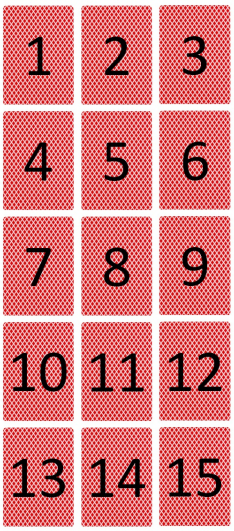
Card dressage

At the start of a **card dressage** game $m \times n$ cards are put face down on the table ordered in m rows of n cards. As shown in the figure below, the cards are numbered from left to right and from top to bottom, starting from one.



Initial configuration of a card dressage game, in which the cards are put face down on the table ordered in five rows of three cards.

solutions

1,3,5,7,9,14

1,2,4,5,12,13

2,3,5,6,10,15

2,7,9,11,13,15

1,6,10,11,13,14

3,4,11,12,14,15

1,2,3,4,6,7,9,10,11,12

4,5,6,7,9,10,12,13,14,15

The aim of the game is to make sure that all cards are turned face up. Cards can only be reversed according to the following rule: if a card is reversed, all its neighbours must be reversed as well. Neighbours are cards to the left, right, top and bottom — if available — but not diagonally. Upon reversal, a card that was face down is turned face up. Conversely, a card that was face up is turned face down after reversal.

Assignment

Define a class CardDressage that can be used to play card dressage games. The objects of this class must at least have the following methods:

- An initialisation method that takes one or two natural numbers. If two numbers are passed to the method, they respectively represent the number of rows and columns in which the cards are ordered. If only a single number is passed to the method, it represents both the number of rows and columns in which the cards are ordered.
- Methods __str__ and __repr__ that both return the same string representation of the object.

This string represents a rectangular grid that indicates the position and current top face of the cards. Cards that are face down are represented with a hash symbol (#). Cards that are face up are represented with a dash symbol (-).

- A method turnCard that can be used to reverse a particular card according to the rules of the card dressage game. The card number must be passed as an argument to the method. In case the card dressage game has no card with the given number, an AssertionError must be raised with the message invalid card number.
- A method turnCards that can be used to reverse a series of cards in a particular order according to the rules of the card dressage game. The numbers of the cards must be passed to the method as a list or tuple of integers. Obviously, the implementation of the method must make use of the method turnCard.
- A method won that takes no arguments. This method must return a Boolean value, which indicates whether or not all cards are face up.

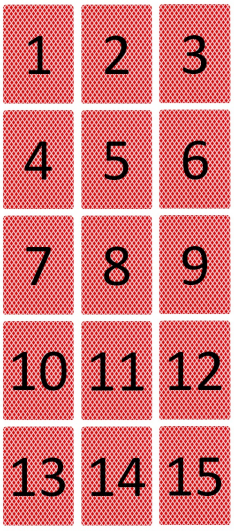
Example

```
>>> game = CardDressage(5, 3)
>>> print(game)
###
###
###
###
###
>>> game.turnCard(1)
>>> game
--#
-##
###
###
###
>>> game.won()
False
>>> game.turnCard(3)
>>> game
-#-
-#-
###
###
###
>>> game.won()
False
>>> game.turnCard(16)
Traceback (most recent call last):
AssertionError: invalid card number
>>> game.turnCard(0)
Traceback (most recent call last):
AssertionError: invalid card number
>>> game.turnCards([5, 7, 9, 14])
>>> game
---
---
---
---
---
>>> game.won()
True
```

```
>>> game = CardDressage(5)
>>> game.turnCards([18, 13, 9, 11, 21, 24, 12, 7, 17, 5, 4, 6, 19, 23, 10])
>>> game
-----
-----
-----
-----
-----
>>> game.won()
True

>>> game = CardDressage(3, 6)
>>> game.turnCards((1, 5, 9, 8, 6, 10))
>>> game
-#-###
##-##-
#---##
>>> game.won()
False
```

Bij de start van een spelletje **kaartdressuur** worden $m \times n$ speelkaarten met hun beeldzijde naar beneden neergelegd in m rijen van n speelkaarten. Zoals aangegeven op onderstaande figuur worden de speelkaarten hierbij startend vanaf één genummerd van links naar rechts en van boven naar onder.



