Exercícios resolvidos com listas 2d representando matrizes

Exercício 1 - Escrever um programa que: (a) crie uma matriz 3 x 4 chamada "w" e armazene o valor 1 em todas as suas posições; (b) imprima a matriz no vídeo, mostrando uma linha embaixo da outra

```
# PASSO 1: cria a matriz 3x4 com 1 em todas as posições
m = 3; n = 4

w = []
for i in range(m):
    w.append([1] * n)

# PASSO 2: imprime a matriz
for i in range(m):
    for j in range(n):
        print(w[i][j], end=" ")    # imprime o valor da célula i,j
                                   # veja que este end = " " foi usado
                                   # para que as células de uma linha i
                                   # sejam impressas lado a lado

    print()  # veja que este "print" pertence (é subordinado)
             # ao comando for i e não ao comando for j!
             # Ele serve para "pular" uma linha na tela, após
             # imprimir todas as células de uma linha da matriz
```

```
Saída:
>>>
1 1 1 1
1 1 1 1
1 1 1 1
```

No PASSO 1, a matriz é preenchida linha por linha, através de um comando append([1] * n) que usa o operador de repetição para repetir 1 por 4 vezes, gerando a lista (linha): [1, 1, 1, 1]. O PASSO 2 imprime a matriz do jeito explicado na aula.

Exercício 2 – Modifique o programa anterior para que ele gere a seguinte matriz:

```
 1   2   3   4
 5   6   7   8
 9  10  11  12
```

```
m = 3; n = 4

w = []
valor = 1
for i in range(m):
    linha = []
    for j in range(n):
        linha.append(valor)
        valor += 1
    w.append(linha)


print(w)  # dessa vez vou printar do jeito feio, só para conferir se deu certo!
```

Nesse exemplo, usamos a variável valor para gerar os valores de 1 a 12. O preenchimento da matriz é feito por linha: cada vez que uma linha é encerrada, ela é "appendada" na matriz w.

Exercício 3 - **Escrever um programa que: crie uma matriz 5 x 5 de inteiros, armazene o valor 1 na diagonal principal e o valor 0 nas demais células.**

```
1 0 0 0 0
0 1 0 0 0
0 0 1 0 0
0 0 0 1 0
0 0 0 0 1
```

```
m=5

a = []
for i in range(m):
    a.append([0] * m) # primeiro insere a linha apenas com 0's
    a[i][i] = 1       # troca o valor da diagonal principal por 1

print(a) # imprimindo apenas para checar, já que o enunciado não pediu
```

Nesse exemplo, basta modificar o valor de "m" de 5 para outro valor (tente 10 ou 20, por exemplo) para que o Python mude o tamanho da matriz quadrada automaticamente. E ele continuará a preencher a diagonal principal com 1 corretamente!

Exercício 4 - **Elaborar um programa que primeiro leia do teclado a matriz quadrada M1 3x3 de reais. Após a leitura ser concluída,gere a matriz M2 com o conteúdo de M1 multiplicado por 2. Depois imprima M1 e M2 com o layout "bonito".**

```
Digite o valor da célula 0:0 = 1
Digite o valor da célula 0:1 = 2
Digite o valor da célula 0:2 = 3
Digite o valor da célula 1:0 = 10
Digite o valor da célula 1:1 = 20
Digite o valor da célula 1:2 = 30
Digite o valor da célula 2:0 = 5
Digite o valor da célula 2:1 = 7
Digite o valor da célula 2:2 = 9

M1
          1.00          2.00          3.00
         10.00         20.00         30.00
          5.00          7.00          9.00

M2
          2.00          4.00          6.00
         20.00         40.00         60.00
         10.00         14.00         18.00
```

```
# 1. recebe os dados de M1 via teclado
# (nesse exemplo, usei a estratégia de criar M1 toda zerada e depois
#  trocar pelos valores digitados pelo usuário)
m = 3
M1 = []
for i in range(m): M1.append([0] * m)

for i in range(m):
    for j in range(m):
        print(f'Digite o valor da célula {i},{j} = ', end="")
        M1[i][j] = float(input())
```

```
# 2. gera M2
M2 = []
for i in range(m):
    linha = []
    for j in range(m):
        linha.append(M1[i][j] * 2)
    M2.append(linha)

# 3. imprime M1 e M2
print('M1')
for i in range(m):
    for j in range(m):
        print(f"{M1[i][j]:>15.2f}", end=" ")
    print()

print('M2')
for i in range(m):
    for j in range(m):
        print(f"{M2[i][j]:>15.2f}", end=" ")
    print()
```

O enunciado dizia para carregar M1 primeiro e depois gerar M2. Então o exercício foi feito dessa forma, que é a mais comum para se trabalhar em estatística: primeiro os dados são colocados na a memória para depois serem processados.

