

# O ecossistema Python para matemática computacional

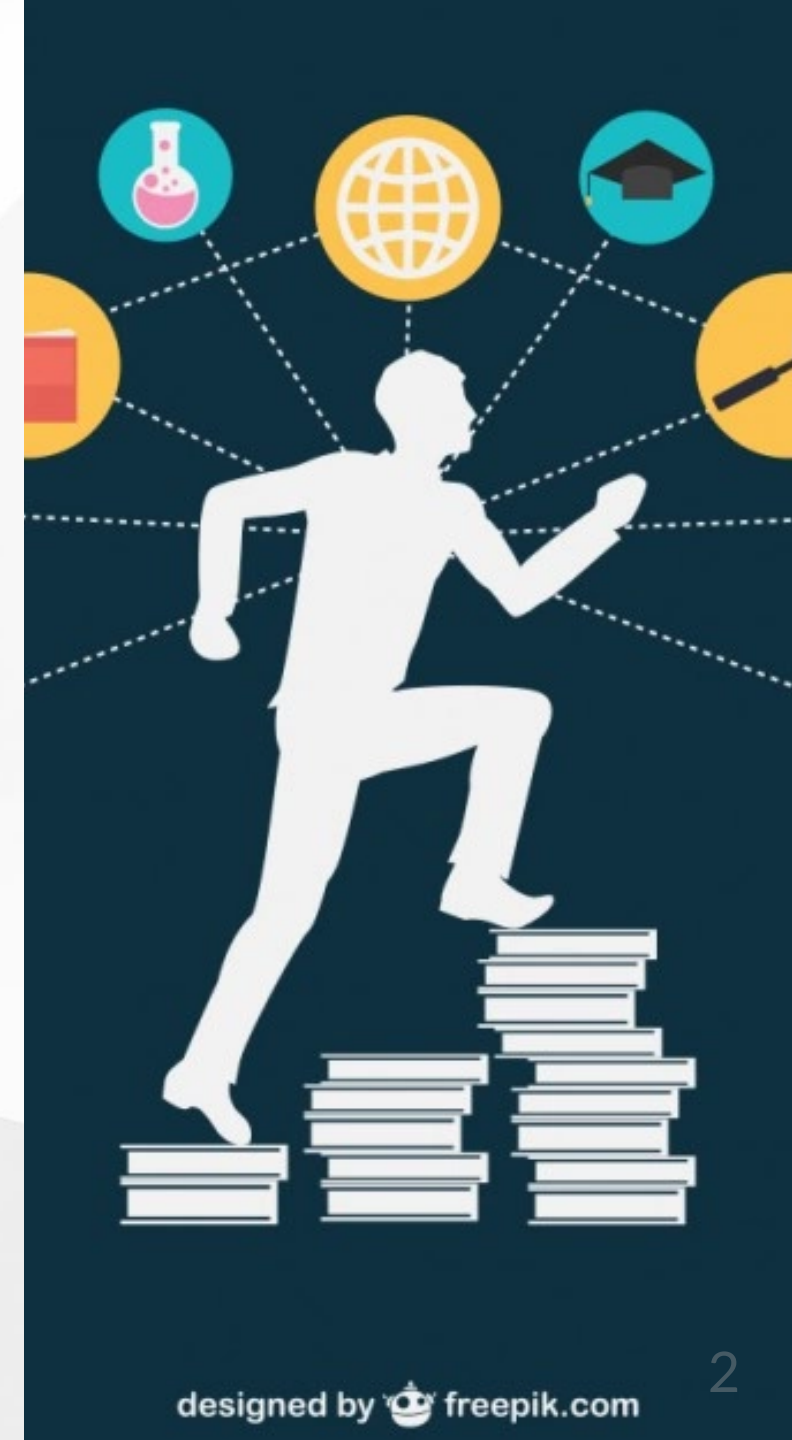
Leon Silva

[leon.silva@ufrpe.br](mailto:leon.silva@ufrpe.br)



## Sobre mim

- Lic. em Matemática (UFRPE)
- Mestre em Matemática (UFC)
- Doutor em Ciência da Computação (UFPE)



# Outline

- Um pouco sobre Python
- Comparações sobre outros softwares
- Um passeio sobre a sintaxe do Python
- Pacotes para Matemática computacional
- Onde usar Python











# Por que Python?



# Python ends C and Java's 20-year reign atop the TIOBE index

 by **Brandon Vigliarolo** in **Developer**   
on October 11, 2021, 6:51 AM PST


There hasn't been any other language atop the monthly index since 2001, making its topping of TIOBE just one more feather in Python's ever-plumier hat.

