Anthropocentrism

A **word square** consists of a list of words written out in a square grid, such that the same words can be read both horizontally and vertically from the corresponding row and column in the grid. The number of words, which is equal to the number of letters in each word, is known as the **order** of the square. For example, this is an order 3 square:

	0	1	2
0	B	I	T
1	I	C	E
2	T	E	N

Large word squares are dramatically harder to make than small ones. To date, the largest anyone has managed to find are composed of 9-letter words:

	0	1	2	3	4	5	6	7	8
0	A	C	H	A	L	A	S	I	A
1	C	R	E	N	I	D	E	N	S
2	H	E	X	A	N	D	R	I	C
3	A	N	A	B	O	L	I	T	E
4	L	I	N	O	L	E	N	I	N
5	A	D	D	L	E	H	E	A	D
6	S	E	R	I	N	E	T	T	E
7	I	N	I	T	I	A	T	O	R
8	A	S	C	E	N	D	E	R	S

Finding a perfect 10×10 word square has been a central goal for wordplay fans for more than 100 years. The task was looking impossible when in 1972 Dmitri Borgmann found an unexpected resource in the African journal of David Livingstone, whose entry for Sept. 26, 1872, reads:

> *Through forest, along the side of a sedgy valley. Cross its head-water, which has rust of iron in it, then west and by south. The forest has very many tsetse. Zebras calling loudly, and Senegal long claw in our camp at dawn, with its cry, 'O-o-o-o-o-o-o-o-o-o.'*

This is exactly what was needed. Given a pen, the yellow-bellied longclaw, *Macronyx flavigaster*, could have drawn for Livingstone a perfect 10×10 square:

O	O	O	O	O	O	O	O	O	O
O	O	O	O	O	O	O	O	O	O
O	O	O	O	O	O	O	O	O	O
O	O	O	O	O	O	O	O	O	O
O	O	O	O	O	O	O	O	O	O
O	O	O	O	O	O	O	O	O	O
O	O	O	O	O	O	O	O	O	O
O	O	O	O	O	O	O	O	O	O
O	O	O	O	O	O	O	O	O	O
O	O	O	O	O	O	O	O	O	O

We ought to consult other species more often. Any longclaw could have given us this contribution — indeed, this is the only word square the bird is capable of making!

Assignment

Because word squares are symmetric with respect to their main diagonal, they can be fully described by reading the letters on or below the main diaginal from left to right and from top to bottom. The order 3 word square that is given as an example in the introduction, can be described in this way as the 6-letter string BICTEN. In general, any order n word square can be described as an $\frac{n(n + 1)}{2}$-letter string containing the letters on or below the main diagonal. Your task is to define a class WordSquare that can be used to represent word squares in Python. This class must support at least the following methods:

- An initialisation method that takes two arguments: the order of the word square and a string of letters on or below the main diagonal, read from left to right and from top to bottom. In case the number of letters on or below the main diagonal does not correspond to the given order of the word square, the method must raise an AssertionError with the message invalid square.
- A method letter that takes two arguments: the row and column index of a cell in the word

square. The method must return the upper case version of the letter at the given cel in the word square. Cells in a word square are located by indexing the rows from top to bottom, and the columns from left to right, each time starting at index zero. In case the given arguments do not point to the location of a cell inside the grid, the method must raise an AssertionError with the message invalid index.

- A method word that takes the index of a row or column in the word square as its argument (using the same indexing as in the previous method). The method must return the upper case version of the word that can be read in the word square at the indicated row or column. In case the given index does not corresspond to the index of a row/column in the word square, the method must raise an AssertionError with the message invalid index.
- A method valid that takes the location of a text file as its argument. This text file must contain a list of words, each on a separate line. The method must return a Boolean value, that indicates whether or not all words in the word square (either read from the rows or the columns) occur in the given list of words. In comparing the words in the word square with the words in the given text file, no distinction should be made between uppercase and lowercase letters.

In addition, it should be possible to use the built-in function print to output a string representation of each object that is an instance of the class WordSquare. In this string representation, the letters of the word square are written out row by row, as illustrated in the examples below.

