Gráficos com o Python

Trabalhando com o Matplotlib

Armando Soares Sousa
Departamento de Computação UFPI
Laboratório de Programação

Matplotlib

O Matplotlib é uma biblioteca de visualização de dados amplamente utilizado em Python.

Ele fornece uma ampla gama de funcionalidades para criar gráficos estáticos, gráficos interativos e até mesmo visualizações 3D.

Logo abaixo seguem alguns dos conceitos básicos e características mais importantes do Matplotlib:

- Figuras e Eixos
- Suporte a vários tipos de gráficos
- Personalização de gráficos
- Criar múltiplos gráficos via Subplots
- Suporte ao Numpy
- Exportação de gráficos para arquivos de imagem
- Interação com o Pandas

Matplotlib (Caracterísitcas)

Figuras e Eixos: O Matplotlib usa o conceito de figuras e eixos para criar gráficos. Uma figura é a janela ou página em branco na qual os gráficos são desenhados, enquanto os eixos são os sistemas coordenados onde os elementos do gráfico são plotados.

Tipos de Gráficos: O Matplotlib suporta uma variedade de tipos de gráficos, incluindo gráficos de linhas, gráficos de dispersão, gráficos de barras, gráficos de pizza, gráficos de caixa, gráficos de histograma, entre outros. Cada tipo de gráfico tem sua própria função correspondente no Matplotlib.

Personalização: O Matplotlib permite personalizar praticamente todos os aspectos dos gráficos. É possível alterar cores, estilos de linha, marcadores, títulos, rótulos dos eixos, legendas, entre outros elementos. A biblioteca oferece uma ampla gama de opções de personalização para atender às necessidades individuais.

Matplotlib (Caracterísitcas)

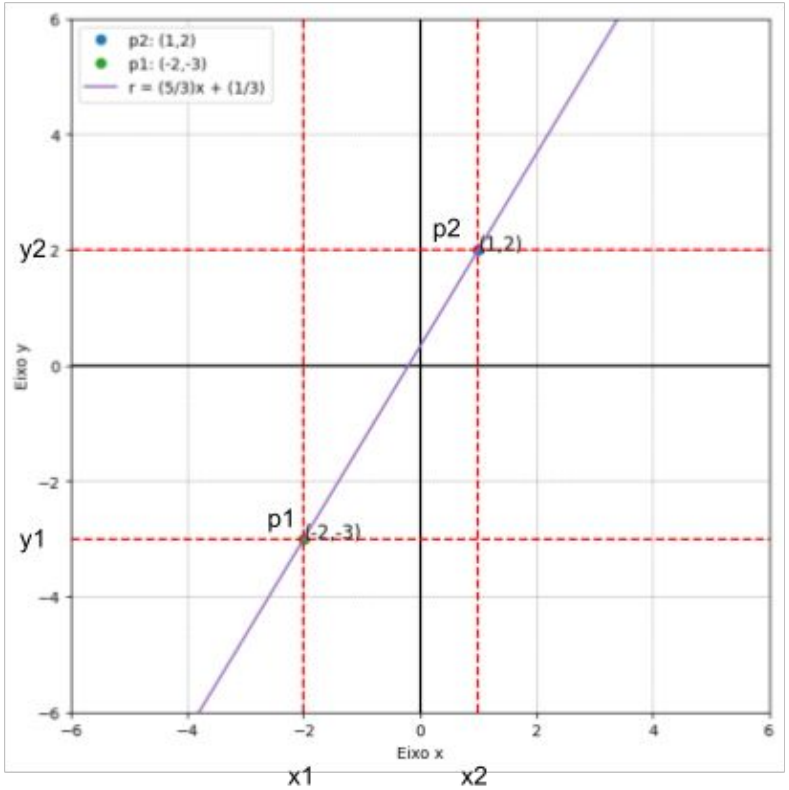
Subplots: O Matplotlib permite criar múltiplos gráficos em uma única figura usando subplots. Isso é útil para comparar diferentes conjuntos de dados ou visualizar diferentes aspectos dos dados em uma mesma figura.

Suporte a NumPy: O Matplotlib funciona bem com arrays NumPy, permitindo que os dados sejam facilmente plotados diretamente a partir de arrays, tornando-o uma escolha natural para visualização de dados científicos.

