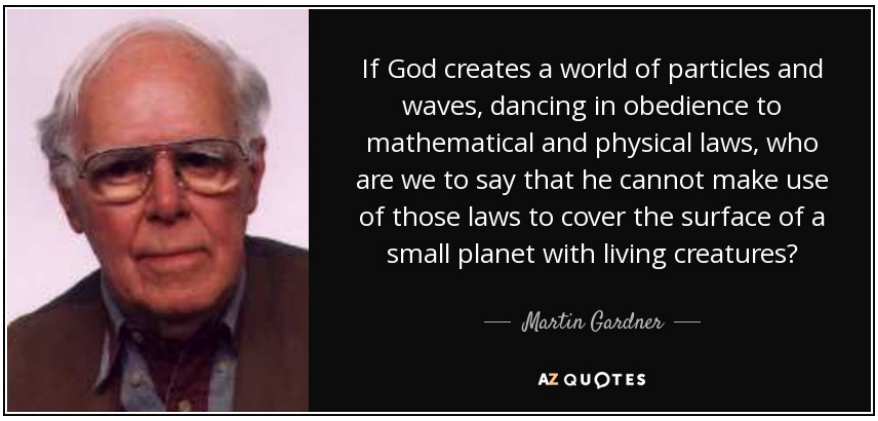
Sorting trick

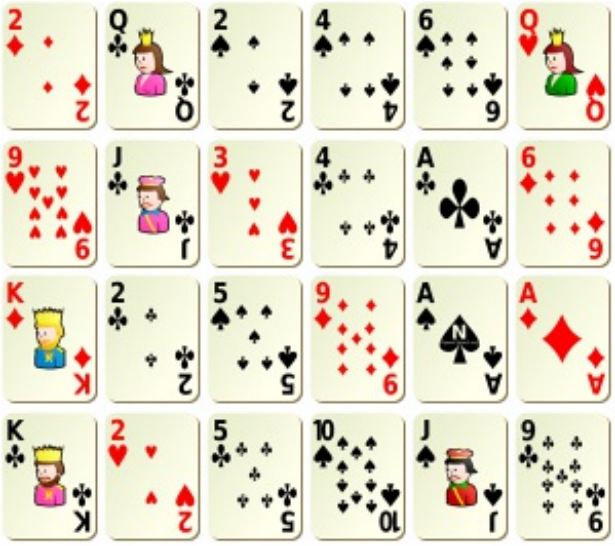
Like good magic tricks, clever puzzles can inspire awe, reveal mathematical truths and prompt important questions. At least that is what Martin Gardner thought. His name is synonymous with the legendary *Mathematical Games* column he wrote for a quarter of a century in Scientific American. The column that began in January 1957 has become a legend in publishing, even though it's been almost 30 years since the last one appeared. Thanks to his own mathemagical skills, Gardner presented noteworthy mathematics every month with all the wonder of legerdemain and — in so doing — captivated a huge readership worldwide. The columns are still considered models of clarity and elegance for introducing fresh and engaging ideas in mathematics in non-technical ways.



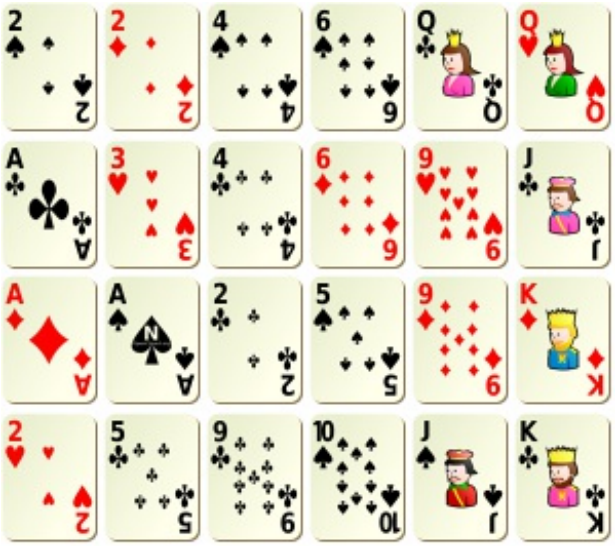
While many articles of the ever-prolific Martin Gardner fall under the umbrella of "recreational mathematics", others touched on cutting-edge concepts involving contributions from some of the world's most creative minds. Even the articles that seemed to be purely for entertainment sometimes inspired important research, some of which led to developments with real impact on science, technology and society. This success is all the more remarkable considering that Gardner had no formal training in mathematics. In his memoirs *Undiluted Hocus-Pocus* (Princeton, 2013), Gardner recalls:

