegen('zINC')
Traceback (most recent call last):
AssertionError: ongeldige naam
>>> chemGen.naamToevoegen('pH')
Traceback (most recent call last):
AssertionError: ongeldige naam

>>> chemGen.naamToevoegen('Bismut')
>>> chemGen.prefixen
{'Osm', 'Bis'}
>>> chemGen.triples
{'mu': {'t'}, 'mi': {'u'}, 'is': {'m'}, 'iu': {'m'}, 'um': {'_'}, 'sm': {'i', 'u'}, 'ut': {'_'}}

>>> chemGen.namenToevoegen('shortlist_elementen.txt')
>>> chemGen.prefixen
{'Lan', 'Tel', 'Unu', 'Plu', 'Osm', 'Rut', 'Bis', 'Tha'}
>>> chemGen.triples
{'el': {'l'}, 'en': {'i'}, 'is': {'m'}, 'iu': {'m'}, 'al': {'l'}, 'an': {'i', 't'}, 'xi': {'u'}, 'ex': {'i'}, 'er': {'f'}, 'nh': {'e'}, 'ni': {'u'}, 'll': {'i', 'u'}, 'di': {'u'}, 'li': {'u'}, 'rd': {'i'}, 'to': {'n'}, 'rf': {'o'}, 'lu': {'u', 't'}, 'th': {'a', 'e'}, 'fo': {'r'}, 'nt': {'h'}, 'nu':
```

```
>>> chemGen.naam()
'Osmuthalluthexium'
>>> chemGen.naam()
'Ruthanthanium'
>>> chemGen.naam()
'Lantherfordium'
>>> chemGen.naam()
'Thanthenium'
```

Bronnen

- **Grant B (2013).** Researchers discover new element. *The Scientist.*