



UNIVERSIDADE  
**VILA VELHA**  
ESPÍRITO SANTO

Laboratório de Programação - PYTHON  
Prof.: Alessandro Bertolani Oliveira  
TRABALHO 2º BIMESTRE

NOME: Ruan Bona Miranda

CRIAR / COMPARTILHAR AQUI UM LINK PARA SEU PRÓPRIO NOTEBOOK COLAB: [Clique Aqui!](#)

O que vamos Praticar:

## ✓ $\longrightarrow$ ESTRUTURA LISTA: list

Clique duas vezes (ou pressione "Enter") para editar


```
1 # Número Sorteado : 2

1 import numpy as np
2 atletas = []
3
4 def get_melhor_nadador(atletas):
5     return min(atletas, key=lambda x: x[5])
6
7 def get_melhor_corredor(atletas):
8     return min(atletas, key=lambda x: x[6])
9
10 def get_melhor_ciclista(atletas):
11     return min(atletas, key=lambda x: x[7])
12
13 def get_melhor_atleta(atletas):
14     return min(atletas, key=lambda x: x[5] + x[6] + x[7])
15
16 def get_media_geral(atletas):
17     if not atletas:
18         return 0
19     return sum(x[5] + x[6] + x[7] for x in atletas) / len(atletas)
20
21 def get_abaixo_media(atletas, media_geral):
22     return [x[0] for x in atletas if (x[5] + x[6] + x[7]) < media_geral]
23 while True:
24     try:
25         print('HUB')
26         print('Digite 0 para encerrar o programa')
27         print('Digite 1 para inserir um atleta')
28         print('Digite 2 para exibir os melhores atletas em cada etapa')
29         print('Digite 3 para exibir o atleta que realizou a competição no menor tempo')
30         print('Digite 4 para exibir a média geral e os atletas que realizaram a prova abaixo dela')
31         opcao = int(input('DIGITE A OPÇÃO DESEJADA: '))
32         if(opcao not in [0, 1, 2, 3, 4]):
33             print('Opção inválida... por favor tente novamente')
34         elif(opcao == 0):
35             break
36         elif(opcao == 1):
37             nome = input('Insira o nome do atleta: ')
38             patrocinador = input('Insira o patrocinador do atleta: ')
39             dia = int(input('insira o dia de nascimento do atleta em formato dd: '))
40             mes = int(input('insira o mês de nascimento do atleta em formato mm: '))
41             ano = int(input('insira o ano de nascimento do atleta em formato aaaa: '))
42             na = int(input('insira a o tempo da prova de natação em segundos: '))
43             co = int(input('insira a o tempo da prova de corrida em segundos: '))
44             ci = int(input('insira a o tempo da prova de ciclismo em segundos: '))
45             atletas.append([nome, patrocinador, dia, mes, ano, na, co, ci])
46         elif(opcao == 2):
47             melhor_nadador = get_melhor_nadador(atletas)
48             melhor_corredor = get_melhor_corredor(atletas)
49             melhor_ciclista = get_melhor_ciclista(atletas)
50             print(f'O nome do melhor nadador é: {melhor_nadador[0]}')
```

```

51     print(f'O nome do melhor corredor é: {melhor_corredor[0]}')
52     print(f'O nome do melhor ciclista é: {melhor_ciclista[0]}')
53     elif(opcao == 3):
54         melhor_atleta = get_melhor_atleta(atletas)
55         print(f'O nome do melhor atleta é: {melhor_atleta[0]}')
56         print(f'O patrocinador do melhor atleta é: {melhor_atleta[1]}')
57     elif(opcao == 4):
58         media_geral = get_media_geral(atletas)
59         print(f'A tempo médio geral de realização do percurso foi de: {media_geral : .2f} segundos')
60         abaixo_media = get_abaixo_media(atletas, media_geral)
61         print(f'Os atletas que tiveram uma performance abaixo da média geral foram: {"", " ".join(abaixo_media) if abaixo_media else "nenh
62 except Exception as erro:
63     print(erro)
64

```

 HUB  
 Digite 0 para encerrar o programa  
 Digite 1 para inserir um atleta  
 Digite 2 para exibir os melhores atletas em cada etapa:  
 Digite 3 para exibir o atleta que realizou a competição no menor tempo  
 Digite 4 para exibir a média geral e os atletas que realizaram a prova abaixo dela

1 Comece a programar ou [gere código](#) com IA.

## ✓ ESTRUTURAS NUMPY: ndarray

### EXERCÍCIO 2

```


1 # NÚMERO SORTEADO 2
2 import numpy as np
3

```

```

1 data = np.load('dataset2.npy')
2 data

```


 array([[8.70200000e+03, 7.78644097e+01, 1.65727848e+00, 1.00000000e+00],  
 [4.40800000e+03, 8.17098036e+01, 1.82968202e+00, 0.00000000e+00],  
 [5.69200000e+03, 8.73402694e+01, 1.82908716e+00, 0.00000000e+00],  
 ...,  
 [5.52500000e+03, 7.60607311e+01, 1.81211849e+00, 0.00000000e+00],  
 [3.14600000e+03, 8.81656660e+01, 1.86525588e+00, 0.00000000e+00],  
 [4.28700000e+03, 8.28435717e+01, 1.89092215e+00, 1.00000000e+00]])