Oct 2021	Oct 2020	Change	Programming Language		Ratings	Change
1	3	▲	 Python		11.27%	-0.00%
2	1	▼	 C		11.16%	-5.79%
3	2	▼	 Java		10.46%	-2.11%
4	4		 C++		7.50%	+0.57%
5	5		 C#		5.26%	+1.10%
6	6		 Visual Basic		5.24%	+1.27%
7	7		 JavaScript		2.19%	+0.05%
8	10	▲	 SQL		2.17%	+0.61%
9	8	▼	 PHP		2.10%	+0.01%
10	17	▲	 Assembly language		2.06%	+0.99%

The top 10 languages in TIOBE's October 2021 ranking


Image: TIOBE

## WHITE PAPERS, WEBCASTS, AND DOWNLOADS

 Ultimate Cisco Certification Super Bundle: Lifetime Access

Training from TechRepublic Academy

[DOWNLOAD NOW](#)

 Four essential downloads for Linux administrators, and those who want to be one


Research from TechRepublic Premium

[DOWNLOAD NOW](#)

 Hiring Kit: Platform Engineer

Research from TechRepublic Premium

[DOWNLOAD NOW](#)

 From video games to mainframes--finding the perfect programmer for the job is easy with the right tools

Research from TechRepublic Premium

[DOWNLOAD NOW](#)

 The Project Management Professional Certification Training Bundle: Lifetime Access

## **Necessidade dos cientistas:**

- Carregar os dados
- Manipular e processar dados
- Manipular e operar com expressões algébricas
- Visualizar dados e resultados
- Alta qualidade e precisão

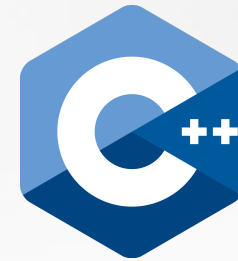
## Pontos fortes do Python:

- Não inventou a roda
- Fácil de aprender
- Legível
- Código eficiente
- Multipropósito



## C, C++, Fortran

- **Prós**
  - Muito rápida. Muito!
  - Grátis
- **Contra**
  - Sintaxe dolorosa
  - Gerenciamento manual da memória
  - Difíceis para não programadores.





# Linguagem Julia

- **Prós**

- Rápido e simples
- Capacidade de integração com Python

- **Contra**

- Limitado a cálculos numéricos
- Pouco testado



# Matlab

- **Prós**

- Muitos algoritmos disponíveis
- Rápido
- Editor integrado e agradável
- Suporte

- **Contra**

- Linguagem pobre
- Código fechado
- Pago



# Maple, Mathematica

- **Prós**

- Especializados em computação algébrica
- Editor próprio e útil
- Documentação profissional
- Suporte

