

# Visualização de Dados

Albert E.F. Muritiba



**A visualização de dados é uma forma de comunicação que usa representações visuais para explorar, interpretar e apresentar dados.**

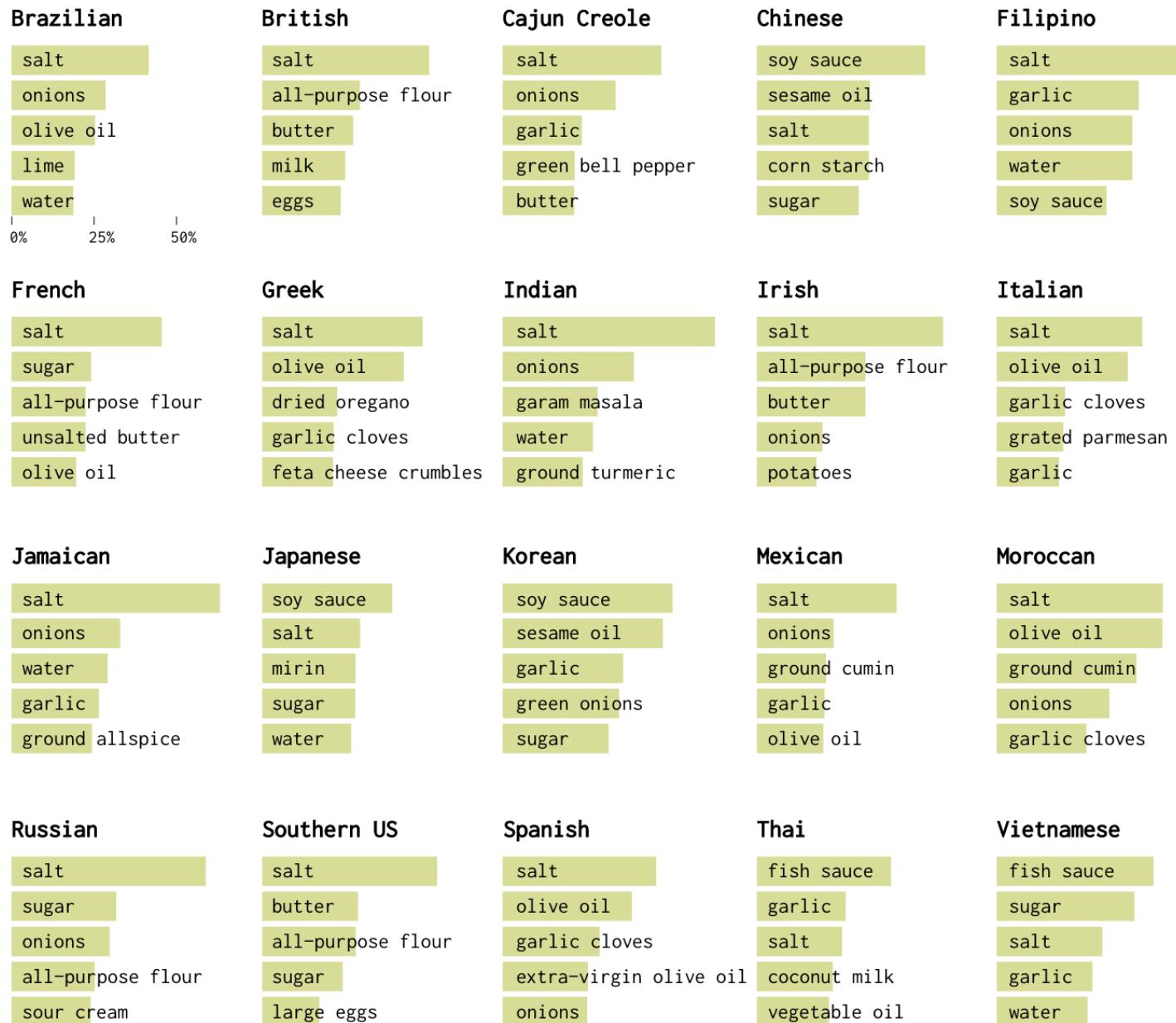
# Por que visualizar dados?

- Facilita a **compreensão** dos dados
- Facilita a identificação de **padrões**
- Facilita a identificação de **outliers**
- Facilita a identificação de **tendências**
- Facilita a identificação de **relações** entre variáveis
- Facilita a identificação de percepções (**insights**)

# MOST USED INGREDIENTS

## Ingredientes culinários

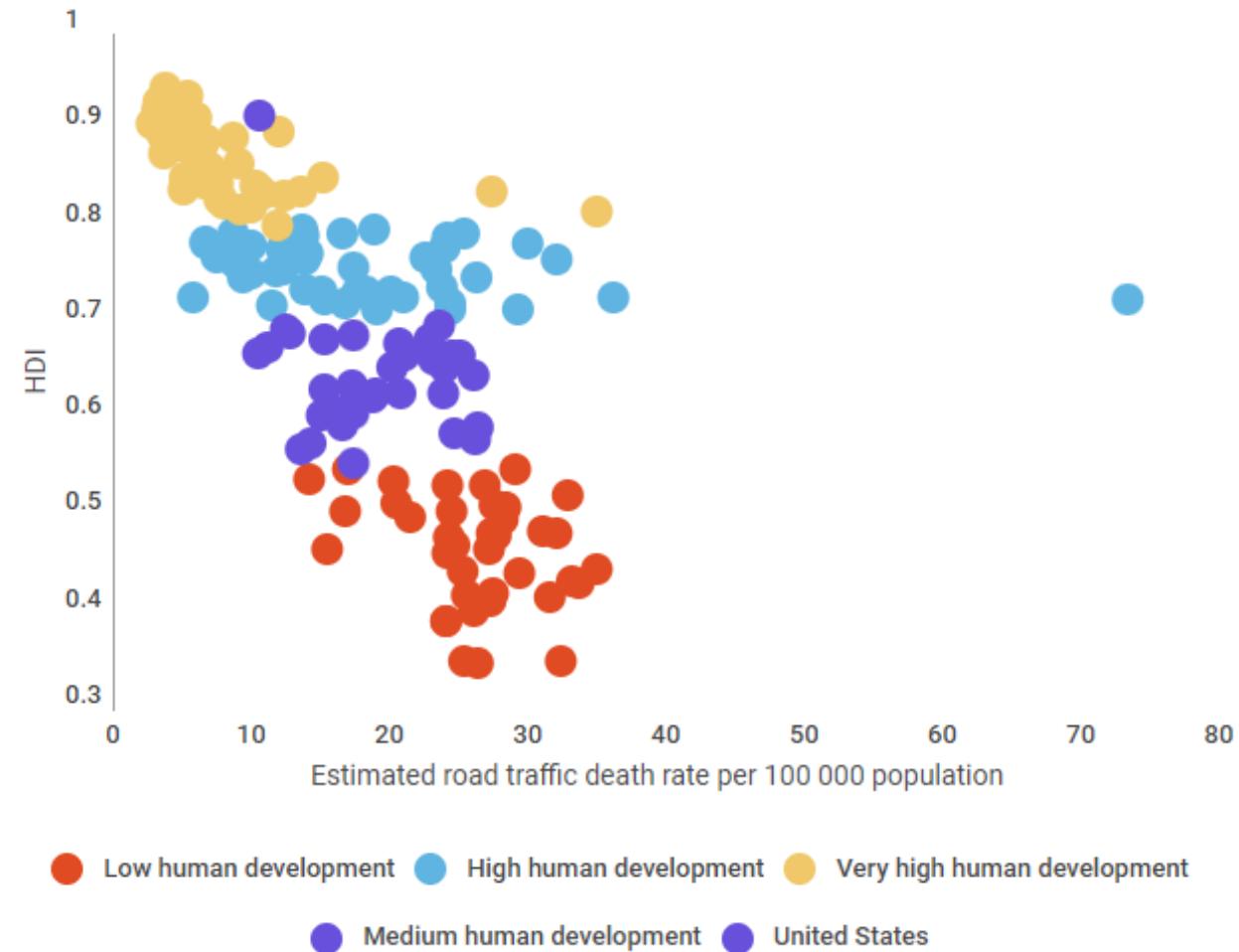
- Qual é o mais utilizado no Brasil?
- e em outros países?
- Onde se usa mais azeite de oliva?
- O que a Jamaica e a Índia têm em comum?
- [fonte](#)



Estimated road traffic death rate vs Human Development Index

## Mortes x IDH

- Algum país se destaca?  
(Líbia)
- Existe alguma relação entre as mortes e o IDH?
- Existe uma faixa de IDH com mais mortes?
- [fonte](#)



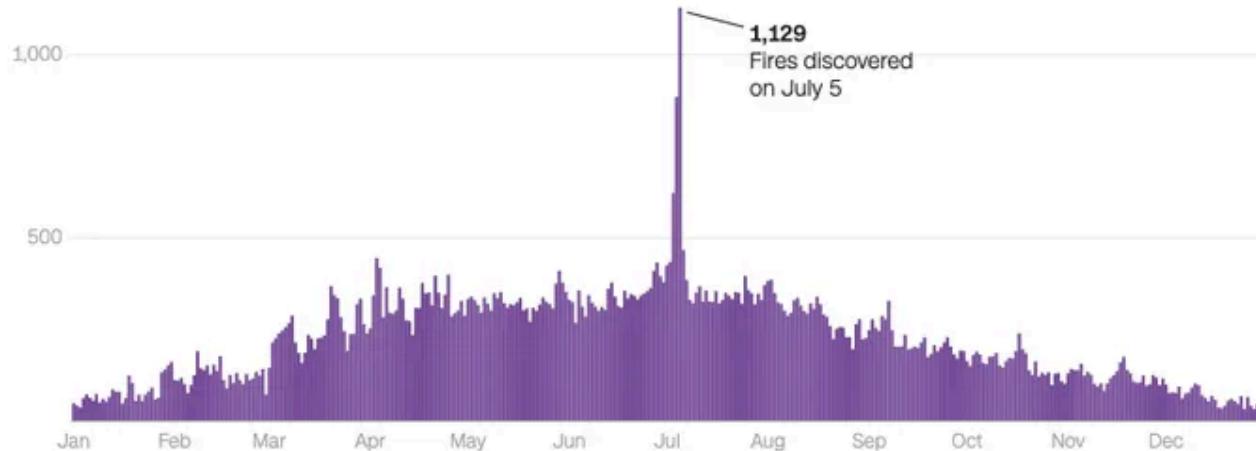
# Incêndios Florestais EUA

- Existem meses com mais incêndios?
- Quais?
- Tem algum *outlier*?
- Qual seria a explicação para esse *outlier*?
- fonte

## Wildfires spike around July 4 holiday

Human-caused wildfires in the United States jump around Independence Day.

Total wildfires discovered each day of the year since 2014



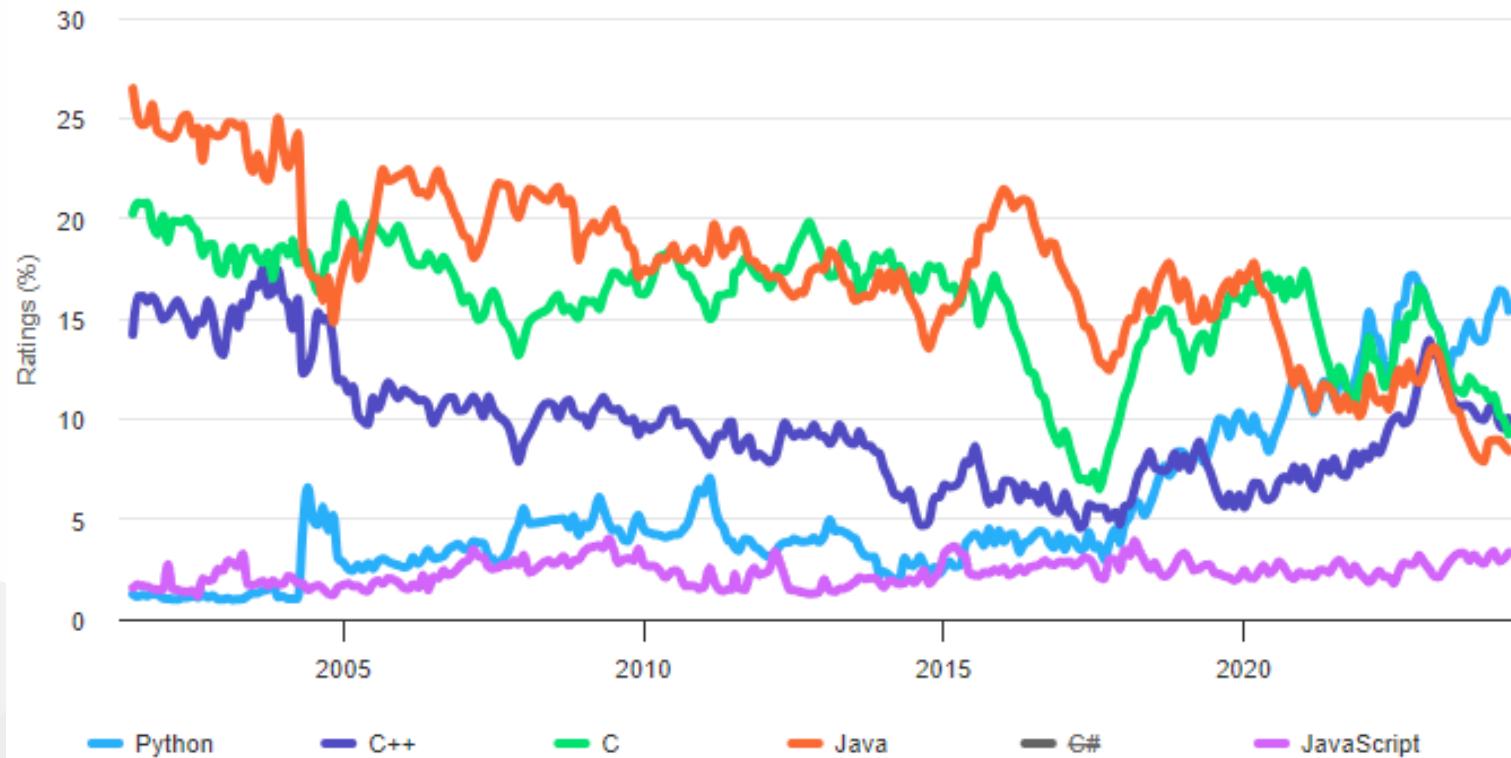
Human-caused fires, excluding prescribed fires. 2022 fires included through June 30. All incident times Eastern.

