Fotocollage - Versie 1 - E2023

LET OP: In deze versie van de opgave is het **NIET** toegelaten om `for`- of `while`-constructies te gebruiken. Gebruik in plaats hiervan het NumPy-taaleigen. Indien je gebruik wil maken van `for` of `while` moet je in versie 2 van de opgave (voor een lager aantal punten) indienen. Het verbeterscript zorgt dat de module `numpy` beschikbaar is via het standaardpatroon `import numpy as np`. Die hoef je NIET in de code op te nemen. Het importeren van NumPy via `from numpy import *` is **EXPLICIET VERBODEN** in deze oefening (levert problemen in het verbeterscript).

Gegeven zijn vier 2D NumPy-tabellen, hierna `A`, `B`, `C` en `D` genoemd, die elk een afbeelding voorstellen (de tabellen bevatten dus gehele getallen tussen 0 en 255, grenzen inbegrepen). We willen deze vier afbeeldingen samenvoegen in 1 tabel, met M rijen en N kolommen. Hiertoe plaatsen we afbeelding `A` in de linkerbovenhoek, afbeelding `B` in de rechterbovenhoek, afbeelding `C` in de linkerbenedenhoek en afbeelding `D` in de rechterbenedenhoek.

Indien er hierdoor gaten zouden ontstaan in de uiteindelijke afbeelding, dan vullen we die gaten op met nul-elementen. Indien afbeeldingen zouden overlappen in de uiteindelijke figuur, dan plaatsen we op de desbetreffende plaatsen de gemiddelde waarde van de overlappende afbeeldingen, naar beneden afgekapt (zodat je steeds een geheel getal uitkomt voor een beeldelement).

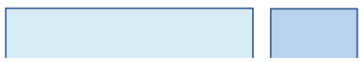
Programmeer de functie `collage()` met als argumenten:

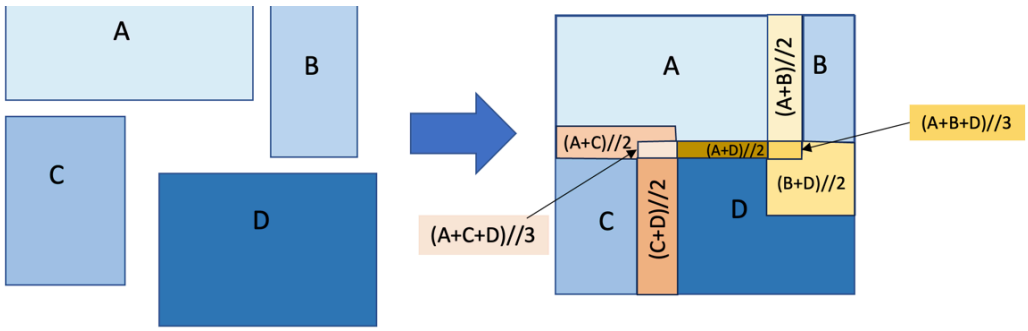
- `l`: een lijst met de afbeeldingen `A`, `B`, `C` en `D`, allen NumPy-tabellen, in die volgorde
- `M`: het aantal rijen in de uiteindelijke afbeelding
- `N`: het aantal kolommen in de uiteindelijke afbeelding

TIP: Het kan handig zijn om bij het construeren van de nieuwe afbeelding per beeldpunt bij te houden hoeveel originele figuren op dit punt overlappen.

Verder mag je aannemen dat M strikt groter is dan het aantal rijen van elk van de originele afbeeldingen. Analoog is N strikt groter dan het aantal kolommen van elk van de originele afbeeldingen.

Onderstaande figuur verduidelijkt het samenvoegen van de vier deelafbeeldingen tot 1 afbeelding.





TIP: een NumPy-tabel `a` die reële getallen bevat, kan je omzetten naar een tabel `b` van gehele getallen (door afkapping) via `np.int_(a)`

Het Dodona-verbeterscript maakt gebruik van de functie `table2str()` die je hieronder ook terugvindt.

```
def table2str(tabel):
    f = '%d\t'
    s = [''.join([f%j for j in i]) + '\n' for i in tabel]
    t = ''.join(s)[:-1]
    return t
```

Voorbeeld

```
A = np.array([[150, 171, 2, 206, 111, 151], [15, 53, 222, 22, 17, 242], [77, 38,
B = np.array([[46, 93, 228, 204, 169], [171, 38, 39, 84, 190], [84, 12, 175, 11,
C = np.array([[88, 97, 15, 85, 169], [69, 24, 99, 227, 239], [83, 201, 39, 145, 3
D = np.array([[219, 75, 64, 115, 211], [118, 122, 44, 101, 129], [94, 242, 124, 2
print(table2str(A))
# 150   171     2       206     111     151
# 15    53      222     22      17      242
# 77    38      189     189     101     15
# 139   108     90      143     10      24
print(table2str(B))
# 46    93      228     204     169
# 171   38      39      84      190
# 84    12      175     11      159
# 16    155     91      32      141
# 143   170     234     138     109
print(table2str(C))
# 88    97      15      85      169
# 69    24      99      227     239
# 83    201     39      145     30
print(table2str(D))
# 219   75      64      115     211
# 118   122     44      101     129
# 94    242     124     246     206
# 216   71      211     146     111
# 124   8       159     59      78
# 148   132     110     212     110
# 106   79      202     234     154
# 119   21      172     139     233
l = [A, B, C, D]
c = collage(l, 14, 11)
print(table2str(c))
# 150   171     2       206     111     151     46      93      228     204     1
# 15    53      222     22      17      242     171     38      39      84      1
# 77    38      189     189     101     15      84      12      175     11      1
# 139   108     90      143     10      24      16      155     91      32      1
# 0     0       0       0       0       0       143     170     234     138     1
# 0     0       0       0       0       0       0       0       0       0       0
# 0     0       0       0       0       0       219     75      64      115     2
# 0     0       0       0       0       0       118     122     44      101     1
```

```
# 0     0       0       0       0       0       94      242     124     246     2
# 0     0       0       0       0       0       216     71      211     146     1
# 0     0       0       0       0       0       124     8       159     59      7
# 88    97      15      85      169     0       148     132     110     212     1
# 69    24      99      227     239     0       106     79      202     234     1
# 83    201     39      145     30      0       119     21      172     139     2
```

```
A = np.array([[85, 177, 243, 136, 21, 14, 151], [20, 223, 91, 248, 194, 113, 8],
B = np.array([[138, 166, 162, 42, 34], [185, 57, 53, 40, 252], [222, 81, 25, 85,
C = np.array([[20, 71, 237, 32], [153, 97, 24, 225], [211, 178, 190, 140], [128,
D = np.array([[72, 223, 144, 88], [159, 66, 216, 97], [53, 10, 28, 99], [171, 48,
print(table2str(A))
# 85    177     243     136     21      14      151
# 20    223     91      248     194     113     8
# 249   218     207     27      151     33      148
# 103   229     111     138     231     232     96
# 227   7       42      144     65      63      63
print(table2str(B))
# 138   166     162     42      34
# 185   57      53      40      252
# 222   81      25      85      18
print(table2str(C))
# 20    71      237     32
# 153   97      24      225
# 211   178     190     140
# 128   162     161     71
```