- **Contra**

- Sintaxe pobre e confusa
- Código fechado
- Não é grátis



# Python

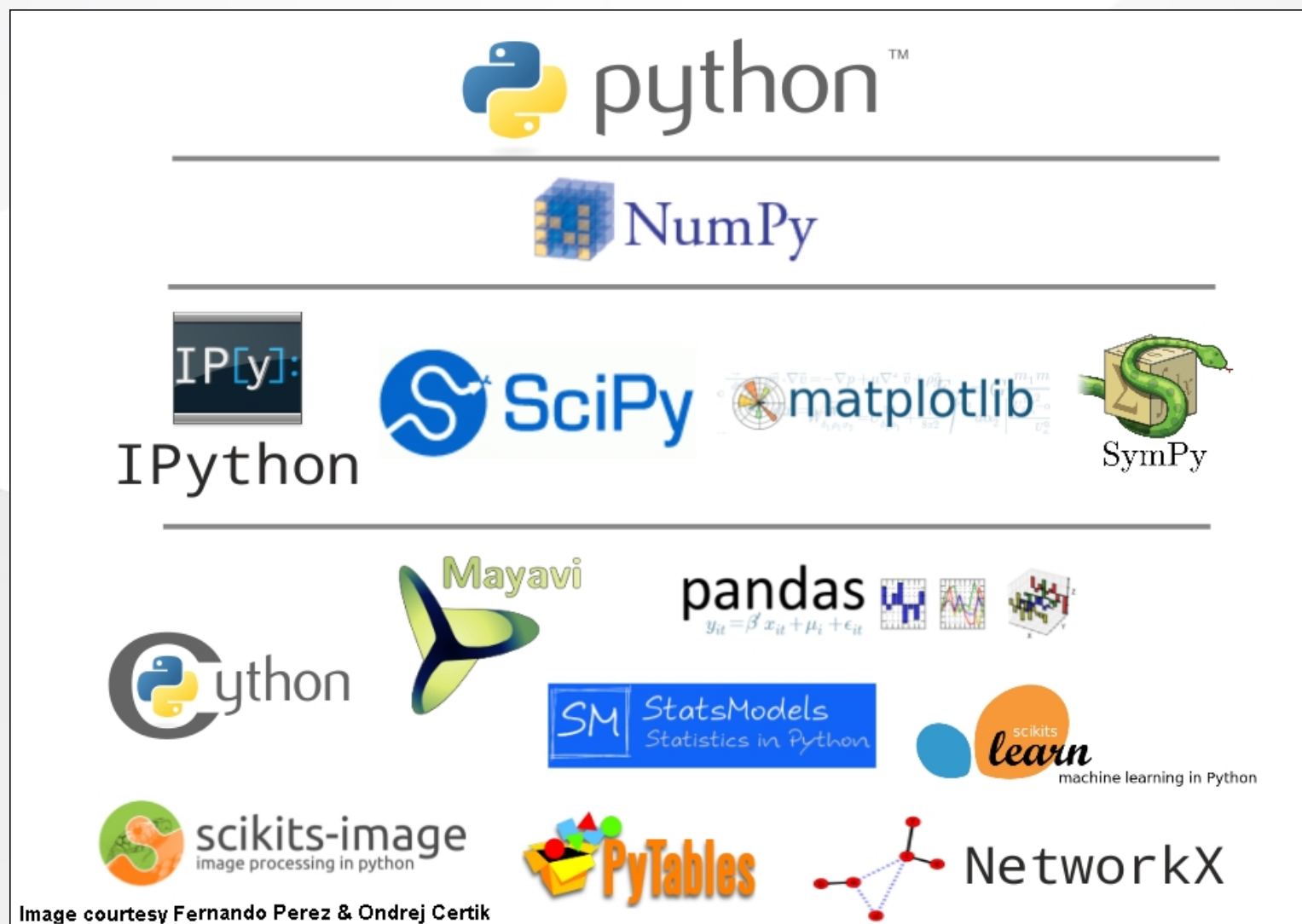
- **Prós**

- Várias bibliotecas para Matemática Computacional
- Linguagem poderosa e simples
- Gratuito e de código aberto
- Variedade de editores disponíveis

- **Contra**

- Nem todos os algoritmos estão disponíveis (ainda)





# Um passeio por Python e Matemática



# Matemática

$$\sum_{n=0}^{10} 3n$$

# Python

```
soma = 0
for n in range(11):
    soma += 3*n
```



# Matemática

$$\prod_{n=1}^{10} 2n$$

# Python

```
produto = 1
for n in range(1, 11):
    produto *= 2*n
```





# Matemática

$$A = \{n^2, \forall n \in \mathbb{N}; 20 \leq n < 100\}$$

# Python

```
a = 20  
b = 100  
A = set([n for n in range(a, b+1)])
```



## Matemática

- Sequência de Fibonacci

$$\begin{cases} F_0 = 0, F_1 = 1 \\ F_n = F_{n-1} + F_{n-2} \quad \text{para } n > 1 \end{cases}$$

## Python

```
def F(n):  
    if n==0:  
        return 0  
    elif n<=2:  
        return 1  
    return F(n-1) + F(n-2)
```