Sources: CNN analysis of data from the National Interagency Fire Center  
Graphic: John Keefe, CNN

CNN

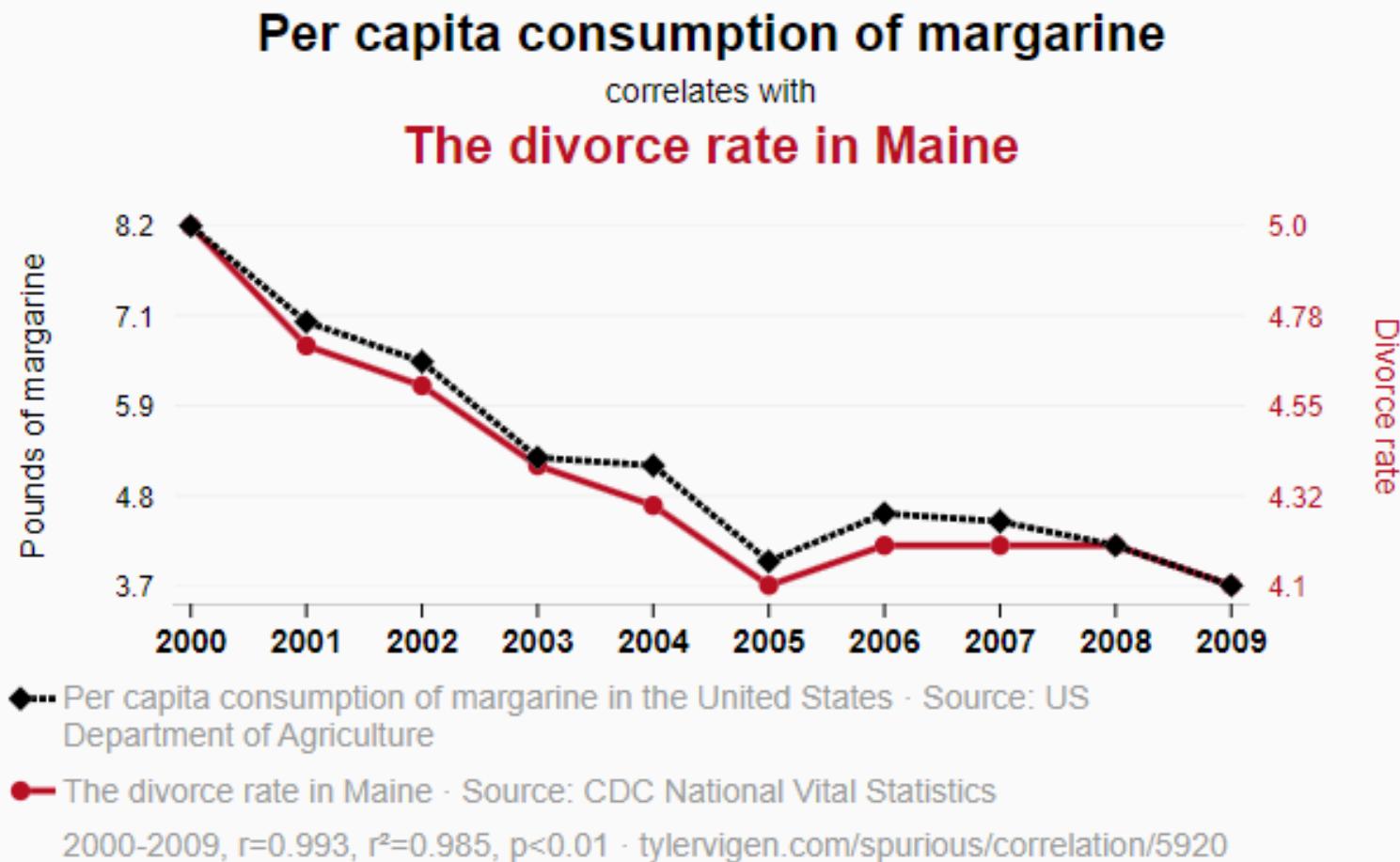
# Popularidade de Linguagens de Programação

- Qual a linguagem mais popular?
- Alguma está crescendo?
- Alguma está em declínio?
- fonte



## Divórcios x Margarina

- O que o consumo de margarina tem a ver com o número de divórcios?
- fonte



# Ferramentas de Visualização em Python

[Matplotlib](#), [Seaborn](#), [Plotly](#), [Bokeh](#), [Altair](#), [Plotnine](#), [Geopandas](#), [Folium](#), [Wordcloud](#), [Networkx](#), [Pydot](#), [Graphviz](#), entre outras.

# 1. Matplotlib

- **Descrição:** Biblioteca de visualização de dados mais antiga e amplamente utilizada.
- **Características:**
  - Suporte para gráficos 2D e 3D.
  - Altamente flexível e personalizável.
  - Base para outras bibliotecas como Seaborn e Pandas.

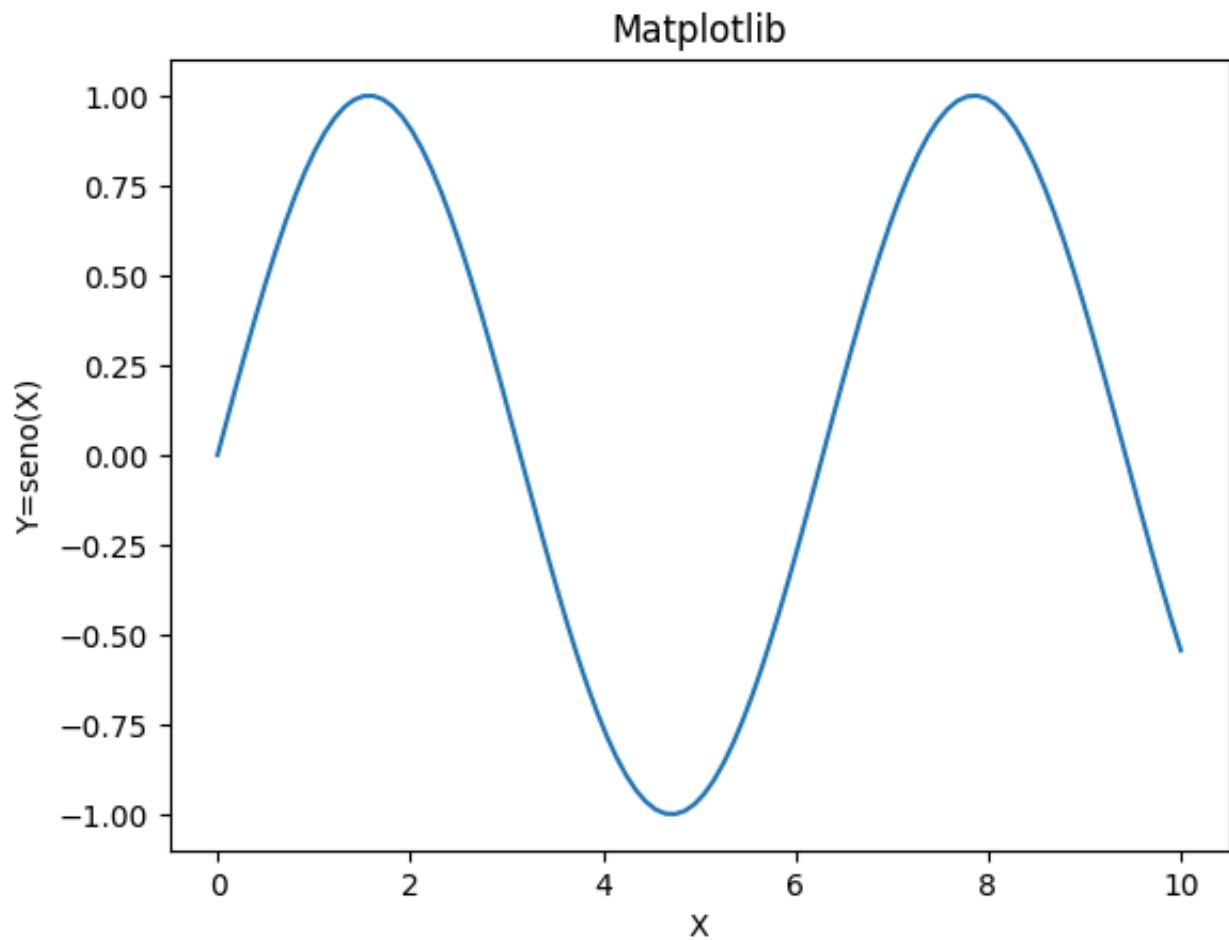


## Exemplo Matplotlib

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

x = np.linspace(0, 10, 100)
y = np.sin(x)

plt.plot(x, y)
plt.xlabel('X')
plt.ylabel('Y=seno(X)')
plt.title('Matplotlib')
plt.show()
```



## 2. Seaborn

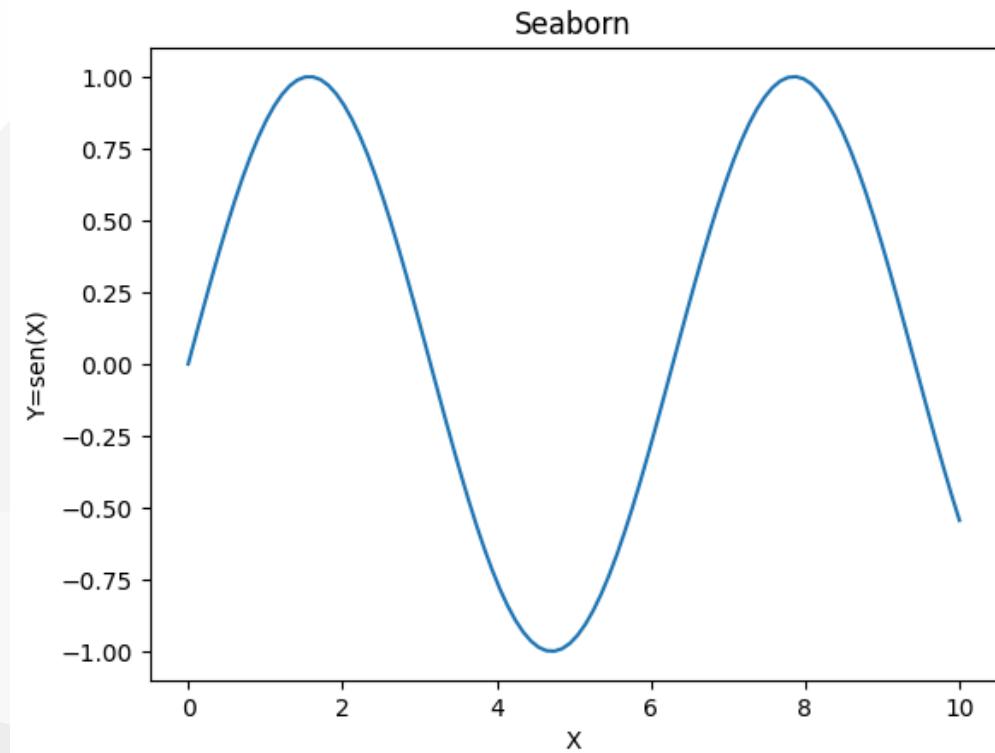
- **Descrição:** Construída sobre Matplotlib, oferece uma interface de alto nível para gráficos estatísticos.
- **Características:**
  - Estilização predefinida de gráficos.
  - Facilidade para criar gráficos complexos.
  - Integração fácil com DataFrames do Pandas.



## Exemplo Seaborn

```
import seaborn as sns
import matplotlib.pyplot as plt
x = np.linspace(0, 10, 100)
y = np.sin(x)
# Dados
data = {'x': x, 'y': y}

# Criação do gráfico
sns.lineplot(x='x', y='y', data=data)
plt.xlabel('X')
plt.ylabel('Y=sen(X)')
plt.title('Seaborn')
plt.show()
```



### 3. Plotly

- **Descrição:** Biblioteca de gráficos interativos para visualizações complexas.
- **Características:**
  - Gráficos interativos e personalizáveis.
  - Suporte para gráficos 3D e mapas.
  - Integração com frameworks web como Dash.



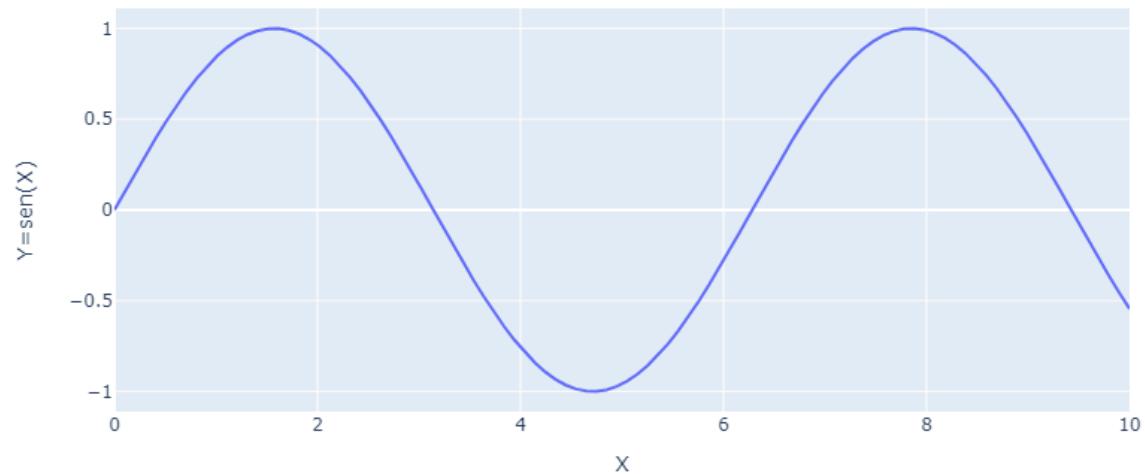
## Exemplo Plotly