Exportação de Gráficos: Os gráficos criados com o Matplotlib podem ser exportados em vários formatos, como PNG, PDF, SVG e EPS. Isso permite que os gráficos sejam utilizados em relatórios, apresentações ou em outros ambientes de visualização.

Integração com Pandas: O Matplotlib se integra bem com a biblioteca Pandas, o que facilita a visualização de dados contidos em estruturas de dados do Pandas, como DataFrames e Series.

Sistema de Coordenadas do Plano Cartesiano



Eixos: O plano cartesiano é dividido em dois eixos perpendiculares, chamados de eixo x e eixo y. O eixo x é horizontal e o eixo y é vertical.

Origem: A origem é o ponto de intersecção dos eixos x e y.

Unidades: Os eixos são divididos em unidades iguais, que podem ser centímetros, metros, polegadas, etc.

Pontos: Um ponto no plano cartesiano é identificado por suas coordenadas, que são os valores em que o ponto intercepta os eixos x e y.

Ponto1: p1=(x1, y1) Ponto2: p2=(x2, y2)

Ponto1: p1=(-2, -3) Ponto2: p2=(1,2)

Equação da reta: y = (5/3)x + 1/3

Módulos mais importantes

Pyplot - é uma interface que plota gráficos.

O Pyplot é uma coleção de funções no estilo de comandos que fazem com que o matplotlib funcione de forma semelhante ao MATLAB. Cada função do pyplot realiza alguma alteração em uma figura: por exemplo, cria uma figura, cria uma área de plotagem em uma figura, plota algumas linhas em uma área de plotagem, decora o gráfico com rótulos.

Numpy - para trabalhar com manipulação de vetores, matrizes multidimensionais, álgebra linear e funções matemáticas de alto nível.

O NumPy é uma biblioteca para a linguagem de programação Python que permite o processamento de grandes arranjos e matrizes multidimensionais. Além disso, o NumPy oferece uma ampla coleção de funções matemáticas de alto nível que facilitam a realização de operações nessas estruturas de dados.

Pandas - biblioteca que permitir manipular e analisar dados.

O Pandas é utilizado para manipulação e análise de dados. Em particular, ele oferece estruturas de dados (DataFrames) e operações para manipulação de tabelas numéricas e séries temporais. É muito usado no ambiente Data Science.

Propriedades mais importantes

```python
# import as bibliotecas matplotlib e numpy
import matplotlib.pyplot as plt
import numpy as np
```

```python
# Criar um conjunto de dados de exemplo
# https://numpy.org/doc/stable/reference/generated/numpy.linspace.html
x = np.linspace(0, 10, 100)
print(f'x: {x} elementos')
print(f'Tamanho de x: {len(x)}')
```

```
x: [ 0.          0.1010101   0.2020202   0.3030303   0.4040404   0.50505051
  0.60606061  0.70707071  0.80808081  0.90909091  1.01010101  1.11111111
  1.21212121  1.31313131  1.41414141  1.51515152  1.61616162  1.71717172
  1.81818182  1.91919192  2.02020202  2.12121212  2.22222222  2.32323232
  2.42424242  2.52525253  2.62626263  2.72727273  2.82828283  2.92929293
  3.03030303  3.13131313  3.23232323  3.33333333  3.43434343  3.53535354
  3.63636364  3.73737374  3.83838384  3.93939394  4.04040404  4.14141414
  4.24242424  4.34343434  4.44444444  4.54545455  4.64646465  4.74747475
  4.84848485  4.94949495  5.05050505  5.15151515  5.25252525  5.35353535
  5.45454545  5.55555556  5.65656566  5.75757576  5.85858586  5.95959596
  6.06060606  6.16161616  6.26262626  6.36363636  6.46464646  6.56565657
  6.66666667  6.76767677  6.86868687  6.96969697  7.07070707  7.17171717
  7.27272727  7.37373737  7.47474747  7.57575758  7.67676768  7.77777778
  7.87878788  7.97979798  8.08080808  8.18181818  8.28282828  8.38383838
  8.48484848  8.58585859  8.68686869  8.78787879  8.88888889  8.98989899
  9.09090909  9.19191919  9.29292929  9.39393939  9.49494949  9.5959596
  9.6969697   9.7979798   9.8989899  10.        ] elementos
Tamanho de x: 100
```