> *One of the pleasures in writing the column was that it introduced me to so many top mathematicians, which of course I was not. Their contributions to my column were far superior to anything I could write, and were a major reason for the column's growing popularity. The secret of its success was a direct result of my ignorance. Even today my knowledge of math extends only through calculus, and even calculus I only dimly comprehend. As a result, I had to struggle to understand what I wrote, and this helped me write in ways that others could understand.*

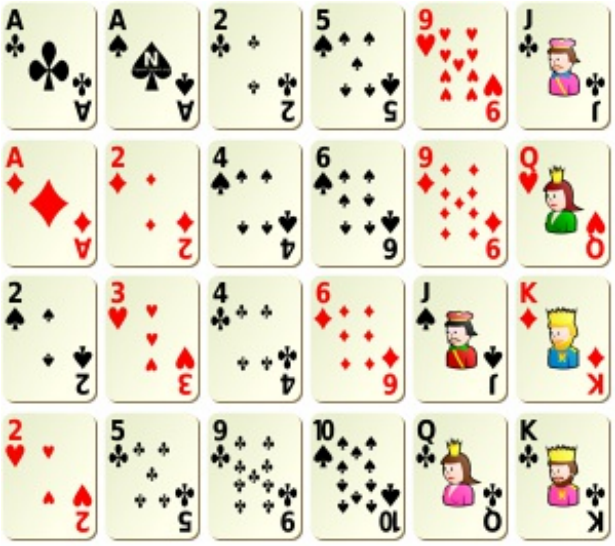
In this assignment we focus on a nice trick that appeared in one of the *Mathematical Games* columns that Martin Gardner wrote together with the mathematician Ross Honsberger. Deal cards into any rectangular array:



Put the cards on each row of the array such that they are sorted in increasing order. In doing so, you only need to take into account the numerical value of the cards.



Now you do the same with the cards in each column of the array.



Surprisingly, that last step hasn't disturbed the preceding one — the cards in each row are still sorted in increasing order. Any idea how this is possible?

Assignment

The above trick works with all objects that can be sorted. In this assignment we keep it simple by considering only integers. This is exemplified using the following integer array. $$ \begin{array}{rrrrrr} 2 & 12 & 2 & 4 & 6 & 12\\ 9 & 11 & 3 & 4 & 1 & 6\\ 13 & 2 & 5 & 9 & 1 & 1\\ 13 & 2 & 5 & 10 & 11 & 9 \end{array} \,\, \stackrel{\text{sort rows}}{\longrightarrow} \,\, \begin{array}{rrrrrr} 2& 2& 4& 6& 12& 12\\ 1& 3& 4& 6& 9& 11 \\ 1& 1& 2& 5& 9& 13\\ 2& 5& 9& 10& 11& 13 \end{array} \,\, \stackrel{\text{sort columns}}{\longrightarrow} \,\, \begin{array}{rrrrrr} 1& 1& 2& 5& 9& 11 \\ 1& 2& 4& 6& 9& 12 \\ 2& 3& 4& 6& 11& 13 \\ 2& 5& 9& 10& 12& 13 \end{array} $$

Define a class `Grid` that can be used to represent rectangular arrays of integers. This class must support at least the following methods:

