

# Programando em Python

## *Aula 7*

Prof. Dr. Marco Antonio Leonel Caetano

# Operações com *Matrizes (array)*

Criando simples matriz

$$\begin{array}{c} \longrightarrow \\ \longrightarrow \\ \longrightarrow \\ \text{Linhas} \end{array} A = \begin{array}{c} \text{Colunas} \\ \downarrow \quad \downarrow \quad \downarrow \\ \begin{pmatrix} \alpha_{11} & \alpha_{12} & \alpha_{13} \\ \alpha_{21} & \alpha_{22} & \alpha_{23} \\ \alpha_{31} & \alpha_{32} & \alpha_{33} \end{pmatrix} \end{array}$$

# Operações com *Matrizes (array)*

## Criando simples matriz

```
1 # Matrizes
2 import numpy as np
3
4 mat=np.array( [ [2,5,8], [1,2,6], [4,9,1] ] )
5 print(mat)
6
```

## Resultado

```
[[2 5 8]
 [1 2 6]
 [4 9 1]]
```

$$mat = \begin{pmatrix} 2 & 5 & 8 \\ 1 & 2 & 6 \\ 4 & 9 & 1 \end{pmatrix}$$

# Operações com *Matrizes (array)*

Primeira linha, segunda coluna

```
In [5]: mat[0,1]  
Out[5]: 5
```

$$mat = \begin{pmatrix} 2 & 5 & 8 \\ 1 & 2 & 6 \\ 4 & 9 & 1 \end{pmatrix}$$

# Operações com *Matrizes (array)*

Primeira linha, segunda coluna

```
In [5]: mat[0,1]  
Out[5]: 5
```

Segunda linha inteira

```
In [6]: mat[1,:]  
Out[6]: array([1, 2, 6])
```

$$mat = \begin{pmatrix} 2 & 5 & 8 \\ 1 & 2 & 6 \\ 4 & 9 & 1 \end{pmatrix}$$

# Operações com *Matrizes (array)*

Primeira linha, segunda coluna

```
In [5]: mat[0,1]  
Out[5]: 5
```

Segunda linha inteira

```
In [6]: mat[1,:]  
Out[6]: array([1, 2, 6])
```

Segunda coluna inteira

```
In [7]: mat[:,1]  
Out[7]: array([5, 2, 9])
```

$$mat = \begin{pmatrix} 2 & 5 & 8 \\ 1 & 2 & 6 \\ 4 & 9 & 1 \end{pmatrix}$$

# Operações com *Matrizes (array)*

Primeira linha, segunda coluna

```
In [5]: mat[0,1]  
Out[5]: 5
```

Segunda linha inteira

```
In [6]: mat[1,:]  
Out[6]: array([1, 2, 6])
```

Segunda coluna inteira

```
In [7]: mat[:,1]  
Out[7]: array([5, 2, 9])
```

Última linha, duas últimas colunas

```
In [8]: mat[-1,-2:]  
Out[8]: array([9, 1])
```

$$mat = \begin{pmatrix} 2 & 5 & 8 \\ 1 & 2 & 6 \\ 4 & 9 & 1 \end{pmatrix}$$

# Operações com *Matrizes (array)*

Cria um vetor e depois transforma em matriz

```
1 # Matrizes
2 import numpy as np
3
4 mat=np.arange(10)
5 print(mat)
6
```

[0 1 2 3 4 5 6 7 8 9]

{cria vetor de 1 a 10}



# Operações com *Matrizes (array)*

Cria um vetor e depois transforma em matriz

```
1 # Matrizes
2 import numpy as np
3
4 mat=np.arange(10)
5 print(mat)
6
```

[0 1 2 3 4 5 6 7 8 9]

{cria vetor de 1 a 10}

```
1 # Matrizes
2 import numpy as np
3
4 mat=np.arange(10)
5
6 mat=mat.reshape((5,2))
7 |
8 print(mat)
```