Example

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

```
>>> square = WordSquare(3, 'bicten')
>>> square.letter(1, 0)
'I'
>>> square.letter(1, 1)
'C'
>>> square.letter(1, 2)
'E'
>>> square.word(1)
'ICE'
>>> print(square)
BIT
ICE
TEN
>>> square.valid('words.txt')
True

>>> square.letter(1, 3)
Traceback (most recent call last):
AssertionError: invalid index
>>> square.word(3)
Traceback (most recent call last):
AssertionError: invalid index

>>> square = WordSquare(3, 'ABCDE')
Traceback (most recent call last):
AssertionError: invalid square

>>> square = WordSquare(9, 'ACRHEXANABLINOLADDLEHSERINETINITIATOASCENDERS')
```

```
>>> square.word(1)
'CRENIDENS'
>>> square.word(7)
'INITIATOR'
>>> print(square)
ACHALASIA
CRENIDENS
HEXANDRIC
ANABOLITE
LINOLENIN
ADDLEHEAD
SERINETTE
INITIATOR
ASCENDERS
>>> square.valid('words.txt')
True

>>> square = WordSquare(4, 'ABCDEFGHIJ')
>>> print(square)
ABDG
BCEH
DEFI
GHIJ
>>> square.valid('words.txt')
False
```

Resources

- **Ross EA (2005)**. A History of the Ten-Square. *In* Cipra B, Demaine ED, Demaine ML, Rodgers T. Tribute To A Mathemagician. Peters AK, Ltd. 85-91. ISBN 978-1-56881-204-5.
- **Gooch R (2003)**. Nine-Square Round-Up. *Word Ways* **36(3)**.
- **Gooch R (2004)**. Hunting The Ten-Square. *Word Ways* **37(2)**.

Een **woordvierkant** bestaat uit een reeks woorden die in een vierkant rooster uitgeschreven worden, zodat hetzelfde woord zowel horizontaal als verticaal kan uitgelezen worden in de overeenkomstige rij en kolom van het rooster. Het aantal woorden — dat gelijk is aan het aantal letters van elk woord — wordt de **orde** van het vierkant genoemd. Dit is bijvoorbeeld een woordvierkant van orde 3:

	0	1	2
0	B	I	T
1	I	C	E
2	T	E	N

In vergelijking met kleine woordvierkanten, zijn grote woordvierkanten verschrikkelijk moeilijk te construeren. Het grootste woordvierkant dat men tot op vandaag heeft kunnen opstellen, bestaat uit 9-letterwoorden:

	0	1	2	3	4	5	6	7	8
0	A	C	H	A	L	A	S	I	A
1	C	R	E	N	I	D	E	N	S
2	H	E	X	A	N	D	R	I	C
3	A	N	A	B	O	L	I	T	E
4	L	I	N	O	L	E	N	I	N
5	A	D	D	L	E	H	E	A	D
6	S	E	R	I	N	E	T	T	E
7	I	N	I	T	I	A	T	O	R
8	A	S	C	E	N	D	E	R	S

Puzzelfanaten zijn al meer dan 100 jaar op zoek naar een perfect 10×10 woordvierkant. Deze taak leek schier onmogelijk, totdat Dmitri Borgmann in 1972 onverwacht botste op een artikel van David Livingstone in een Afrikaanse krant, waarvan de editie van 26 september 1872 het volgende tekstfragment bevatte:

> *Through forest, along the side of a sedgy valley. Cross its head-water, which has rust of iron in it, then west and by south. The forest has very many tsetse. Zebras calling loudly, and Senegal long claw in our camp at dawn, with its cry, 'O-o-o-o-o-o-o-o-o-o.'*

Meer was er niet nodig. Voorzien van een pen kon de geelbuiklangkauw (*Macronyx flavigaster*) voor Livingstone een perfect 10×10 woordvierkant neergeschreven hebben:

O	O	O	O	O	O	O	O	O	O
O	O	O	O	O	O	O	O	O	O
O	O	O	O	O	O	O	O	O	O
O	O	O	O	O	O	O	O	O	O
O	O	O	O	O	O	O	O	O	O
O	O	O	O	O	O	O	O	O	O
O	O	O	O	O	O	O	O	O	O
O	O	O	O	O	O	O	O	O	O
O	O	O	O	O	O	O	O	O	O
O	O	O	O	O	O	O	O	O	O

We moesten vaker te rade gaan bij andere soorten. Elke langkauw had ons deze oplossing op een dienblaadje kunnen aanbieden — inderdaad, het is het enige woordvierkant dat die vogel kan maken!