```
import plotly.express as px
import numpy as np

x = np.linspace(0, 10, 100)
y = np.sin(x)

fig = px.line(x=x, y=y, title='Plotly')
fig.show()
```

Plotly



## 4. Bokeh

- **Descrição:** Biblioteca de visualização interativa para gráficos elegantes e concisos.
- **Características:**
  - Gráficos interativos e responsivos.
  - Integração com notebooks Jupyter.
  - Suporte para gráficos em tempo real.

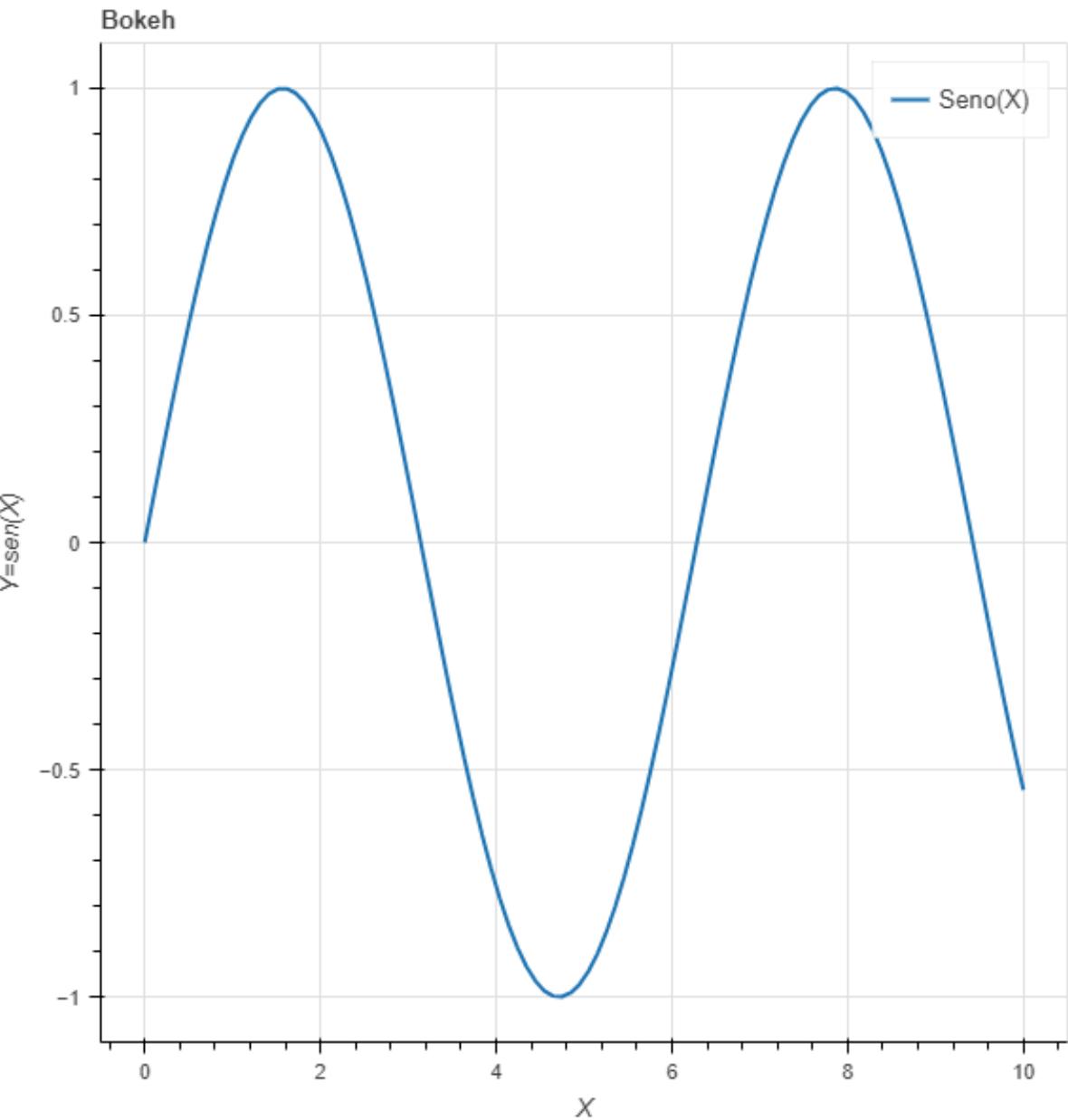


# Exemplo Bokeh

```
from bokeh.plotting import figure, show
import numpy as np

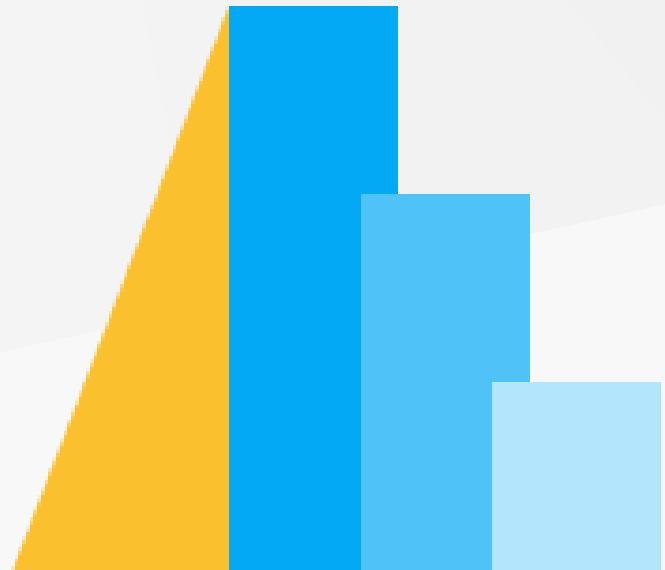
x = np.linspace(0, 10, 100)
y = np.sin(x)

p = figure(title='Bokeh', x_axis_label='X', y_axis_label='Y=sen(X)')
p.line(x, y, legend_label='Seno(X)', line_width=2)
show(p)
```



## 5. Altair

- **Descrição:** Biblioteca declarativa para gráficos estatísticos, baseada na linguagem Vega-Lite.
- **Características:**
  - Sintaxe declarativa simples.
  - Integração com DataFrames do Pandas.
  - Suporte para visualizações interativas.



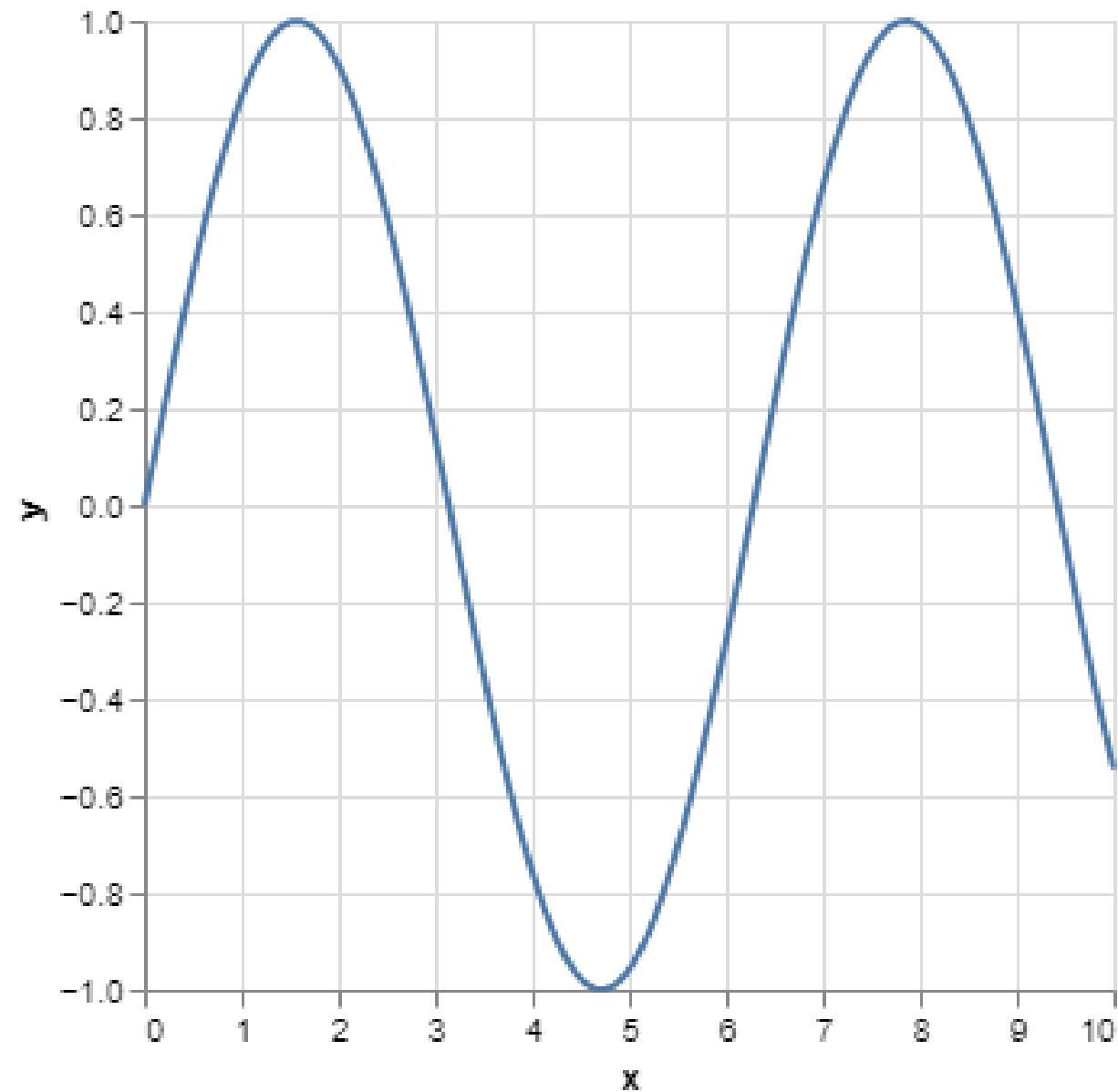
## Altair

# Exemplo Altair

```
import altair as alt
import numpy as np
import pandas as pd

x = np.linspace(0, 10, 100)
y = np.sin(x)
data = pd.DataFrame({'x': x, 'y': y})

alt.Chart(data).mark_line().encode(
    x='x',
    y='y'
).properties(
    title='Altair'
)
```



Neste curso, vamos focar na utilização da biblioteca **Matplotlib**. Por ser a mais antiga e amplamente utilizada, é importante conhecer seus recursos e funcionalidades.

# Matplotlib

## Figuras e Eixos

- A **figura** é a área total onde o gráfico será desenhado. Em Matplotlib, a figura é representada pelo objeto Figure.
- O que são Eixos?
- Os **eixos** são a área onde os dados são plotados. Eles representam o sistema de coordenadas do gráfico.

```
import matplotlib.pyplot as plt  
fig, ax = plt.subplots()
```

`subplots()` é uma função que cria uma nova figura e um novo conjunto de eixos.

# Elementos Básicos de um Gráfico

- **Título:** Título do gráfico.

```
ax.set_title('Título')
```

- **Rótulos dos Eixos:** Rótulos dos eixos x e y.

```
ax.set_xlabel('Eixo X')
ax.set_ylabel('Eixo Y')
```

- **Legenda:** Legenda do gráfico.

```
ax.legend(['Série 1', 'Série 2'])
```

- **Grade:** Linhas de grade no gráfico.

```
ax.grid(True)
```

## Tipos de Gráficos

- **Gráfico de Linhas:** Gráfico que exibe a relação entre duas variáveis.