- An initialization method that takes a single argument. In case a string is passed as an argument, it must contain the location of a text file from which a rectangular integer array can be read. Each line of the file must contain the integers on a single row of the array, separated from each other by one or more spaces or tabs. As an alternative, a list of tuple containing the rows of the array can be passed as an argument, where each row is itself represented as a list or tuple of the integers on that row. In either case, a rectangular integer array must be passed with at least one row and at least one column, for which holds that each row contains the same number of integers. If this is not the case, the method must raise an `AssertionError` with the message `invalid grid`.
- A method `largest` that returns the largest integer in the array.
- A method `smallest` that returns the smallest number in the array.
- A method `__str__` that returns a string representation of the array. Each integer in this string representation must be right aligned over p positions, where p is the number of characters needed to represent the widest integer in the array. The columns of the array must be separated by a single space in the string representation.
- A method `sorted` having two optional parameters `decreasing` and `columns` that both take Boolean values (default value: `False`). The method must return a Boolean value that indicates whether or not the rows (`columns = False`) or columns (`columns = True`) are sorted in increasing (`decreasing = False`) or decreasing (`decreasing = True`) order.
- A method `sort` having two optional parameters `decreasing` and `columns` that both take Boolean values (default value: `False`). The method must make sure that the integers on each row (`columns = False`) or column (`columns = True`) of the array are sorted in increasing (`decreasing = False`) or decreasing (`decreasing = True`) order. The method must return a reference to the object on which the method was called.

Make sure that the operators `-` and `&` can both be used to compose two grids (objects of the class `Grid`) into a grid that is formed by putting the rows of the second grid underneath the rows of the first grid. The end result must be a new object of the class `Grid`. If the grids that are composed do not have the same number of columns, an `AssertionError` must be raised with the message `different number of columns`.

Make sure that the operators `+` and `|` can both be used to compose two grids (objects of the class `Grid`) into a grid that is formed by putting the columns of the second grid after the columns of the first grid. The end result must be a new object of the class `Grid`. If the grids that are composed do not have the same number of rows, an `AssertionError` must be raised with the message `different number of rows`.

Example

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

```
>>> grid = Grid('grid.txt')
>>> grid.largest()
16
>>> grid.smallest()
1
>>> print(grid)
16  3  2 13
 5 10 11  8
 9  6  7 12
 4 15 14  1
>>> grid.sorted()
False
>>> grid.sorted(columns=True)
False

>>> print(grid.sort())
 2  3 13 16
 5  8 10 11
 6  7  9 12
 1  4 14 15
>>> grid.sorted()
True
>>> grid.sorted(columns=True)
False

>>> print(grid.sort(columns=True))
 1  3  9 11
 2  4 10 12
 5  7 13 15
 6  8 14 16
>>> grid.sorted()
True
>>> grid.sorted(columns=True)
True

>>> print(grid.sort(decreasing=True))
11  9  3  1
12 10  4  2
15 13  7  5
16 14  8  6
>>> grid.sorted()
False
>>> grid.sorted(decreasing=True)
True

>>> print(grid.sort(columns=True, decreasing=True))
16 14  8  6
15 13  7  5
12 10  4  2
11  9  3  1
>>> grid.sorted(columns=True)
False
>>> grid.sorted(columns=True, decreasing=True)
True
```