Opgave

Omdat woordvierkanten symmetrisch zijn ten opzichte van de hoofddiagonaal, volstaat het om een woordvierkant te beschrijven door de letters op en onder de hoofddiagonaal uit te lezen van links naar rechts, en van boven naar onder. Het woordvierkant van orde 3 dat in de inleiding als voorbeeld gegeven werd, kan op die manier beschreven worden aan de hand van de zes letters BICTEN. Algemeen geldt dat een woordvierkant van orde n kan beschreven worden aan de hand van $\frac{n(n + 1)}{2}$ letters op of onder de hoofddiagonaal. Je opdracht bestaat erin een klasse Woordvierkant te definiëren waarmee woordvierkanten kunnen voorgesteld worden in Python. Deze klasse moet minstens de volgende methoden ondersteunen:

- Een initialisatiemethode waaraan twee argumenten moeten doorgegeven worden: de orde van het woordvierkant en de letters op en onder de hoofddiagonaal, uitgelezen van links naar rechts en van boven naar onder. Indien het gegeven aantal letters op en onder de hoofddiagonaal niet correspondeert met de gegeven orde, dan moet de methode een AssertionError opwerpen met de boodschap ongeldig vierkant.
- Een methode letter waaraan de rij- en kolomindex van een positie binnen het woordvierkant moeten doorgegeven worden. De methode moet de letter (in hoofdlettervorm) teruggeven die gevonden wordt op de aangegeven positie in het woordvierkant. Hierbij worden de rijen van boven naar onder, en de kolommen van links naar rechts geïndexeerd vanaf nul. Indien de gegeven argumenten niet corresponderen met een positie binnen het vierkant rooster, dan moet de methode een AssertionError opwerpen met de boodschap ongeldige index.
- Een methode woord waaraan een rij- of kolomindex van het woordvierkant moet doorgegeven worden (zelfde indexering als bij voorgaande methode). De methode moet het woord (in hoofdletters) teruggeven dat in het woordvierkant gevonden wordt op de aangegeven rij/kolom. Indien de gegeven index niet correspondeert met een rij/kolom in het woordvierkant, dan moet de methode een AssertionError opwerpen met de boodschap ongeldige index.
- Een methode geldig waaraan de locatie van een tekstbestand moet doorgegeven worden. Dit tekstbestand bevat een lijst van woorden, elk op een afzonderlijke regel. De methode moet een Booleaanse waarde teruggeven, die aangeeft of alle woorden uit het woordvierkant voorkomen in de gegeven lijst van woorden. Bij de vergelijking van de woorden uit het woordvierkant en de gegeven woordenlijst mag geen onderscheid gemaakt worden tussen hoofdletters en kleine letters.

Bovendien moet het mogelijk zijn om met de ingebouwde functie print een stringvoorstelling uit te schrijven van elk object van de klasse Woordvierkant. In deze stringvoorstelling worden de letters van het woordvierkant rij per rij uitgeschreven, zoals aangegeven in onderstaande voorbeelden.

Voorbeeld

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

```
>>> vierkant = Woordvierkant(3, 'bicten')
>>> vierkant.letter(1, 0)
'I'
>>> vierkant.letter(1, 1)
'C'
>>> vierkant.letter(1, 2)
'E'
>>> vierkant.woord(1)
'ICE'
>>> print(vierkant)
BIT
ICE
TEN
>>> vierkant.geldig('words.txt')
True

>>> vierkant.letter(1, 3)
Traceback (most recent call last):
AssertionError: ongeldige index
>>> vierkant.woord(3)
Traceback (most recent call last):
AssertionError: ongeldige index

>>> vierkant = Woordvierkant(3, 'ABCDE')
Traceback (most recent call last):
AssertionError: ongeldig vierkant

>>> vierkant = Woordvierkant(9, 'ACRHEXANABLINOLADDLEHSERINETINITIATOASCENDERS')
>>> vierkant.woord(1)
'CRENIDENS'
>>> vierkant.woord(7)
'INITIATOR'
>>> print(vierkant)
ACHALASIA
CRENIDENS
HEXANDRIC
ANABOLITE
LINOLENIN
ADDLEHEAD
SERINETTE
INITIATOR
ASCENDERS
>>> vierkant.geldig('words.txt')
True

>>> vierkant = Woordvierkant(4, 'ABCDEFGHIJ')
>>> print(vierkant)
ABDG
BCEH
```

```
DEFI
GHIJ
>>> vierkant.geldig('words.txt')
False
```

Bronnen

- **Ross EA (2005)**. A History of the Ten-Square. *In* Cipra B, Demaine ED, Demaine ML, Rodgers T. Tribute To A Mathemagician. Peters AK, Ltd. 85-91. ISBN 978-1-56881-204-5.
- **Gooch R (2003)**. Nine-Square Round-Up. *Word Ways* **36(3)**.
- **Gooch R (2004)**. Hunting The Ten-Square. *Word Ways* **37(2)**.