```
ax.plot([1, 2, 3, 4], [10, 20, 25, 30])
```

Um gráfico de linha é adequado para exibir dados ordenados e contínuos, ou seja, se supõe que haja valores intermediários entre os pontos.

## Principais Parâmetros

- `x` : Valores do eixo x.
- `y` : Valores do eixo y.
- `label` : Rótulo da série.
- `color` : Cor da linha.
- `linestyle` : Estilo da linha.
- `linewidth` : Largura da linha.
- `marker` : Marcador dos pontos.

[saiba mais](#)

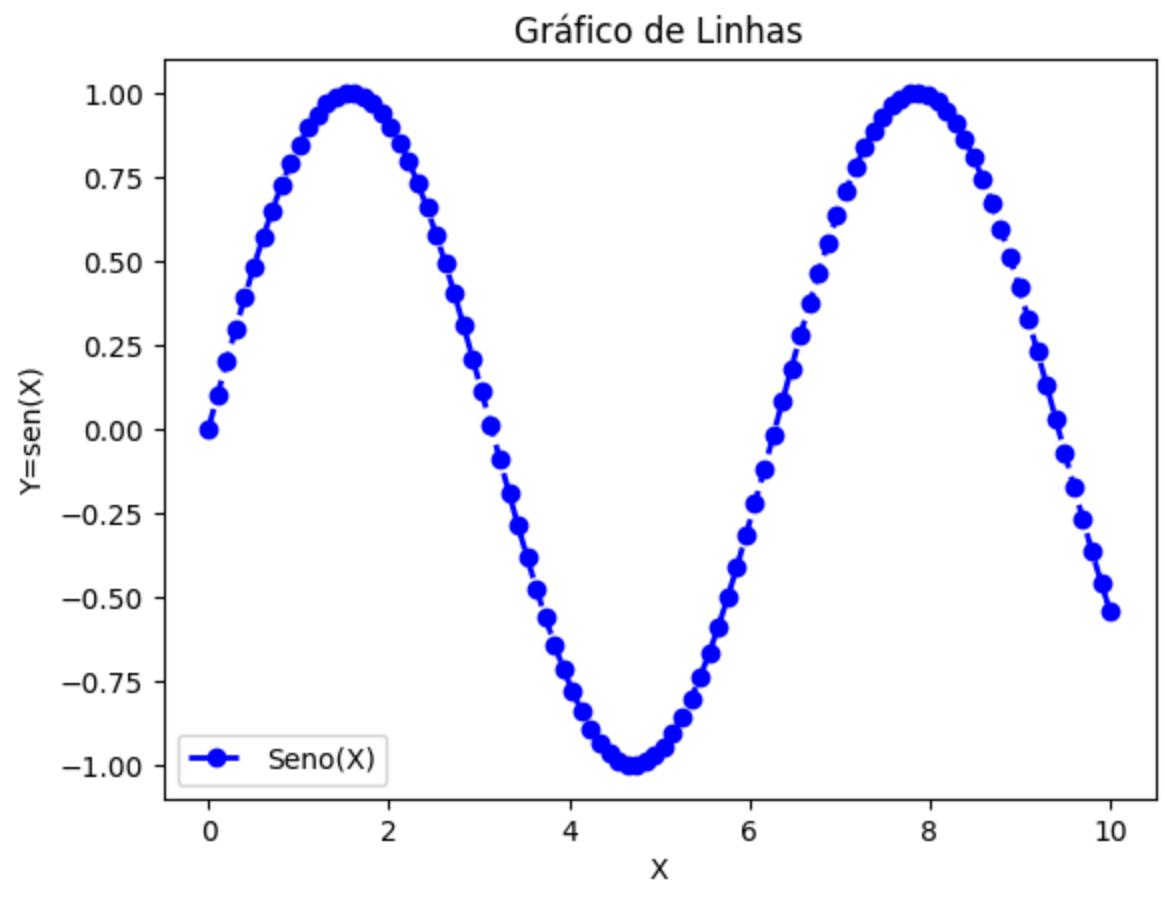
- Os valores x/y podem ser listas, arrays, séries ou DataFrames do Pandas.
- A cor da linha pode ser especificada por nome ou código hexadecimal. Ex: 'red', '#FF0000'.
- O estilo da linha pode ser sólido ('-'), tracejado ('--'), [outros](#).
- O marcador dos pontos pode ser 'o' (círculo), 's' (quadrado), '^' (triângulo), [outros](#).

## Exemplo Gráfico de Linhas

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

x = np.linspace(0, 10, 100)
y = np.sin(x)

fig, ax = plt.subplots()
ax.plot(x, y, label='Seno(X)', color='blue', linestyle='--', linewidth=2, marker='o')
ax.set_title('Gráfico de Linhas')
ax.set_xlabel('X')
ax.set_ylabel('Y=sen(X)')
ax.legend()
plt.show()
```

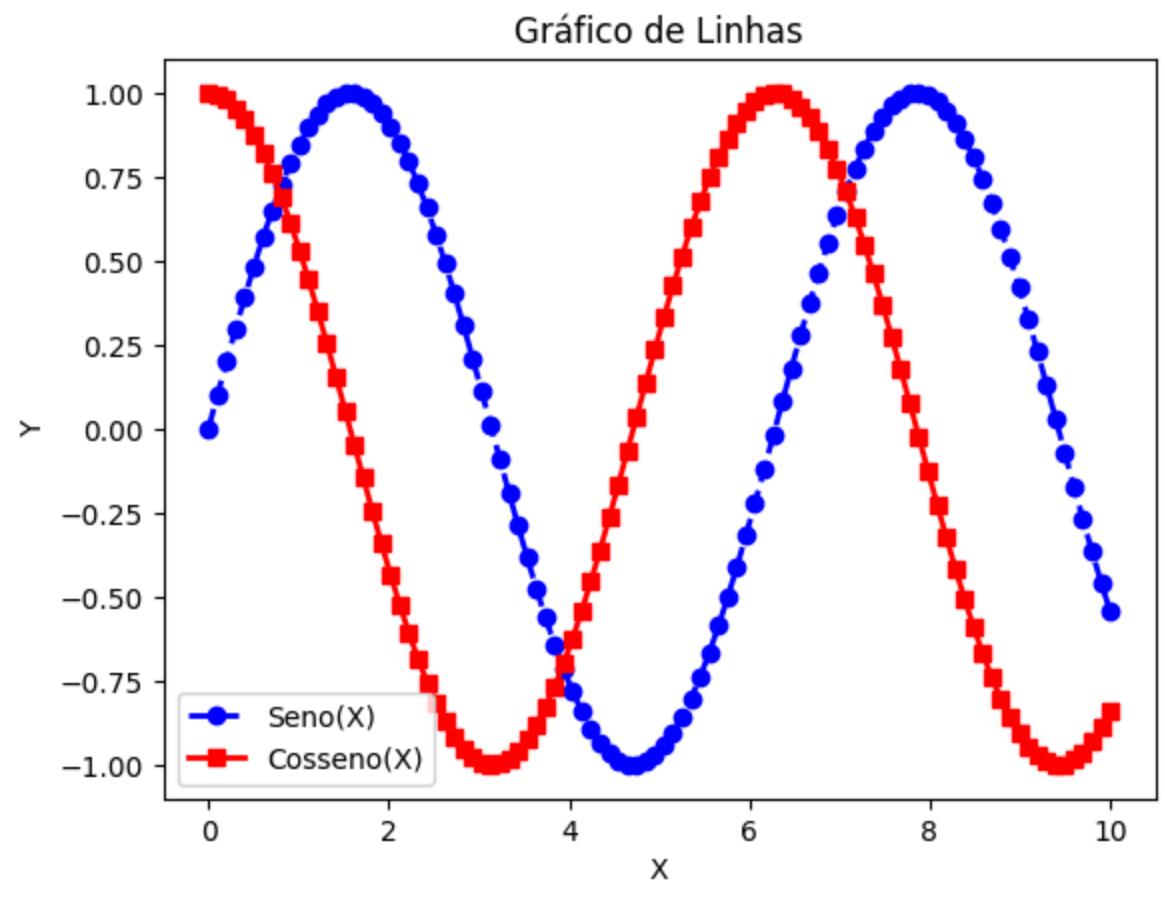


## Gráfico de linhas com mais de uma série

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

x = np.linspace(0, 10, 100)
y1 = np.sin(x)
y2 = np.cos(x)

fig, ax = plt.subplots()
ax.plot(x, y1, label='Seno(X)', color='blue', linestyle='--', linewidth=2, marker='o')
ax.plot(x, y2, label='Cosseno(X)', color='red', linestyle='-', linewidth=2, marker='s')
ax.set_title('Gráfico de Linhas')
ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.legend()
plt.show()
```



## Gráfico de Barras

- **Gráfico de Barras:** Gráfico que exibe a comparação entre diferentes categorias.

```
ax.bar(['A', 'B', 'C', 'D'], [10, 20, 15, 30])
```

Um gráfico de barras é adequado para exibir dados categóricos ou discretos.

## Principais Parâmetros

- `x` : Categorias.
- `height` : Altura das barras.
- `color` : Cor das barras.
- `edgecolor` : Cor da borda das barras.
- `linewidth` : Largura da borda das barras.
- `label` : Rótulo da série.
- `align` : Alinhamento das barras.

[saiba mais](#)

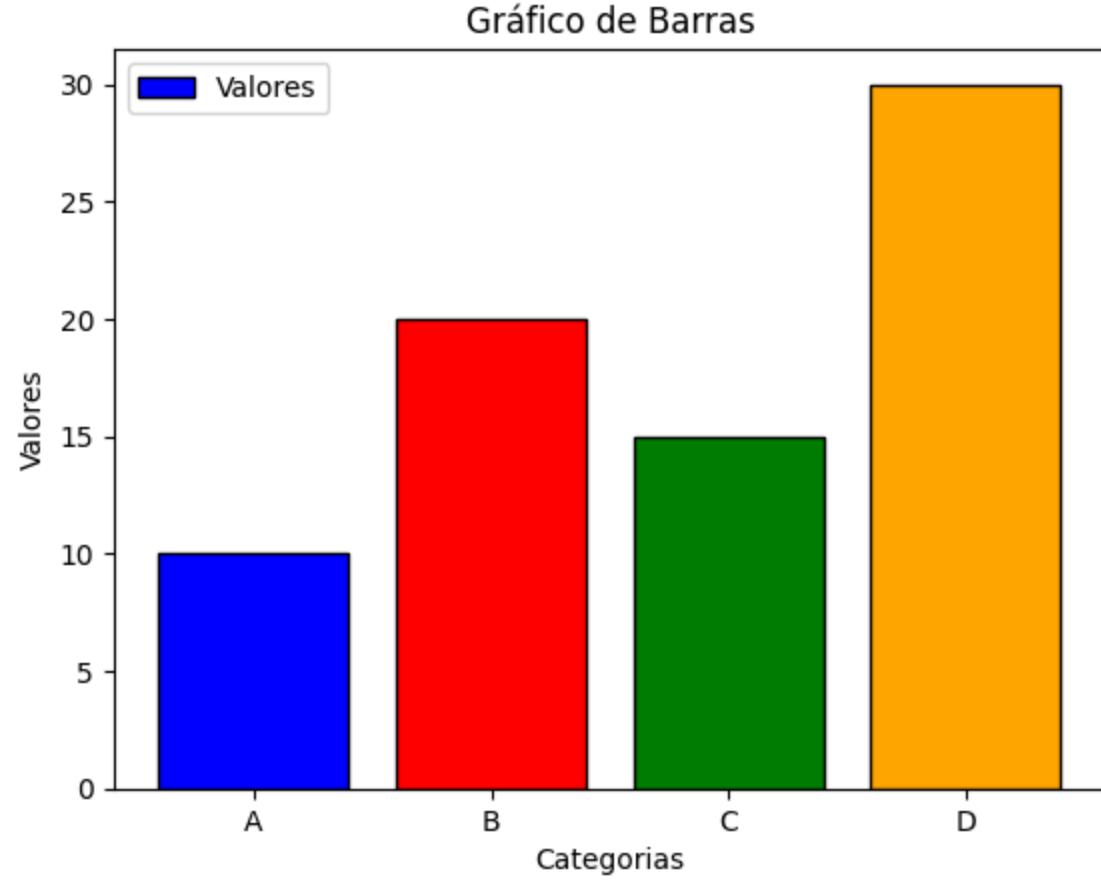
- As categorias x podem ser listas, arrays, séries ou DataFrames do Pandas.
- A cor das barras pode ser uniforme ou variável. Ex: 'blue', ['blue', 'red', 'green'].
- A largura das barras é definida automaticamente, mas pode ser ajustada.

## Exemplo Gráfico de Barras

```
import matplotlib.pyplot as plt

categorias = ['A', 'B', 'C', 'D']
valores = [10, 20, 15, 30]
cores = ['blue', 'red', 'green', 'orange']

fig, ax = plt.subplots()
ax.bar(categorias, valores, color=cores,
       edgecolor='black', linewidth=1, label='Valores')
ax.set_title('Gráfico de Barras')
ax.set_xlabel('Categorias')
ax.set_ylabel('Valores')
ax.legend()
plt.show()
```



## Gráfico de Barras Horizontais

- **Gráfico de Barras Horizontais:** Gráfico de barras com as categorias no eixo y.

```
ax.barh(['A', 'B', 'C', 'D'], [10, 20, 15, 30])
```

Tem a mesma finalidade do gráfico de barras, mas com as categorias no eixo y.

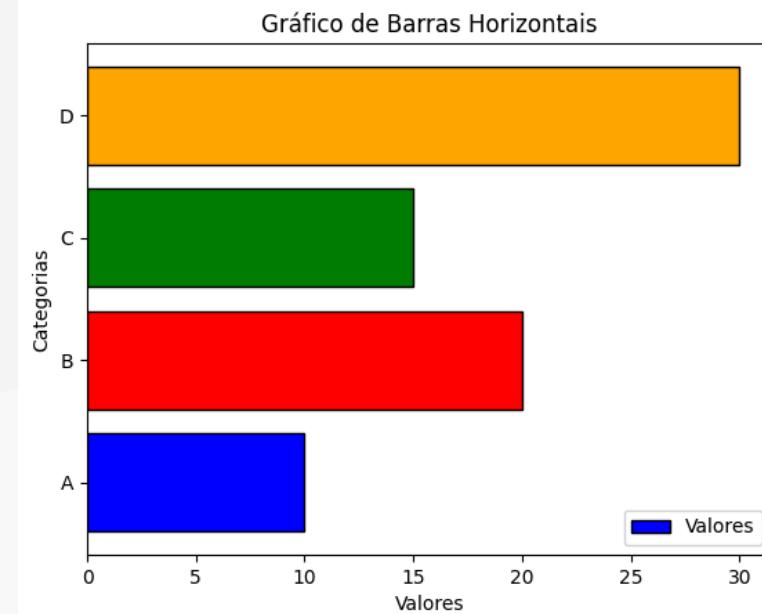
Pode ser útil para exibir muitas categorias.

## Exemplo Gráfico de Barras Horizontais

```
import matplotlib.pyplot as plt

categorias = ['A', 'B', 'C', 'D']
valores = [10, 20, 15, 30]
cores = ['blue', 'red', 'green', 'orange']

fig, ax = plt.subplots()
ax.barh(categorias, valores, color=cores,
         edgecolor='black', linewidth=1, label='Valores')
ax.set_title('Gráfico de Barras Horizontais')
ax.set_xlabel('Valores')
ax.set_ylabel('Categorias')
ax.legend()
plt.show()
```



## Gráfico de Pizza

- **Gráfico de Pizza:** Gráfico circular que exibe a proporção de cada categoria.

