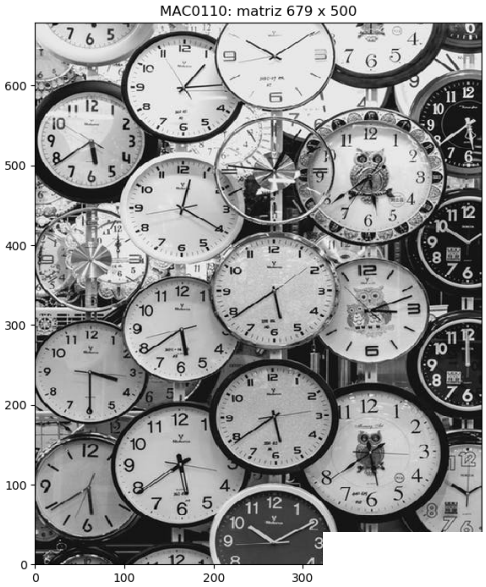
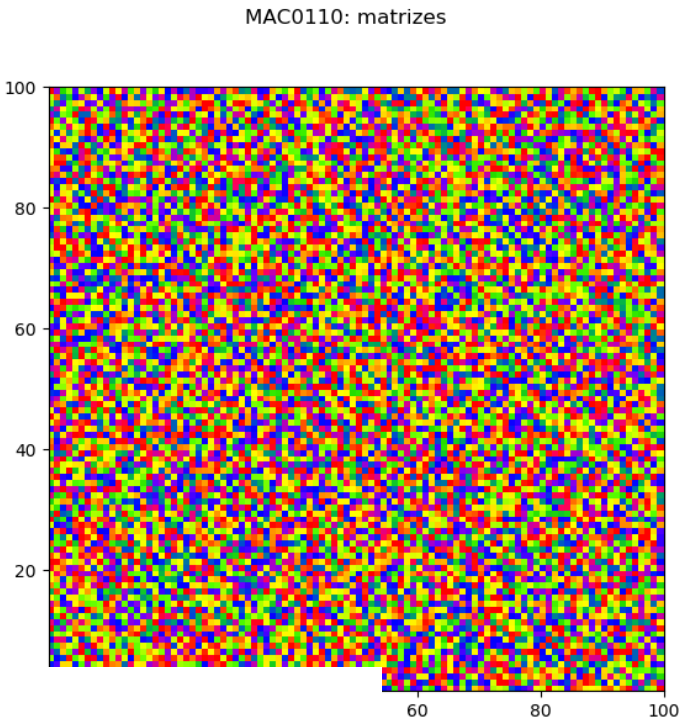
26 Reunião 26: 20/JUL/2021

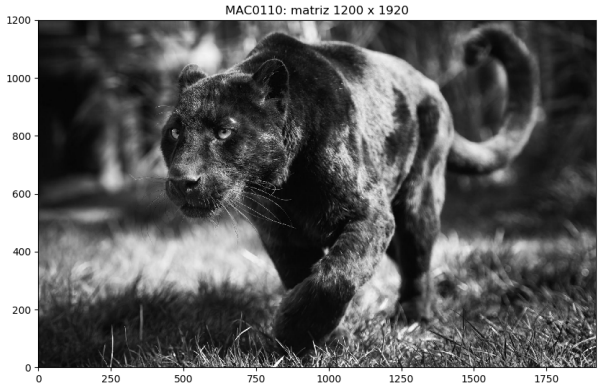
Figure 1: Fonte: Calvi e Hobbes por Bill Watterson

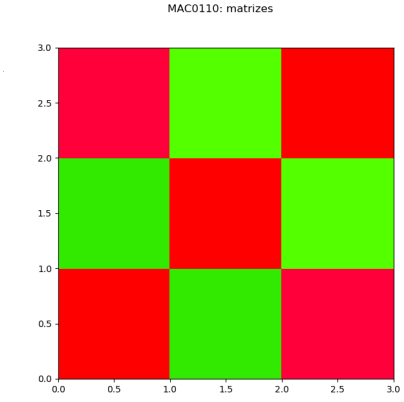
26.1 Reunião passada

Matrizes









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)
    ncols = 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

	0	1	2	3	4
0		a	b	d	g
1	a		c	e	h
2	b	c		f	i
3	d	e	f		j
4	g	h	i	j	

26.2 Hoje

Continuaremos a conversar sobre matrizes (=tabelas bidimensionais).