Letra A) A altura média dos 200 entrevistados mais altos.

```

1 altura = data[:, 2]
2 altura_sorted = np.sort(altura)
3 altura_top200 = altura_sorted[-200]
4 altura_200 = np.mean(altura_top200)
5 print(f'A altura média dos 200 entrevistados mais altos é: {altura_top200: .2f}')

```


 A altura média dos 200 entrevistados mais altos é: 1.88

Letra B) A quantidade: Absoluta e relativa (%) dos entrevistados sedentários e não sedentários.

```

1 sed = data[:, 3]
2 abs1 = np.count_nonzero(sed)
3 rel = (abs1 * 100) / len(sed)
4 print(f'O número absoluto de entrevistados que são sedentários é: {abs1}, esse valor relativo ao total é {rel}%')
5 print(f'Os não sedentários são em valores absolutos: {1000 - abs1}, e em valor relativo é {100 - rel}%')

```

 O número absoluto de entrevistados que são sedentários é: 494, esse valor relativo ao total é 49.4%  
 Os não sedentários são em valores absolutos: 506, e em valor relativo é 50.6%

Letra C) A quantidade: Absoluta e relativa (%) dos entrevistados saudáveis com IMC entre [18.5, 25.0].

```

1 massa = data[:, 1]
2 imc = massa / altura ** 2
3 saudavel = (imc >= 18,5) and (imc <= 25)
4 saud_abs = np.count_nonzero(saudavel)
5 saud_rel = saud_abs / len(data) * 100
6 print(f'Quantidade total de entrevistados saudáveis: {saud_abs}, o valor relativo comparado ao total é {saud_rel: .2f}%')

```

Quantidade total de entrevistados saudáveis: 511, o valor relativo comparado ao total é 51.10%

Letra D) O valor da média do IMC entre os sedentários e não sedentários:

```
1 sedentario = (sed == 1)
2 saudavel = (sed == 0)
3 imc_sed = massa[sedentario] / altura[sedentario] ** 2
4 imc_sau = massa[saudavel] / altura[saudavel] ** 2
5 med_sed = np.mean(imc_sed)
6 med_sau = np.mean(imc_sau)
7 print(f'O IMC médio dos sedentários é {med_sed: .1f}, para os não sedentários é {med_sau: .1f}.')
```

O IMC médio dos sedentários é 25.2, para os não sedentários é 25.0.

Letra E) O código dos 100 entrevistados com os piores índices de IMC: Magreza e Obesidade

```
1 np.random.shuffle(data)
2 maiores_imc = imc[:100]
3 menores_imc = imc[-100:]
4
5
6 print("Os códigos dos 100 maiores IMCs são:")
7 for indice, codigo in enumerate(maiores_imc):
8     print(f'ENTREVISTADO: {indice + 1} CÓDIGO: {codigo}')
9
10 print("\nOs códigos dos 100 menores IMCs são:")
11 for indice, codigo in enumerate(menores_imc):
12     print(f'ENTREVISTADO: {indice + 1} CÓDIGO: {codigo}')
```



```
ENTREVISTADO: 98 CÓDIGO: 35.5609/0498544/0
ENTREVISTADO: 99 CÓDIGO: 40.37814766375428
ENTREVISTADO: 100 CÓDIGO: 42.96671584763834
```

## EXERCÍCIO 3

```
1 data2 = np.load('dataset3.npy')
```

Letra A) O Gerente de CRM (Customer Relationship Management - Gerente de Relacionamento com o Cliente) da Empresa quer fazer uma Distribuição de Cartão de Fidelidade, de acordo com a tabela de compras.

Qual a quantidade: Absoluta e relativa (%) de clientes por cartão.

```
1 gasto = data2[:,1]
2 bronzeabs = np.sum(gasto <= 100)
3 prataabs = np.sum(np.logical_and(gasto > 100, gasto <= 500))
4 ouroabs = np.sum(np.logical_and(gasto > 500, gasto <= 1000))
5 diamanteabs = np.sum(gasto > 1000)
6 bronze_rel = bronzeabs / len(gasto) * 100
7 prata_rel = prataabs / len(gasto) * 100
8 ouro_rel = ouroabs / len(gasto) * 100
9 diamante_rel = diamanteabs / len(gasto) * 100
10 print(f'abs Bronze: {bronzeabs}, rel: {bronze_rel}%')
11 print(f'abs Prata: {prataabs}, rel: {prata_rel}%')
12 print(f'abs Ouro: {ouroabs}, rel: {ouro_rel}%')
13 print(f'abs Diamante: {diamanteabs}, rel: {diamante_rel:.1f}%')
14
```

```
➞ abs Bronze: 1, rel: 0.1%
abs Prata: 5, rel: 0.5%
abs Ouro: 52, rel: 5.2%
abs Diamante: 942, rel: 94.2%
```

