ALGORITMOS

LISTA DE EXERCÍCIOS

Versão em Python

Instituto Federal do Espírito Santo

Coordenadoria de Física, *Campus* Cariacica

Wesley Spalenza

CAPÍTULO 3

LISTAS E ARRAYS

Exercício 3.1 – Faça um programa de um aluno que tenha cinco notas, [6,7,5,8,9], e deseja calcular a média. Use o conceito de listas.

Exercício 3.2 – Faça um programa de um aluno que tenha cinco notas, [6,7,5,8,9], e deseja calcular a média. Use o conceito de listas. Porém agora, o usuário deve entrar com o valor das notas na lista.

Exercício 3.3 – Imprima a lista `[8,1,3,12,45,"isto é uma lista"]`, usando (a) "for" e (b) "while"

Exercício 3.4 – (a) Faça um código para enumerar a lista `[8,1,3,12,45,100].` (b) Faça agora usando o comando "enumerate".

Exercício 3.5 – Pesquise dentro um elemento dentro da lista `[8,1,3,12,45,100],` e diga se não houver o elemento que você digitar.

Exercício 3.6 – Faça um código que leia a lista `[2,8,1,3,4,12,33,45,22,100,75]` e separe em duas outras listas, uma com valores pares e outra com valores ímpares.

Exercício 3.7 – Faça um programa que leia (o usuário insira os produtos) e imprima uma lista de compras e quando o usuário digitar "fim", o programa termine.

Exercício 3.8 – Faça um programa que controle a utilização das salas de um cinema. Suponha a lista com as salas: `salas = [10,2,1,3,0]` , contenha o número de lugares vagos nas salas 1, 2, 3, 4 e 5, respectivamente. Esse programa lerá o número da sala e a quantidade de lugares solicitados e deve informar se é possível vender o número de lugares solicitados, ou seja, se ainda há lugares livres. Caso seja possível vender os bilhetes, atualizará o número de lugares livres. Defina a saída quando se digita zero "0".

Exercício 3.9 – Defina uma lista com produtos que contenha

```
["tipo de produto", quantidade, preço]
```

Faça um programa que (a) imprima as características dos produtos que você comprou:

```
p1 = ["maçã", 10, 0.30],  p2 = ["pera", 5, 0.75], p3 = ["kiwi", 4, 0.98].
```

(b) Imprima agora discriminando o produto, a quantidade e o preço.

Exercício 3.10 – Escreva um programa completo, capaz de perguntar o nome do produto, quantidade e preço e, no final, imprimir uma lista de compras completa.

Exercício 3.11 – Escreva um programa para ordenação de valores a partir do método de Bolhas. Este método consiste em comparar dois elementos a cada vez, se o valor do primeiro elemento for maior que o do segundo, eles trocarão de posição, de modo que se repita até o fim da lista.

Exercício 3.12 – Escreva um programa para ordenação, como o do problema anterior, mas de valores no sentido decrescente.

Exercício 3.13 – Crie um dicionário com os dados:

```
{"Alface": 0.45,
"Batata": 1.20,
"Tomate": 2.30,
"Feijão": 1.50 }
```

e (a) imprima. (b) Imprima, por exemplo, somente o produto Tomate. (c) Verifique se Manga está na tabela. (d) Verfique se Batata está na tabela.

Exercício 3.14 – Crie um dicionário com os dados:

```
{ "Alface": 0.45,
"Batata": 1.20,
"Tomate": 2.30,
```

```
"Feijão": 1.50 }
```

e que exiba o preço do produto, se você digitar o nome do produto.

Exercício 3.15 – Crie um dicionário com os dados:

```
{"Alface": 0.45,
"Batata": 1.20,
"Tomate": 2.30,
"Feijão": 1.50 }
```

e exclua o produto Tomate, por exemplo, e imprima na tela a nova tabela.

Exercício 3.16 – Crie um dicionário com listas dentro, com os dados:

```
estoque={ "tomate": [ 1000, 2.30],
"alface": [ 500, 0.45],
"batata": [ 2001, 1.20],
"feijão": [ 100, 1.50] }
```

e imprima.

