Domino tiles

Dominoes is a game played with rectangular domino 'tiles'. Today the tiles are often made of plastic or wood, but in the past, they were made of real stone or ivory. They have a rectangle shape and are divided in 2 square fields, each marked with zero to six pips. In a standard set, all 28 combinations of stones with 0 to 6 pips occur exactly once.

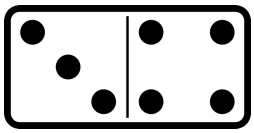
Domino tiles.

The genesis of the game is unclear. Possibly, dominoes originates from China and the stones were brought here by Marco Polo, but this is uncertain.

Assignment

Define a class `Domino` to represent domino tiles in Python. We choose a design where the tiles are *immutable* objects. This way, not a single method may change the internal state of an object after is was initialized. The objects of this class should have at least the following methods:

- An initializing method to which two numbers must be given. We represent a domino tile as two square halves that are next to each other, and the given numbers respectively are the amount of pips on the left and right half.

  The initializing method must print an `AssertionError` with the message `invalid number of pips`, if the number of pips on both halves is not between 0 and 6 (boundaries included).
- A method `__repr__` that prints a string representation of the domino tile. This string representation reads as a Python-expression that makes a new object of the class `Domino` that has the same state as the domino tiles on which the method was called.
- A method `__str__` that prints a string representation of the domino tile. This string representation uses the templates below to compose both halves of the tile based on their number of pips.

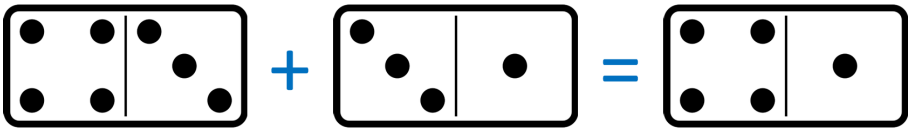
```
+---+   +---+   +---+   +---+   +---+   +---+   +---+
|   |   |   |   |o  |   |o  |   |o o|   |o o|   |ooo|
|   |   | o |   |   |   | o |   |   |   | o |   |   |
|   |   |   |   |  o|   |  o|   |o o|   |o o|   |ooo|
+---+   +---+   +---+   +---+   +---+   +---+   +---+

  0       1       2       3       4       5       6
```

Here, a dot is represented by a lowercase letter `o`. The templates that correspond with the left and right half are put next to each other in the string representation, where the right border of the left half coincides with the left border or the right half.

- A method `rotate` that prints a new tile (object of the class `Domino`), of which the number of pips on the left and right half were switched with regard to the tile on which the method was called.

- A method that allows to add two domino tiles based on the `+` operator. Two tiles can be added if the number of pips on the right of the first stone is equal to the amount of pips on the left half of the second stone. If this isn't the case, the method must print an `AssertionError` with the message `domino tiles do not match`.



Adding two domino tiles results in a new domino tile (object of the class `Domino`) of which the number of pips on the left half is equal to the number of pips on the left half of the first tile, and the number of pips on the right half is equal to the right half of the second tile.

Example

```
>>> tile1 = Domino(3, 4)
>>> Domino(-1, 7)
Traceback (most recent call last):
AssertionError: invalid number of pips

>>> tile1
Domino(3, 4)
>>> print(tile1)
+---+---+
|o  |o o|
| o |   |
|  o|o o|
+---+---+
>>> print(tile1.rotate())
+---+---+
|o o|o  |
|   | o |
|o o|  o|
+---+---+
>>> print(tile1)
+---+---+
|o  |o o|
| o |   |
|  o|o o|
+---+---+

>>> tile2 = Domino(1, 3)
>>> tile1 + tile2
Traceback (most recent call last):
AssertionError: domino tiles do not match
>>> print(tile2 + tile1)
```

```
+---+---+
|   |o o|
| o |   |
|   |o o|
+---+---+
>>> tile3 = tile1.rotate() + tile2.rotate()
>>> tile3
Domino(4, 1)
>>> print(tile3)
+---+---+
|o o|   |
|   | o |
|o o|   |
+---+---+
```

Domino is een legspel dat met speciale dominostenen gespeeld wordt. De stenen zijn tegenwoordig vaak van plastic of hout gemaakt, maar vroeger werden ze ook vervaardigd van echte steen of ivoor. Ze hebben een rechthoekige vorm en zijn verdeeld in 2 vierkante veldhelften, ieder gemerkt met nul tot zes ogen. In de standaardset komen alle 28 combinaties van stenen met 0 tot 6 ogen juist eenmaal voor.

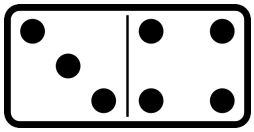
Dominostenen.