Sempre ter em mente que: crucial... difícil com listas de listas

- atribuções 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 **no andar**"

26.3 Exercício: `str_matriz()`

Escreva uma função `str_matriz()` 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:

Exemplos

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

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

Solução

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

    RECEBE uma matriz mt.
    RETORNA uma string que para ser usada por print() para
    exibir a matriz.
    '''
    s = ""
    nlins = len(mt)
    ncols = 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: {nlins} x {ncols}"
    for i in range(0, nlins, +1):
        for j in range(0, ncols, +1):
            s +=  f"{mt[i][j]:{max_len}}"
        # pule uma linha
        s += "\n"

    return s
```

26.4 Exercício: `init_matriz()`

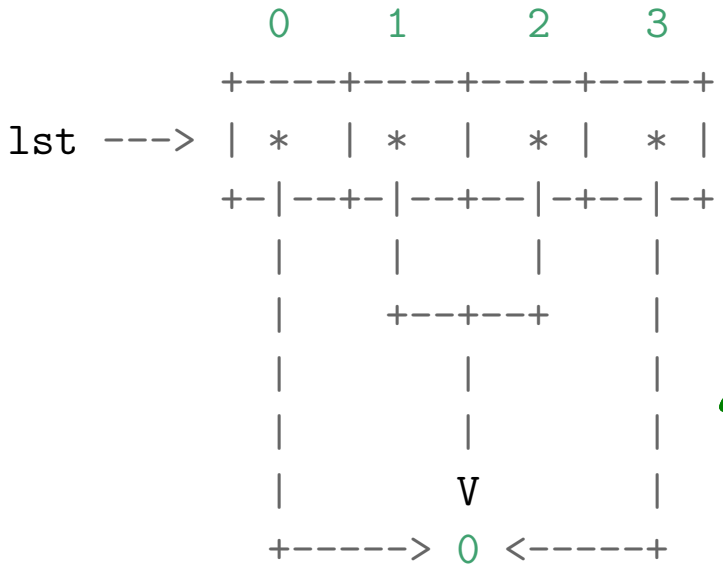
Escreva uma função `init_matriz()` que **recebe** dois inteiros `nlins` e `ncols` e um valor `val` e **cria** e retorna uma matriz de dimensão `nlins x ncols` com o valor `val` em cada posição.

Absolutamente fundamental para manipular matrizes representadas por listas de listas

Exemplos

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

In [7]: print(str_matriz(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(str_matriz(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