Exercício 3.17 – Crie um dicionário com listas dentro, com os dados:

```
estoque={ "tomate": [ 1000, 2.30],
"alface": [ 500, 0.45],
"batata": [ 2001, 1.20],
"feijão": [ 100, 1.50] }
```

Com ela definida, calcule o preço total de venda, atualizando também a quantidade de estoque. Tome uma lista de vendas como: venda = [["tomate", 5], ["batata", 10], ["alface",5]], onde temos as sublistas como [produto,quantidade].

Exercício 3.18 – (a) Crie um vetor (array de rank 1) com elementos `x = (3,2)`. (b) Faça a transposição de x, usando o comando "transpose". Dica: Você precisará usar o pacote/bibliotca do python chamado numpy e chamar os comandos:

```
from numpy import mat, zeros, transpose
```

Exercício 3.19 – Crie um vetor `v = (1,2,3,4)` usando o comando do numpy, np.array . Aplique o atributo shape para verficar a forma do array. Defina agora a matriz

```
A=np.array([[1,2],[3,4]])
```

e verifique `A.shape`.

Exercício 3.20 – (a) Crie um vetor `v = (1,2,3,4)` usando o comando do numpy, np.array, e apartir dele use o comando v.shape = (2,2) para transformá-lo em um Array de rank 2, isto é, uma matriz, aqui no caso 2 x2. (b) Use agora o comando `v = np.array([1,2,3,4]).reshape(2,2)` e analise a resposta, comparando com a letra anterior.

Exercício 3.21 – Defina um Array (tensor) de rank 3

```
v = np.array(range(50)).reshape(2,5,5)
```

Aplique os atributos `shape, ndim` e `size`. O número de eixos (ou dimensões) de um tensor é dado pelo atributo `ndim`, enquanto o número total de elementos é dado por `size`. (b) Use o método `flatten`, pois ele retorna uma cópia do tensor com todos os elementos em apenas uma dimensão.

Exercício 3.22 – Defina arrays de rank 2, 3x3, com as funções `zeros` (array só com elementos nulos), `ones` (array só com elementos 1) e `diag` (array só com elementos na diagonal, coloque aqui 10 para todos, por exemplo). Eles são muito convenientes para a criação de tensores.

Exercício 3.23 – Defina vetores a partir de sequências criadas usando as funções arange e linspace, que permitem a criação de sequências como tensores NumPy. Crie por exemplo

```
v = [0.  0.5  1.0  1.5  2.0  2.5  3.0  3.5  4.0  4.5]
u = [0.        , 0.55555556, 1.11111111, 1.66666667, 2.22222222,
       2.77777778, 3.33333333, 3.88888889, 4.44444444,   5.       ]
```

Dica: Imprima os vetores abaixo e analise a resposta:

```
v = np.arange(0, 5, 0.5)
u = np.linspace(0, 5, 10)
```

Exercício 3.24 – (a) Definindo o vetor como

```
v = np.array([[1, 2, 3, 4],[5,6,7,8],[9,10,11,12],[13,14,15,16]])
```

acesse o elemento com índice 0 no primeiro eixo e índice 1 no segundo eixo, ou seja, o elemento da primeira linha e segunda coluna.

(b) Imprima, analise e explique cada caso:

```
v[3,1], v[:],  v[0:] , v[:0], v[0:1], v[1:], v[:1], v[2:3], v[0:2,1:3], v[1:2,2:3]
```

(c) Definindo o vetor como `v = np.arange(8).reshape(2,2,2)`, acesse o elemento com índice 0 no primeiro eixo e índice 1 no segundo eixo.

(d) Definindo o vetor `v = np.arange(10)`, acesse os elementos das posições 1 até a 3, e defina este acesso como um novo vetor. Lembre-se que no python o primeiro elemento começa com 0.

(e) Definindo uma matriz do tipo , acesse os elemento `A = np.arange(15).reshape(3,5)`, selecione somente a submatriz correspondente às linhas 0 e 1, e colunas 2 e 3.

Exercício 3.25 – Realize as operações com os vetores,

```
v = np.array([10,20,30]),    u = np.array([2,2,2])
```

(a) w = u + v, (b) w = u*v, (c) w = u/v, (d) w = v**2, (e) w = sin(v)

Exercício 3.26 – Calcule, para o vetor `x = np.arange(10)`, operadores unários como a média, menor valor, maior valor.