[[0 1]  
[2 3]  
[4 5]  
[6 7]  
[8 9]]

{transforma em matriz 5x2}

# Operações com *Matrizes (array)*

Cria matriz identidade 5x5

```
In [12]: np.identity(5)
```

# Operações com *Matrizes (array)*

## Cria matriz identidade 5x5

```
In [12]: np.identity(5)
```

```
array([[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.]])
```

## Cria matriz nula 3x3

```
In [13]: np.zeros((3,3))
```

# Operações com *Matrizes (array)*

## Cria matriz identidade 5x5

```
In [12]: np.identity(5)
```

```
array([[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.]])
```

## Cria matriz nula 3x3

```
In [13]: np.zeros((3,3))
```

```
array([[0., 0., 0.],  
       [0., 0., 0.],  
       [0., 0., 0.]])
```

## Cria matriz unitária 2x2

```
In [14]: np.ones((2,2))
```

# Operações com *Matrizes (array)*

## Cria matriz identidade 5x5

```
In [12]: np.identity(5)
```

```
array([[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.]])
```

## Cria matriz nula 3x3

```
In [13]: np.zeros((3,3))
```

```
array([[0., 0., 0.],  
       [0., 0., 0.],  
       [0., 0., 0.]])
```

## Cria matriz unitária 2x2

```
In [14]: np.ones((2,2))
```

```
array([[1., 1.],  
       [1., 1.]])
```

# Operações com *Matrizes (array)*

Matriz com números aleatórios uniforme( *precisa da biblioteca **numpy***)

```
aleat=np.random.rand(n,n)  
print(aleat)
```

Para n =3:

```
[ [0.38678315 0.69233713 0.67609648]  
  [0.5715626  0.10641785 0.76833536]  
  [0.07774609 0.27812185 0.47500038]]
```

# Operações com *Matrizes (array)*

Descobrir o número de linhas e colunas da matriz

```
1 # Descobrindo o número de Linhas e Colunas
2 import numpy as np
3
4 mat=np.array( [ [2,5,8], [1, 2, 6]])
5 ordem=mat.shape
6
7 print('-----')
8 print(ordem)
9 print('----- LINHAS -----')
10 print(ordem[0])
11 print('----- COLUNAS -----')
12 print(ordem[1])
```

$$mat = \begin{pmatrix} 2 & 5 & 8 \\ 1 & 2 & 6 \end{pmatrix}$$

# Operações com *Matrizes (array)*

Descobrir o número de linhas e colunas da matriz

$$mat = \begin{pmatrix} 2 & 5 & 8 \\ 1 & 2 & 6 \end{pmatrix}$$

```
1 # Descobrindo o número de Linhas e Colunas
2 import numpy as np
3
4 mat=np.array( [ [2,5,8], [1, 2, 6]] )
5 ordem=mat.shape
6
7 print('-----')
8 print(ordem)
9 print('----- LINHAS -----')
10 print(ordem[0])
11 print('----- COLUNAS -----')
12 print(ordem[1])
```

(2, 3)



# Operações com *Matrizes (array)*

Descobrir o número de linhas e colunas da matriz

$$mat = \begin{pmatrix} 2 & 5 & 8 \\ 1 & 2 & 6 \end{pmatrix}$$

```
1 # Descobrindo o número de Linhas e Colunas
2 import numpy as np
3
4 mat=np.array( [ [2,5,8], [1, 2, 6]] )
5 ordem=mat.shape
6
7 print('-----')
8 print(ordem)
9 print('----- LINHAS -----')
10 print(ordem[0])
11 print('----- COLUNAS -----')
12 print(ordem[1])
```