## Matemática

- Resolver:  $ax^2 + bx + c = 0$

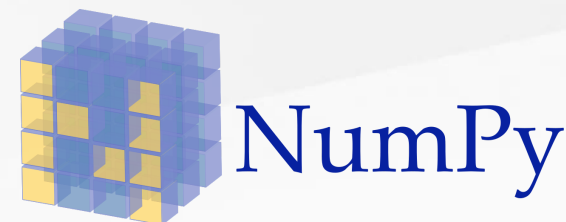
## Python

```
x = (b**2)-(4*a*c)
if x < 0 :
    print ("Raiz negativa nao pode ser extraida.")
else :
    x = math.sqrt(x)
    x1 = (-b + x)/(2*a);
    x2 = (-b - x)/(2*a)
    print ('x1 = ',x1, '\nx2 = ', x2)
```



## O que é o NumPy?

- Pacote de extensão para Python para matrizes multidimensionais
- Mais perto do hardware (eficiência)
- Projetado para computação científica (conveniência)
- Também conhecido como computação orientada a array



**nature**


[Explore content](#) ▾ [About the journal](#) ▾ [Publish with us](#) ▾

---

[nature](#) > [review articles](#) > [article](#)

Review Article | [Open Access](#) | [Published: 16 September 2020](#)

## Array programming with NumPy

[Charles R. Harris](#), [K. Jarrod Millman](#)  [\[...\]](#) [Travis E. Oliphant](#)

[Nature](#) **585**, 357–362 (2020) | [Cite this article](#)

[239k](#) Accesses | [1019](#) Citations | [1993](#) Altmetric | [Metrics](#)

---

### Abstract

Array programming provides a powerful, compact and expressive syntax for accessing, manipulating and operating on data in vectors, matrices and higher-dimensional arrays. NumPy is the primary array programming library for the Python language. It has an essential role in research analysis pipelines in fields as diverse as physics, chemistry, astronomy, geoscience, biology, psychology, materials science, engineering, finance and economics. For example, in astronomy, NumPy was an important part of the software stack used in the discovery of gravitational waves<sup>1</sup> and in the first imaging of a black hole<sup>2</sup>. Here we review how a few fundamental array concepts lead to a simple and powerful programming paradigm for organizing, exploring and analysing scientific data. NumPy is the foundation

## Python

```
soma = 0
for n in range(11):
    soma += 3*n
```

## NumPy

```
import numpy

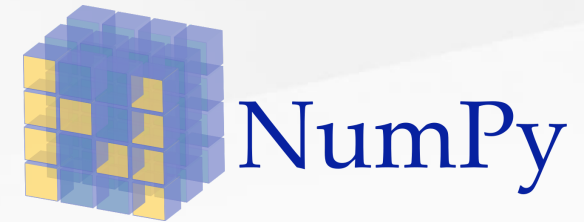
soma = numpy.sum(3*numpy.arange(11));
```



## Matemática

- Dada uma matriz  $A$  inversível:
  - Computar a inversa, autovetores e resolver o sistema  $Ax = b$ .

## NumPy



```
from numpy import linalg

linalg.inv(A)      #inversa de A

linalg.eigvals(A)  #autovalores

linalg.solve(A, b) #Resolve Ax=b
```

## Matemática

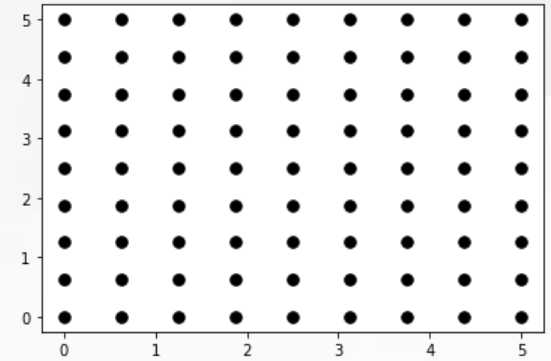
- Gerar coordenadas de 81 pontos na região  $R = [0, 5] \times [0, 5]$ , distribuídos uniformemente.