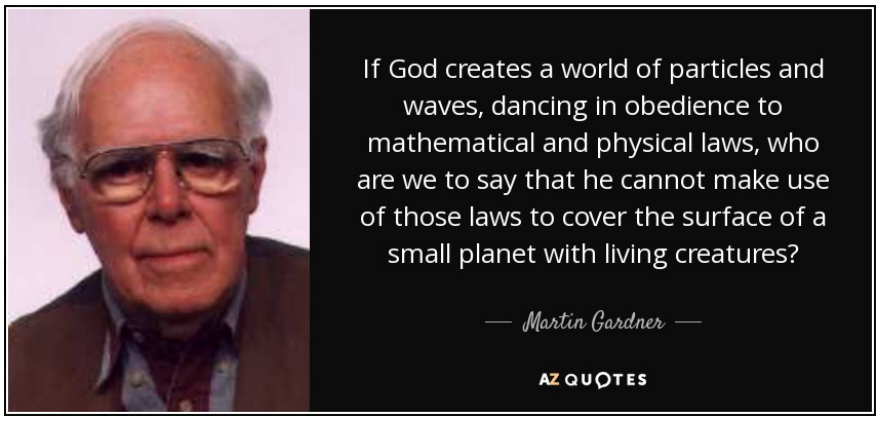
```
>>> row = Grid([[17, 18, 19, 20]])
>>> print(grid - row)
16 14  8  6
15 13  7  5
12 10  4  2
11  9  3  1
17 18 19 20

>>> col = Grid([[-21], [-22], [-23], [-24]])
>>> print(col + grid)
-21  16  14   8   6
-22  15  13   7   5
-23  12  10   4   2
-24  11   9   3   1

>>> print(grid & row)
16 14  8  6
15 13  7  5
12 10  4  2
11  9  3  1
17 18 19 20
>>> print(col | grid)
-21  16  14   8   6
-22  15  13   7   5
-23  12  10   4   2
-24  11   9   3   1

>>> Grid([[1], [2, 3], [4, 5, 6]])
Traceback (most recent call last):
AssertionError: invalid grid
>>> Grid([])
Traceback (most recent call last):
AssertionError: invalid grid
>>> Grid([[]])
Traceback (most recent call last):
AssertionError: invalid grid

>>> grid & col
Traceback (most recent call last):
AssertionError: different number of columns
>>> grid | row
Traceback (most recent call last):
AssertionError: different number of rows
```

Resources

- **Gardner M (2013)**. Undiluted Hocus-Pocus: The Autobiography of Martin Gardner. *Princeton University Press*.
- **Honsberger R (1990)**. More Mathematical Morsels. *The Mathematical Association of America*.

Net als goeie goocheltrucs kunnen vernuftige denkpuzzels nieuwe wiskundige inzichten blootleggen of interessante vragen poneren. Dat was althans een belangrijke drijfveer in het werk van Martin Gardner (1914-2010). Zijn naam is zowat synoniem geworden met de legendarische *Mathematical Games* columns in het tijdschrift *Scientific American*. De eerste uit de reeks verscheen in 1957, en hoewel het al meer dan 30 jaar geleden is dat de laatste

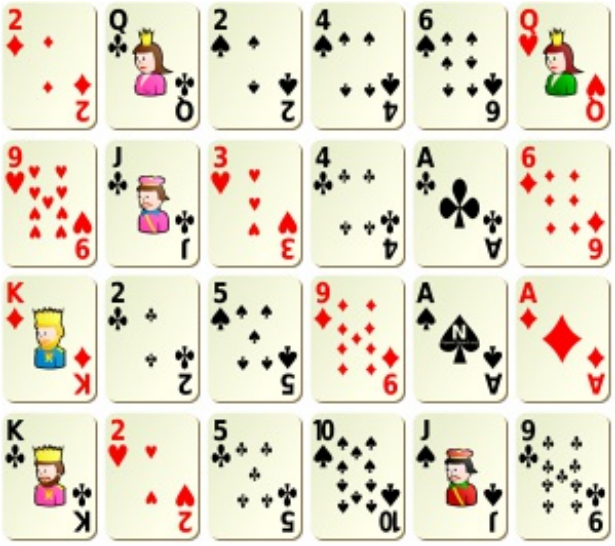
verscheen, zijn ze een legende geworden in de uitgeverswereld. Maandelijks schreef Gardner met al zijn mathemagische vaardigheden en de vingervlugheid van een goochelaar een stuk over betoverende wiskunde, waarmee hij een immens lezerspubliek van over de hele wereld in de ban hield. De columns staan vandaag de dag dan ook nog steeds model voor het duidelijk en elegant aanbrengen van nieuwe en boeiende wiskundige ideeën, zonder te vervallen in techniciteiten.