## RESULTADO NO CONSOLE

```
-----
(2, 3)
----- LINHAS -----
2
----- COLUNAS -----
3
```

# Operações com *Matrizes (array)*

## Soma de matrizes

```
1 # Matrizes
2 import numpy as np
3
4 m1=np.array( [ [2,5], [3,-2] ])
5 m2=np.array( [ [4,1], [-3,-7] ])
6
7 soma=m1+m2
8
9 print(soma)
10
```

$$\begin{pmatrix} 2 & 5 \\ 3 & -2 \end{pmatrix} + \begin{pmatrix} 4 & 1 \\ -3 & -7 \end{pmatrix} =$$

# Operações com *Matrizes (array)*

## Soma de matrizes

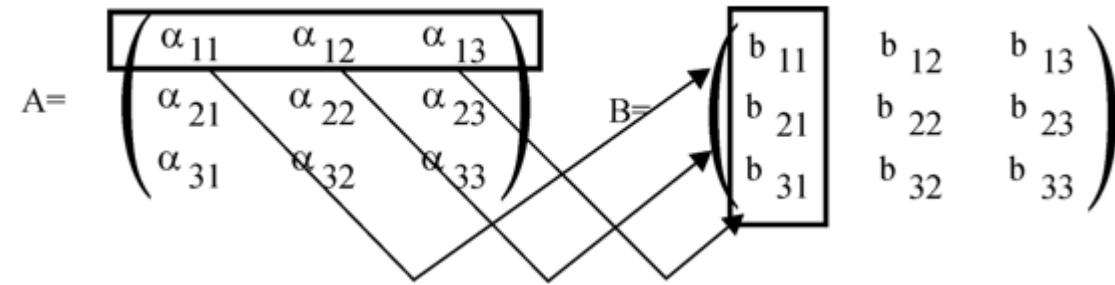
```
1 # Matrizes
2 import numpy as np
3
4 m1=np.array( [ [2,5], [3,-2] ])
5 m2=np.array( [ [4,1], [-3,-7] ])
6
7 soma=m1+m2
8
9 print(soma)
10
```

$$\begin{pmatrix} 2 & 5 \\ 3 & -2 \end{pmatrix} + \begin{pmatrix} 4 & 1 \\ -3 & -7 \end{pmatrix} = \begin{pmatrix} 6 & 6 \\ 0 & -9 \end{pmatrix}$$

$$\begin{bmatrix} 6 & 6 \\ 0 & -9 \end{bmatrix}$$

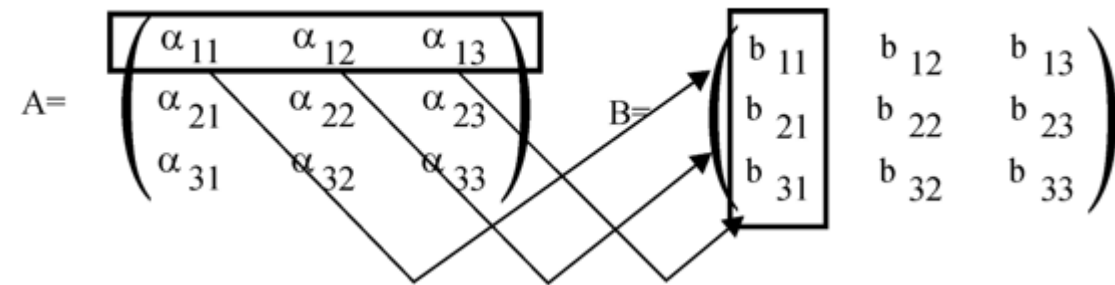
# Operações com *Matrizes (array)*

## Produto de matrizes



# Operações com *Matrizes (array)*

## Produto de matrizes



## Produto de matrizes (com np.matmul)

```
In [36]: np.matmul(m1,m2)
```

# Operações com *Matrizes (array)*

$$\begin{pmatrix} 2 & 5 \\ 3 & -2 \end{pmatrix} \times \begin{pmatrix} 4 & 1 \\ -3 & -7 \end{pmatrix} = \begin{pmatrix} -7 & -33 \\ 18 & 17 \end{pmatrix}$$

