25 Reunião 25: 15/JUL/2021

Figure 1: Fonte: Calvi e Hobbes por Bill Watterson

25.1 Reuniões passadas

Figure 2: Desenho copiado daqui

Um **dicionário** (tipo `dict`) é um conjunto de objetos ou itens cada um dotado de uma `chave` e de um `valor`.

txt = arq.read()

for linha in arq:

Também andamos manipulando **arquivos**. Fizemos a **leitura** de arquivos de arquivos e **criamos** um arquivo `.csv` com o conteúdo de um dicionário.

arq.write(f"{linha}\n")

25.2 Hoje

Começaremos a conversar sobre como representar matriz (=tabelas bidimensionais em Python). Hoje, nosso principal foco será em **percorrer matrizes** e nos habituarmos com seus índices e suas posições.

Sempre ter em mente que:

- atribuições não criam nada, apenas criam ou alteram um apelido;
- posições de listas e de matrizes são apelidos para coisas/objetos;
- "**antes** de entrar no elevador, verifique se ele está parado **neste andar**"

25.3 Matrizes

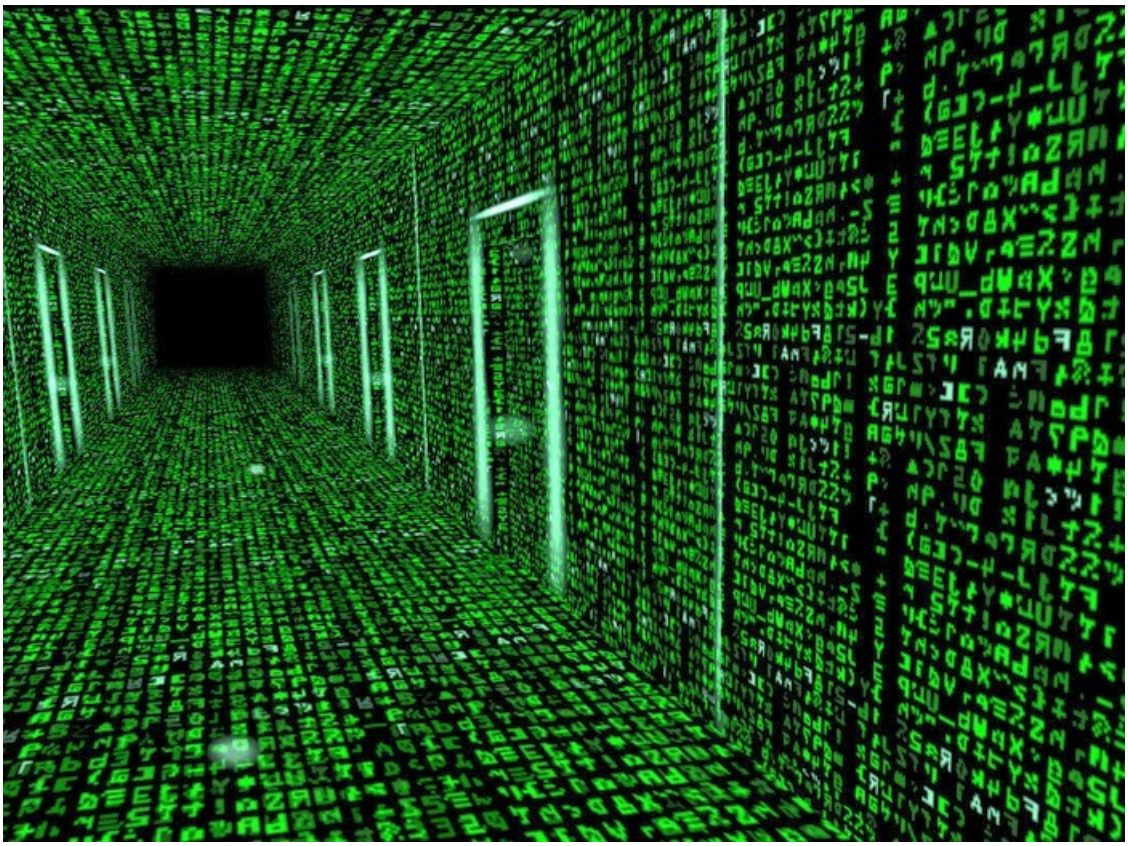
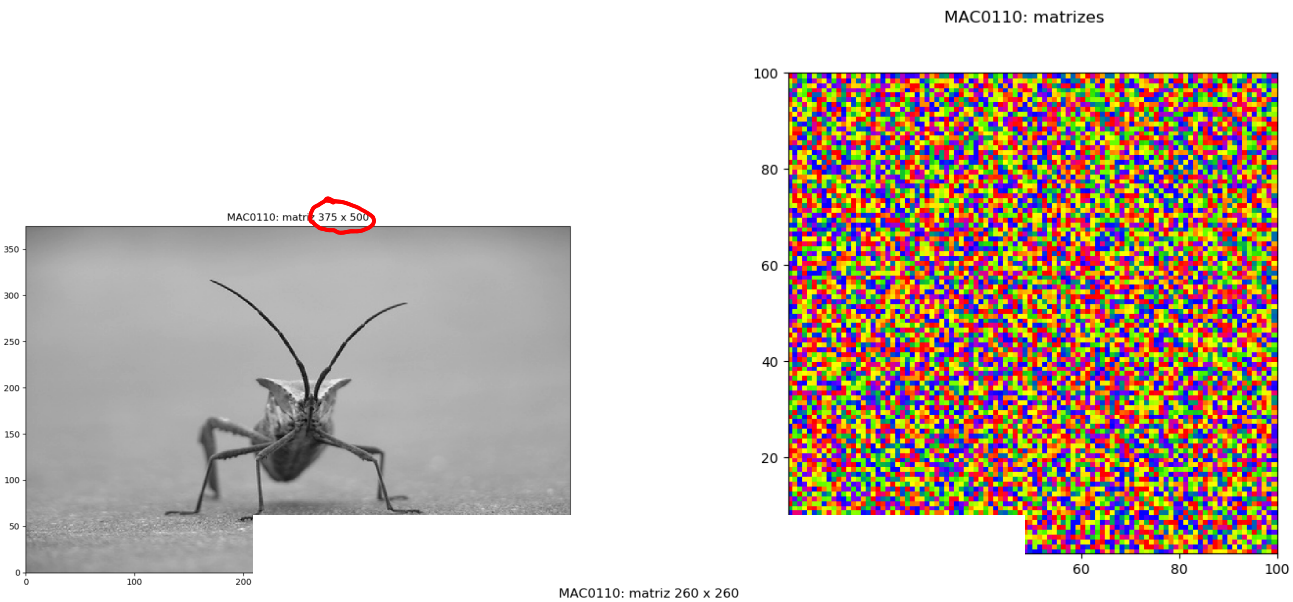
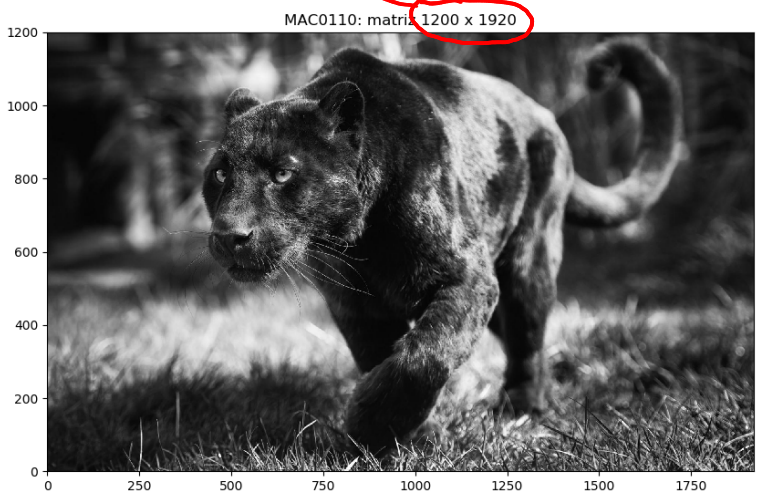


