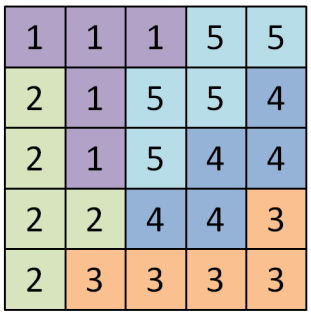
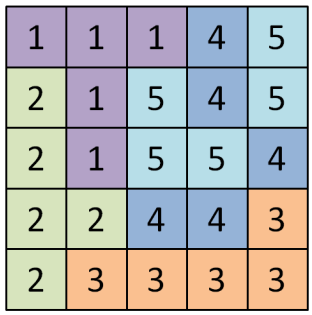
Equidivision

Consider a square $n \times n$ grid in which each cell is labeled. Labels are either integers or strings. Cells with the same label form a group. Labeling the cells results in an **equidivision** if each group has n cells and is connected. Two cells are contiguous when they have a common side. A group is connected if all its cells thus form a contiguous unit. Below, we show two 5×5 grids whose cells have been labeled with the integers in the range 1 to 5. All groups have 5 cells, but all groups are connected only in the left grid. The left grid thus is an equidivision, where the right grid is not (groups 4 and 5 are not connected).





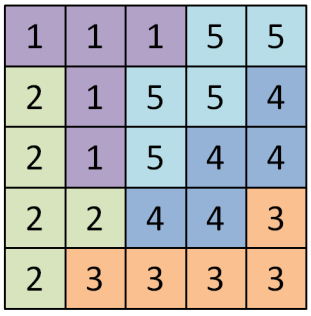
Assignment

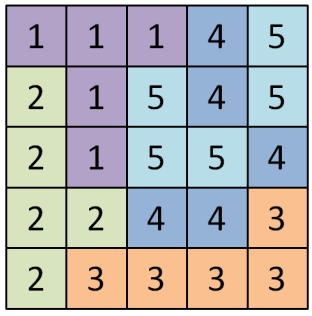
- Write a function `groups` that takes a square grid of labeled cells as an argument. The grid is described by a list of lists, where the inner lists represent the rows, and the elements of those lists represent the labels of the corresponding cells. The function must return a dictionary that maps each label used in the grid to the set of positions of cells in the grid that carry the label. The position of a cell is described by a tuple (r, c), with r the row number and c the column number of the cell in the grid. Rows and columns are numbered starting from 0.
- Write a function `connected` that takes a container type object (list, tuple, set, …) as an argument. This container describes the positions of the cells in a the row number and c the column number of the cell in the grid. The function must return a Boolean value that indicates whether or not the given group is connected.
- Use the functions `groups` and `connected` to write a function `equidivision`. The function must return a Boolean value that indicates whether or not a given square grid of labeled cells is an equidivision. The given grid must be passed as an argument to the function, and is described in the same way as the argument of the function `groups`.

Example

```
>>> grid = [[1, 1, 1], [2, 2, 3], [2, 3, 3]]
>>> groups(grid)
{1: {(0, 1), (0, 0), (0, 2)}, 2: {(2, 0), (1, 0), (1, 1)}, 3: {(1, 2), (2, 1), (2, 2)}}

>>> grid = [[1, 1, 1, 5, 5], [2, 1, 5, 5, 4], [2, 1, 5, 4, 4], [2, 2, 4, 4, 3], [2, 3, 3, 3, 3]]
>>> groups(grid)
{1: {(0, 1), (0, 0), (0, 2), (2, 1), (1, 1)}, 2: {(3, 0), (2, 0), (1, 0), (3, 1), (4, 0)}, 3: {(4, 1), (4, 4), (3, 4), (4, 3), (4, 2)}, 4: {(3, 2), (2, 3), (2, 4), (1, 4), (3, 3)}, 5: {(1, 2), (0, 3), (1, 3), (2, 2), (0, 4)}}

>>> connected({(1, 2), (1, 3), (2, 2), (0, 3), (0, 4)})
True
>>> connected({(1, 2), (1, 4), (2, 2), (0, 3), (0, 4)})
False

>>> grid = [[1, 1, 1], [2, 2, 3], [2, 3, 3]]
>>> equidivision(grid)
True

>>> grid = [[1, 1, 1], [2, 2, 3], [3, 3, 2]]
>>> equidivision(grid)
False

>>> grid = [[1, 1, 1, 5, 5], [2, 1, 5, 5, 4], [2, 1, 5, 4, 4], [2, 2, 4, 4, 3], [2, 3, 3, 3, 3]]
>>> equidivision(grid)
True
```