## Produto de matrizes (com np.matmul)

```
In [36]: np.matmul(m1,m2)
```

```
array([[ -7, -33],  
       [ 18,  17]])
```

# Operações com *Matrizes (array)*

## Inversa(inv)

Para inverter uma matriz o programa precisa importar a biblioteca de álgebra linear, chamada de “**linalg**”

```
1 # Matrizes
2 import numpy as np
3 from numpy.linalg import inv
4
```

# Operações com *Matrizes (array)*

## Inversa(inv)

Para inverter uma matriz o programa precisa importar a biblioteca de álgebra linear, chamada de “**linalg**”

```
1 # Matrizes
2 import numpy as np
3 from numpy.linalg import inv
4
5 m1=np.array( [ [2,5], [3,-2] ] )
6
```



# Operações com *Matrizes (array)*

## Inversa(inv)

Para inverter uma matriz o programa precisa importar a biblioteca de álgebra linear, chamada de “**linalg**”

```
1 # Matrizes
2 import numpy as np
3 from numpy.linalg import inv
4
5 m1=np.array( [ [2,5], [3,-2] ])
6
7 inversa= inv(m1)
8
9 print(inversa)
```

```
[[ 0.10526316  0.26315789]
 [ 0.15789474 -0.10526316]]
```

# Operações com *Matrizes (array)*

## Inversa(inv)

Produto da matriz pela inversa:  $A \cdot A^{-1}$  {usar *np.matmul*}

$$M1 = \begin{bmatrix} 2 & 5 \\ 3 & -2 \end{bmatrix}$$

$$\text{Inv}(M1) = \begin{bmatrix} 0.10526316 & 0.26315789 \\ 0.15789474 & -0.10526316 \end{bmatrix}$$

# Operações com *Matrizes (array)*

## Inversa(inv)

Produto da matriz pela inversa:  $A \cdot A^{-1}$  {usar *np.matmul*}

$$M1 = \begin{bmatrix} 2 & 5 \\ 3 & -2 \end{bmatrix}$$

$$\text{Inv}(M1) = \begin{bmatrix} 0.10526316 & 0.26315789 \\ 0.15789474 & -0.10526316 \end{bmatrix}$$

```
1 # Matrizes
2 import numpy as np
3 from numpy.linalg import inv
4
5 m1=np.array( [ [2,5], [3,-2] ])
6
7 inversa= inv(m1)
8
9 verif=np.matmul(m1,inversa)
10
11 print(verif)
```

# Operações com *Matrizes (array)*

## Inversa(inv)

Produto da matriz pela inversa:  $A \cdot A^{-1}$  {usar *np.matmul*}

$$M1 = \begin{bmatrix} 2 & 5 \\ 3 & -2 \end{bmatrix}$$

$$\text{Inv}(M1) = \begin{bmatrix} 0.10526316 & 0.26315789 \\ 0.15789474 & -0.10526316 \end{bmatrix}$$

```
1 # Matrizes
2 import numpy as np
3 from numpy.linalg import inv
4
5 m1=np.array( [ [2,5], [3,-2] ])
6
7 inversa= inv(m1)
8
9 verif=np.matmul(m1,inversa)
10
11 print(verif)
```

## Resultado

$$M1 * \text{Inv}(M1) = \begin{bmatrix} 1. & 0. \\ 0. & 1. \end{bmatrix}$$

# Operações com *Matrizes (array)*

Transposta (np.transpose)

$$m1 = \begin{bmatrix} 2 & 5 \\ 3 & -2 \end{bmatrix}$$

$$M = \begin{pmatrix} 2 & 5 \\ 3 & -2 \end{pmatrix}$$

$$M^T = \begin{pmatrix} 2 & 3 \\ 5 & -2 \end{pmatrix}$$

# Operações com *Matrizes (array)*

## Transposta (np.transpose)

m1=  $\begin{bmatrix} 2 & 5 \\ 3 & -2 \end{bmatrix}$

$$M = \begin{pmatrix} 2 & 5 \\ 3 & -2 \end{pmatrix}$$

$$M^T = \begin{pmatrix} 2 & 3 \\ 5 & -2 \end{pmatrix}$$

```
In [42]: np.transpose(m1)
```