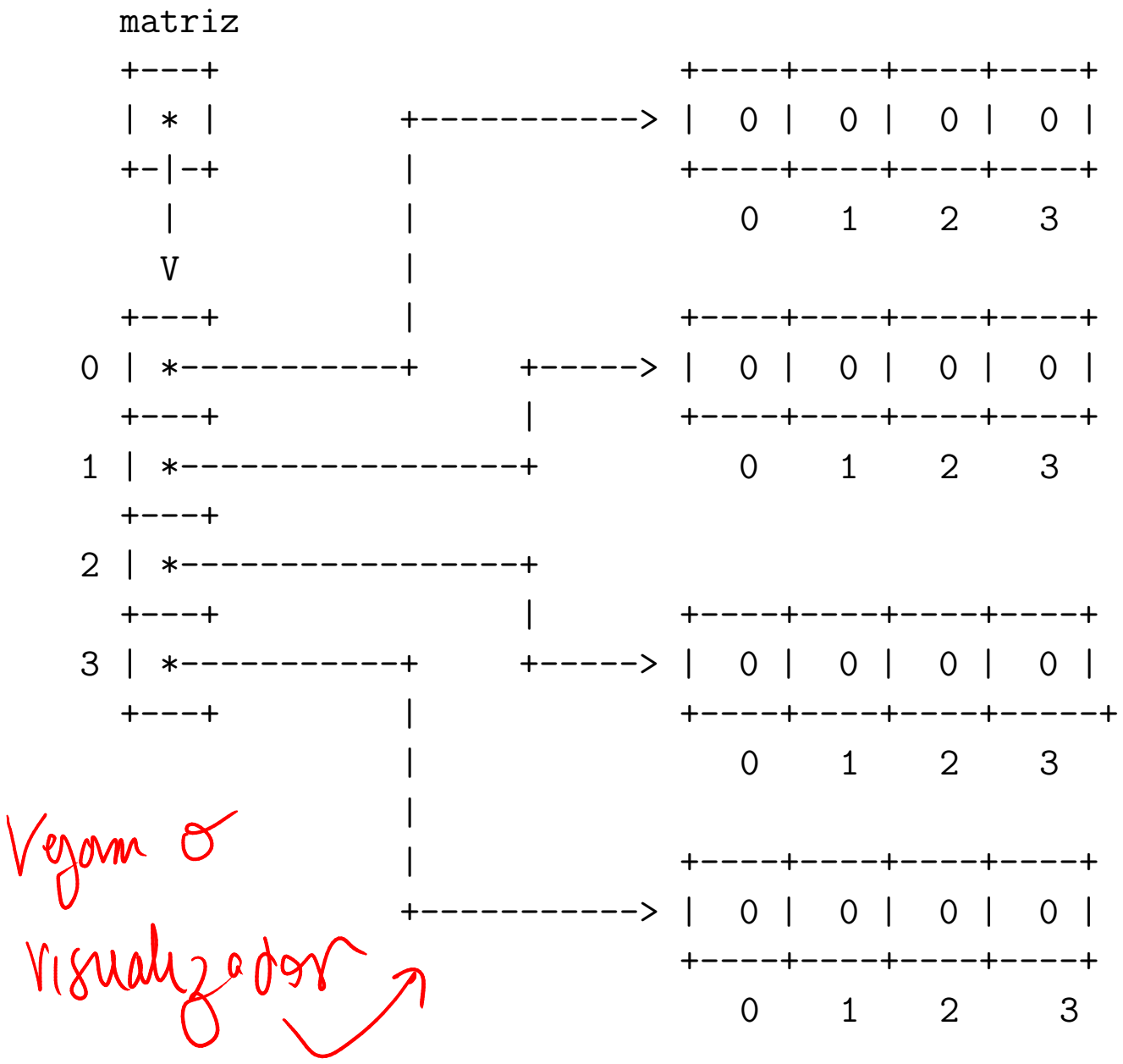


A vida como ela é

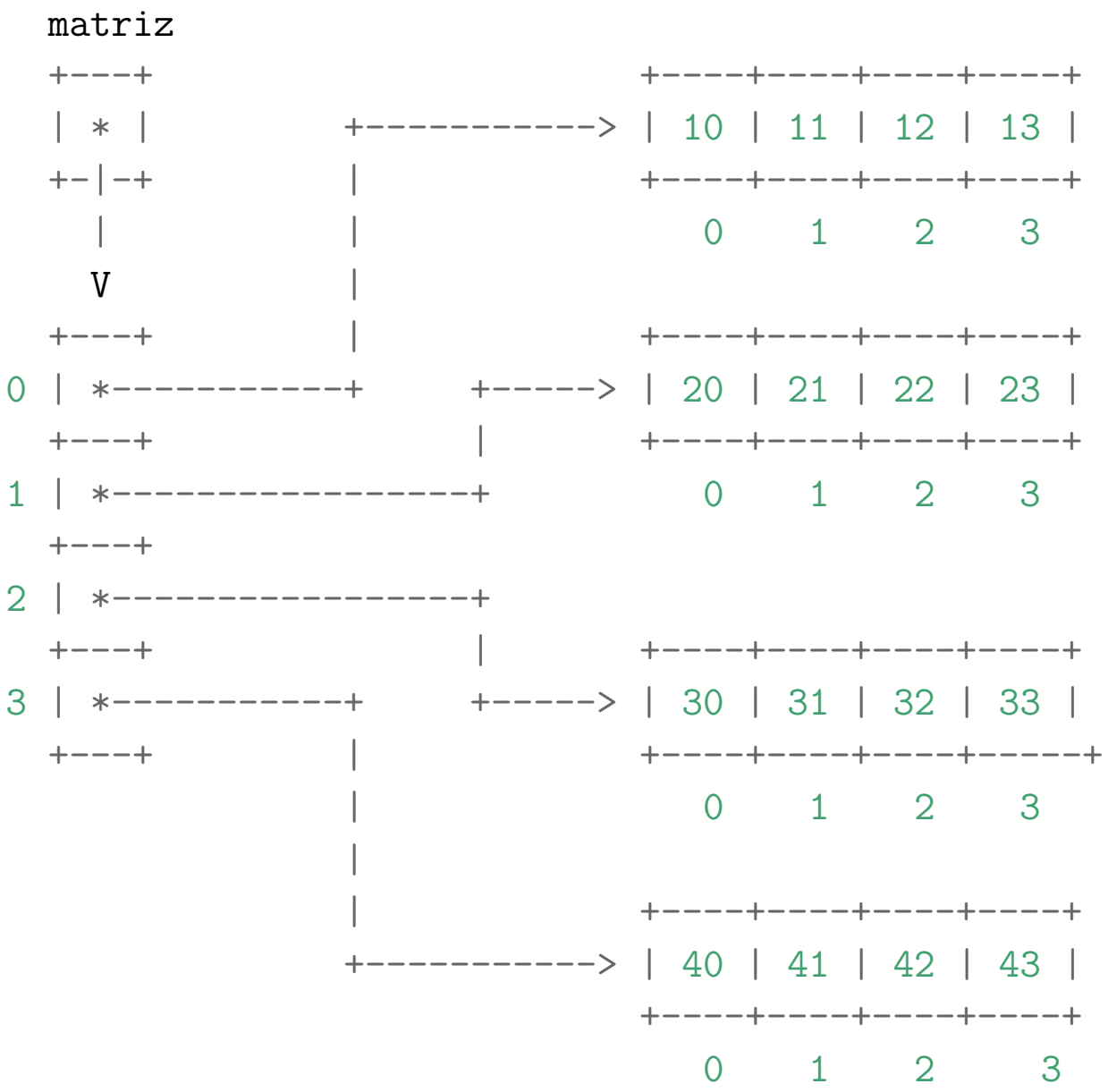
```
lst = [ 0,   0,    0,   0 ]
```

Rascunhos

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

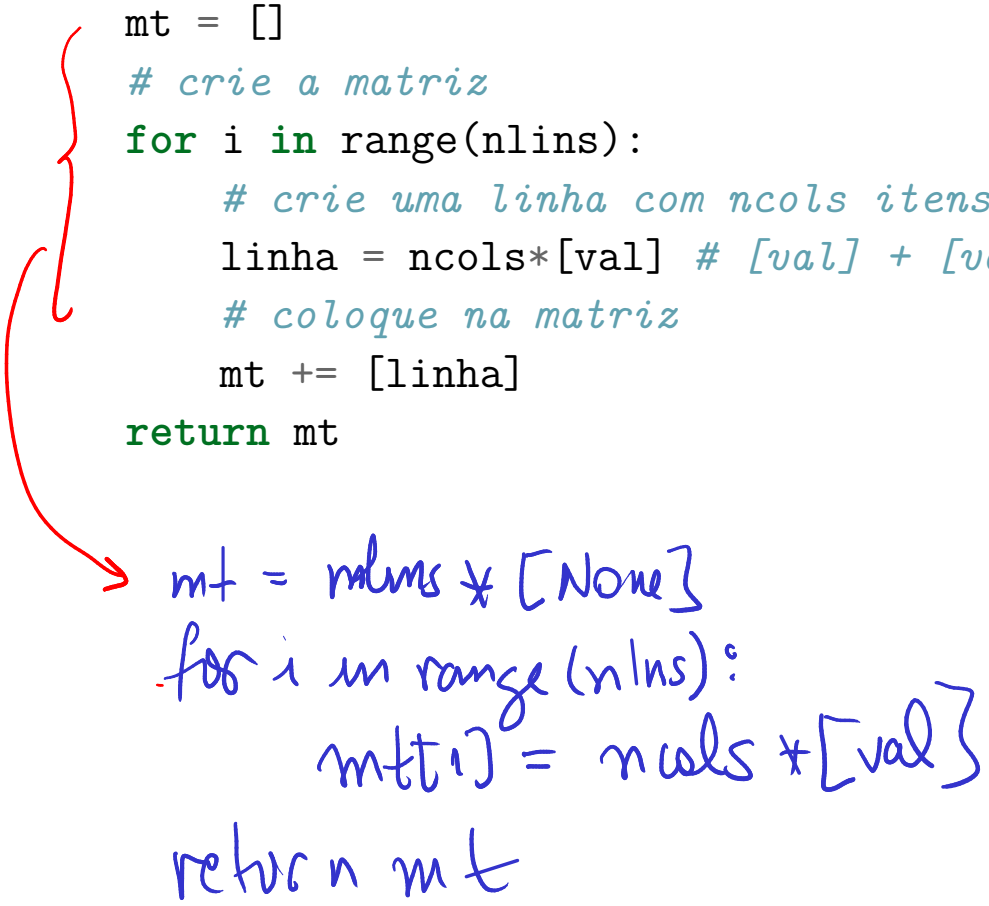


Rascunhos


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(nlins, ncols, val=0):
    ''' (int, int, obj) -> matriz (list de list)

    RECEBE dois inteiros `nlins`, `ncols` e um valor `val`.
    RETORNA uma matriz de dimensão `nlins` x `ncols` 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(nlins):
        # crie uma linha com ncols itens
        linha = ncols*[val] # [val] + [val] +...+[val]
        # coloque na matriz
        mt += [linha]
    return mt
```



