Mad Libs

Mad Libs is word game that starts from a quote or a short story where some keywords have been replaced with blanks. Beneath each blank is specified a lexical or other category, such as "*noun*", "*verb*", "*place*", or "*part of the body*". One player then asks the other players, in turn, to suggest a word for the category specified under the next blank, but without revealing the context for that word. Finally, the filled quote or the completed story is read aloud. The result is usually comic, surreal and somewhat nonsensical.

Say we start for example from the following partial quote.

_____ created _____ so that _____ would learn _____.
Name *thing* *CITIZENS* *discipline*

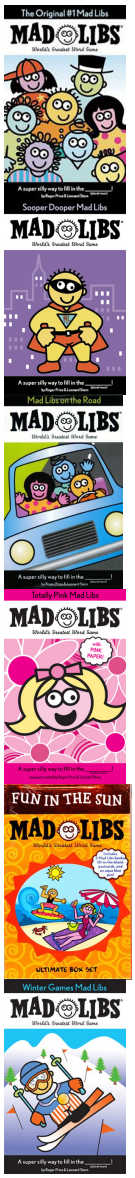
The missing words could be completed as follows

God created ***war*** so that ***AMERICANS*** would learn ***geography***.

or equally well in the following way

Mercator created ***maps*** so that ***BELGIANS*** would learn ***navigation***.

The game was invented by Leonard Stern en Roger Price, and since the series was first published in 1958 more than 110 million copies of the *Mad Libs* books have been sold in the US alone.



Assignment

Write a class MadLibs whose objects can be used to complete a given quote or story in which some keywords have been replaced with blanks, by randomly picking words from the indicates categories. Each object of this class must have a property vocabulary that points to a dictionary. This dictionary represents the vocabulary the object has learned so far. Initially the dictionary is empty, but throughout the object's lifetime new information can be added, or the information in the dictionary can be consulted by calling the following methods that must be supported by the class MadLibs:

- A method learn that can be used to add one or more word form a certain category to the object's vocabulary. This method takes two

arguments: *i*) the category of the new words and *ii*) one or more words from this category. In case a single words is passed as the second argument, this word may be passed as a string. In case multiple words are passed as the the second argument, the words must be passed as a list, a tuple or a set. The method must make sure that the dictionary referenced by the property `vocabulary` is extended by adding the words to the set to which the given category is mapped by the dictionary. In case the dictionary did not have a key corresponding to the given category, a new key-value pair must be added to it, with the key corresponding to the given category and the value being a set containing the given words. All categories and words included in the dictionary must always be converted into lowercase.
- A method `suggest` that can be used to suggest a word from a given category. The category must be passed as an argument to the method. The method must randomly pick one of the words from the set to which the given category is mapped by the property `vocabulary`. In case the given category does not occur as a key in the property `vocabulary`, the method must raise an `AssertionError` with the message `unknown category`. If all letters contained in the given category are capitals, all letters in the selected word must also be converted into uppercase. If the first letter contained in the given category is a capital and all successive letters are lowercase letters, the first letter of the selected word must also be converted into uppercase. Finally, the method must return the (adjusted version of the) selected word.
- A method `fill` that takes a string. This string may contain fragments enclosed in between two underscores (_). These fragments specify the name of a category, and represent keywords from that category that have been replaced with blanks in the given string. The method must replace each of these fragments (including the leading and trailing underscores) with a randomly selected word from the specified category. This must be done by calling the method `suggest`, which also applies the rules for adjusting the selected word as implemented by this method. The string in which all blanked keywords have been replaced by randomly chosen words from the specified category, must be returned by the method. In case the given string contains a fragment that does not correspond to a category in the property `vocabulary`, the method must raise an `AssertionError` with the message `unknown category`.

