28 Reunião 28: 03/DEZ/2020

Figure 1: Yoda, copiado daqui

28.1 Reuniões passadas

Matrizes

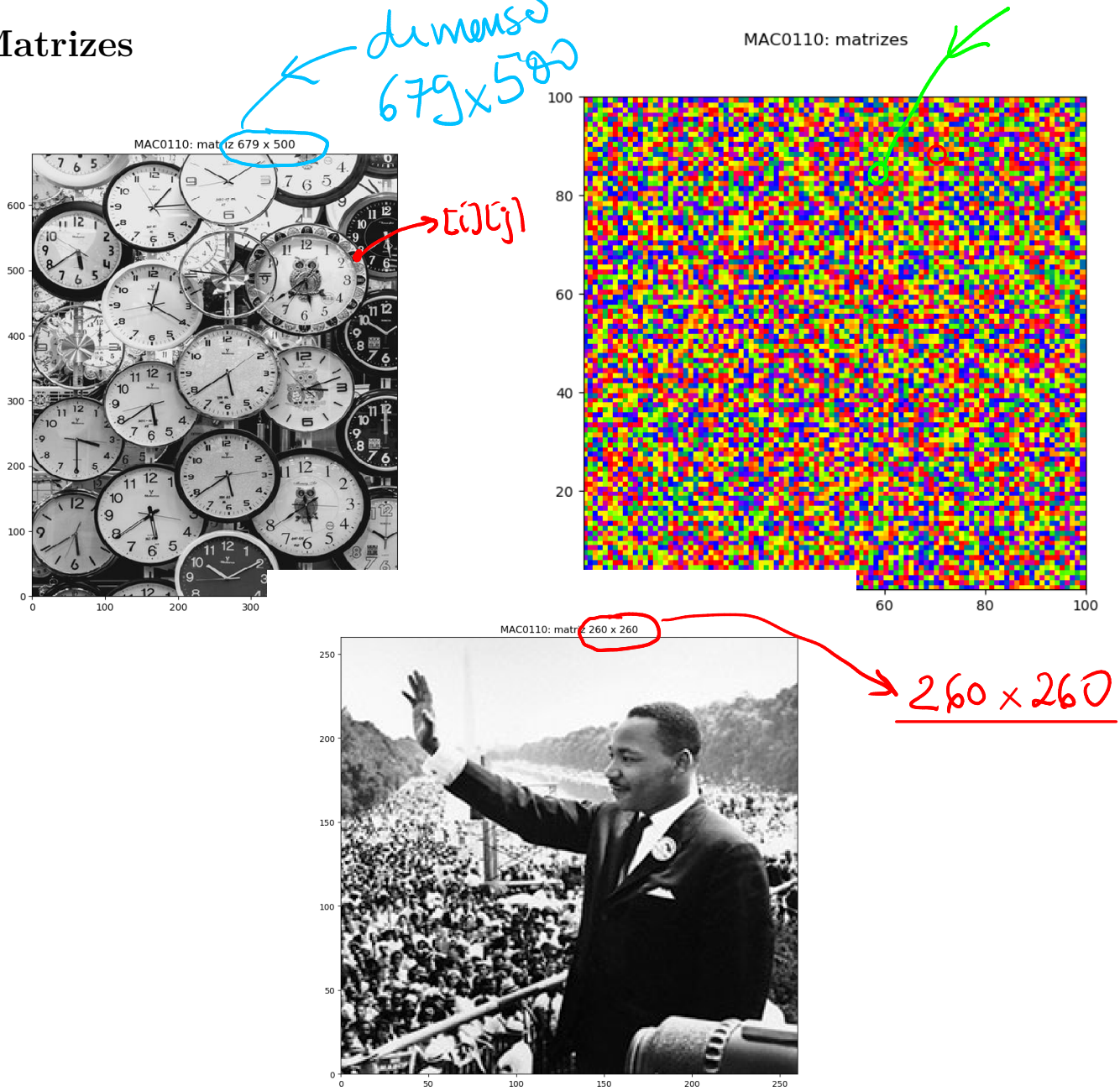