26.5 Exercício: `leia_matriz()`

```
#-------------------------------------------------
def leia_matriz():
    '''(None) -> matriz (list de list)
    Lê uma matriz através do teclado e retorna uma
    lista de lista representando a matriz
    '''
    # 1 leia as dimensões
    nlins = int(input("Digite o no. de linha: "))
    ncols = int(input("Digite o no. de colunas: "))

    # 2 crie a matriz
    matriz = crie_matriz(nlins, ncols) #!!!!!!!! <3

    # 3 preencha a matriz
    for i in range(nlins):
        # leia linha i
        for j in range(ncols):
            valor = int(input(f"Digite elem [{i}][{j}]: "))
            matriz[i][j] = valor

    return matriz
```

exiba_matriz() e str_matriz()

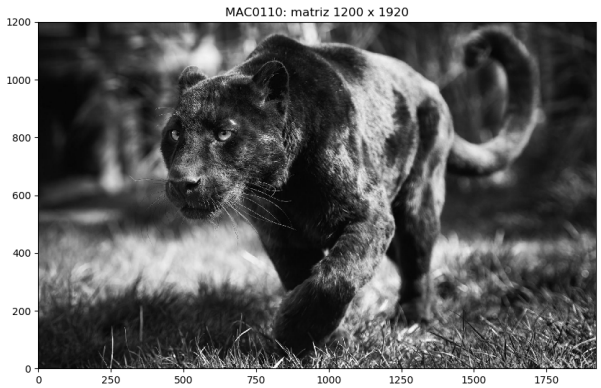
```
#-----------------------------------------------------------
def exiba_matriz(matriz):
    '''(matriz) -> None

    RECEBE e exibe uma matriz.
    '''
    print(str_matriz(matriz)) # NOVO

#-----------------------------------------------------------
def str_matriz(matriz):
    '''(matriz) -> str

    Recebe e cria um string que representa a matriz.

    O string retornado é usado para imprimir a matriz.
    '''
    s = ""
    nlins = len(matriz)
    ncols = len(matriz[0])

    # pegue o maior número caracteres para escreve um valor
    max_len = max_len_valor(matriz) + 1 # EXERCÍCIO

    s += "Matriz: {nlins} x {ncols}"
    for i in range(0, nlins, +1):
        for j in range(0, ncols, +1):
            s +=  f"{matriz[i][j]:{max_len}}" # NOVO
        # pule uma linha
        s += "\n"

    return s
```

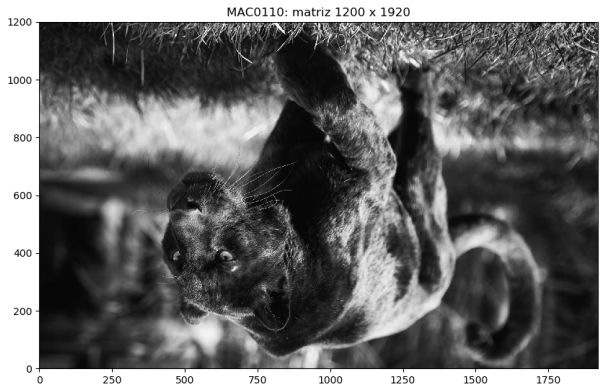
26.6 gire_horizontal()