```
ax.pie([10, 20, 15, 30], labels=['A', 'B', 'C', 'D'])
```

Um gráfico de pizza é adequado para exibir a distribuição de categorias **em um todo**, ou seja, a proporção de cada categoria em relação ao total.

## Principais Parâmetros

- `x` : Valores das categorias.
- `labels` : Rótulos das categorias.
- `colors` : Cores das categorias.
- `explode` : Destaque de uma ou mais categorias.
- `autopct` : Formato dos percentuais.
- `shadow` : Sombra do gráfico.

[saiba mais](#)

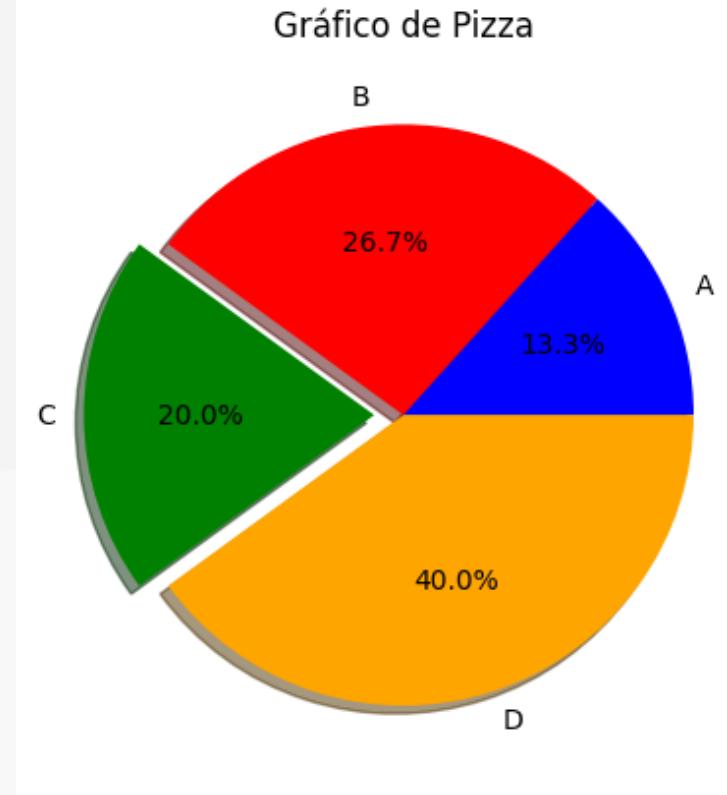
- O destaque de uma ou mais categorias é feito por meio de uma lista de valores.
- O formato dos percentuais pode ser uma string de formatação. Ex: '%.1f%%'.

## Exemplo Gráfico de Pizza

```
import matplotlib.pyplot as plt

valores = [10, 20, 15, 30]
categorias = ['A', 'B', 'C', 'D']
cores = ['blue', 'red', 'green', 'orange']
destaque = (0, 0, 0.1, 0)

fig, ax = plt.subplots()
ax.pie(valores, labels=categorias, colors=cores,
       explode=destaque, autopct='%.1f%%', shadow=True)
ax.set_title('Gráfico de Pizza')
plt.show()
```



## Gráfico de Dispersão

- **Gráfico de Dispersão:** Gráfico que exibe a relação entre duas variáveis.

```
ax.scatter([1, 2, 3, 4], [10, 20, 15, 30])
```

Um gráfico de dispersão é adequado para exibir a relação entre duas variáveis contínuas, onde não se supõe que haja qualquer função entre elas.

## Principais Parâmetros

- `x` : Valores do eixo x.
- `y` : Valores do eixo y.
- `s` : Tamanho dos pontos.
- `c` : Cor dos pontos.
- `marker` : Marcador dos pontos.
- `alpha` : Transparência dos pontos.
- `label` : Rótulo da série.
- `cmap` : Mapa de cores.

[saiba mais](#)

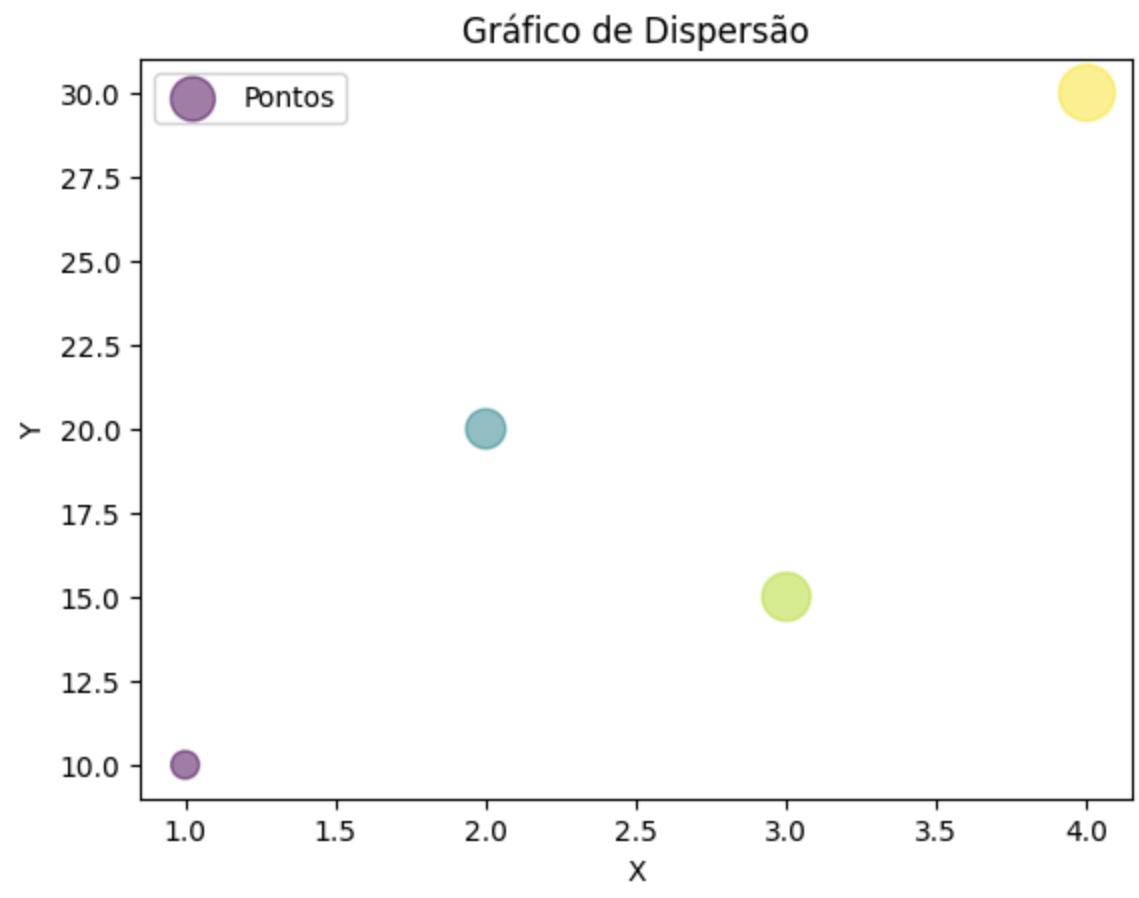
- Os valores x/y podem ser listas, arrays, séries ou DataFrames do Pandas.
- O tamanho dos pontos pode ser uniforme ou variável. Ex: 100, [10, 20, 30].
- A cor dos pontos pode ser uniforme ou variável. Ex: 'blue', ['blue', 'red', 'green'].
- O marcador dos pontos pode ser 'o' (círculo), 's' (quadrado), '^' (triângulo), [outros](#).
- A transparência dos pontos varia de 0 (transparente) a 1 (opaco).
- O mapa de cores é uma paleta de cores. Ex: 'viridis', 'plasma', 'inferno', 'magma'.
- A paleta de cores é útil para representar uma terceira variável.

## Exemplo Gráfico de Dispersão

```
import matplotlib.pyplot as plt

x = [1, 2, 3, 4]
y = [10, 20, 15, 30]
tamanhos = [100, 200, 300, 400]
cores = [0.1, 0.5, 0.9, 1]

fig, ax = plt.subplots()
ax.scatter(x, y, s=tamanhos, c=cores, cmap='viridis', alpha=0.5, label='Pontos')
ax.set_title('Gráfico de Dispersão')
ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.legend()
plt.show()
```



## Gráfico de Histograma

- **Gráfico de Histograma:** Gráfico que exibe a distribuição de uma variável contínua.

```
ax.hist([1, 2, 2, 3, 3, 3, 4, 4, 4, 4])
```

Um gráfico de histograma é adequado para exibir a distribuição de uma variável contínua em intervalos.

A altura das barras representa a frequência dos valores.

## Principais Parâmetros

- `x` : Valores da variável.
- `bins` : Número de intervalos.
- `range` : Intervalo dos valores.
- `color` : Cor das barras.
- `edgecolor` : Cor da borda das barras.
- `linewidth` : Largura da borda das barras.
- `label` : Rótulo da série.
- `density` : Densidade das barras.

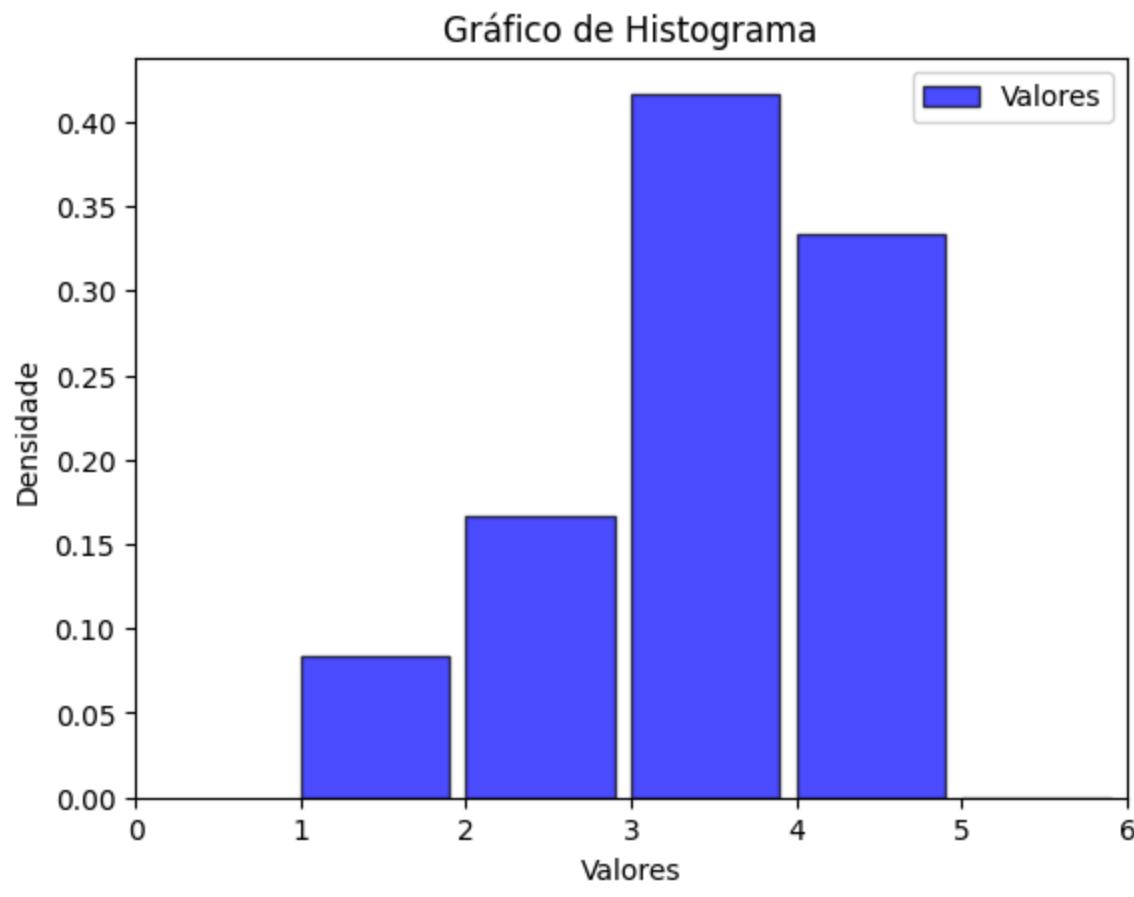
[saiba mais](#)

- Os valores x podem ser listas, arrays, séries ou DataFrames do Pandas.
- O número de intervalos é definido automaticamente, mas pode ser ajustado.
- A cor das barras pode ser uniforme ou variável. Ex: 'blue', ['blue', 'red', 'green'].
- A largura das barras é definida automaticamente, mas pode ser ajustada.
- A densidade das barras é a proporção de valores em cada intervalo.

## Exemplo Gráfico de Histograma