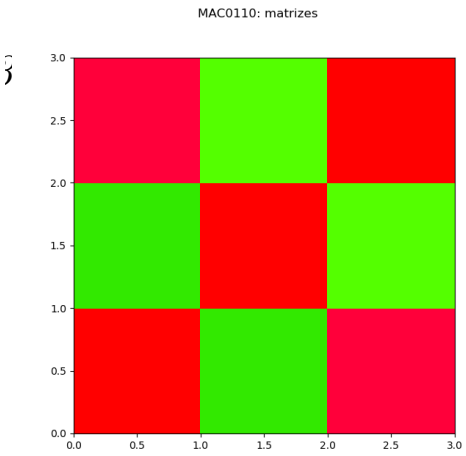
Figure 3: *Matrix*: matriz 522 x 700

Matrizes são estruturas bidimensionais (tabelas) com `nlins` linhas por `ncols` colunas muito utilizadas para a resolução de sistemas de equações, transformações lineares, representar imagens, etc.



2 304 000 ~ 2.3 milhões de valores





matriz 3x3

25.4 Matrizes em Python

Em Python, a maneira mais básica de se representar uma matriz é através de uma lista de listas (`list[list]`). Cada elemento da lista é um apelido para uma lista que representa uma linha da matriz. Cada linha da matriz é uma lista com os elementos de cada coluna na linha matriz.

Nota: a representação de matrizes através de lista de listas, dependendo da aplicação, pode ter problemas sérios de desempenho. Por esse motivo, é comum que seja usado o módulo numpy para processar enormes matrizes de números. Este módulo é conveniente para computação científica e evita muitas das ineficiências da representação através de lista de lista. Este será um dos primeiros tópicos que abordaremos em MAC0122 no próximo semestre.