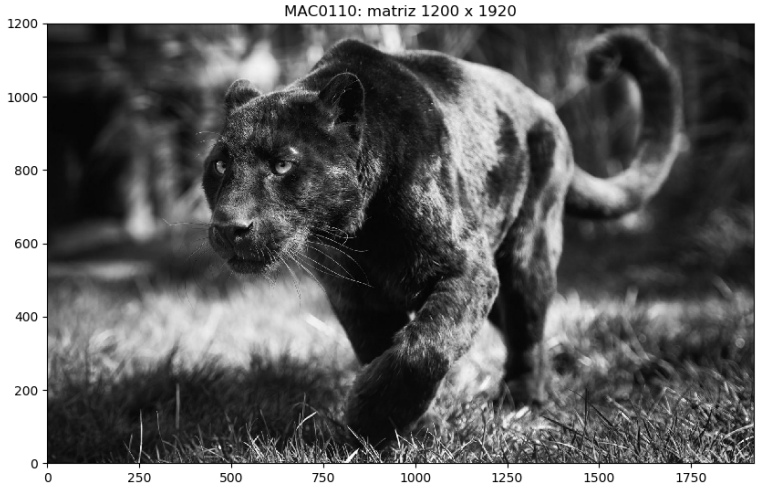
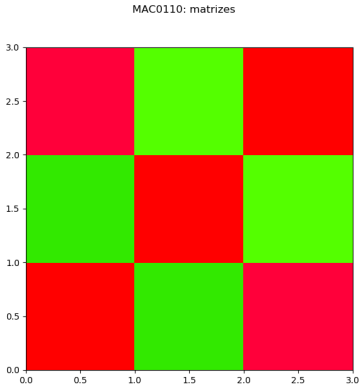
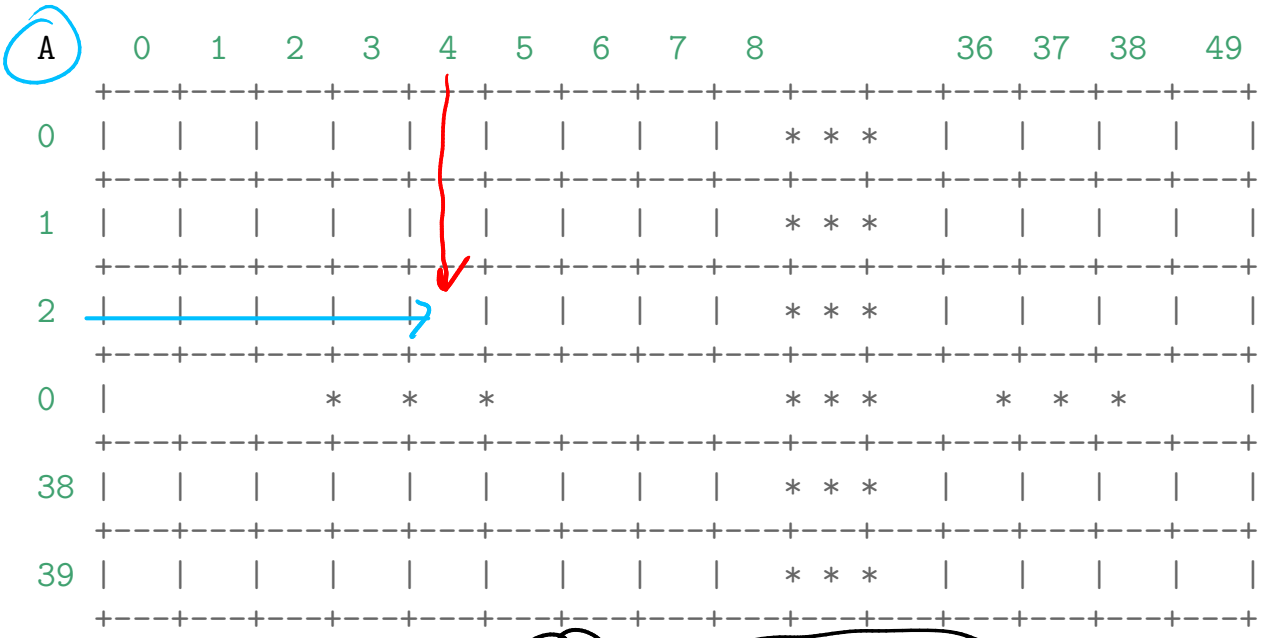