### Resultado

```
array([[ 2,  3],  
       [ 5, -2]])
```

# Operações com *Matrizes (array)*

## Determinante (det)

```
a=np.array( [ [2,5], [3,-2] ] )
```

# Operações com *Matrizes (array)*

## Determinante (det)

```
a=np.array( [ [2,5], [3,-2] ])
```

## Resultado

```
In [52]: np.linalg.det(a)  
Out[52]: -18.999999999999996
```



# Operações com *Matrizes (array)*

## Sistema Linear

$$\begin{cases} 2x + 5y = -1 \\ 3x - 2y = 5 \end{cases}$$

## Transformar em matrizes

```
a= array([[ 2,  5],  
          [ 3, -2]])      b= array([-1,  5])
```

# Operações com *Matrizes (array)*

## Sistema Linear

$$\begin{cases} 2x + 5y = -1 \\ 3x - 2y = 5 \end{cases}$$

```
1 # Matrizes
2 import numpy as np
3 from numpy.linalg import inv
4
5 a=np.array( [ [2,5], [3,-2] ] )
6 b=np.array([-1,5])
```

# Operações com *Matrizes (array)*

## Sistema Linear

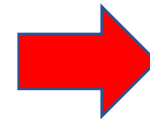
$$\begin{cases} 2x + 5y = -1 \\ 3x - 2y = 5 \end{cases}$$

```
1 # Matrizes
2 import numpy as np
3 from numpy.linalg import inv
4
5 a=np.array( [ [2,5], [3,-2] ])
6 b=np.array([-1,5])
7 inversa= inv(a)
8 x=np.matmul(inversa,b)
9
10 print(x)
```

## Solução do sistema linear

$$A \cdot x = B$$

$$x = A^{-1}B$$



$$\mathbf{x} = \text{inv}(\mathbf{a}) * \mathbf{b}$$

# Operações com *Matrizes (array)*

## Sistema Linear

$$\begin{cases} 2x + 5y = -1 \\ 3x - 2y = 5 \end{cases}$$

```
1 # Matrizes
2 import numpy as np
3 from numpy.linalg import inv
4
5 a=np.array( [ [2,5], [3,-2] ] )
6 b=np.array([-1,5])
7 inversa= inv(a)
8 x=np.matmul(inversa,b)
9
10 print(x)
```

## Solução do sistema linear

$$x = \text{inv}(a) * b$$

[ 1.21052632 -0.68421053]



x

# Operações com *Matrizes (array)*

## Sistema Linear

$$\begin{cases} 2x + 5y = -1 \\ 3x - 2y = 5 \end{cases}$$

```
1 # Matrizes
2 import numpy as np
3 from numpy.linalg import inv
4
5 a=np.array( [ [2,5], [3,-2] ] )
6 b=np.array([-1,5])
7 inversa= inv(a)
8 x=np.matmul(inversa,b)
9
10 print(x)
```

## Solução do sistema linear

$$\mathbf{x} = \text{inv}(\mathbf{a}) * \mathbf{b}$$

[ 1.21052632 -0.68421053]



y

# Operações com *Matrizes (array)*

Sistema Linear (resolução rápida com “solve”)

$$\begin{cases} 2x + 5y = -1 \\ 3x - 2y = 5 \end{cases}$$

```
In [49]: np.linalg.solve(a,b)  
Out[49]: array([ 1.21052632, -0.68421053])
```