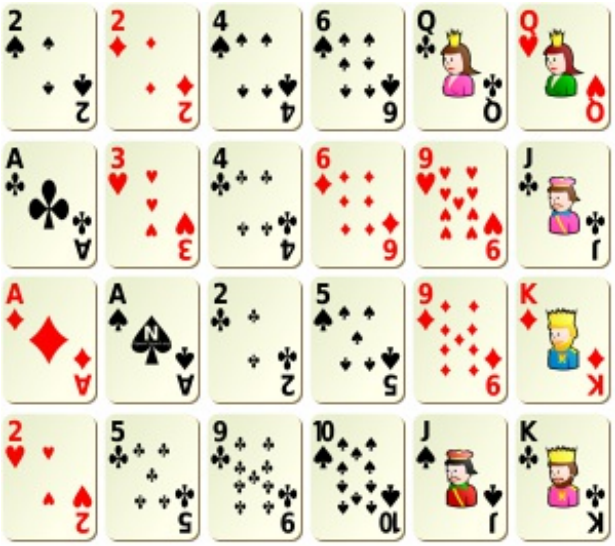
Veel artikels van de immer productieve Martin Gardner vallen onder de noemer "recreatieve wiskunde", maar hij besprak ook ontzettend graag gloednieuwe concepten met bijdragen van 's werelds meest creatieve geesten. Zelfs columns die enkel voor de fun geschreven leken te zijn, bleken een inspiratie voor baanbrekend onderzoek dat geleid heeft tot ontwikkelingen met een grote wetenschappelijke, technologische of maatschappelijke impact. Dit succes is des te opmerkelijker omdat Gardner zelf geen wiskundige opleiding genoten heeft. In zijn memoires *Undiluted Hocus-Pocus* (Princeton, 2013) schreef hij hierover:

> *One of the pleasures in writing the column was that it introduced me to so many top mathematicians, which of course I was not. Their contributions to my column were far superior to anything I could write, and were a major reason for the column's growing popularity. The secret of its success was a direct result of my ignorance. Even today my knowledge of math extends only through calculus, and even calculus I only dimly comprehend. As a result, I had to struggle to understand what I wrote, and this helped me write in ways that others could understand.*

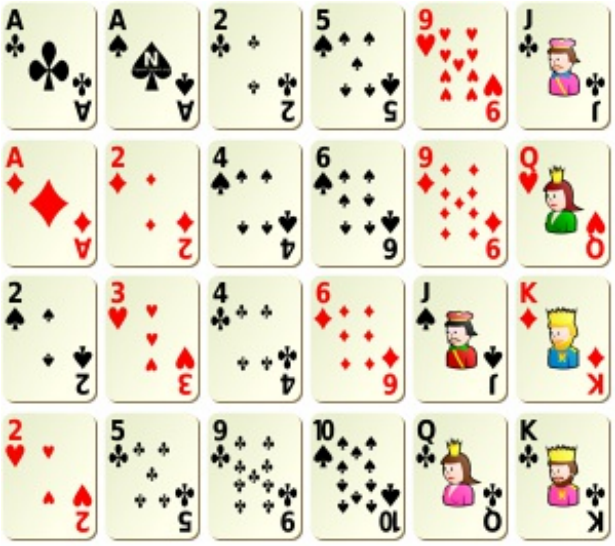
In deze opgave belichten we een wistjedatje dat opdook in één van de *Mathematical Games* columns die Martin Gardner samen schreef met de wiskundige Ross Honsberger. Leg een aantal speelkaarten in een rechthoekig rooster.



Herschik de speelkaarten op elke rij van het rooster, zodat ze van klein naar groot gerangschikt zijn. Hierbij hou je enkel rekening met de numerieke waarde van de speelkaarten.