```
import matplotlib.pyplot as plt

valores = [1, 2, 2, 3, 3, 3, 4, 4, 4, 4, 3, 3]
fig, ax = plt.subplots()
ax.hist(valores, bins=[1, 2, 3, 4, 5, 6], width=0.9, color='blue',
        edgecolor='black', linewidth=1, label='Valores', density=True, alpha=0.7)
ax.set_xlim(0, 6) # Limites do eixo x
ax.set_title('Gráfico de Histograma')
ax.set_xlabel('Valores')
ax.set_ylabel('Densidade')
ax.legend()
plt.show()
```

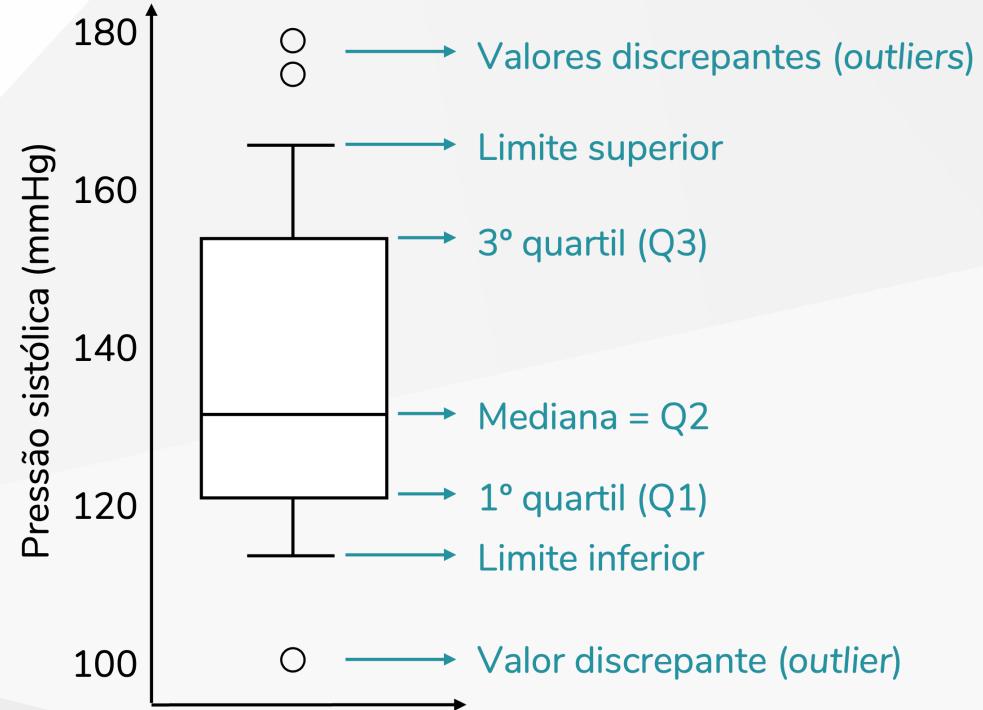


## Gráfico de Boxplot

- **Gráfico de Boxplot:** Gráfico que exibe a distribuição de uma variável contínua.

Um gráfico de boxplot é adequado para exibir a distribuição de uma variável contínua em quartis.

[fonte imagem](#)



## Principais Parâmetros

- `x` : Valores da variável.
- `vert` : Orientação do boxplot.
- `patch_artist` : Estilo do boxplot.
- `notch` : Intervalo de confiança.
- `showmeans` : Média dos valores.
- `showfliers` : Outliers.
- `showcaps` : Extremidades dos whiskers.
- `showbox` : Caixa do boxplot.

[saiba mais](#)

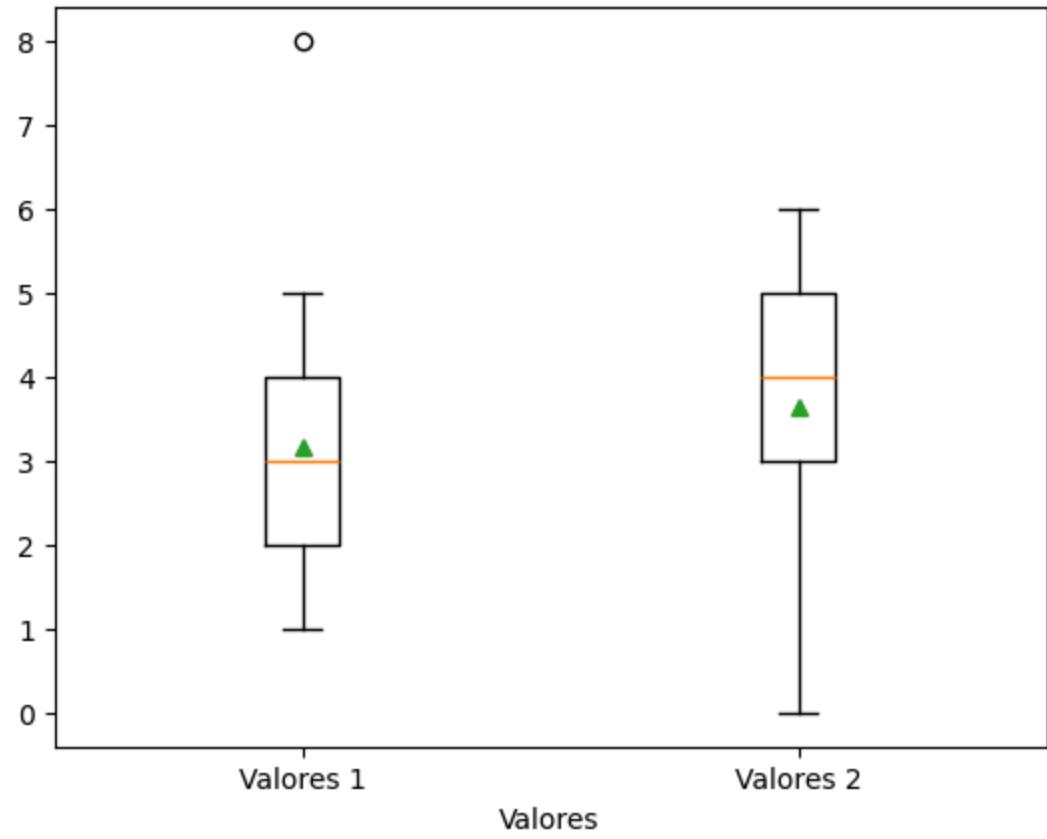
- Os valores x podem ser listas, arrays, séries ou DataFrames do Pandas.
- A orientação do boxplot pode ser vertical ('vert') ou horizontal ('horiz').
- O estilo do boxplot pode ser simples ou colorido.
- As extremidades são os valores mínimo e máximo.
- A caixa do boxplot é a região entre o primeiro e terceiro quartis.

## Exemplo Gráfico de Boxplot

```
import matplotlib.pyplot as plt

valores1 = [1, 2, 2, 3, 3, 3, 4, 4, 4, 4, 4, 5, 3, 3, 2, 2, 1, 8]
valores2 = [2, 3, 3, 4, 4, 4, 5, 5, 5, 5, 6, 4, 4, 3, 3, 2, 0]
fig, ax = plt.subplots()
ax.boxplot([valores1, valores2], labels=['Valores 1', 'Valores 2'], showmeans=True)
ax.set_title('Gráfico de Boxplot')
ax.set_xlabel('Valores')
plt.show()
```

### Gráfico de Boxplot



## Gráfico de Área

- Gráfico de Área: Gráfico que exibe a relação entre duas variáveis.

```
ax.fill_between([1, 2, 3, 4], [10, 20, 15, 30])
```

Um gráfico de área é adequado para exibir a relação entre duas variáveis contínuas, onde a área sob a curva é preenchida.

## Principais Parâmetros

- `x` : Valores do eixo x.
- `y1` : Valores da curva inferior.
- `y2` : Valores da curva superior (opcional).
- `color` : Cor da área.
- `alpha` : Transparência da área.

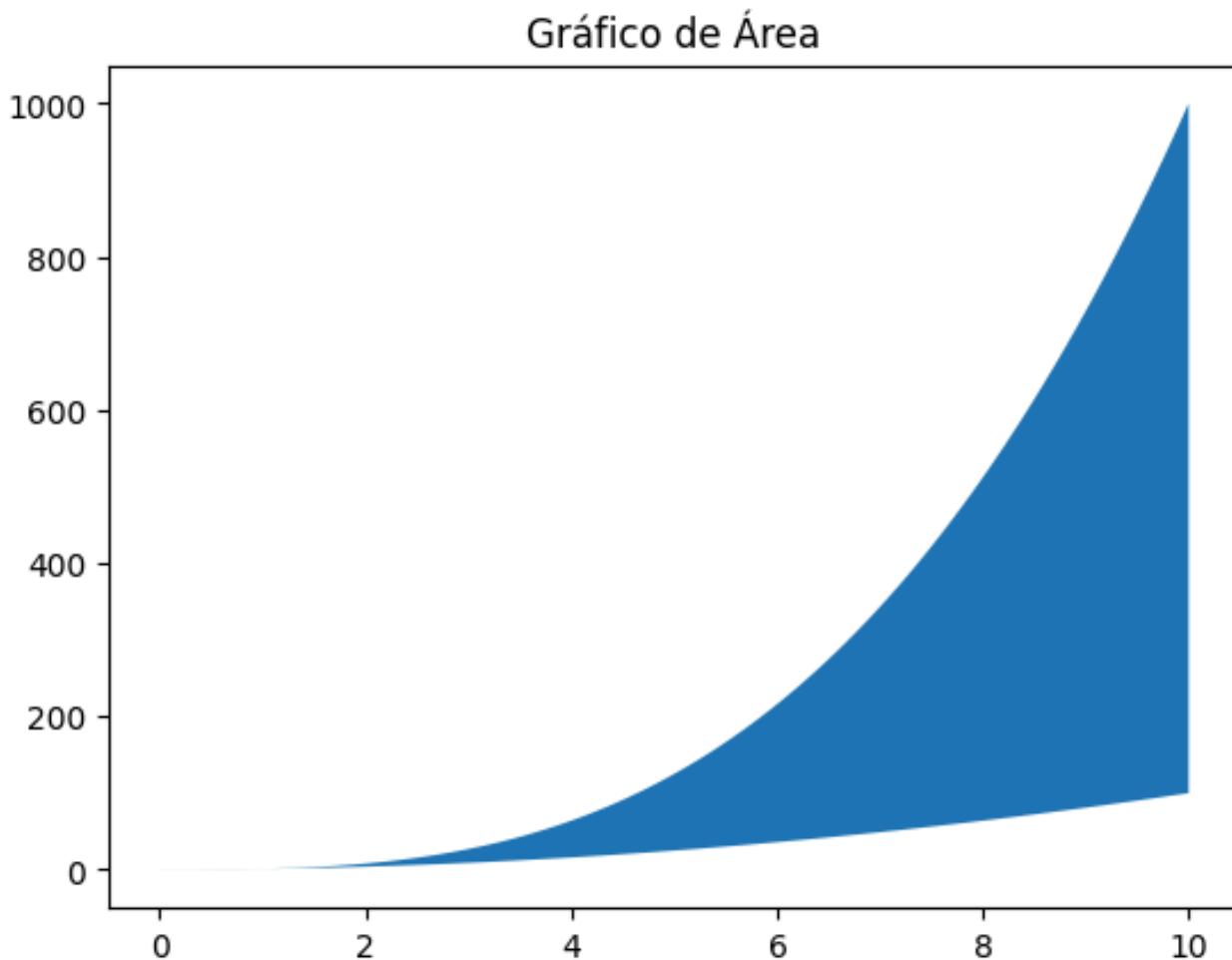
[saiba mais](#)

- Os valores x/y podem ser listas, arrays, séries ou DataFrames do Pandas.
- A cor da área pode ser uniforme ou variável. Ex: 'blue', ['blue', 'red', 'green'].
- A transparência da área varia de 0 (transparente) a 1 (opaca).
- Se não for especificado y2, a área é preenchida até o eixo x.
- Se for especificado y2, a área é preenchida entre y1 e y2.
- Preencher a área entre duas curvas é útil para **evidenciar a diferença** entre elas.

## Exemplo Gráfico de Área

```
import matplotlib.pyplot as plt
import numpy as np
x = np.linspace(0, 10, 100)
y1 = x**2
y2 = x**3

fig, ax = plt.subplots()
ax.fill_between(x, y1, y2, label='Área')
ax.set_title('Gráfico de Área')
plt.show()
```



# **Outros elementos**

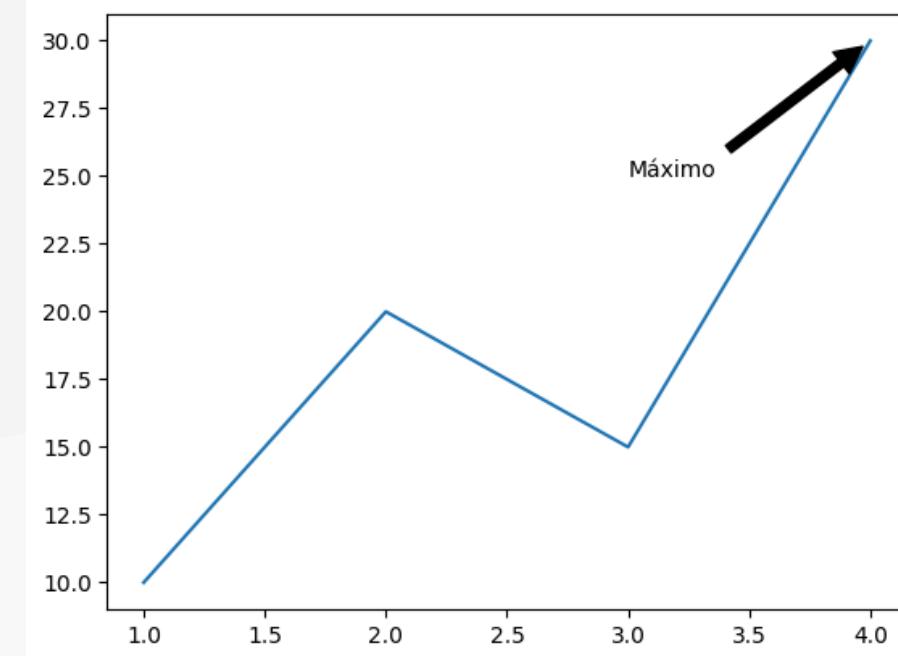
## Anotações

Texto e setas para destacar informações no gráfico.