Exercício 5 - Escrever um programa que leia e depois imprima uma matriz *m* x *n* do tipo string, onde o tamanho máximo de *m* e *n* deve ser 100.

```
Digite a quantidade de linhas da matriz (MAX = 100): 2
Digite a quantidade de colunas da matriz (MAX = 100): 3
Elemento[0 , 0] = F
Elemento[0 , 1] = L
Elemento[0 , 2] = A
Elemento[1 , 0] = F
Elemento[1 , 1] = L
Elemento[1 , 2] = U

F L A
F L U
```

```
# PASSO 1: pergunta para o usuário qual a quantidade de linhas
#          e qual a quantidade de colunas da matriz, onde o máximo é 100.
#          Usa um loop para garantir que informação correta será digitada
MAX = 100
m = 0
n = 0

while (m == 0) or (m > 100):
    print('Digite a quantidade de linhas da matriz (MAX = ', MAX, '): ')
    m = int(input())

while (n == 0) or (n > 100):
    print('Digite a quantidade de colunas da matriz (MAX = ', MAX, '): ')
    n = int(input())

# PASSO 2: carrega a matriz, usando os dados entrados pelo usuário
# como os limites dos comandos for-range
```

```
mat = []
for i in range(m): mat.append([0] * n)

for i in range(m):
    for j in range(n):
           print('Elemento[', i, ',', j, '] = ', end="")
           mat[i][j] = input()

# PASSO 3: - imprime a matriz
for i in range(m):
    for j in range(n):
        print(mat[i][j], end=' ')
    print()
```

<u>Exercício 6</u> - Dada uma matriz A m x n de inteiros em memória, gere e imprima ^{t}A (a matriz transposta de A).

Exemplo: Se

```
A = | 2   0  3 |
    | 1  -1  2 |

então tA =  | 2   1 |
            | 0  -1 |
            | 3   2 |
```

```
# aqui um exemplo
A = [[2, 0, 3],
     [1, -1, 2]]

m = len(A); n = len(A[0])


# ---------------------------------------------------------
# aqui se obtém a transposta de A.
# ---------------------------------------------------------

# PASSO 1: cria a transposta em memória com todas as células 0
# Ela é n x m -> n linhas e m colunas.
# Por isso o for é range(n), jogando m colunas 0
tA = []
for i in range(n): tA.append([0] * m)

# PASSO 2: joga os valores corretos nas células de tA
for i in range(m):
    for j in range(n):
        tA[j][i] = A[i][j]    # se você tiver dificuldade em
                              # entender esse passo, tente fazer um desenho
                              # no papel simulando um exemplo

# PASSO 3: imprime as duas matrizes, A e sua transposta
print('\nMatriz A')
for i in range(m):
    for j in range(n):
        print(f"{A[i][j]:>9}", end=" ")
    print()

print('\nMatriz Transposta de A')
for i in range(n):
    for j in range(m):
        print(f"{tA[i][j]:>9}", end=" ")
    print()
```

Exercício 7 - Dadas duas matrizes 4x3 A e B em memória, ambas preenchidas, faça um programa que calcule e mostre:
– A soma das duas matrizes, resultando em uma nova matriz C.
– A diferença das duas matrizes, resultando em uma nova matriz D.

```
# aqui um exemplo só para testar o programa
A = [[1, 2, 3], [4, 5, 6]]
B = [[10, 20, 30], [40, 50, 60]]

# aqui começa o que foi pedido no enunciado... gera C = A + B e D = A - B
m = len(A); n = len(A[0])

C = []; D = []
for i in range(m):
    soma_linha = []
    sub_linha = []

    for j in range(n):
        soma_linha.append(A[i][j] + B[i][j])
        sub_linha.append(A[i][j] - B[i][j])

    C.append(soma_linha)
    D.append(sub_linha)

print(C); print(D)
```

Exercício 8 – Faça um programa que imprima os elementos da diagonal secundária de uma matriz quadrada de ordem m, já carregada em memória.

```
# * * Solução 1: percorrendo todas as células e usando "equação"
# para detectar a célula da diagonal secundária em cada linha

# aqui um exemplo só para testar o programa
mat = [[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9]]

# aqui começaria o seu programa...
m = len(mat)
diag_sec = []
for i in range(m):
    for j in range(m):
        if i == m - j - 1:  # faça a verificação no papel !!!
            diag_sec.append(mat[i][j])

print(diag_sec)


# * * Solução 2: percorre apenas as linhas usando indexação negativa

# aqui um exemplo só para testar o programa
mat = [[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9]]

# aqui começa o que foi pedido no enunciado...
m = len(mat)
diag_sec = []
for i in range(m):
    diag_sec.append(mat[i][-(i+1)]) # faça a verificação no papel !!!

print(diag_sec)
```

Exercício 9 – **Dada uma matriz quadrada A de ordem m já carregada em memória, faça um programa capaz de alterar A da seguinte forma: multiplique cada elemento de uma linha pelo elemento da diagonal principal da linha em questão, exceto o elemento que já está na diagonal principal !!!. Mostre a matriz após as multiplicações.**

```
# aqui um exemplo só para testar o programa
A = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]

# aqui começa o que foi pedido no enunciado...
m = len(A)
for i in range(m):
    elemento_da_diagonal = A[i][i]
    for j in range(m):
        if i != j:
            A[i][j] *= elemento_da_diagonal

print(A)
```

Exercício 10 - Elabore um programa onde o usuário possa digitar o nome de um país e o sistema possa informar qual é a capital do país. Apenas os seguintes conjuntos de países/capitais devem estar cadastrados no programa (esses dados são fixos do programa e não precisam ser lidos do teclado).