Doe nu hetzelfde met de kaarten in elke kolom van het rooster.



Verrassend genoeg verstoort deze laatste stap de eerste niet — op elke rij liggen de kaarten nog steeds van klein naar groot gerangschikt. Heb je enig idee hoe dat komt?

Opgave

Het bovenstaande trucje werkt voor alle objecten die kunnen gesorteerd worden. In deze opgave houden we het voor de eenvoud bij gehele getallen. Hieronder geven we daarvan een voorbeeld. $$ \begin{array}{rrrrrr} 2 & 12 & 2 & 4 & 6 & 12\\ 9 & 11 & 3 & 4 & 1 & 6\\ 13 & 2 & 5 & 9 & 1 & 1\\ 13 & 2 & 5 & 10 & 11 & 9 \end{array} \,\, \stackrel{\text{rijen sorteren}}{\longrightarrow} \,\, \begin{array}{rrrrrr} 2& 2& 4& 6& 12& 12\\ 1& 3& 4& 6& 9& 11 \\ 1& 1& 2& 5& 9& 13\\ 2& 5& 9& 10& 11& 13 \end{array} \,\, \stackrel{\text{kolommen sorteren}}{\longrightarrow} \,\, \begin{array}{rrrrrr} 1& 1& 2& 5& 9& 11 \\ 1& 2& 4& 6& 9& 12 \\ 2& 3& 4& 6& 11& 13 \\ 2& 5& 9& 10& 12& 13 \end{array} $$

Definieer een klasse Rooster waarmee rechthoekige roosters van gehele getallen kunnen voorgesteld worden. Deze klasse moet minstens de volgende methoden implementeren:

- Een initialisatiemethode waaraan één enkel argument moet doorgegeven worden. Indien een string als argument wordt doorgegeven, dan moet die de locatie van een tekstbestand bevatten waaruit het rooster moet ingelezen worden. Elke regel van het tekstbestand moet de getallen van één rij uit het rooster bevatten, van elkaar gescheiden door één of meer spaties of tabs. Anderzijds kan als argument ook een lijst of een tuple van de rijen van het rooster doorgegeven worden, waarbij elke rij wordt voorgesteld als een lijst of een tuple van de getallen op die rij. In beide gevallen moet aan de methode een rooster van gehele getallen doorgegeven worden met minstens één rij en één kolom, waarbij geldt dat elke rij evenveel getallen bevat. Indien dit niet het geval is, dan moet de methode een AssertionError opwerpen met de boodschap ongeldig rooster.
- Een methode grootste die het grootste getal uit het rooster teruggeeft.
- Een methode kleinste die het kleinste getal uit het rooster teruggeeft.
- Een methode __str__ die een stringvoorstelling van het rooster teruggeeft. In deze stringvoorstelling moet elk getal rechts uitgelijnd worden over p posities, waarbij p gelijk is aan het aantal karakters dat nodig is om het breedste getal uit het rooster uit te schrijven. De kolommen van het rooster moeten in de stringvoorstelling van elkaar gescheiden worden door één enkele spatie.
- Een methode gesorteerd met twee optionele parameters dalend en kolommen waaraan een Booleaanse waarde kan doorgegeven worden (standaardwaarde: False). De methode moet een Booleaanse waarde teruggeven die aangeeft of de rijen (kolommen = False) of kolommen (kolommen = True) al dan niet van klein naar groot (dalend = False) of van groot naar klein (dalend = True) gerangschikt zijn.
- Een methode sorteer met twee optionele parameters dalend en kolommen waaraan een Booleaanse waarde kan doorgegeven worden (standaardwaarde: False). De methode moet ervoor zorgen dat de getallen op elke rij (kolommen = False) of kolom (kolommen = True) van het rooster van klein naar groot (dalend = False) of van groot naar klein (dalend = True) gerangschikt worden. De methode moet een verwijzing teruggeven naar het object waarop de methode werd aangeroepen.