Beschouw een vierkant $n \times n$ rooster waarvan elke cel gelabeld is. De labels kunnen zowel integers als strings zijn. Cellen die hetzelfde label dragen vormen een groep. De labels vormen een **evendeling** van het rooster als elke groep bestaat uit n cellen en als de groep samenhangend is. Twee cellen worden aangrenzend genoemd als ze een gemeenschappelijke zijde hebben, en op die manier moeten de cellen van een samenhangende groep een aangrenzend geheel vormen. Onderstaande figuur toont twee 5×5 roosters, waarvan de cellen gelabeld zijn met de getallen tussen 1 en 5. Hierbij ontstaan telkens groepen van 5 cellen, maar enkel in het linker rooster zijn die allemaal samenhangend. Het linker rooster is dus een evendeling, het rechter rooster is dat niet (groepen 4 en 5 zijn niet samenhangend).





Opgave

- Schrijf een functie groepen waaraan een vierkant gelabeld rooster moet doorgegeven worden. Dit rooster wordt beschreven door een lijst van lijsten, waarbij de binnenste lijsten de rijen vormen, en waarvan de elementen de labels van de corresponderende cellen voorstellen. De functie moet een dictionary teruggeven die elk label van het rooster afbeeldt op een verzameling met de posities van de cellen in het rooster die dat label dragen. De positie van een cel wordt beschreven door een tuple (r, k), waarbij r het rijnummer van de cel in het rooster aangeeft, en k het kolomnummer. Rijen en kolommen worden genummerd vanaf 0.
- Schrijf een functie samenhangend waaraan een container (lijst, tuple, verzameling, …) moet doorgegeven worden. Deze container omschrijft de posities van de cellen in een groep. De positie van een cel wordt beschreven door een tuple (r, k), waarbij r het rijnummer van de cel in het rooster aangeeft, en k het kolomnummer. De functie moet een Booleaanse waarde teruggeven, die aangeeft of de cellen van de gegeven groep al dan niet samenhangend zijn.
- Gebruik de functies groepen en samenhangend om daarmee een functie evendeling te schrijven. Deze functie moet een Booleaanse waarde teruggeven die aangeeft of een gegeven vierkant gelabeld rooster een evendeling vormt of niet. Het gegeven rooster moet als argument aan de functie doorgegeven worden, en wordt op dezelfde manier omschreven als bij de functie groepen.

Voorbeeld

```
>>> rooster = [[1, 1, 1], [2, 2, 3], [2, 3, 3]]
>>> groepen(rooster)
{1: {(0, 1), (0, 0), (0, 2)}, 2: {(2, 0), (1, 0), (1, 1)}, 3: {(1, 2), (2, 1), (2, 2)}}

>>> rooster = [[1, 1, 1, 5, 5], [2, 1, 5, 5, 4], [2, 1, 5, 4, 4], [2, 2, 4, 4, 3], [2, 3, 3, 3, 3]]
>>> groepen(rooster)
{1: {(0, 1), (0, 0), (0, 2), (2, 1), (1, 1)}, 2: {(3, 0), (2, 0), (1, 0), (3, 1), (4, 0)}, 3: {(4, 1), (4, 4), (3, 4), (4, 3), (4, 2)}, 4: {(3, 2), (2, 3), (2, 4), (1, 4), (3, 3)}, 5: {(1, 2), (0, 3), (1, 3), (2, 2), (0, 4)}}

>>> samenhangend({(1, 2), (1, 3), (2, 2), (0, 3), (0, 4)})
True
>>> samenhangend({(1, 2), (1, 4), (2, 2), (0, 3), (0, 4)})
False

>>> rooster = [[1, 1, 1], [2, 2, 3], [2, 3, 3]]
>>> evendeling(rooster)
True

>>> rooster = [[1, 1, 1], [2, 2, 3], [3, 3, 2]]
>>> evendeling(rooster)
False

>>> rooster = [[1, 1, 1, 5, 5], [2, 1, 5, 5, 4], [2, 1, 5, 4, 4], [2, 2, 4, 4, 3], [2, 3, 3, 3, 3]]
>>> evendeling(rooster)
True
```