# Operações com *Matrizes (array)*

Sistema Linear (*IMPORT* “solve”)

$$\begin{cases} 2x + 5y = -1 \\ 3x - 2y = 5 \end{cases}$$

$$A = \begin{pmatrix} 2 & 5 \\ 3 & -2 \end{pmatrix}$$

$$B = \begin{pmatrix} -1 \\ 5 \end{pmatrix}$$


# Operações com *Matrizes (array)*

Sistema Linear (*IMPORT* “solve”)

$$\begin{cases} 2x + 5y = -1 \\ 3x - 2y = 5 \end{cases}$$

$$A = \begin{pmatrix} 2 & 5 \\ 3 & -2 \end{pmatrix}$$

$$B = \begin{pmatrix} -1 \\ 5 \end{pmatrix}$$



```
1 # matriz
2 import numpy as np
3 from numpy.linalg import solve
4
5 A = np.array( [ [2,5],[3,-2] ] )
6 B = np.array( [-1,5])
7 X=solve(A,B)
8 print('+++++')
9 print('X = ',X[0])
10 print('Y = ',X[1])
11
```

```
+++++
X =  1.2105263157894737
Y = -0.6842105263157895
```



# APLICAÇÃO – SISTEMAS LINEARES

Um clube de investimento tem um montante em dinheiro para investimentos em ações. Para aceitar um risco relativo, os formadores do fundo decidiram dividir os investimentos em **alto risco** ( $x_a$ ), **médio risco** ( $x_m$ ) e **baixo risco** ( $x_b$ ). O fundo estima um retorno de 15% ao ano em ações de **alto risco**, 10% ao ano em ações de **médio risco** e 6% ao ano em investimentos de **baixo risco**. Os membros decidiram que investimentos de baixo risco devem ser iguais a soma das outras duas categorias. Determine quanto o clube deve investir em cada tipo de ações seguindo o seguinte cenário traçado por eles:

*O clube tem R\$ 200.000,00 para investir e o objetivo do investimento é ter um retorno de R\$ 20.000/ano sobre o total de investimentos.*

# APLICAÇÃO – SISTEMAS LINEARES

Um clube de investimento tem um montante em dinheiro para investimentos em ações. Para aceitar um risco relativo, os formadores do fundo decidiram dividir os investimentos em **alto risco** ( $x_a$ ), **médio risco** ( $x_m$ ) e **baixo risco** ( $x_b$ ). O fundo estima um retorno de 15% ao ano em ações de **alto risco**, 10% ao ano em ações de **médio risco** e 6% ao ano em investimentos de **baixo risco**. Os membros decidiram que investimentos de baixo risco devem ser iguais a soma das outras duas categorias. Determine quanto o clube deve investir em cada tipo de ações seguindo o seguinte cenário traçado por eles:

*O clube tem R\$ 200.000,00 para investir e o objetivo do investimento é ter um retorno de R\$ 20.000/ano sobre o total de investimentos.*

Com as taxas de retorno estimadas e o retorno desejada de R\$ 20.000/ano:  $0,15x_a + 0,1x_m + 0,06x_b = 20.000$

# APLICAÇÃO – SISTEMAS LINEARES

Um clube de investimento tem um montante em dinheiro para investimentos em ações. Para aceitar um risco relativo, os formadores do fundo decidiram dividir os investimentos em alto risco ( $x_a$ ), médio risco ( $x_m$ ) e baixo risco ( $x_b$ ). O fundo estima um retorno de 15% ao ano em ações de alto risco, 10% ao ano em ações de médio risco e 6% ao ano em investimentos de baixo risco. Os membros decidiram que investimentos de baixo risco devem ser iguais a soma das outras duas categorias. Determine quanto o clube deve investir em cada tipo de ações seguindo o seguinte cenário traçado por eles:

*O clube tem R\$ 200.000,00 para investir e o objetivo do investimento é ter um retorno de R\$ 20.000/ano sobre o total de investimentos.*

