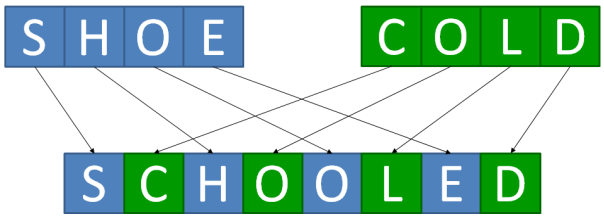
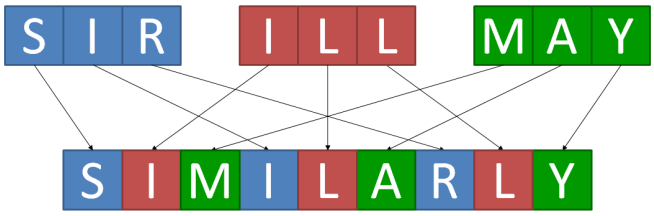
Alternades

Interleave the letters of the words SHOE and COLD and you get the word SCHOOLED.

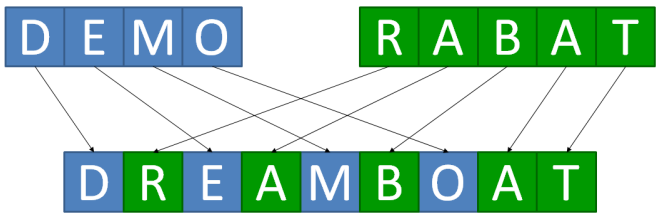


SHOE + COLD = SCHOOLED

The same procedure can also be applied to interleave the letters of three words to produce a fourth.



SIR + ILL + MAY = SIMILARLY
However, interleaving the letters of a series of words is not limited to words having equal length. The general procedure repeatedly takes the next letter of the next word, where the last word is again followed by the first word. If all letters of a given word have been taken, that word is ignored in the remainder of the procedure. The procedure ends if all letters of all words have been taken.



DEMO + RABAT = DREAMBOAT

Assignment

Define a class Alternade that can be used to interleave the letters of a series of words. The objects of the class Alternade must at least support the following methods:

- An initialization method that takes a string or a list of strings as its argument. After initialization, each newly created object must have a property words that refers to a **copy** of the given list of strings. If the argument passed to the initialization method is a string, this is just a shorthand notation for a list that contains only a single string.
- A method __repr__ that returns a string representation of the object. This string representations contains a Python expression that can be used to instantiate a new object having the same value as the current object. The string representation is formatted as Alternade(*li*), where *li* is the string representation of the list of strings that is referenced by the property words of the object. However, if the list of strings contains only a single string, *li* is the string representation of the only string in the list.

- A method __add__ that allows to use the + operator to add the current object s to another object o of the class Alternade. The result of the expression $s + o$ is a **new object** of the class Alternade whose property words refers to the list of strings that results from concatenating the lists of words of the objects s and o.
- A method __str__ that returns the string that results from interleaving the letters of the list of strings referenced by the property words of the current object.

Preparation

To determine whether a given object has a certain type of specifications, you can of course use the built-in function type(o), that prints the type of specification of the object o. However, it is better to use the built-in function isinstance(o, t). This function prints a Boolean value that indicates whether the object o belongs to type t, or not.

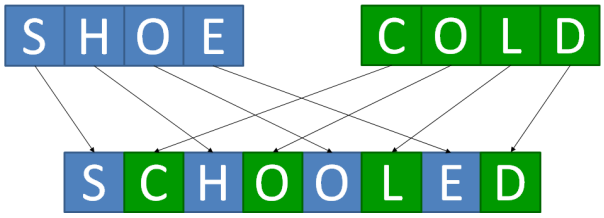
```
>>> type(3) == int
True
>>> isinstance(3.14, int)
False
>>> isinstance(3.14, float)
True
>>> isinstance([1, 2, 3], list)
True
```

Example

```
>>> word1 = Alternade('SHOE')
>>> word1.words
['SHOE']
>>> word1
Alternade('SHOE')
>>> print(word1)
SHOE

>>> words = ['COLD']
>>> word2 = Alternade(words)
>>> word2.words
['COLD']
>>> id(words) != id(word2.words)
True
>>> word2
Alternade('COLD')

>>> word3 = word1 + word2
>>> isinstance(word3, Alternade)
True
>>> word3.words
['SHOE', 'COLD']
>>> word3
Alternade(['SHOE', 'COLD'])
>>> print(word3)
SCHOOLED
>>> word1
Alternade('SHOE')
>>> word2
Alternade('COLD')
```

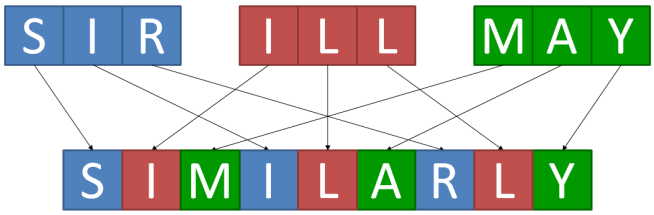
```
>>> word4 = Alternade('DEMO') + Alternade('RABAT')
>>> word4
Alternade(['DEMO', 'RABAT'])
>>> print(word4)
DREAMBOAT

>>> word5 = Alternade('SIR') + Alternade('ILL') + Alternade('MAY')
>>> word5.words
['SIR', 'ILL', 'MAY']
>>> word5
Alternade(['SIR', 'ILL', 'MAY'])
>>> print(word5)
SIMILARLY
```

Voeg alternerend de letters van de woorden SHOE en COLD samen en je krijgt het woord SCHOOLED.