PAÍS	CAPITAL
Nigéria	Abuja
Somália	Mogadíscio
Quênia	Nairóbi
Argélia	Argel
Moçambique	Maputo
Madagascar	Antananarivo
Cabo Verde	Praia
Uganda	Campala
Zâmbia	Lusaca

Exemplo de Saída:

```
>>> %Run paises_capitais.py
 PAÍSES E CAPITAIS
 =================

 Digite o nome de um pais (use apenas as iniciais em maiúsculo)
 Cabo Verde

 PAIS =  Cabo Verde  - CAPITAL =  Praia
```

```
# 1. cria a matriz onde a col. 0 tem o nome do país e a col. 1 a capital
paises = [
['Nigéria', 'Abuja'],
['Somália', 'Mogadíscio'],
['Quênia', 'Nairóbi'],
['Argélia', 'Argel'],
['Moçambique', 'Maputo'],
['Madagascar', 'Antananarivo'],
['Cabo Verde', 'Praia'],
['Uganda', 'Campala'],
['Zâmbia', 'Lusaca']
]

NUM_LIN = 9

# 2. SOLICITA O NOME DE UM PAÍS AO USUÁRIO
```

```
print('PAÍSES E CAPITAIS');
print('================');
print()
print('Digite o nome de um pais (use apenas as iniciais em maiúsculo)')
busca = input()

# 3. PROCURA PELO PAÍS NA MATRIZ (OLHANDO A COLUNA 0).
#              SE ENCONTRAR EXIBE A CAPITAL (COLUNA 1)
#              SE NAO ENCONTRAR EXIBE A MENSAGEM: "PAIS NAO CADASTRADO"}

achou = False  # supõe que país não será achado
for i in range(NUM_LIN): # laço que percorre as linhas da matriz

    if (busca == paises[i][0]): # testa se valor digitado é igual ao
                                # conteúdo armazenado na linha i, coluna 0 da
                                # matriz
        achou = True  # se isso ocorre, então achei o país
        print()
        print('PAIS = ', busca, ' - CAPITAL = ',paises[i][1])
        print()

# atenção, aqui o loop for já acabou! O único comando subordinado ao
# for é "if busca == paises[i][0]"

# se o país não tiver sido achado, manda mensagem ao usuário
if (not achou): # OBS: isso é o mesmo que if achou == False
    print('DESCULPE, o país',busca, ' não está cadastrado em nosso sistema.')
```

Exercício 11 - **Dada uma matriz W carregada em memória, faça um programa que gere uma lista contendo o maior elemento de cada coluna dessa matriz. Por exemplo:**

```
W = [[1, 200, 3, 77],
     [4, 5,  16, -1],  → [7, 200, 16, 77]
     [7, 8,   9, 12]]
```

Solução:

```
# aqui um exemplo só para testar o programa
W = [[1, 200, 3, 77],
       [4, 5, 16, -1],
       [7, 8, 9, 12]]

# aqui começa o que foi pedido no enunciado...
m = len(W); n = len(W[0])
l = []
for j in range(n): # percorre a matriz por colunas (fixa coluna e percorre todas
                   # as linhas dela). Basta colocar o for das colunas (for j)
                   # antes do for das linhas (for i)

    maior = W[0][j] # assume que o maior da coluna é o que está na linha 0
    for i in range(m): # para a coluna j, percorre as linhas
        if W[i][j] > maior: maior = W[i][j] # troca se achar alguém maior
    l.append(maior)

print(l)
```

Exercício 12 - **Dada uma matriz W carregada em memória, faça um programa que identifique o elemento MINMAX dessa matriz, onde o elemento MINMAX corresponde ao menor elemento da linha em que se encontra o menor elemento. Por exemplo:**

W = [[10, 60, 7],
[100, 1, 8], → MINMAX = 1, pois esse é o menor elemento da linha em
[50, -3, 9]] que está o maior elemento (100, na linha 1)

Solução:

```
# aqui um exemplo só para testar o programa
W = [[10, 60, 7],[100, 1, 8],[50, -3, 9]]

# aqui começa o que foi pedido no enunciado...
m = len(W); n = len(W[0])

# ---------------------------------------------------------------------
# 1. percorre todas as células da matriz para achar o maior elemento
#    (veja que vou guardar quem é o maior e qual a sua linha
#     para facilitar minha vida na parte 2)
# ---------------------------------------------------------------------
maior = W[0][0]  # assume que maior é o primeiro
linha = 0
for i in range(m):
    for j in range(n):
        if W[i][j] > maior:
            maior = W[i][j]
            linha = i

# ---------------------------------------------------------------------
# 2. acha o menor elemento da linha do maior
# ---------------------------------------------------------------------
menor = min(W[linha])

print(menor)
```