Figure 2: *Dr King*


MAC0110: matriz 1200 x 1920
1200
1000
800
600
400
200
0
0
250
500
750
1000
1250
1500
1750


MAC0110: matrizes
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0.0
0.5
1.0
1.5
2.0
2.5
3.0

MATRIZES



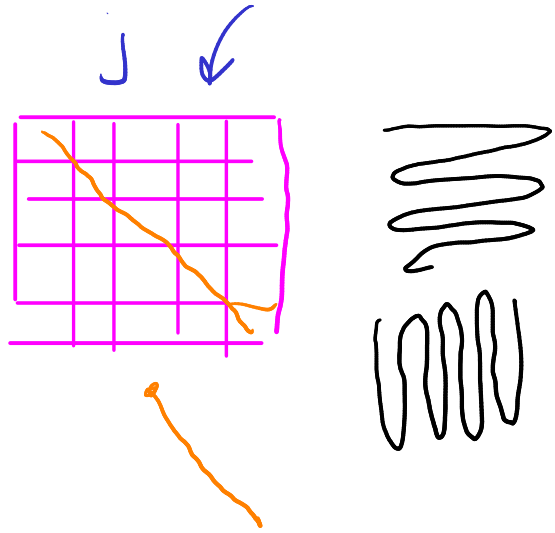
nlins = 40
ncols = 50

Matrizes em Python

Em Python, uma matriz pode ser representada como uma lista de listas, onde elemento da lista contém uma linha da matriz, que por sua vez corresponde a uma lista com os elementos da coluna da matriz.

Exemplos

```
In [1]:  matriz = [[10, 11, 12, 13],[20, 21, 22, 23], [30, 31, 32
In [2]: matriz
Out[2]: [[10, 11, 12, 13], [20, 21, 22, 23], [30, 31, 32, 33]]

In [3]: matriz[1]
Out[3]: [20, 21, 22, 23]

In [4]: matriz[1][0]
Out[4]: 21

In [5]: matriz[1][-1]
Out[5]: 23

In [6]: matriz[1][5]
---------------------------------------------------------------
IndexError                                Traceback (most recent
<ipython-input-6-fa3662a2b8eb> in <module>
----> 1 matriz[1][5]
IndexError: list ind
```

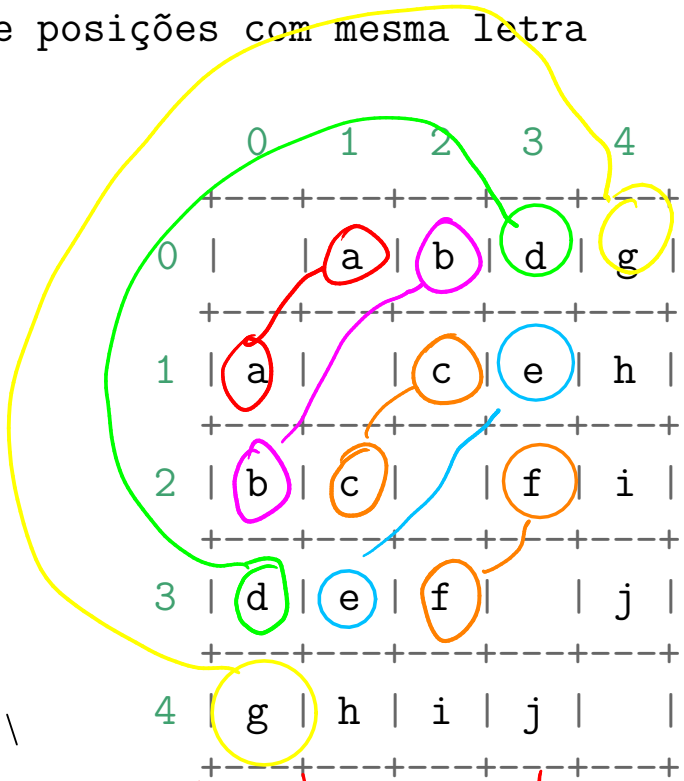


simetrica()

```
#-----------------------------------------------
def simetrica(mt):
    '''(matriz) -> bool
    RETORNA True se matriz é simétria e False em
    '''
    # pegue a dimensão da matriz
    nlin = len(mt)
    ncol = len(mt[0])

    # percorrer matriz linha por linha
    for i in range(1, nlin):
        for j in range(i):
            if mt[i][j] != mt[j][i]: #(*)
                print(f"{mt[i][j]}!={mt[j][i]}")
                return False # acaba aqui
    return True
```