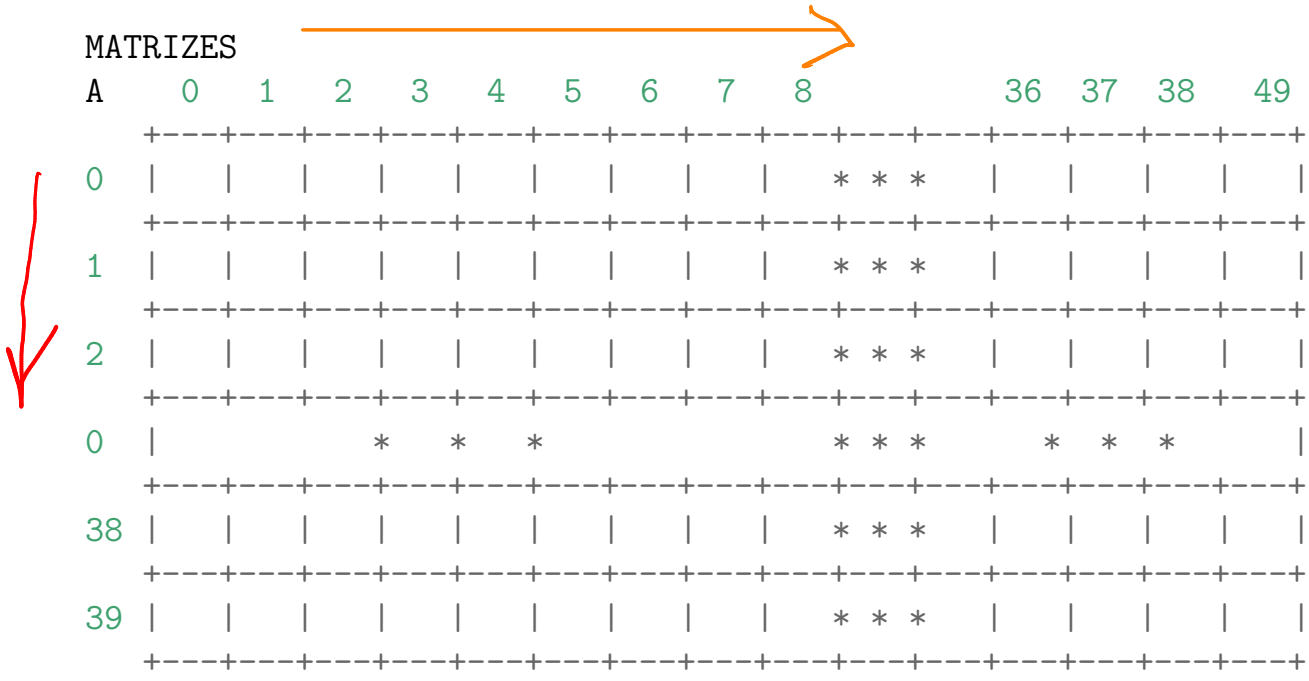
25.4.1 Exemplos

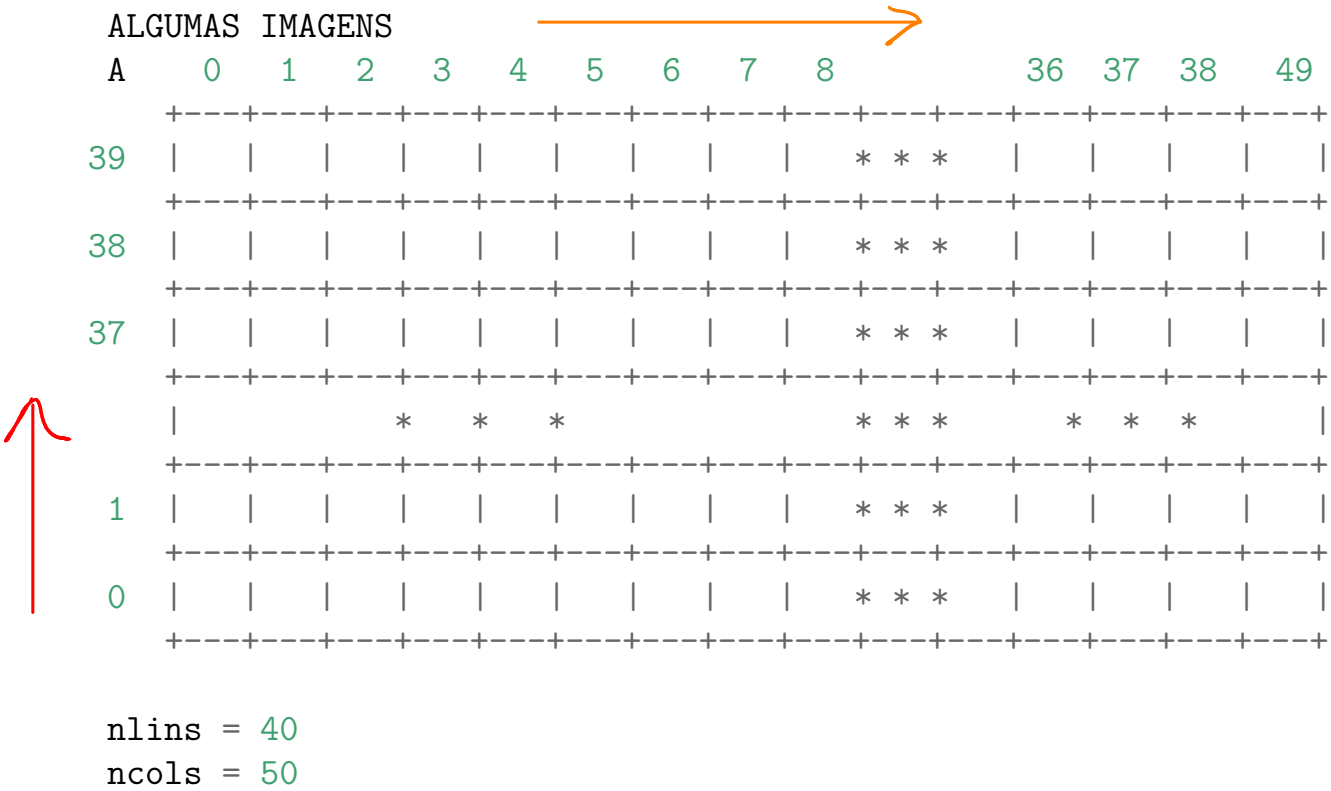
```
In [1]:  matriz = [[1,2,3,4],[5,6,7,8],[9,10,11,12]]

In [2]: matriz
Out[2]: [[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]]

In [3]: matriz[1]
Out[3]: [5, 6, 7, 8]

In [4]: matriz[1][0]
Out[4]: 5

In [5]: matriz[1][-1]
Out[5]: 8

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