```
import matplotlib.pyplot as plt

fig, ax = plt.subplots()
ax.plot([1, 2, 3, 4], [10, 20, 15, 30])
ax.annotate('Máximo', xy=(4, 30), xytext=(3, 25),
            arrowprops={'facecolor': 'black', 'shrink': 0.05})
plt.show()
```

[saiba mais](#)



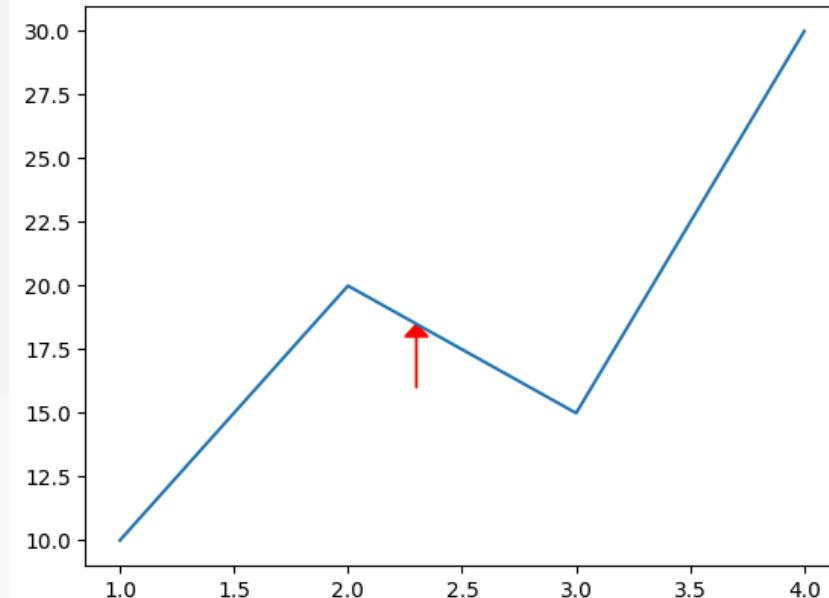
# Setas

```
import matplotlib.pyplot as plt

fig, ax = plt.subplots()
ax.plot([1, 2, 3, 4], [10, 20, 15, 30])
ax.arrow(2.3, 16, 0, 2,
         head_width=0.1, head_length=0.5,
         fc='red', ec='red')
plt.show()
```

```
ax.arrow(x, y, dx, dy, head_width, head_length)
```

- As coordenadas x/y são o início da seta.
- Os valores dx/dy são o comprimento da seta.
- `head_width` e `head_length` são a largura e o comprimento da cabeça da seta.
- `fc` e `ec` são as cores da face e da borda da seta.
- [saiba mais](#)



# Linhas de Referência

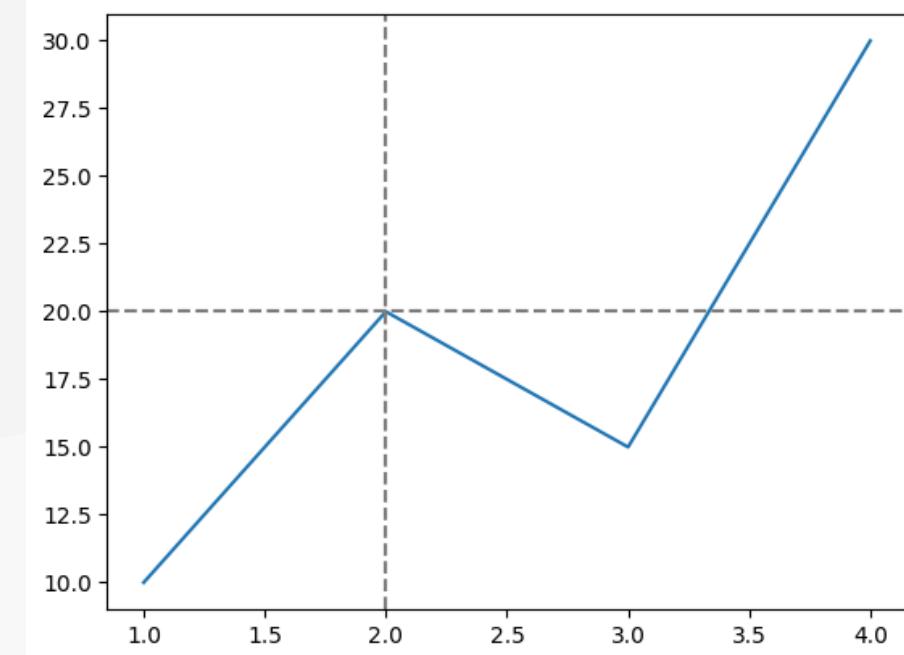
```
import matplotlib.pyplot as plt

fig, ax = plt.subplots()
ax.plot([1, 2, 3, 4], [10, 20, 15, 30])

ax.axhline(y=20, color='gray',
            linestyle='--', label='Referência h')

ax.axvline(x=2, color='gray',
            linestyle='--', label='Referência v')

plt.show()
```



[saiba mais](#)

# Imagens

```
from PIL import Image
import matplotlib.pyplot as plt

fig, ax = plt.subplots()
img = Image.open('email.jpg')
ax.imshow(img, extent=[0, 30, 0, 30],
          aspect='auto', alpha=0.5)
ax.plot([1, 2, 3, 4, 8, 15, 30],
        [10, 20, 15, 30, 10, 20, 15])
ax.set_xlim(0, 30)
ax.set_ylim(0, 30)
plt.show()
```

PIL é uma biblioteca para manipulação de imagens em Python.

[saiba mais imshow](#)

[saiba mais PIL](#)



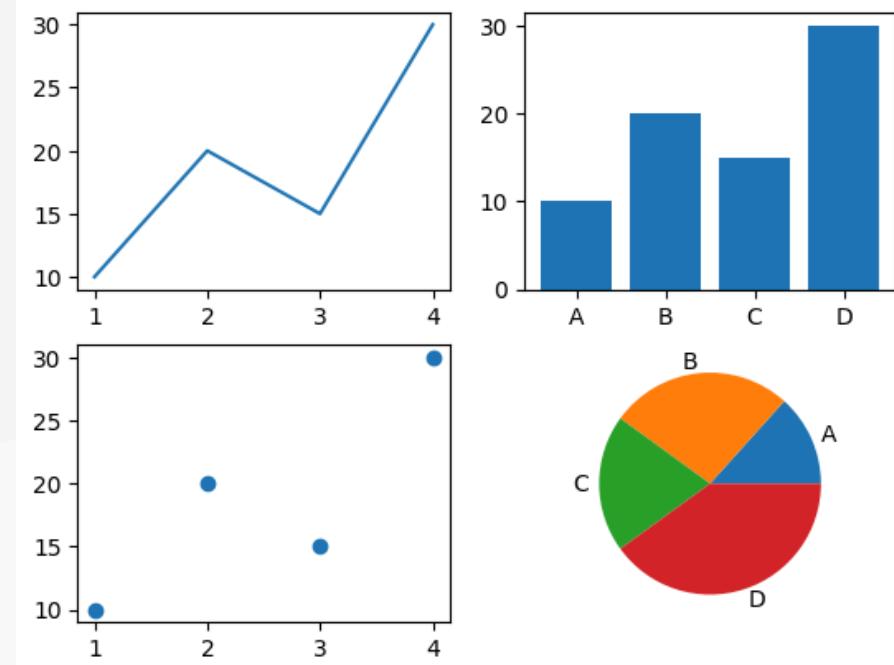
# Subgráficos

Múltiplos gráficos em uma única figura.

```
import matplotlib.pyplot as plt

fig, axs = plt.subplots(2, 2)
axs[0, 0].plot([1, 2, 3, 4], [10, 20, 15, 30])
axs[0, 1].bar(['A', 'B', 'C', 'D'], [10, 20, 15, 30])
axs[1, 0].scatter([1, 2, 3, 4], [10, 20, 15, 30])
axs[1, 1].pie([10, 20, 15, 30], labels=['A', 'B', 'C', 'D'])
plt.show()
```

`subplots(2,2)` cria uma matriz de 2x2 subgráficos na figura.

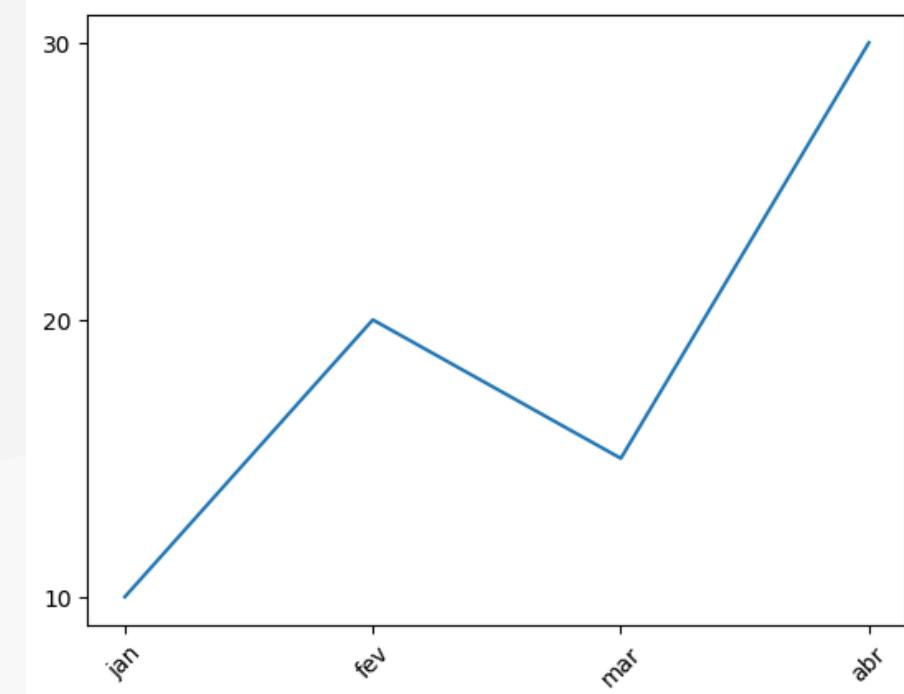


## Ticks

Ticks são as marcas nos eixos que indicam os valores.

```
import matplotlib.pyplot as plt

fig, ax = plt.subplots()
ax.plot([1, 2, 3, 4], [10, 20, 15, 30])
ax.set_xticks([1, 2, 3, 4],
    labels=['jan', 'fev', 'mar', 'abr'],
    rotation=45)
ax.set_yticks([10, 20, 30])
plt.show()
```



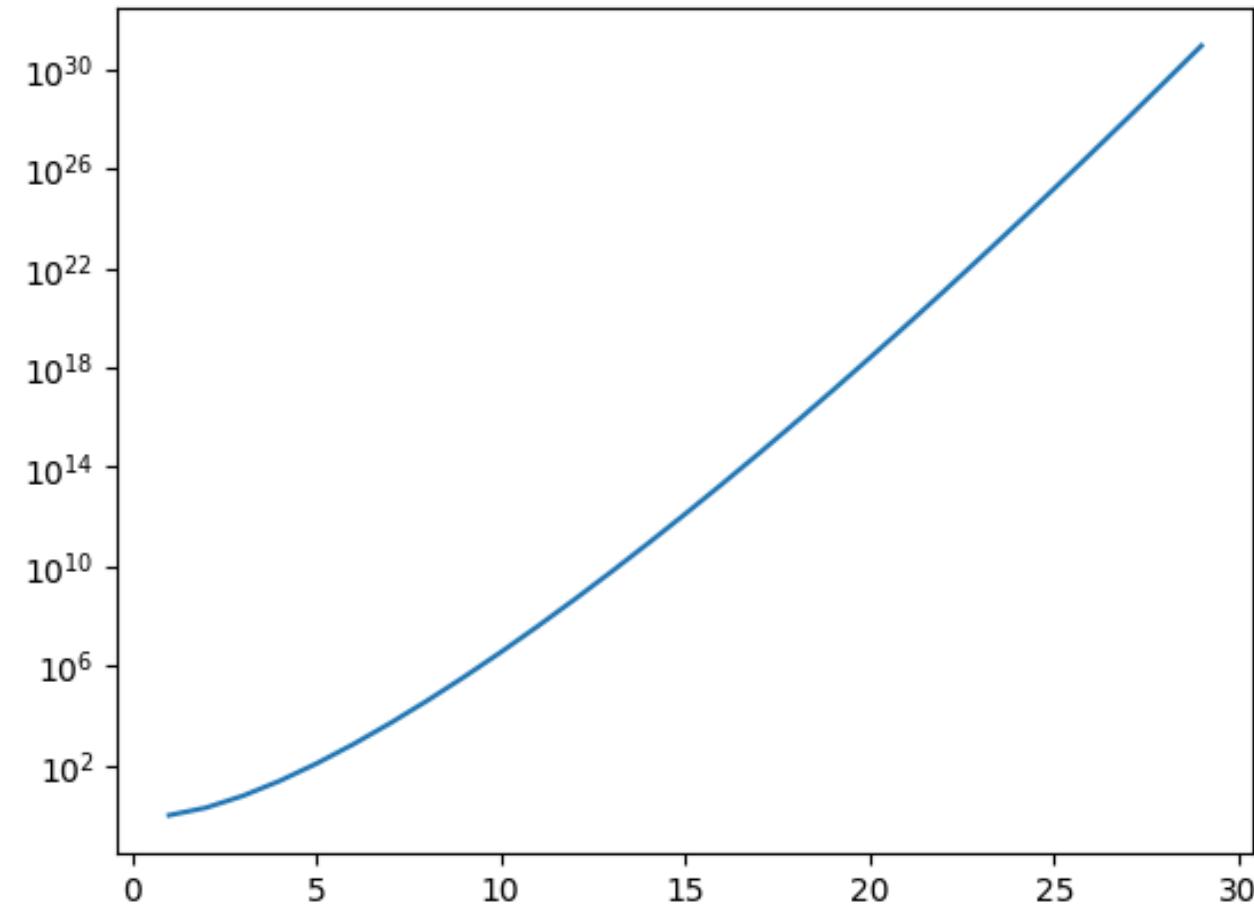
[saiba mais](#)

## Escala