Com as taxas de retorno estimadas e o retorno desejada de R\$ 20.000/ano:  $0,15x_a + 0,1x_m + 0,06x_b = 20.000$

total de investimentos dos ativos de baixo risco:  $x_b = x_a + x_m$

# APLICAÇÃO – SISTEMAS LINEARES

Um clube de investimento tem um montante em dinheiro para investimentos em ações. Para aceitar um risco relativo, os formadores do fundo decidiram dividir os investimentos em alto risco ( $x_a$ ), médio risco ( $x_m$ ) e baixo risco ( $x_b$ ). O fundo estima um retorno de 15% ao ano em ações de alto risco, 10% ao ano em ações de médio risco e 6% ao ano em investimentos de baixo risco. Os membros decidiram que investimentos de baixo risco devem ser igual a soma das outras duas categorias. Determine quanto o clube deve investir em cada tipo de ações seguindo o seguinte cenário traçado por eles:

*O clube tem R\$ 200.000,00 para investir e o objetivo do investimento é ter um retorno de R\$ 20.000/ano sobre o total de investimentos.*

Com as taxas de retorno estimadas e o retorno desejada de R\$ 20.000/ano:  $0,15x_a + 0,1x_m + 0,06x_b = 20.000$

total de investimentos dos ativos de baixo risco:  $x_b = x_a + x_m$

total de investimentos necessários:  $x_a + x_m + x_b = 200.000$

# APLICAÇÃO – SISTEMAS LINEARES

Um clube de investimento tem um montante em dinheiro para investimentos em ações. Para aceitar um risco relativo, os formadores do fundo decidiram dividir os investimentos em alto risco ( $x_a$ ), médio risco ( $x_m$ ) e baixo risco ( $x_b$ ). O fundo estima um retorno de 15% ao ano em ações de alto risco, 10% ao ano em ações de médio risco e 6% ao ano em investimentos de baixo risco. Os membros decidiram que investimentos de baixo risco devem ser igual a soma das outras duas categorias. Determine quanto o clube deve investir em cada tipo de ações seguindo o seguinte cenário traçado por eles:

*O clube tem R\$ 200.000,00 para investir e o objetivo do investimento é ter um retorno de R\$ 20.000/ano sobre o total de investimentos.*

Com as taxas de retorno estimadas e o retorno desejada de R\$ 20.000/ano:  $0,15x_a + 0,1x_m + 0,06x_b = 20.000$

total de investimentos dos ativos de baixo risco:  $x_b = x_a + x_m$

total de investimentos necessários:  $x_a + x_m + x_b = 200.000$

# APLICAÇÃO – SISTEMAS LINEARES

Um clube de investimento tem um montante em dinheiro para investimentos em ações. Para aceitar um risco relativo, os formadores do fundo decidiram dividir os investimentos em alto risco ( $x_a$ ), médio risco ( $x_m$ ) e baixo risco ( $x_b$ ). O fundo estima um retorno de 15% ao ano em ações de alto risco, 10% ao ano em ações de médio risco e 6% ao ano em investimentos de baixo risco. Os membros decidiram que investimentos de baixo risco devem ser igual a soma das outras duas categorias. Determine quanto o clube deve investir em cada tipo de ações seguindo o seguinte cenário traçado por eles:

*O clube tem R\$ 200.000,00 para investir e o objetivo do investimento é ter um retorno de R\$ 20.000/ano sobre o total de investimentos.*

$$\begin{cases} 0,15x_a + 0,1x_m + 0,06x_b = 20.000 \\ -x_a - x_m + x_b = 0 \\ x_a + x_m + x_b = 200.000 \end{cases}$$

# APLICAÇÃO – SISTEMAS LINEARES

Um clube de investimento tem um montante em dinheiro para investimentos em ações. Para aceitar um risco relativo, os formadores do fundo decidiram dividir os investimentos em alto risco ( $x_a$ ), médio risco ( $x_m$ ) e baixo risco ( $x_b$ ). O fundo estima um retorno de 15% ao ano em ações de alto risco, 10% ao ano em ações de médio risco e 6% ao ano em investimentos de baixo risco. Os membros decidiram que investimentos de baixo risco devem ser igual a soma das outras duas categorias. Determine quanto o clube deve investir em cada tipo de ações seguindo o seguinte cenário traçado por eles:

*O clube tem R\$ 200.000,00 para investir e o objetivo do investimento é ter um retorno de R\$ 20.000/ano sobre o total de investimentos.*

$$\begin{cases} 0,15x_a + 0,1x_m + 0,06x_b = 20.000 \\ -x_a - x_m + x_b = 0 \\ x_a + x_m + x_b = 200.000 \end{cases}$$

$$A = \begin{pmatrix} 0,15 & 0,1 & 0,06 \\ -1 & -1 & 1 \\ 1 & 1 & 1 \end{pmatrix} \quad X = \begin{pmatrix} x_a \\ x_m \\ x_b \end{pmatrix} \quad B = \begin{pmatrix} 20.000 \\ 0 \\ 200000 \end{pmatrix}$$

# APLICAÇÃO – SISTEMAS LINEARES

Um clube de investimento tem um montante em dinheiro para investimentos em ações. Para aceitar um risco relativo, os formadores do fundo decidiram dividir os investimentos em alto risco ( $x_a$ ), médio risco ( $x_m$ ) e baixo risco ( $x_b$ ). O fundo estima um retorno de 15% ao ano em ações de alto risco, 10% ao ano em ações de médio risco e 6% ao ano em investimentos de baixo risco. Os membros decidiram que investimentos de baixo risco devem ser igual a soma das outras duas categorias. Determine quanto o clube deve investir em cada tipo de ações seguindo o seguinte cenário traçado por eles:

*O clube tem R\$ 200.000,00 para investir e o objetivo do investimento é ter um retorno de R\$ 20.000/ano sobre o total de investimentos.*

```
1 # matriz
2 import numpy as np
3 from numpy.linalg import solve
4
5 A = np.array( [ [0.15,0.1,0.06],[-1,-1,1],[1,1,1] ] )
6 B = np.array( [20000,0,200000] )
7 X=solve(A,B)
8 print('+++++')
9 print('Alto Risco = ',X[0])
10 print('Médio Risco = ',X[1])
11 print('Baixo Risco = ',X[2])
```

No Console,

```
+++++
Alto Risco =  80000.000000000001
Médio Risco =  19999.999999999999
Baixo Risco =  100000.0
```



# APLICAÇÃO – SISTEMAS LINEARES

O setor de transporte de cargas de uma cooperativa agrícola, que opera em São Paulo, dispõe de 3 modelos de caminhões A, B e C. Existe uma carga com no máximo 165 toneladas para ser remetida para o RS, outra no máximo com 300 toneladas para o MT e outra no máximo com 290 toneladas para MG. A capacidade de transporte por tonelada por caminhão é:

SP-RS: Modelo A: 20 ton.

Modelo B: 12 ton.

Modelo C: 1,5 ton.

SP-MT: Modelo A: 35 ton.

Modelo B: 22 ton.

Modelo C: 3,5

SP-MG: Modelo A: 40 ton.

Modelo B: 20 ton.

Modelo C: 2,0

*Quantos caminhões devem ser enviados para RS, MT e MG*

# Solução

```
1 ##### aula de matrizes #####
2 import numpy as np
3 from numpy.linalg import inv,det,solve
4
5 A=np.array([ [20,12,1.5],[35,22,3.5],[40,20,2]])
6 B=np.array([165,300,290])
7
8
9 x=solve(A,B)
10
11 print('Solução da Análise Logística')
12 print('+++++')
13 print('Qtde caminhoes A = %7.3f' % x[0])
14 print('Qtde caminhoes B = %7.3f' % x[1])
15 print('Qtde caminhoes C = %7.3f' % x[2])
```

## Solução da Análise Logística

+++++

Qtde caminhoes A = 2.400

Qtde caminhoes B = 9.500

Qtde caminhoes C = 2.000