Em (*) comparações apenas entre posições com mesma letra

i	for j in range(i)
1	0
2	0 1
3	0 1 2
4	0 1 2 3
5	0 1 2 3 4



Vá para o laboratório "importação de módulos" (simetrica.py)

import matriz as mat

28.2 Exercício: `to_str()`

Escreva uma função `to_str()` que recebe uma matriz (= list[list]) `mt` e retorna uma string que para ser usada por `print()` para exibir a matriz de um forma estruturada:

matriz.py

Exemplos

```
In [2]: mt = [[ 1, 22, 333], [33, 8, -555], [1, 2, 3, 4]]
In [3]: print(to_str(mt))
Matriz: 3 x 3
   1   22  333
  33    8 -555
   1    2    3
```

Vá para o laboratório "Exercício: to_str()"

```
In [4]: mt = [[ 13], [33], [-111]]
In [5]: print(to_str(mt))
Matriz: 3 x 1
   13
   33
 -111
```

"Matriz: 3 x 1\n—13—33—-111"

Solução

```
def to_str(mt):
    '''(matriz) -> str

    RECEBE uma matriz `mt`.
    RETORNA uma string que para ser usada por print() para
    exibir a matriz.
    '''
    s = ""
    nlin = len(mt)
    ncol = len(mt[0])

    # pegue o maior número caracteres para escreve um valor
    max_len = max_len_valor(mt) + 1 # mais 1 para um espaço

    s += "Matriz: %d x %d\n" %(nlin, ncol)
    for i in range(0, nlin, +1):
        for j in range(0, ncol, +1):
            s +=  f"{mt[i][j]:{max_len}}"
        # pule uma linha
        s += "\n"

    return s
```

28.3 Exercício: `init_matriz()`

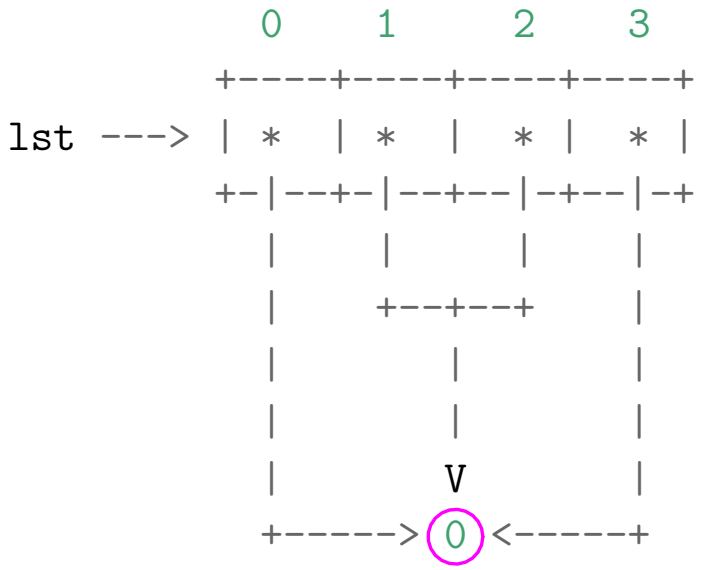
Escreva uma função `init_matriz()` que **recebe** dois inteiros `nlins` e `ncols` e um valor `val` e **cria** e retorna uma matriz de dimensão `nlins x ncols` com o valor `val` em cada posição.

Exemplos

```
In [6]: mt = init_matriz(3, 5)

In [7]: print(to_str(mt))
Matriz: 3 x 5
 0 0 0 0 0
 0 0 0 0 0
 0 0 0 0 0


In [8]: mt = init_matriz(3, 5, 1.2)

In [9]: print(to_str(mt))
Matriz: 3 x 5
 1.2 1.2 1.2 1.2 1.2
 1.2 1.2 1.2 1.2 1.2
 1.2 1.2 1.2 1.2 1.2

n [10]: mt
Out[10]:
[[1.2, 1.2, 1.2, 1.2, 1.2],
 [1.2, 1.2, 1.2, 1.2, 1.2],
 [1.2, 1.2, 1.2, 1.2, 1.2]]

In [11]: mt = init_matriz(1, 6, 'a')

In [12]: mt
```

```
Out[12]: [['a', 'a', 'a', 'a', 'a', 'a']]
```