Zorg ervoor dat de operatoren - en & beide kunnen gebruikt worden om twee roosters (objecten van de klasse Rooster) samen te voegen tot een rooster dat gevormd wordt door de rijen van het tweede rooster onder die van het eerste rooster te plaatsen. Het resultaat moet een nieuw object van de klasse Rooster zijn. Indien de twee roosters die samengevoegd worden niet hetzelfde aantal kolommen hebben, dan moet een AssertionError opgeworpen worden met de boodschap aantal kolommen is verschillend.

Zorg er ook voor dat de operatoren `+` en `|` beide kunnen gebruikt worden om twee roosters (objecten van de klasse `Rooster`) samen te voegen tot een rooster dat gevormd wordt door de kolommen van het tweede rooster achter die van het eerste rooster te plaatsen. Het resultaat moet een nieuw object van de klasse `Rooster` zijn. Indien de twee roosters die samengevoegd worden niet hetzelfde aantal rijen hebben, dan moet een `AssertionError` opgeworpen worden met de boodschap `aantal rijen is verschillend`.

Voorbeeld

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

```
>>> rooster = Rooster('rooster.txt')
>>> rooster.grootste()
16
>>> rooster.kleinste()
1
>>> print(rooster)
16  3  2 13
 5 10 11  8
 9  6  7 12
 4 15 14  1
>>> rooster.gesorteerd()
False
>>> rooster.gesorteerd(kolommen=True)
False

>>> print(rooster.sorteer())
 2  3 13 16
 5  8 10 11
 6  7  9 12
 1  4 14 15
>>> rooster.gesorteerd()
True
>>> rooster.gesorteerd(kolommen=True)
False

>>> print(rooster.sorteer(kolommen=True))
 1  3  9 11
 2  4 10 12
 5  7 13 15
 6  8 14 16
>>> rooster.gesorteerd()
True
>>> rooster.gesorteerd(kolommen=True)
True

>>> print(rooster.sorteer(dalend=True))
11  9  3  1
12 10  4  2
15 13  7  5
16 14  8  6
>>> rooster.gesorteerd()
False
>>> rooster.gesorteerd(dalend=True)
True
```

```
>>> print(rooster.sorteer(kolommen=True, dalend=True))
16 14  8  6
15 13  7  5
12 10  4  2
11  9  3  1
>>> rooster.gesorteerd(kolommen=True)
False
>>> rooster.gesorteerd(kolommen=True, dalend=True)
True

>>> rij = Rooster([[17, 18, 19, 20]])
>>> print(rooster - rij)
16 14  8  6
15 13  7  5
12 10  4  2
11  9  3  1
17 18 19 20

>>> kolom = Rooster([[-21], [-22], [-23], [-24]])
>>> print(kolom + rooster)
-21  16  14   8   6
-22  15  13   7   5
-23  12  10   4   2
-24  11   9   3   1

>>> print(rooster & rij)
16 14  8  6
15 13  7  5
12 10  4  2
11  9  3  1
17 18 19 20
>>> print(kolom | rooster)
-21  16  14   8   6
-22  15  13   7   5
-23  12  10   4   2
-24  11   9   3   1

>>> Rooster([[1], [2, 3], [4, 5, 6]])
Traceback (most recent call last):
AssertionError: ongeldig rooster
>>> Rooster([])
Traceback (most recent call last):
AssertionError: ongeldig rooster
>>> Rooster([[]])
Traceback (most recent call last):
AssertionError: ongeldig rooster

>>> rooster & kolom
Traceback (most recent call last):
AssertionError: aantal kolommen is verschillend
>>> rooster | rij
Traceback (most recent call last):
AssertionError: aantal rijen is verschillend
```

Bronnen

- **Gardner M (2013)**. Undiluted Hocus-Pocus: The Autobiography of Martin Gardner. *Princeton University Press*.

- **Honsberger R (1990)**. More Mathematical Morsels. *The Mathematical Association of America*.