	0	1	2	3
0	a	b	c	d
1	e	f	g	h
2	i	j	k	l
3	m	n	o	p
4	q	r	s	t

flipH() ----->

	0	1	2	3
0	q	r	s	t
1	m	n	o	p
2	i	j	k	l
3	e	f	g	h
4	a	r	s	t



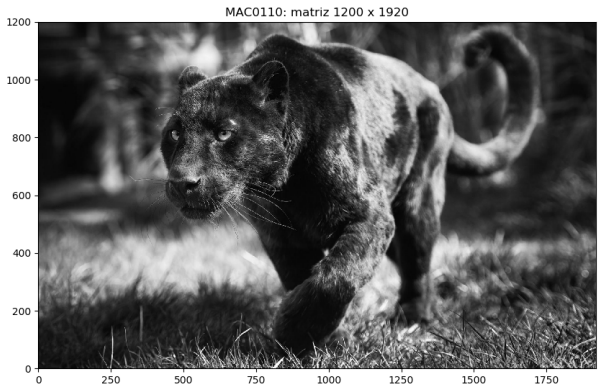


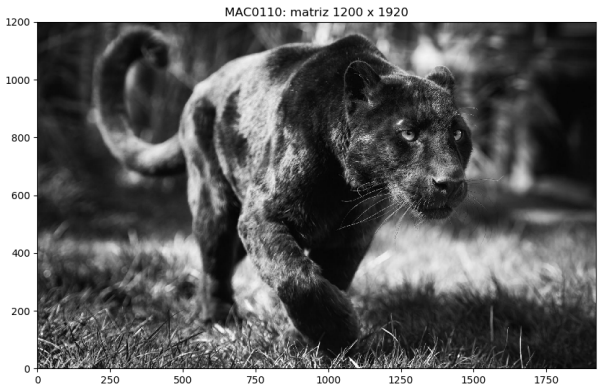
Solução

```
def gire_horizontal(mt):
    '''(matriz) -> matriz

    RECEBE uma matriz mt.
    RETORNA a mt refletida horizontalmente (cima-baixo).
    Essa função NÃO é mutadora.
    '''
    nlins = len(mt)
    ncols = len(mt[0])
    horizontal = init_matriz(nlins, ncols)
    for i in range(nlins):
        for j in range(ncols):
            horizontal[i][j] = mt[nlins-i-1][j]
    return horizontal
```

26.7 gire_vertical()

```
    0   1   2   3                          0   1   2   3
  +---+---+---+---+                      +---+---+---+---+
0 | a | b | c | d |                    0 | d | c | b | a |
  +---+---+---+---+                      +---+---+---+---+
1 | e | f | g | h |                    1 | h | g | f | e |
  +---+---+---+---+      flipV()        +---+---+---+---+
2 | i | j | k | l |      ----->        2 | l | k | j | i |
  +---+---+---+---+                      +---+---+---+---+
3 | m | n | o | p |                    3 | p | o | n | m |
  +---+---+---+---+                      +---+---+---+---+
4 | q | r | s | t |                    4 | t | s | r | q |
  +---+---+---+---+                      +---+---+---+---+
```



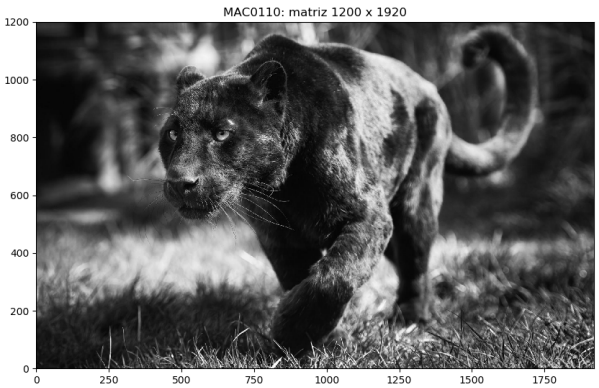


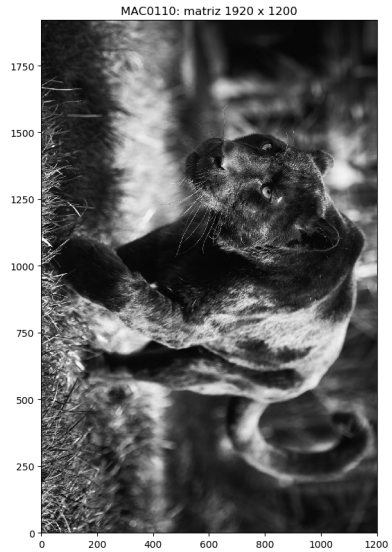
Solução

```
def gire_vertical(mt):
    '''(matriz) -> matriz

    RECEBE uma matriz mt.
    RETORNA mt refletida verticalmente (esquerda-direita).
    Essa função NÃO é mutadora.
    '''
    nlins = len(mt)
    ncols = len(mt[0])
    vertical = init_matriz(nlins, ncols)
    for i in range(nlins):
        for j in range(ncols):
            vertical[i][j] = mt[i][ncols-j-1]
    return vertical
```