```
import matplotlib.pyplot as plt
import numpy as np
from math import factorial
x = np.arange(1, 30, dtype=int)
y = [factorial(i) for i in x]

fig, ax = plt.subplots()
ax.plot(x,y)
ax.set_yscale('log')
plt.show()
```

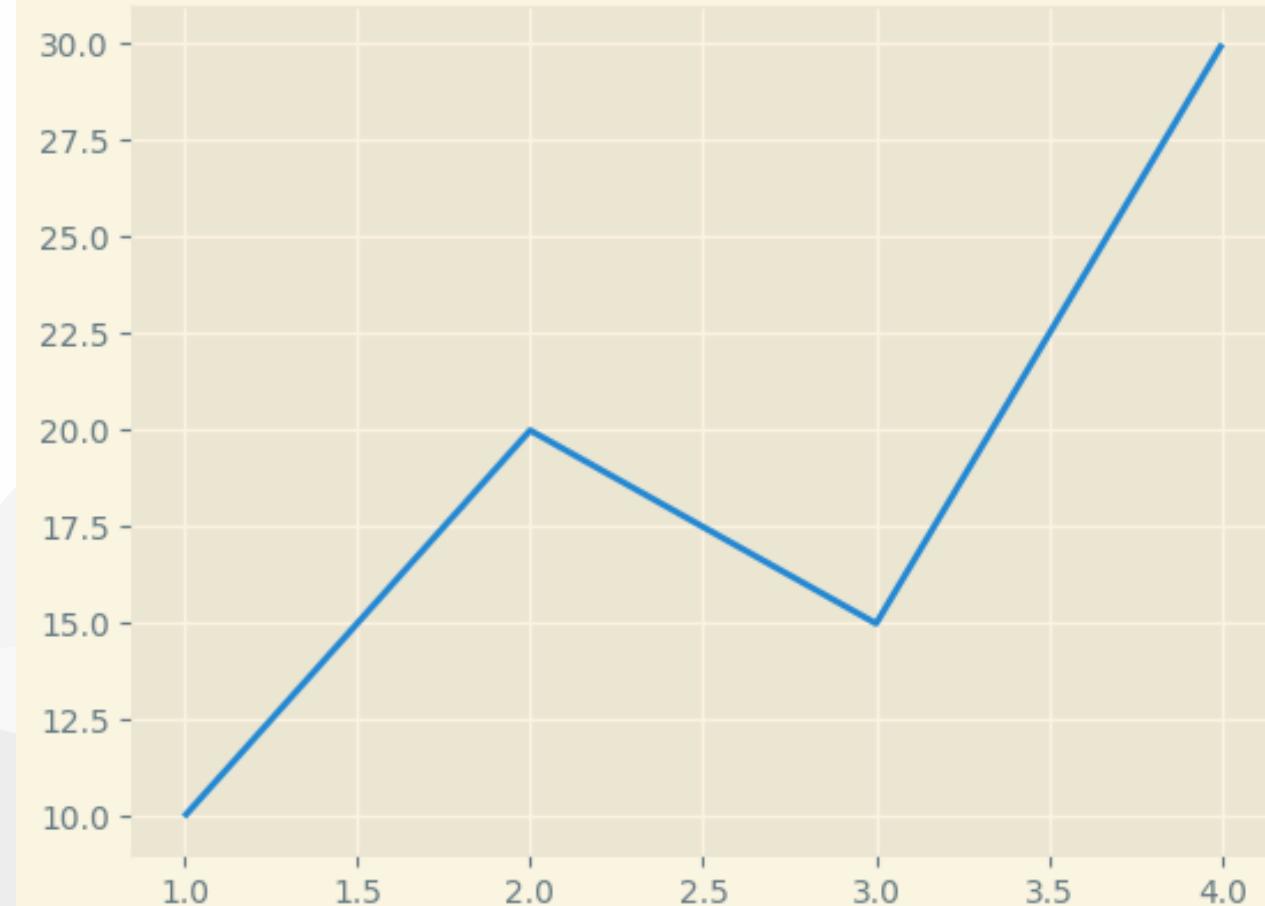
- As escalas disponíveis são 'linear', 'log', 'symlog', 'logit'.
- [saiba mais](#)



## Estilos

```
import matplotlib.pyplot as plt  
  
plt.style.use('Solarize_Light2')  
fig, ax = plt.subplots()  
ax.plot([1, 2, 3, 4], [10, 20, 15, 30])  
plt.show()
```

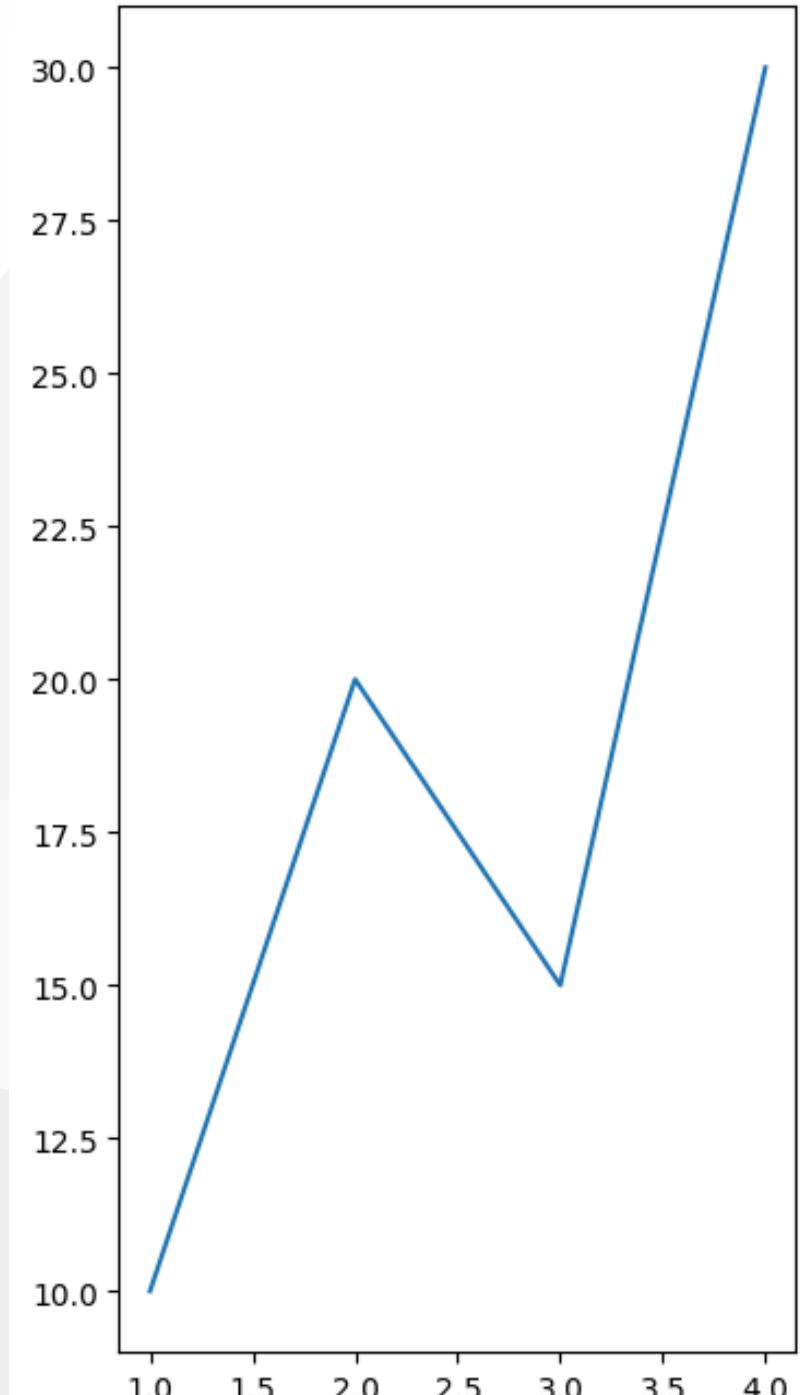
[saiba mais](#)



## Dimensões e Resolução

```
import matplotlib.pyplot as plt
fig, ax = plt.subplots(figsize=(4, 8), dpi=100)
ax.plot([1, 2, 3, 4], [10, 20, 15, 30])
plt.show()
```

- `figsize` define a largura e a altura da figura em polegadas.
- `dpi` define a resolução da figura em pontos por polegada.
- A resolução é importante para gráficos que serão impressos ou exibidos em alta definição.

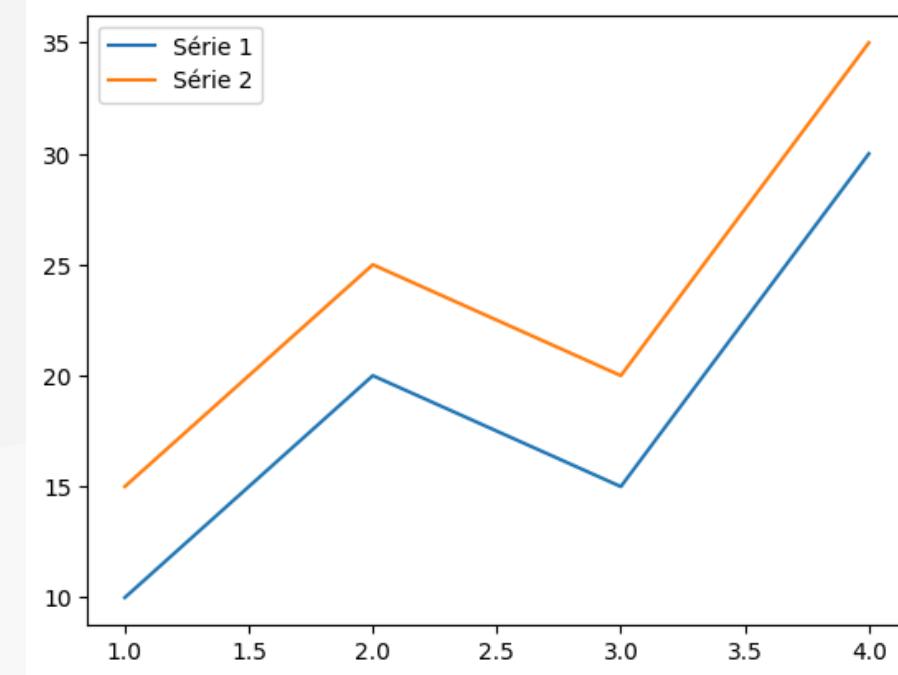


## Legenda

```
import matplotlib.pyplot as plt

fig, ax = plt.subplots()
ax.plot([1, 2, 3, 4], [10, 20, 15, 30],
        label='Série 1')
ax.plot([1, 2, 3, 4], [15, 25, 20, 35],
        label='Série 2')
ax.legend(loc='upper left')
plt.show()
```

- `loc='upper left'` define a posição da legenda no gráfico.
- `loc` também pode receber coordenadas (x, y).



# Mapa de Cores

Com o uso de mapas de cores, é possível representar uma terceira variável em um gráfico. Bem como melhorar o aspecto visual do gráfico.

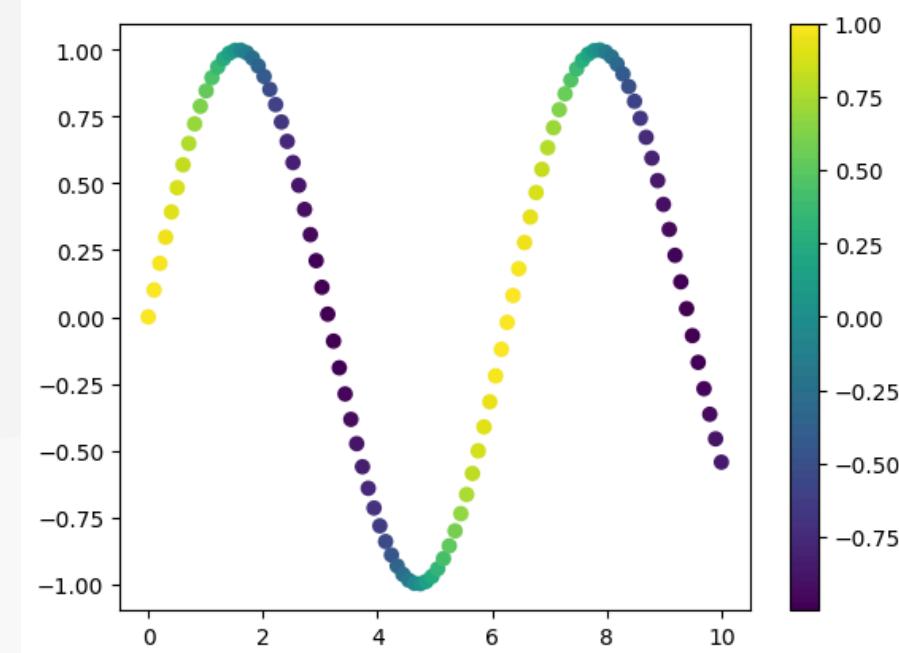
```
import matplotlib.pyplot as plt
import numpy as np

x = np.linspace(0, 10, 100)
y = np.sin(x)
z = np.cos(x)

fig, ax = plt.subplots()
sc = ax.scatter(x, y, c=z, cmap='viridis')

fig.colorbar(sc)

plt.show()
```



- Mapas de cores são gradientes ou sequências de cores que representam valores numéricos ou categorias.
- Para usar um mapa de cores, você precisa indicar a variável `c` e o `cmap`.
- `c` é a variável que será representada no mapa de cores. A escala de cores é definida automaticamente.
- `cmap` é o mapa de cores que será utilizado. Ex: 'viridis', 'plasma', 'inferno', 'magma'.

[saiba mais](#)

Alguns tipos de gráficos, como o de barras, não possuem um parâmetro `c` para a variável de cor. Nesses casos, é possível usar um mapa de cores para gerar a lista de cores.

```
import matplotlib.pyplot as plt
import numpy as np
x = np.arange(10)
y = np.random.randint(1, 10, 10)
z = 1 - y / y.max() # Normaliza os valores
cores = plt.cm.Wistia(z)
fig, ax = plt.subplots()
ax.bar(x, y, color=cores)
plt.show()
```

Obs.: os valores de `z` devem seguir a escala adequada para o mapa de cores. No exemplo, os valores foram normalizados entre 0 e 1 e invertidos para o que notas (`y`) mais altas tenham cores mais claras.

- Matplotlib também permite a elaboração de gráficos 3D, [saiba mais](#)
- Matplotlib também permite a elaboração de gráficos de mapas, [saiba mais](#)