## NumPy

```
import numpy as np

nx, ny = (9, 9)
x = np.linspace(0, 5, nx)
y = np.linspace(0, 5, ny)

xv, yv = np.meshgrid(x, y)
```





# O que é o Matplotlib?

- Muito usado para gráficos 2d
- Fornece dados e figuras de qualidade de publicação
- Exporta figuras para diversos formatos
- Suporta simulações dinâmicas
- Está integrado ao NumPy

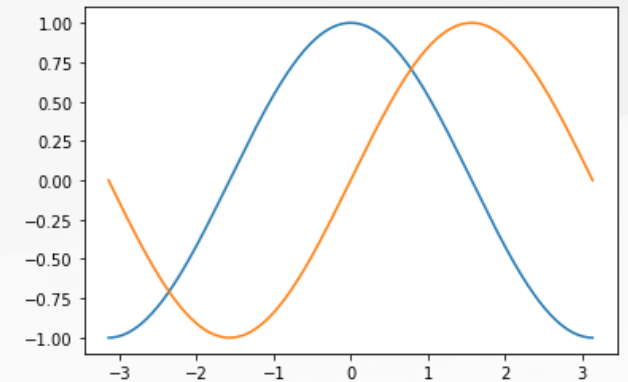


## Matplotlib: plot $y = \sin x$ e $y = \cos x$

```
# Gráficos 2D
```

```
import numpy as np  
import matplotlib.pyplot as plt
```

```
X = np.linspace(-np.pi, np.pi, 256)  
C, S = np.cos(X), np.sin(X)  
plt.plot(X, C)  
plt.plot(X, S)  
plt.show()
```

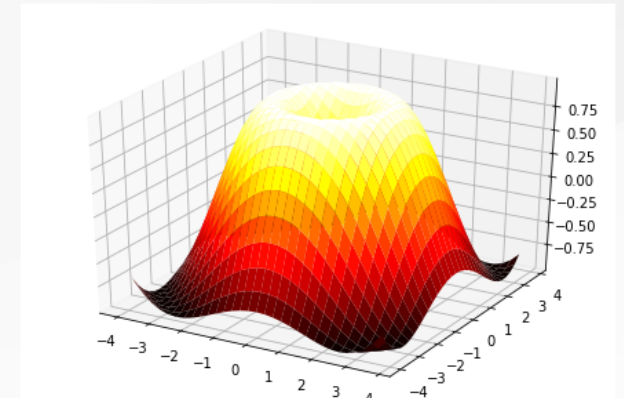


## Matplotlib: plot $z = \sin(\sqrt{x^2 + y^2})$

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
```

```
fig = plt.figure()
ax = Axes3D(fig)
X = np.arange(-4, 4, 0.25)
Y = np.arange(-4, 4, 0.25)
X, Y = np.meshgrid(X, Y)
R = np.sqrt(X ** 2 + Y ** 2)
Z = np.sin(R)

ax.plot_surface(X, Y, Z, cmap=plt.cm.hot)
plt.show()
```

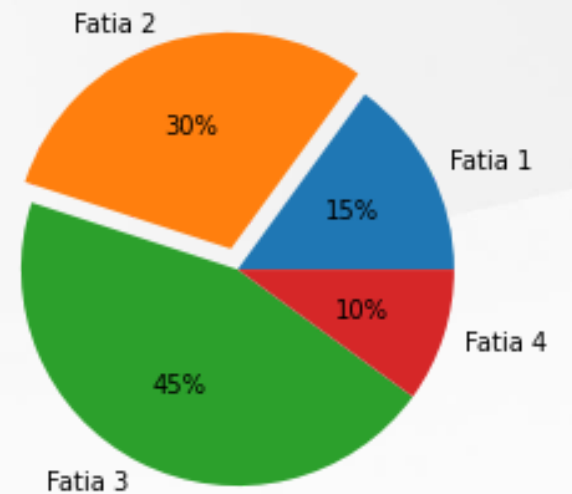


# Matplotlib: Gráfico de Pizza

```
import matplotlib.pyplot as plt

legendas = 'Fatia 1', 'Fatia 2', 'Fatia 3', 'Fatia 4'
porcentagem = [15, 30, 45, 10]
estourar = (0, 0.1, 0, 0)

fig, ax = plt.subplots()
ax.pie(porcentagem, estourar, legendas, autopct='%1.0f%%')
ax.axis()
plt.show()
```



## O que é o SciPy?

- Pacote principal para algoritmos científicos em Python
- Opera com eficiência matrizes do NumPy
- Dedicadas a muitas aplicações científicas



## Matemática

- Encontrar mínimo da função escalar
$$f(x) = x^2 + 10 \sin x, 0 < x < 10.$$

## SciPy