```
----> 1 matriz[1][5]

IndexError: list ind
```

Vizualização

É comum vermos matrizes como tabelas bidimensionais.





25.5 Exercício: matrizes simétricas

Escreva uma função `simetrica()` que **recebe** um matriz *quadrada* e **retorna** `True` se a matriz é simétrica e `False` em caso contrários.

Exemplo

Para a matriz a seguir a resposta é `True`

```
     0     1    2    3
   +-----+-----+----+----+
0  | 11  | -3  | 4  | 8  |
   +-----+-----+----+----+
1  | -3  | 12  | 6  | 11 |
   +-----+-----+----+----+
2  |  4  |  6  | 5  | 13 |
   +-----+-----+----+----+
3  |  8  | 11  | 13 | 5  |
   +-----+-----+----+----+
```

Solução

```
def simetrica(mt):
    '''(matriz) -> bool

    RECEBE uma matriz quadrada representada por lista de
           listas (list[list]).
    RETORNA True se a matriz for simétrica, em caso
           contrário retorna False.

    Pré-condição: a função supões que a matriz e quadrada

    In [10]: a = [[1,2,3],[2,1,4],[3,4,1]]
    In [11]: a
    Out[11]: [[1, 2, 3], [2, 1, 4], [3, 4, 1]]
    In [12]: simetrica(a)
    Out[12]: True
    '''
    n = len(mt)
    e_simetrica = True

    i = 0
    while i < n and e_simetrica:
        j = 0
        while j < i and e_simetrica:
            if mt[i][j] != mt[j][i]:
                e_simetrica = False
                print("mt[{i}][{j}]={mt[i][j]}!={mt[j][i]}="\
                      +"mt[{j}][{i}]")
            j = j + 1
        i = i + 1

    return e_simetrica
```

versão com while

na reunião usamos for

25.6 Exercício: linhas com `val`

Escreva uma função `linhas_val()` que **recebe** um matriz `mt` e um valor `val` e **retorna** o número de linhas em que todos os valores são iguais a `val` 

Exemplo

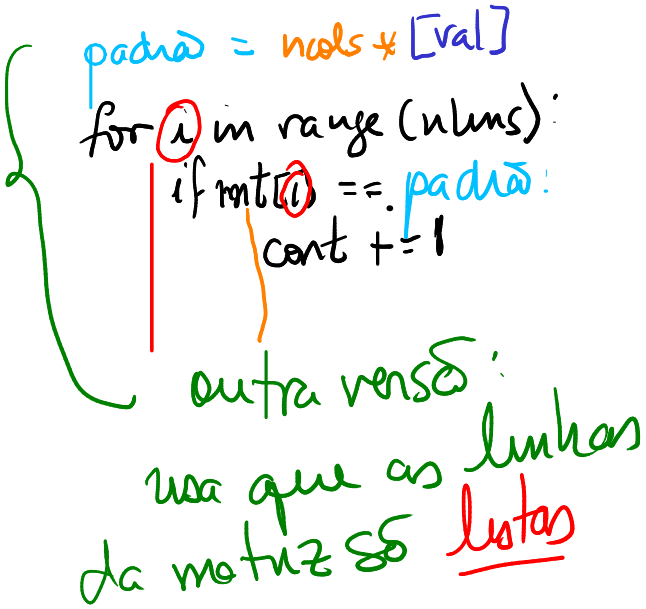
Para a matriz a seguir

- se `val = 11` a resposta é `2` e
- se `val = 5` a resposta é `0`.

```
     0     1    2    3     4
   +-----+-----+----+----+----+
0  | 11  | 11  | 11 | 11 | 11 |
   +-----+-----+----+----+----+
1  | -3  | 12  | 6  | 11 | 34 |
   +-----+-----+----+----+----+
2  | 11  | 11  | 11 | 11 | 11 |
   +-----+-----+----+----+----+
3  |  8  | 11  | 13 | 5  |  7 |
   +-----+-----+----+----+----+
```

Solução

```
#-----------------------------------------------------------
def linhas_val(mt, val):
    '''(matriz, obj) -> int

    RECEBE uma matriz mt representada por lista de
           listas (list[list]) e um valor val.
    RETORNA o número de linhas da mt em que todos os valores
           são iguais a val.