De ontstaansgeschiedenis van het spel is onduidelijk. Mogelijk vindt het dominospel zijn oorsprong in China en zijn de stenen door Marco Polo daarvandaan meegenomen, maar dit valt niet met zekerheid vast te stellen.

Opgave

Definieer een klasse `Domino` waarmee dominostenen kunnen voorgesteld worden in Python. We kiezen voor een ontwerp waarbij dominostenen onveranderlijke (*immutable*) objecten zijn. Daardoor mag geen enkele methode de interne toestand van een object aanpassen nadat het werd geïnitialiseerd. De objecten van deze klasse moeten minstens de volgende methoden hebben:

- Een initialisatiemethode waaraan twee getallen moeten doorgegeven worden. We stellen een dominosteen voor als twee vierkante helften die naast elkaar staan, en de gegeven getallen stellen dan respectievelijk het aantal ogen op de linker- en de rechterhelft voor.

  De initialisatiemethode moet een `AssertionError` opwerpen met de boodschap `ongeldig aantal ogen`, indien het aantal ogen op beide helften van de dominosteen niet telkens tussen 0 en 6

(grenzen inbegrepen) gelegen is.

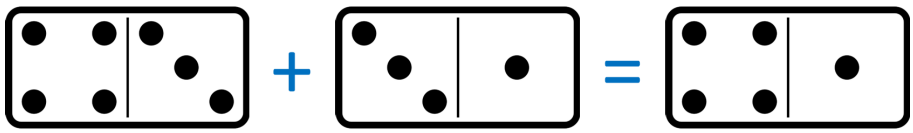
- Een methode `__repr__` die een stringvoorstelling van de dominosteen teruggeeft. Deze stringvoorstelling leest als een Python-expressie die een nieuw object aanmaakt van de klasse `Domino` dat dezelfde toestand heeft als de dominosteen waarop de methode aangeroepen wordt.
- Een methode `__str__` die een stringvoorstelling van de dominosteen teruggeeft. Deze stringvoorstelling maakt gebruik van onderstaande sjablonen om elk van beide helften van de steen voor te stellen op basis van hun aantal ogen.

```
+---+   +---+   +---+   +---+   +---+   +---+   +---+
|   |   |   |   |o  |   |o  |   |o o|   |o o|   |ooo|
|   |   | o |   |   |   | o |   |   |   | o |   |   |
|   |   |   |   |  o|   |  o|   |o o|   |o o|   |ooo|
+---+   +---+   +---+   +---+   +---+   +---+   +---+

  0       1       2       3       4       5       6
```

  Hierbij wordt een oog voorgesteld door de kleine letter `o`. De sjablonen die overeenkomen met de linker- en de rechterhelft worden naast elkaar gezet in de stringvoorstelling, waarbij de rechterrand van de linkerhelft samenvalt met de linkerrand van de rechterhelft.
- Een methode `draai` die een nieuwe dominosteen (object van de klasse `Domino`) teruggeeft, waarvan het aantal ogen op de linker- en rechthelft omgewisseld werd ten opzichte van de dominosteen waarop de methode werd aangeroepen.

- Een methode die toelaat om twee dominostenen bij elkaar op te tellen aan de hand van de `+` operator. Twee dobbelstenen kunnen enkel bij elkaar opgeteld worden als het aantal ogen aan de rechterkant van de eerste steen gelijk is aan het aantal ogen aan de linkerkant van de tweede steen. Indien dit niet het geval is, dan moet de methode een `AssertionError` opwerpen met de boodschap `dominostenen passen niet`.



  De optelling van twee dominostenen resulteert in een nieuwe dominosteen (object van de klasse `Domino`) waarvan het aantal ogen op de linkerhelft gelijk is aan het aantal ogen op de linkerhelft van de eerste steen, en het aantal ogen op de rechterhelft gelijk is aan het aantal ogen op de rechterhelft van de tweede steen.

Voorbeeld

```
>>> steen1 = Domino(3, 4)
>>> Domino(-1, 7)
Traceback (most recent call last):
AssertionError: ongeldig aantal ogen

>>> steen1
Domino(3, 4)
>>> print(steen1)
+---+---+
|o  |o o|
```

```
| o |   |
|  o|o o|
+---+---+
>>> print(steen1.draai())
+---+---+
|o o|o  |
|   | o |
|o o|  o|
+---+---+
>>> print(steen1)
+---+---+
|o  |o o|
| o |   |
|  o|o o|
+---+---+

>>> steen2 = Domino(1, 3)
>>> steen1 + steen2
Traceback (most recent call last):
AssertionError: dominostenen passen niet
>>> print(steen2 + steen1)
+---+---+
|   |o o|
| o |   |
|   |o o|
+---+---+
>>> steen3 = steen1.draai() + steen2.draai()
>>> steen3
Domino(4, 1)
>>> print(steen3)
+---+---+
|o o|   |
|   | o |
|o o|   |
+---+---+
```