```
import numpy as np
from scipy import optimize

def f(x):
    return x**2 + 10*np.sin(x)

optimize.minimize(f, x0=0)
```



## Matemática

- Calcular  $\int_1^{10} \frac{\sin x}{x} dx$

## SciPy

```
import numpy as np
from scipy.integrate import quadrature
```

```
f = lambda x: np.sin(x)/x
quadrature(f, 1, 10)
```



## Matemática

- Resolver a EDO 
$$\begin{cases} \frac{dy}{dt} = 2y & 0 \leq t \leq 4 \\ y(0) = 1 \end{cases}$$

## SciPy



```
def calc_derivada(ypos, tempo):  
    return -2 * ypos  
  
from scipy.integrate import odeint  
  
t = np.linspace(0, 4, 40)  
y = odeint(calc_derivada, y0=1, t)
```



## O que é o SymPy?

- Operações algébricas em expressões algébricas
- Operação exatas do Cálculo
- Resolve equações algébricas
- Resolve EDO's



# Matemática

- Expandir a expressão  $(x + 1)^6$

## SymPy

```
import sympy  
  
x = sympy.symbols("x")  
  
sympy.expand((x+1)**6)
```



## Matemática

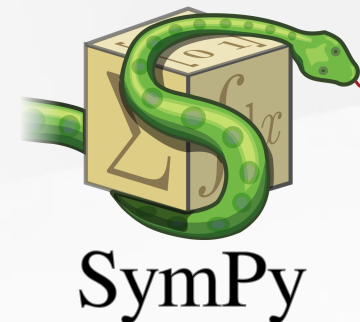
- $\lim_{x \rightarrow 0} \frac{\sin x}{x}$
- $\frac{d}{dx} [\sin x]$
- $\int \log x \, dx$

## SymPy

```
import sympy
# cálculos exatos
sympy.limit(sympy.sin(x)/x, x, 0) #limite

sympy.diff(sympy.sin(x), x) #derivada

sympy.integrate(sympy.log(x), x) #integral
```



## Matemática

- Resolver 
$$\begin{cases} x + 5y = 2 \\ -3x + 6y = 15 \end{cases}$$

## SymPy

```
import sympy as sym

x, y = sym.symbols('x, y')

sym.solve((x + 5*y-2, -3*x + 6*y-15), (x, y))
```



## Matemática

- Resolver a EDO  $y'' + 9y = 0$

## SymPy

```
from sympy import Function, dsolve, symbols  
  
x = symbols('x')  
y = Function('y')  
  
ddy = sympy.diff(y(x), x, x)  
  
dsolve(ddy + 9*y(x), y(x))
```



## O que é o Scikit-image?

- Processamento de imagens
  - Algoritmos:
    - segmentação
    - transformações geométricas
    - manipulação de cor
    - filtros



## Processamento

- Lendo arquivos da web (logo DM-UFRPE)

```
from skimage import io
import matplotlib.pyplot as plt

url = 'https://pymat.com.br/assets/images/logos/logo_dm.png'

logo_dm = io.imread(url)
plt.imshow(logo_dm)
plt.axis('off')
plt.show()
```



## Processamento

- RGB para tons de cinza

## Scikit-image

```
from skimage import color

logo_dm_grayscale = color.rgb2gray(logo_dm)
plt.imshow(logo_dm_grayscale, cmap=plt.cm.gray)
plt.axis('off')
plt.show()
```





## Processamento

- Trocar o primeiro fundo com o segundo

## Scikit-mage

```
from skimage import filters

# Otsu's method.
val = filters.threshold_otsu( grayscale )
plt.imshow( grayscale < val, cmap=plt.cm.gray )
plt.axis('off')
plt.show()
```



## Outros pacotes do ecossistema



# SageMath: o Capitão Planeta

- Inclui os pacotes Python:
  - NumPy
  - SciPy
  - SymPy
  - Matplotlib
  - NetworkX
- Além do:
  - R
  - Maxima, GAP e outros



# E agora?



# Onde usar o Python e seus ecossistema



Obrigado