26.8 rode_direita()

```
    0   1   2   3                          0   1   2   3
  +---+---+---+---+                      +---+---+---+---+---+
0 | a | b | c | d |                    0 | q | m | i | e | a |
  +---+---+---+---+                      +---+---+---+---+---+
1 | e | f | g | h |                    1 | r | n | j | f | b |
  +---+---+---+---+    rotateR()         +---+---+---+---+---+
2 | i | j | k | l |     ----->         2 | s | o | k | g | c |
  +---+---+---+---+                      +---+---+---+---+---+
3 | m | n | o | p |                    3 | t | p | l | h | d |
  +---+---+---+---+                      +---+---+---+---+---+
4 | q | r | s | t |
  +---+---+---+---+
```



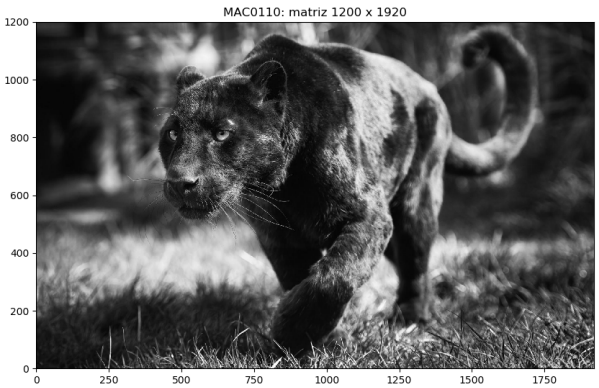


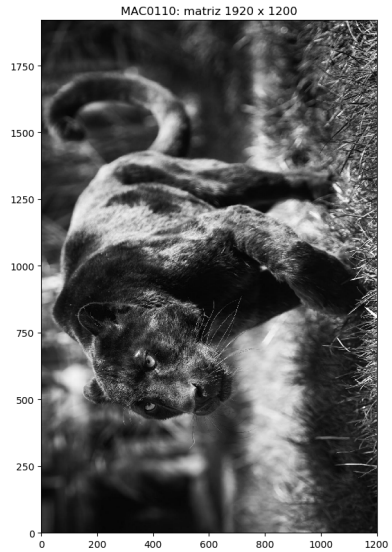
Solução

```
def rode_dir(mt):
    '''(matriz) -> matriz

    RECEBE uma matriz mt.
    RETORNA mt rotacionada para a direita (horário)
    Essa função NÃO é mutadora.
    '''
    nlins = len(mt)
    ncols = len(mt[0])
    direita = init_matriz(ncols, nlins)
    for i in range(ncols):
        for j in range(nlins):
            direita[i][j] = mt[nlins-j-1][i]
    return direita
```

26.9 rode_esquerda()

```
     0   1   2   3                          0   1   2   3
   +---+---+---+---+                      +---+---+---+---+---+
0  | a | b | c | d |                   0  | d | h | l | p | t |
   +---+---+---+---+                      +---+---+---+---+---+
1  | e | f | g | h |                   1  | c | g | k | o | s |
   +---+---+---+---+    rotateL()        +---+---+---+---+---+
2  | i | j | k | l |     ----->        2  | b | f | j | n | r |
   +---+---+---+---+                      +---+---+---+---+---+
3  | m | n | o | p |                   3  | a | e | i | m | q |
   +---+---+---+---+                      +---+---+---+---+---+
4  | q | r | s | t |
   +---+---+---+---+
```





Solução

```
def rode_esq(mt):
    '''(matriz) -> matriz

    RECEBE uma matriz `mt`.
    RETORNA a mt rotacionada para a esquerda (anti-horário).
    Essa função NÃO é mutadora.
    '''
    nlins = len(mt)
    ncols = len(mt[0])
    esquerda = init_matriz(ncols, nlins)
    for i in range(ncols):
        for j in range(nlins):
            esquerda[i][j] = mt[j][ncols-i-1]
    return esquerda
```

26.10 Arquivos

Em Python há pelo menos quatro maneira diferentes de lermos um arquivo. Cada uma delas tem sua utilidade. Até o momento vimos duas delas.

Nos exemplos a seguir utilizaremos o arquivo `jeff.txt`.

```
We hold these truths to be self-evident:
that all men are created equal;
that they are endowed by their Creator
with certain unalienable rights;
that among these are life, liberty,
and the pursuit of happiness.
```

`read()`

Se `arq` é um arquivo, `arq.read()` retorna uma string com todo o conteúdo do arquivo. Os caracteres de *nova linha* `'\n'` no arquivo estão na string.