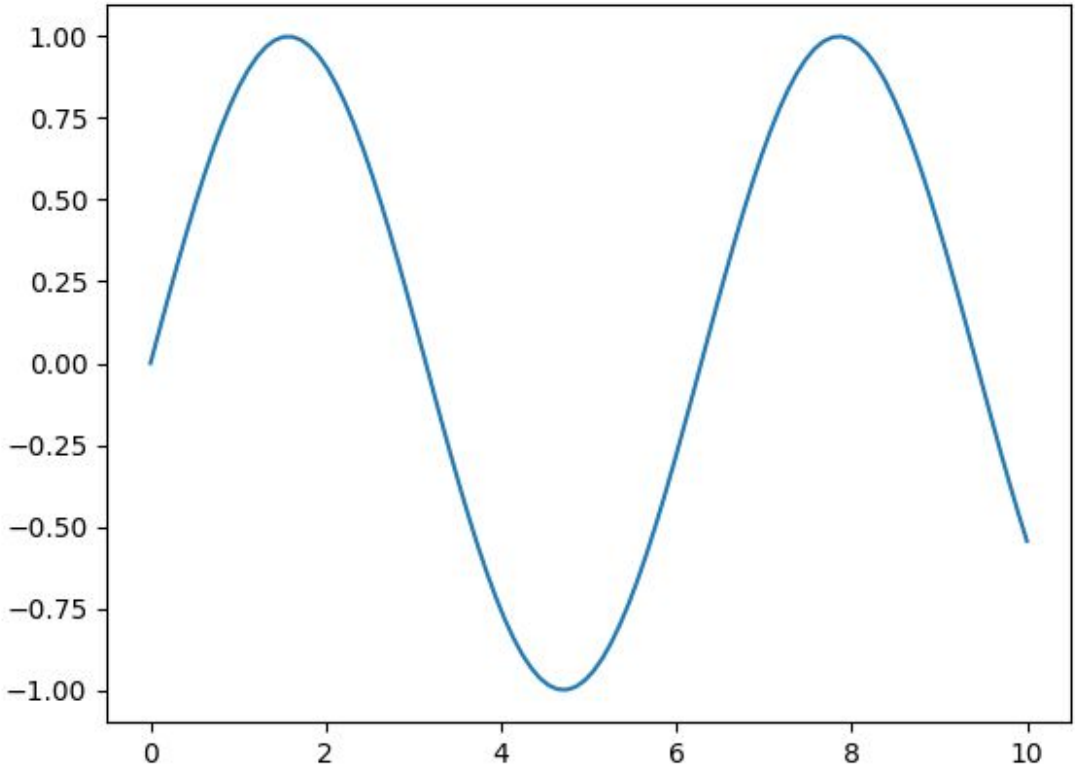
Propriedades mais importantes

```
1 # https://numpy.org/doc/stable/reference/generated/numpy.sin.html
2 y = np.sin(x)
3 print(f'y: {y} elementos')
4 print(f'Tamanho de y: {len(y)}')
```

```
y: [ 0.          0.10083842  0.20064886  0.2984138   0.39313661  0.48385164
  0.56963411  0.64960951  0.72296256  0.78894546  0.84688556  0.8961922
  0.93636273  0.96698762  0.98775469  0.99845223  0.99897117  0.98930624
  0.96955595  0.93992165  0.90070545  0.85230712  0.79522006  0.73002623
  0.65739025  0.57805259  0.49282204  0.40256749  0.30820902  0.21070855
  0.11106004  0.01027934 -0.09060615 -0.19056796 -0.28858706 -0.38366419
 -0.47483011 -0.56115544 -0.64176014 -0.7158225  -0.7825875  -0.84137452
 -0.89158426 -0.93270486 -0.96431712 -0.98609877 -0.99782778 -0.99938456
 -0.99075324 -0.97202182 -0.94338126 -0.90512352 -0.85763861 -0.80141062
 -0.73701276 -0.66510151 -0.58640998 -0.50174037 -0.41195583 -0.31797166
 -0.22074597 -0.12126992 -0.0205576   0.0803643   0.18046693  0.27872982
  0.37415123  0.46575841  0.55261747  0.63384295  0.7086068   0.77614685
  0.83577457  0.8868821   0.92894843  0.96154471  0.98433866  0.99709789
  0.99969234  0.99209556  0.97438499  0.94674118  0.90944594  0.86287948
  0.8075165   0.74392141  0.6727425   0.59470541  0.51060568  0.42130064
  0.32770071  0.23076008  0.13146699  0.03083368 -0.07011396 -0.17034683
 -0.26884313 -0.36459873 -0.45663749 -0.54402111] elementos
Tamanho de y: 100
```