Letra B) Custo da campanha CRM desta empresa por grupo e total (Custo de confecção unitário do cartão: R\$ 3.87).

```
1 custototal = len(data2) * 3.87
2 custob = bronze * 3.87
3 custop = prata * 3.87
4 custoo = ouro * .87
5 custod = diamante * 3.87
6 print(f'Custo total: {custototal}')
7 print(f'Custo Bronze: {custob}')
8 print(f'Custo Prata: {custop}')
9 print(f'Custo Ouro: {custoo}')
10 print(f'Custo Diamante: {custod}')
```

```
➞ Custo total: 3870.0
Custo Bronze: 3.87
Custo Prata: 19.35
Custo Ouro: 45.24
Custo Diamante: 3645.54
```

Letra C) O perfil dos clientes por sexo, com os seguintes dados;

Mínimo gasto

Máximo gasto

Média de gastos

```
1 sex = data2[:, 2]
2 gastom = gasto[sex == 0]
3 gastow = gasto[sex == 1]
4
5 minm = np.min(gastom)
6 maxm = np.max(gastom)
7 mediam = np.mean(gastom)
8
9 minw = np.min(gastow)
10 maxw = np.max(gastow)
11 mediaw = np.mean(gastow)
12
13 print(f'Homem - Min: R${minm:.1f}, max: R${maxm:.1f}, média: R${mediam:.1f}')
14 print(f'Mulher - Min: R${minw:.1f}, max: R${maxw:.1f}, média: R${mediaw:.1f}')
```

```
➞ Homem - Min: R$ 509.2, max: R$ 3761.1, média: R$ 1826.5
Mulher - Min: R$ 59.8, max: R$ 3387.0, média: R$ 1821.2
```

Letra D) O perfil dos clientes por forma de pagamento, com os seguintes dados;

Mínimo gasto

Máximo gasto

Média de gastos

```
1 pag = data2[:, 3]
2
3 gastod = gasto[pag == 0]
4 gastoc = gasto[pag == 1]
5
6 minm = np.min(gastod)
7 maxm = np.max(gastod)
8 mediam = np.mean(gastod)
9
10 minw = np.min(gastoc)
11 maxw = np.max(gastoc)
12 mediaw = np.mean(gastoc)
13
14 print(f'Débito - Min: R${minm:.1f} , max: R${maxm:.1f}, média: R${mediam:.1f}')
15 print(f'Crédito - Min: R${minw:.1f}, max: R${maxw:.1f}, média: R${mediaw:.1f}')
```

➦ Débito - Min: R\$59.8 , max: R\$3761.1, média: R\$1809.8  
Crédito - Min: R\$299.1, max: R\$3387.0, média: R\$1839.4

Letra E) O perfil dos clientes por tipo de cartão, com os seguintes dados;

Mínimo gasto

Máximo gasto

Média de gastos

```
1 gastobr = gasto[gasto <= 100]
2 gastopr = gasto[(gasto > 100) & (gasto <= 500)]
3 gastoou = gasto[(gasto > 500) & (gasto <= 1000)]
4 gastodi = gasto[gasto > 1000]
5
6 minbr = np.min(gastobr)
7 maxbr = np.max(gastobr)
8 mediabr = np.mean(gastobr)
9
10 minpr = np.min(gastopr)
11 maxpr = np.max(gastopr)
12 mediapr = np.mean(gastopr)
13
14 minou = np.min(gastoou)
15 maxou = np.max(gastoou)
16 mediaou = np.mean(gastoou)
17
18 mindi = np.min(gastodi)
19 maxdi = np.max(gastodi)
20 mediadi = np.mean(gastodi)
21
22 print(f'Bronze - Min: R${minbr:.1f} , max: R${maxbr:.1f}, média: R${mediabr:.1f}')
23 print(f'Prata - Min: R${minpr:.1f} , max: R${maxpr:.1f}, média: R${mediapr:.1f}')
24 print(f'Ouro - Min: R${minou:.1f} , max: R${maxou:.1f}, média: R${mediaou:.1f}')
25 print(f'Diamante - Min: R${mindi:.1f} , max: R${maxdi:.1f}, média: R${mediadi:.1f}')
26
```

➦ Bronze - Min: R\$59.8 , max: R\$59.8, média: R\$59.8  
Prata - Min: R\$299.1 , max: R\$470.4, média: R\$409.3  
Ouro - Min: R\$509.2 , max: R\$998.8, média: R\$825.8  
Diamante - Min: R\$1019.5 , max: R\$3761.1, média: R\$1888.4