Exemplo de leitura com `read()`

```
def leitura_com_read():
    ''' (None) -> None

    Exemplo de leitura de um arquivo com read()
    '''
    print("Leitura com read()")
    # 1 abra o arquivo
    arq = open(NOME_ARQUIVO, 'r', encoding='utf-8')
    # 2 leia o conteúdo do arquivo
    txt = arq.read() # retorna uma string
    # 3 feche o arquivo
    arq.close()
    # 4 exiba o conteúdo do arquivo
    print(f"Conteúdo do arquivo '{NOME_ARQUIVO}'")
    print(txt)
```

Saída produzida

```
Leitura com read()
Conteúdo do arquivo 'jeff.txt'
We hold these truths to be self-evident:
that all men are created equal;
that they are endowed by their Creator
with certain unalienable rights;
that among these are life, liberty,
and the pursuit of happiness.
```

`for linha in arq:`

Se **`arq`** é um arquivo, **`for linha in arq:`** lê o conteúdo de um arquivo linha a linha. No caso, a variável **`linha`** recebe em cada iteração uma string com o conteúdo de cada linha do arquivo. Os caracteres de *nova linha* **`'\n'`** no arquivos estão nas strings.

Exemplo de leitura com **`for ...`**

```
def leitura_com_for():
    '''(None) -> None

    Exemplo de leitura de um arquivo percorrendo suas
    linhas com for ...
    '''
    print("Leitura com for ..")
    # 1 abra o arquivo
    arq =open(NOME_ARQUIVO, 'r', encoding='utf-8')
    # 2 leia o arquivo linha a linha
    # 1 abra o arquivo
    arq = open(NOME_ARQUIVO, 'r', encoding='utf-8')
    # 2 leia o conteúdo do arquivo linha a linha
    print(f"Conteúdo do arquivo '{NOME_ARQUIVO}'")
    i = 0
    for linha in arq:
        print(f"{i}: {linha}")
        i += 1
    # 3 feche o arquivo
    arq.close()
```

Saída produzida

```
Leitura com for ..
Conteúdo do arquivo 'jeff.txt'
0: We hold these truths to be self-evident:
```

```
1: that all men are created equal;

2: that they are endowed by their Creator

3: with certain unalienable rights;

4: that among these are life, liberty,

5: and the pursuit of happiness.
```

readline()

Se **arq** é um arquivo, **arq.reasline()** retorna uma string com o conteúdo de uma linha do arquivo. Um novo **arq.readline()** retorna a proxima linha, e mais um novo **arq.readline()** retorna a próxima linha e assim até o final do arquivo se atingido. Os caracteres de *nova linha* **'\n'** no arquivo estão nas strings.

Exemplo de leitura com **readline()**

```
def leitura_com_readline():
    '''(None) -> None

    Exemplo de leitura de um arquivo com readline()
    '''
    print("Leitura com readline()")
    # 1 abra o arquivo
    arq = open(NOME_ARQUIVO, 'r', encoding='utf-8')
    # 2 leia o conteúdo do arquivo linha a linha
    print(f"Conteúdo do arquivo '{NOME_ARQUIVO}'")
    i = 0
    linha = arq.readline()
    while linha != '':
        print(f"{i}: {linha}")
        i += 1
        linha = arq.readline()
    # 3 feche o arquivo
    arq.close()
```

Saída produzida

```
Leitura com readline()
Conteúdo do arquivo 'jeff.txt'
0: We hold these truths to be self-evident:
```

```
1: that all men are created equal;

2: that they are endowed by their Creator

3: with certain unalienable rights;

4: that among these are life, liberty,

5: and the pursuit of happiness.
```

readlines()

Se **arq** é um arquivo, **arq.reaslines()** retorna uma lista em que cada item é uma string com o conteúdo de uma linha do arquivo. Os caracteres de *nova linha* **\n** no arquivo estão nas strings.

Exemplo de leitura com **readlines()**

```
def leitura_com_readlineS():
    '''(None) -> None

    Exemplo de leitura de um arquivo com readline()
    '''
    print("Leitura com readlines()")
    # 1 abra o arquivo
    arq = open(NOME_ARQUIVO, 'r', encoding='utf-8')
    # 2 leia o conteúdo do arquivo linha a linha
    lst_linhas = arq.readlines()
    # 3 feche o arquivo
    arq.close()
    # 4 exiba o conteúdo do arquivo
    for i in range(len(lst_linhas)):
        linha = lst_linhas[i]
        print(f"{i}: {linha}")
```

Saída produzida

```
Leitura com readlines()
0: We hold these truths to be self-evident:

1: that all men are created equal;

2: that they are endowed by their Creator

3: with certain unalienable rights;
```

```
4: that among these are life, liberty,

5: and the pursuit of happiness.
```