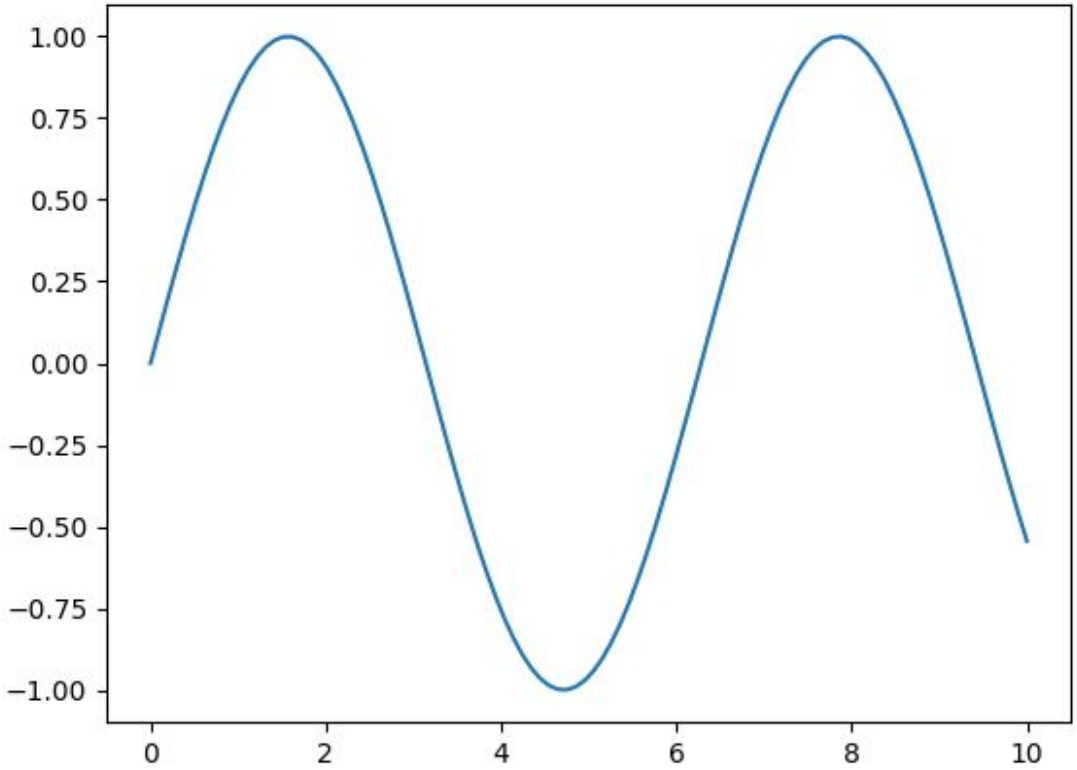
Propriedades mais importantes

```
1 # Plotar os dados
2 plt.plot(x, y)
3
4 # Mostrar o gráfico
5 plt.show()
```



Propriedades mais importantes

```
1 import matplotlib.pyplot as plt
2 import numpy as np
3
4 x = np.linspace(0, 10, 100)
5 y = np.sin(x)
6
7 plt.plot(x, y)
8
9 plt.show()
```