Example

```
>>> madlib = MadLibs()
>>> madlib.vocabulary
{}

>>> madlib.learn('name', 'God')
>>> madlib.learn('thing', 'war')
>>> madlib.learn('citizens', 'Americans')
>>> madlib.learn('discipline', 'geography')
>>> madlib.vocabulary
{'discipline': {'geography'}, 'thing': {'war'}, 'citizens': {'americans'}, 'name': {'god'}}
>>> madlib.suggest('name')
'god'
>>> madlib.suggest('NAME')
'GOD'
>>> madlib.suggest('Name')
'God'
>>> madlib.fill('_Name_ created _thing_ so that _CITIZENS_ would learn _discipline_.')
'God created war so that AMERICANS would learn geography.'

>>> madlib.learn('name', ('Mercator', 'Caesar'))
>>> madlib.learn('thing', ['maps', 'coordinates'])
>>> madlib.learn('citizens', {'Belgians', 'Martians', 'Germans'})
>>> madlib.learn('discipline', 'navigation')
>>> madlib.learn('discipline', 'colonisation')
>>> madlib.vocabulary
{'discipline': {'colonisation', 'navigation', 'geography'}, 'thing': {'maps', 'war', 'coordinates'}, 'citizens': {'belgians', 'americans', 'germans', 'martians'}, 'name': {'god', 'caesar', 'mercator'}}
>>> madlib.fill('_Name_ created _thing_ so that _CITIZENS_ would learn _discipline_.')
'Mercator created maps so that BELGIANS would learn geography.'
>>> madlib.fill('_Name_ created _thing_ so that _CITIZENS_ would learn _discipline_.')
'Mercator created war so that BELGIANS would learn navigation.'

>>> madlib.suggest('country')
Traceback (most recent call last):
AssertionError: unknown category

>>> madlib.suggest('_CITIZENS_ live in _Country_.')
Traceback (most recent call last):
AssertionError: unknown category
```

Mad Libs is een taalspelletje waarbij wordt vertrokken van een quote of een kort verhaaltje waarin bepaalde sleutelwoorden zijn weggelaten. Onder elk weggelaten woord wordt een lexicale of andere categorie aangegeven, zoals "*zelfstandig naamwoord*", "*werkwoord*" , "*plaats*" of "*lichaamsdeel*" . Een speler vraagt de andere spelers beurtelings om een bepaald woord te suggereren binnen de categorie die onder het volgende weggelaten woord staat, zonder daarbij de context te onthullen waarin het woord moet gebruikt worden. Op het einde wordt de ingevulde quote of het vervolledigde verhaal luidop voorgelezen. Het resultaat is vaak komisch, surrealistisch en ietwat onzinnig.

Stel dat we bijvoorbeeld vertrekken van de volgende gedeeltelijke quote.

_____ created _____ so that _____ would learn _____.
Naam *ding* *INWONERS* *discipline*

De ontbrekende woorden zouden dan op de volgende manier kunnen ingevuld worden

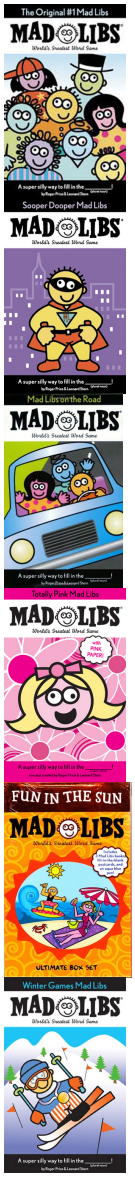
God created ***war*** so that ***AMERICANS*** would learn ***geography***.

of evengoed op de volgende manier

Mercator created ***maps*** so that ***BELGIANS*** would learn ***navigation***.

Dit spelletje werd in 1953 uitgevonden door Leonard Stern en Roger Price, en sinds de reeks voor het eerst werd uitgegeven in 1958 werden

alleen al binnen de Verenigde Staten meer dan 110 miljoen exemplaren verkocht van de *Mad Libs* boeken.



Opgave

Schrijf een klasse MadLibs waarvan de objecten kunnen gebruikt worden om een gegeven quote of verhaal waarin sleutelwoorden weggelaten werden, aan te vullen op basis van woorden die willekeurig gekozen werden uit bepaalde categorieën. Elk object van deze klasse moet beschikken over een eigenschap woordenschat die verwijst naar een dictionary. Deze dictionary stelt de woordenschat voor die het object aangeleerd gekregen heeft. Initieel is deze dictionary nog leeg, maar tijdens de levensduur van het object kan er nieuwe informatie aan toegevoegd worden, of kan de informatie in de dictionary geraadpleegd worden bij het aanroepen van de volgende methoden die minimaal door de klasse MadLibs moeten ondersteund worden:

- Een methode leren waarmee bijkomende woorden in een bepaalde categorie aan de woordenschat van het object kunnen toegevoegd worden. Aan deze methode moeten twee argumenten doorgegeven worden: *i*) de categorie van de nieuwe woorden en *ii*) één of meerdere woorden binnen deze categorie. Indien er als tweede argument één woord wordt doorgegeven, dan mag dat woord als een string doorgegeven worden. Indien er als tweede argument meerdere woorden doorgegeven worden, dan moeten deze woorden onder de vorm van een lijst, een tuple of een verzameling doorgegeven worden. De methode moet ervoor zorgen dat de dictionary van de eigenschap woordenschat uitgebreid wordt, door de gegeven woorden toe te voegen aan de verzameling waarop de gegeven categorie door de dictionary wordt afgebeeld. Indien er in de dictionary nog geen sleutel voorkwam die correspondeert met de gegeven categorie, dan moet een nieuw sleutel-waarde paar aangemaakt worden, waarbij de sleutel correspondeert met de gegeven categorie, en de waarde gevormd wordt door de verzameling van de gegeven woorden. Alle categorieën en woorden die in de dictionary opgenomen worden, moeten steeds omgezet worden naar kleine letters.
- Een methode suggereren waarmee een woord uit een bepaalde categorie kan gesuggereerd worden. Deze categorie moet als argument aan de methode doorgegeven worden. De methode moet een willekeurig woord kiezen uit de verzameling waarop de gegeven categorie wordt afgebeeld door de eigenschap woordenschat. Indien de gegeven categorie niet als sleutel voorkomt in de eigenschap woordenschat, dan moet de methode een AssertionError opwerpen met de boodschap onbekende categorie. Als alle letters die voorkomen in de gegeven categorie hoofdletters zijn, dan moeten ook de letters in het gekozen woord omgezet worden naar hoofdletters. Als de eerste letter van de gegeven categorie een hoofdletter is en alle volgende letters kleine letters zijn, dan moet ook de eerste letter van het gekozen woord naar een hoofdletter omgezet worden. Finaal moet de methode (de aangepaste versie van) het gekozen woord als resultaat teruggeven.
- Een methode invullen waaraan een string moet doorgegeven worden. Deze string mag fragmenten bevatten die ingesloten zitten tussen twee underscores (_). Deze fragmenten geven de naam van een categorie aan, en stellen weggelaten sleutelwoorden van die categorie voor. De methode moet elk van deze fragmenten (inclusief de underscore vooraan en achteraan) vervangen door een willekeurig gekozen

woord van die categorie. Hiervoor moet een woord gesuggereerd worden door gebruik te maken van de methode suggereren, waarbij dus ook de regels voor het aanpassen van het gesuggereerde woord moeten gelden die geïmplementeerd worden door deze methode. De string waarin alle weggelaten sleutelwoorden zijn vervangen door willekeurig gekozen woorden, moet als resultaat teruggeven worden. Indien de gegeven string een fragment bevat dat niet correspondeert met een categorie in de eigenschap woordenschat, dan moet de methode een AssertionError opwerpen met de boodschap onbekende categorie.

Voorbeeld

```
>>> madlib = MadLibs()
>>> madlib.woordenschat
{}

>>> madlib.leren('naam', 'God')
>>> madlib.leren('ding', 'war')
>>> madlib.leren('inwoners', 'Americans')
>>> madlib.leren('discipline', 'geography')
>>> madlib.woordenschat
{'ding': {'war'}, 'naam': {'god'}, 'discipline': {'geography'}, 'inwoners': {'americans'}}
>>> madlib.suggereren('naam')
'god'
>>> madlib.suggereren('NAAM')
'GOD'
>>> madlib.suggereren('Naam')
'God'
>>> madlib.invullen('_Naam_ created _ding_ so that _INWONERS_ would learn _discipline_.')
'God created war so that AMERICANS would learn geography.'

>>> madlib.leren('naam', ('Mercator', 'Caesar'))
>>> madlib.leren('ding', ['maps', 'coordinates'])
>>> madlib.leren('inwoners', {'Belgians', 'Martians', 'Germans'})
>>> madlib.leren('discipline', 'navigation')
>>> madlib.leren('discipline', 'colonisation')
>>> madlib.woordenschat
{'ding': {'maps', 'war', 'coordinates'}, 'naam': {'god', 'caesar', 'mercator'}, 'discipline': {'colonisation', 'navigation', 'geography'}, 'inwoners': {'belgians', 'americans', 'germans', 'martians'}}
>>> madlib.invullen('_Naam_ created _ding_ so that _INWONERS_ would learn _discipline_.')
'Mercator created maps so that BELGIANS would learn geography.'
>>> madlib.invullen('_Naam_ created _ding_ so that _INWONERS_ would learn _discipline_.')
'Mercator created war so that BELGIANS would learn navigation.'

>>> madlib.suggereren('land')
Traceback (most recent call last):
AssertionError: onbekende categorie

>>> madlib.suggereren('_INWONERS_ live in _Land_.')
Traceback (most recent call last):
AssertionError: onbekende categorie
```