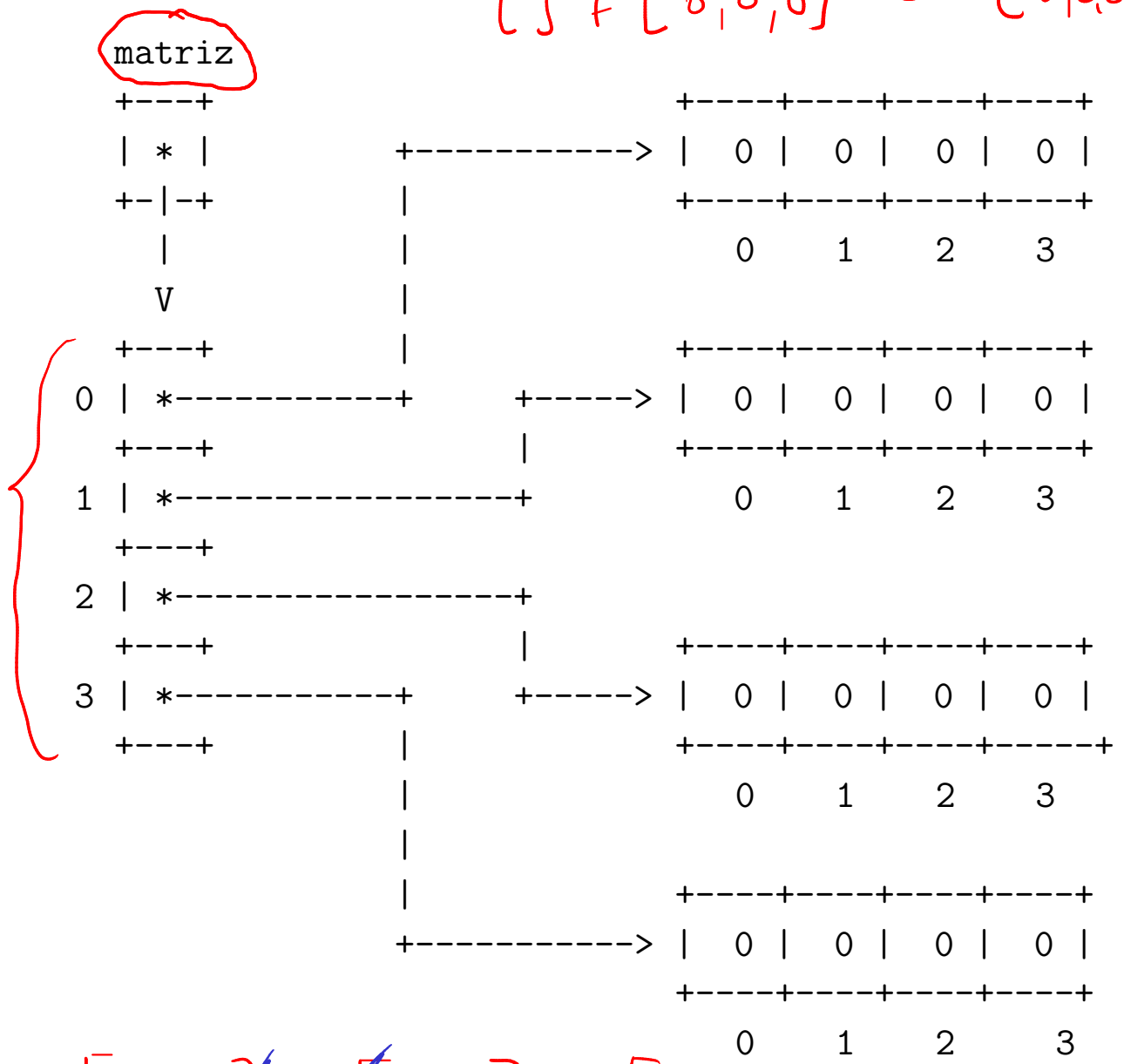
Rascunhos



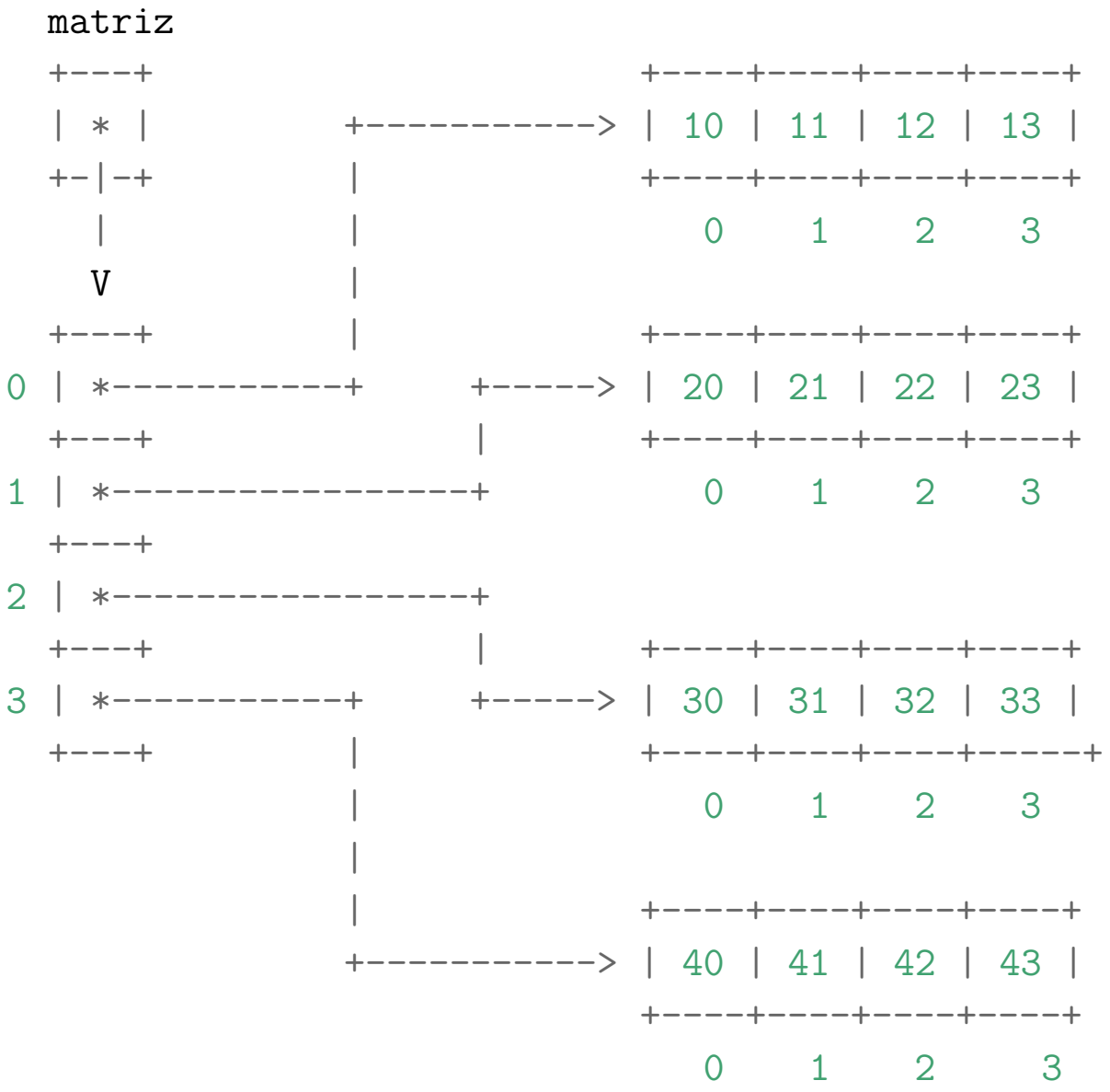

```
lst = [ 0,   0,    0,   0 ]
```

Rascunhos

```
init_matriz(4, 4, 0)
```



Rascunhos


matriz
0
1
2
3
10 11 12 13
0 1 2 3
20 21 22 23
0 1 2 3
30 31 32 33
0 1 2 3
40 41 42 43
0 1 2 3

Solução

```
def init_matriz(nlin, ncol, val=0):
    ''' (int, int, obj) -> matriz (list de list)

    RECEBE dois inteiros `nlin`, `ncol` e um valor `val`.
    RETORNA uma matriz de dimensão `nlin` x `ncol` em que
    todas as posições tem `val`
    Exemplo:
    In [1]: mat = init_matriz(2,3)
    In [2]: mat
    Out[2]: [[0, 0, 0], [0, 0, 0]]
    '''
    mt = []
    # crie a matriz
    for i in range(nlin):
        # crie uma linha com ncol itens
        linha = ncol*[val] # [val] + [val] +...+[val]
        # coloque na matriz
        mt += [linha]
    return mt
```