Exercício 3.27 – Dados os vetores e matrizes

```
a = np.array([4, 0])
b = np.array([-1, 1])
A = np.array([[2, 0], [0, 5]])
```

Usando o pacote numpy, encontre (a) produto escalar entre a e b, (b) produto entre A e a, (c) inversa de A, (d) o determinante de A, (e) autovalores e autovetores de A, (f) produto vetorial entre [1, 1, 1] e [0, 0, 1]. (g)

Encontre o ângulo, em grau, entre os vetores a e b, via produto escalar, usando o atributo norma/módulo `np.linalg.norm( )`. Encontre a solução do sistema linear para

```
A = np.array([10, 20, 30, 40]).reshape(2,2)
b = np.array([5,10])
```

Exercício 3.28 – Encontre a solução do sistema linear para

```
A = np.array([10, 20, 30, 40]).reshape(2,2)
b = np.array([5,10])
```

Exercício 3.29 – O NumPy possui um submódulo chamado random que possui diversas funções para a geração de números (pseudo)aleatórios. Embora o Python possua uma biblioteca padrão também chamada random, a biblioteca do NumPy tem mais funcionalidades e gera diretamente tensores aleatórios. Crie um tensor segundo uma distribuição uniforme no intervalo [0,1).

Exercício 3.30 – Crie um tensor em que cada elemento segue uma distribuição normal com as características de $\mu = 10.0$ e $\sigma = 1.0$.

Exercício 3.31 – Note que toda vez que rodarmos o código, os tensores terão valores diferentes. Podemos evitar esse comportamento, de forma que toda vez que o código é executado o tensor aleatório tenha o mesmo valor por meio da função seed, cujo argumento é a semente para o gerador de números aleatórios do Python.

Exercício 3.32 – (a) Crie o vetor `v1 = array([1,2,3,4])`. (b) imprima o primeiro emento do vetor. (c) defina o vetor v2, que é o dobro de v1. (d) Imprima a ordem do vetor, a dimensão, ou seja, o número de índices, e em seguida o número de elementos. (e) Faça o produto entre as componentes dos vetores. (f) Faça o produto escalar entre os vetores.

Exercício 3.33 – (a) Crie um vetor de 12 posições, começando pelo zero, usando o comando “arange” do Numpy. (b) Reescreva-a, a partir do comando “reshape”, com 3 linhas e 4 colunas. (c) Imprima a ordem da matriz, a dimensão, ou seja, o número de índices, e em seguida o número de elementos. (d) Realize as impressões

```
primeira linha              A[:1,:]
primeiras duas linhas       A[:2,:]
As colunas 2 e 3            A[:,1:3]
```

e analise cada caso.

Exercício 3.34 – (a) Defina a matriz

```
A = array([[1,2],[3,4]])
```

e imprima-a. (b) Imprima cada elemento de A e escreva a ordem da matriz. (c) Calcule o produto dos elementos de A com

```
B = array([[-1,3],[0,-2]])
```

(d) Realize a multiplicação matricial entre A e B.

Exercício 3.35 – (a) Escreva as três matrizes de Pauli e (b) teste a relação $[\sigma_x, \sigma_y] = 2i\sigma_z$, porém, no python, o número complexo é j.

Exercício 3.36 – (a) Use a matriz do exercício 3.32 e faça a transposição de v e de A. (b) Imprima a ordem, o número de índices e o número de elementos.

Exercício 3.37 – Seja as matrizes

```
A = array([[1,2,3],[22,32,42],[55,66,100]])
```

```
B = array([1,2,3])
```

Usando o pacote Numpy, (a) resolva o sistema A X = B e encontre a matriz de erro E = AX – B . (b) Encontre a matriz inversa de A e depois resolva o mesmo sistema da letra anterior.

Exercício 3.38 – Elabore um programa que descreva o movimento uniforme planar de um sistema de duas partículas, na forma de gráficos, tomando como entrada as posições e velocidades iniciais de cada partícula. Defina as condições iniciais no código e o tempo do período do movimento.

Exercício 3.39 – Elabore um programa que descreva o movimento uniformemente variado espacial de um sistema de duas partículas, na forma de gráficos, tomando como entrada as posições, velocidades e acelerações iniciais de cada partícula. Use o pacote "`Matplotlib`" em três dimensões.