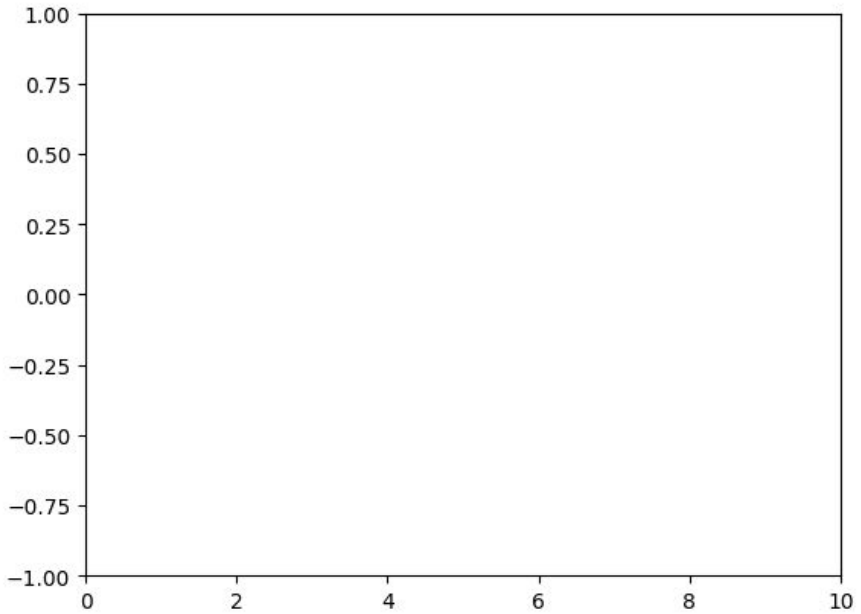


Eixo x e Eixo y

Os eixos x e y são essenciais em qualquer gráfico e indicam as dimensões de seus dados.

Você pode personalizar esses eixos para mostrar intervalos específicos ou números de etiquetas.

```
# Personalizar os eixos
plt.xlim([0, 10])
plt.ylim([-1, 1])

# Mostrar o gráfico
plt.show()
```



A função xlim() define os limites do eixo x e a função ylim() define os limites do eixo y.

No exemplo acima, definimos o intervalo do eixo x para [0, 10] e o intervalo do eixo y para [-1, 1].

Isso significa que o eixo x vai de 0 a 10 e o eixo y de -1 a 1.

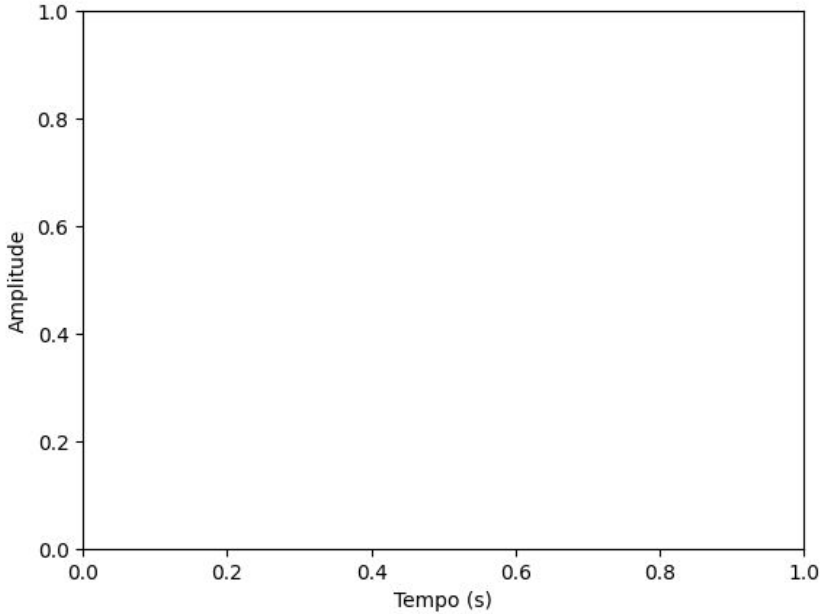
Rotulando os eixos x e y

Os rótulos dos eixos x e y informam ao leitor o que está sendo medido em cada eixo.