Beginopstelling van een spelletje kaartdressuur, waarbij de kaarten met de beeldzijde naar onder neergelegd worden in vijf rijen van drie speelkaarten.

oplossingen

1,3,5,7,9,14

1,2,4,5,12,13

2,3,5,6,10,15

2,7,9,11,13,15

1,6,10,11,13,14

3,4,11,12,14,15

1,2,3,4,6,7,9,10,11,12

4,5,6,7,9,10,12,13,14,15

De bedoeling van het spel is om ervoor te zorgen dat alle kaarten met hun beeldzijde naar boven

komen te liggen. Hierbij mogen kaarten alleen omgedraaid worden volgens de spelregel: als een kaart omgedraaid wordt, dan moeten ook al zijn buren omgedraaid worden. Buren zijn kaarten links, rechts, boven en onder — indien aanwezig — maar niet diagonaal. Na het omdraaien komt een kaart die met zijn rug naar boven lag, met zijn beeldzijde naar onder te liggen. Omgekeerd wordt een kaart die met zijn beeldzijde naar boven lag omgedraaid zodat zijn rug naar boven komt te liggen.

Opgave

Definieer een klasse Kaartdressuur waarmee spelletjes kaartdressuur kunnen gespeeld worden. De objecten van deze klasse moeten minstens de volgende methoden hebben:

- Een initialisatiemethode waaraan één of twee natuurlijke getallen moeten doorgegeven worden. Indien er twee getallen aan de methode doorgegeven worden, dan stellen deze respectievelijk het aantal rijen en kolommen voor waarin de kaarten neergelegd worden. Indien er slechts één getal aan de methode doorgegeven wordt, dan geeft die zowel het aantal rijen als het aantal kolommen aan waarin de kaarten worden neergelegd.
- Methoden __str__ en __repr__ die beide dezelfde stringvoorstelling van het object teruggeven. Deze stringvoorstelling vormt een rechthoekig rooster dat de positie en huidige bovenzijde van de kaarten weergeeft. Kaarten die met hun rug naar boven liggen, worden aangegeven met een hekje (#). Kaarten die met hun beeldzijde naar boven liggen, worden aangegeven met een koppelteken (-).
- Een methode draaiKaart waarmee een bepaalde kaart kan omgedraaid worden volgens de regels van het spelletje kaartdressuur. Het nummer van de kaart moet als argument aan de methode doorgegeven worden. Indien het spelletje kaartdressuur geen kaart heeft met het opgegeven nummer, dan moet een AssertionError opgeworpen worden met de boodschap ongeldig kaartnummer.
- Een methode draaiKaarten waarmee een reeks kaarten na elkaar kunnen omgedraaid worden volgens de regels van het spelletje kaartdressuur. De nummers van de kaarten moeten als een lijst of tuple aan de methode doorgegeven worden. Uiteraard moet je bij de implementatie van deze methode gebruik maken van de methode draaiKaart.
- Een methode gewonnen waaraan geen argumenten kunnen doorgegeven worden. De methode moet een Booleaanse waarde teruggeven, die aangeeft of alle kaarten van het spelletje kaartdressuur met hun beeldzijde naar boven liggen.

Voorbeeld

```
>>> spel = Kaartdressuur(5, 3)
>>> print(spel)
###
###
###
###
###
>>> spel.draaiKaart(1)
>>> spel
--#
-##
###
###
###
```

```
>>> spel.gewonnen()
False
>>> spel.draaiKaart(3)
>>> spel
-#-
-#-
###
###
###
>>> spel.gewonnen()
False
>>> spel.draaiKaart(16)
Traceback (most recent call last):
AssertionError: ongeldig kaartnummer
>>> spel.draaiKaart(0)
Traceback (most recent call last):
AssertionError: ongeldig kaartnummer
>>> spel.draaiKaarten([5, 7, 9, 14])
>>> spel
---
---
---
---
---
>>> spel.gewonnen()
True

>>> spel = Kaartdressuur(5)
>>> spel.draaiKaarten([18, 13, 9, 11, 21, 24, 12, 7, 17, 5, 4, 6, 19, 23, 10])
>>> spel
-----
-----
-----
-----
-----
>>> spel.gewonnen()
True

>>> spel = Kaartdressuur(3, 6)
>>> spel.draaiKaarten((1, 5, 9, 8, 6, 10))
>>> spel
-#-###
##-##-
#---##
>>> spel.gewonnen()
False
```