    In [10]: a = [[1,1,1],[2,1,4],[1,1,1]]
    In [11]: a
    [[1, 1, 1], [2, 1, 4], [1, 1, 1]]
    In [12]: linhas_val(a, 1)
    Out[12]: 2
    '''
    cont = 0
    nlins = len(mt)
    ncols = len(mt[0])

    for i in range(nlins):
        iguais = True
        for j in range(ncols):
            if mt[i][j] != val:
                iguais = False
        if iguais:
            cont += 1

    return cont
```



25.7 Exercício: colunas com `val`

Escreva uma função `colunas_val()` que **recebe** um matriz `mt` e um valor `val` e **retorna** o número de colunas em que todos os valores são iguais a `val`.

Exemplo

Para a matriz e seguir

- se `val = 11` a resposta é `2` e
- se `val = 5` a resposta é `1`.

```
     0     1    2    3     4
   +-----+-----+----+----+----+
0  | 11  | 11  |  5 | 11 |  7 |
   +-----+-----+----+----+----+
1  |  5  | 12  |  5 | 11 |  6 |
   +-----+-----+----+----+----+
2  | 11  | 11  |  5 | 11 | 17 |
   +-----+-----+----+----+----+
3  |  5  | 11  |  5 | 11 | 11 |
   +-----+-----+----+----+----+
```

Solução

```
def colunas_val(mt, val):
    '''(matriz, obj) -> int

    RECEBE uma matriz `mat` representada por lista de
           listas (list[list]) e um valor `val`.
    RETORNA o número de colunas de `mt` em que todos os valores
           são iguais a `val`.

    In [10]: a = [[1,1,1],[2,1,4],[1,1,1]]
    In [11]: a
    [[1, 1, 1], [2, 1, 4], [1, 1, 1]]
    In [12]: colunas_val(a, 1)
    Out[12]: 1
    '''
    cont = 0
    nlins = len(mt)
    ncols = len(mt[0])

    for j in range(ncols):
        iguais = True
        for i in range(nlins):
            if mt[i][j] != val:
                iguais = False
        if iguais:
            cont += 1

    return cont
```

25.8 Exercício: diagonais com `val`

Escreva uma função `diagonais_val()` que **recebe** um matriz *quadrada* `mt` e um valor `val` e **retorna** `True` se a matriz é a diagonal principal e secundária de `mt` tem todos seus valores iguais a `val`. Em caso contrário a função **retorna** `False`.

não fizemos no resumo

Exemplos

Para a matriz a seguir

- se `val = 11` a resposta é `True` e
- para qualquer outro valor a resposta é `False`.

```
      0     1    2    3
    +-----+-----+----+----+
0   | 11  | -3  | 4  | 11 |
    +-----+-----+----+----+
1   | -3  | 11  | 11 | 45 |
    +-----+-----+----+----+
2   |  4  | 11  | 11 |  8 |
    +-----+-----+----+----+
3   | 11  | 11  | -1 | 11 |
    +-----+-----+----+----+
```

Solução

```
def diagonais_val(mt, val):
    '''(matriz, obj) -> int

    RECEBE uma matriz quadrada `mt` representada por lista de
           listas (list[list]) e um valor `val`.
    RETORNA True se todos os valores da diagonal principal e sec
           de `mt` são iguais a `val`. Em caso contrário a funçã
           False.

    In [10]: a = [[1,1,1],[2,1,4],[1,1,1]]
    In [11]: a
    [[1, 1, 1], [2, 1, 4], [1, 1, 1]]
    In [12]: diagonais_val(a, 1)
    Out[12]: True
    '''
    iguais = True
    nlins = len(mt)

    for i in range(nlins):
        if mt[i][i] != val or mt[nlins-1-i][i] != val:
            iguais = False

    return iguais
```