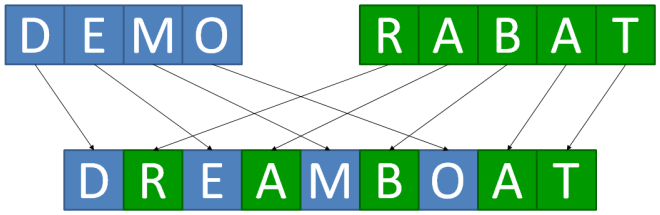
SHOE + COLD = SCHOOLED

Dezelfde procedure kan ook toegepast worden om de letters van drie woorden alternerend samen te voegen.



SIR + ILL + MAY = SIMILARLY

Het alternerend samenvoegen van de letters van een reeks woorden blijft ook niet beperkt tot woorden van dezelfde lengte. De algemene procedure neemt telkens de volgende letter van het volgende woord, waarbij na het laatste woord terug het eerste volgt. Als alle letters van een woord zijn opgebruikt, dan wordt dat woord genegeerd in de rest van de procedure. De procedure eindigt als alle letters van alle woorden zijn opgebruikt.



DEMO + RABAT = DREAMBOAT

Opgave

Definieer een klasse Woordrits waarmee de letters van een reeks woorden alternerend kunnen samengevoegd worden. De objecten van de klasse Woordrits moeten minstens over de volgende methoden beschikken:

- Een initialisatiemethode waaraan ofwel een string of een lijst van strings als argument kan doorgegeven worden. Na initialisatie moet het nieuw aangemaakte object beschikken over een eigenschap `woorden` die verwijst naar een **kopie** van de gegeven lijst van strings. Als het argument dat aan de methode wordt doorgegeven een string is, dan is dit een verkorte notatie van een lijst die slechts één enkele string bevat.
- Een methode `__repr__` die een stringvoorstelling van het object teruggeeft. Deze stringvoorstelling bestaat uit een Python expressie die kan gebruikt worden om een nieuw object met eenzelfde waarde als het huidige object aan te maken. Deze voorstelling heeft de vorm `Woordrits(`*`li`*`)`, waarbij *li* bestaat uit de stringvoorstelling van de lijst van strings waarnaar de eigenschap `woorden` van het object verwijst. Wanneer de lijst van strings echter slechts één enkel element bevat, dan bestaat *li* uit de voorstelling van de enige string in de lijst.
- Een methode `__add__` die ervoor zorgt dat het huidige object s met de `+` operator kan opgeteld worden bij een ander object o van de klasse `Woordrits`. Het resultaat van de expressie $s + o$ is een **nieuw object** van de klasse `Woordrits` waarvan de eigenschap `woorden` verwijst naar de lijst van strings die ontstaat uit het samenvoegen van de woordenlijsten van de objecten s en o.
- Een methode `__str__` die de string teruggeeft die gevormd wordt door het alternerend samenvoegen van de letters van de strings in de lijst waar de eigenschap `woorden` van het huidige object naar verwijst.

Voorbereiding

Om na te gaan of een gegeven object een bepaald gegevenstype heeft, kan je natuurlijk gebruik maken van de ingebouwde functie `type(o)` die het gegevenstype van het object `o` teruggeeft. Het is echter beter om hiervoor de ingebouwde functie `isinstance(o, t)` te gebruiken. Deze functie geeft een Booleaanse waarde terug die aangeeft of het object `o` al dan niet behoort tot het type `t`.

```
>>> type(3) == int
True
>>> isinstance(3.14, int)
False
>>> isinstance(3.14, float)
True
>>> isinstance([1, 2, 3], list)
True
```

Voorbeeld

```
>>> rits1 = Woordrits('SHOE')
>>> rits1.woorden
['SHOE']
>>> rits1
Woordrits('SHOE')
>>> print(rits1)
SHOE

>>> woorden = ['COLD']
>>> rits2 = Woordrits(woorden)
>>> rits2.woorden
['COLD']
>>> id(woorden) != id(rits2.woorden)
True
```

```
>>> rits2
Woordrits('COLD')

>>> rits3 = rits1 + rits2
>>> isinstance(rits3, Woordrits)
True
>>> rits3.woorden
['SHOE', 'COLD']
>>> rits3
Woordrits(['SHOE', 'COLD'])
>>> print(rits3)
SCHOOLED
>>> rits1
Woordrits('SHOE')
>>> rits2
Woordrits('COLD')

>>> rits4 = Woordrits('DEMO') + Woordrits('RABAT')
>>> rits4
Woordrits(['DEMO', 'RABAT'])
>>> print(rits4)
DREAMBOAT

>>> rits5 = Woordrits('SIR') + Woordrits('ILL') + Woordrits('MAY')
>>> rits5.woorden
['SIR', 'ILL', 'MAY']
>>> rits5
Woordrits(['SIR', 'ILL', 'MAY'])
>>> print(rits5)
SIMILARLY
```