Você pode personalizar esses rótulos usando as funções xlabel() e ylabel().

```
1 # Personalizar os rótulos dos eixos
2 plt.xlabel('Tempo (s)')
3 plt.ylabel('Amplitude')
4
5 # Mostrar o gráfico
6 plt.show()
```

No exemplo acima, adicionamos rótulos aos eixos x e y. O rótulo do eixo x é "Tempo (s)" e o rótulo do eixo y é "Amplitude".

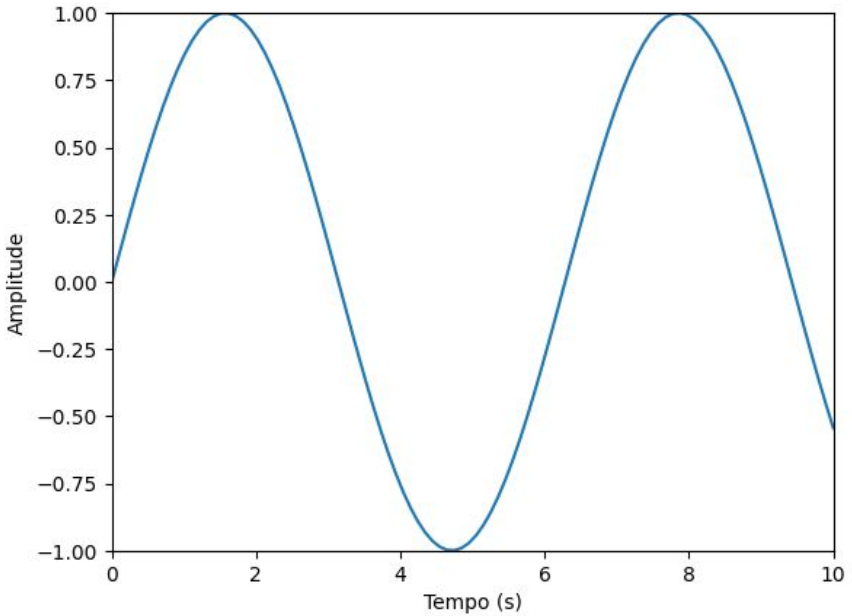


Rotulando os eixos x e y

Os rótulos dos eixos x e y informam ao leitor o que está sendo medido em cada eixo.

Você pode personalizar esses rótulos usando as funções xlabel() e ylabel().

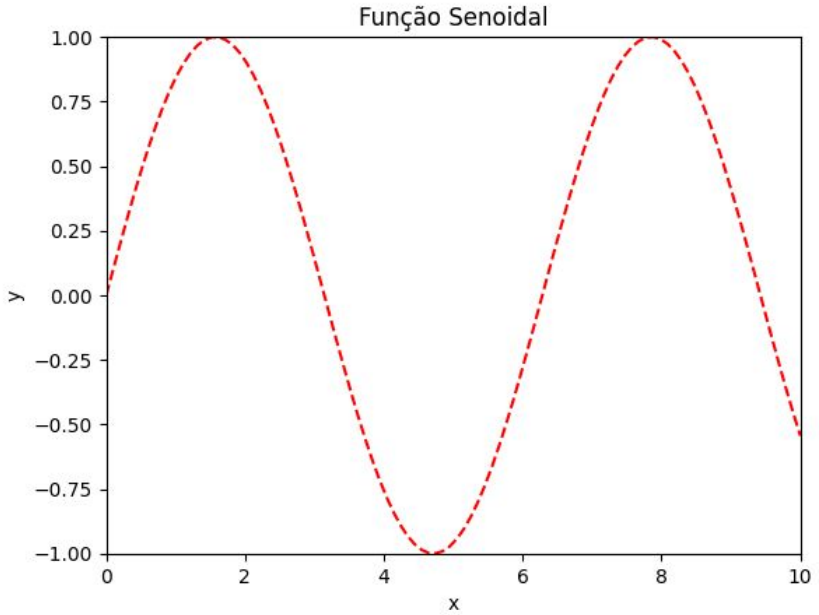
```
x = np.linspace(0, 10, 100)
y = np.sin(x)

# Plotar os dados
plt.plot(x, y)
plt.xlim([0, 10])
plt.ylim([-1, 1])

# Personalizar os rótulos dos eixos
plt.xlabel('Tempo (s)')
plt.ylabel('Amplitude')

# Mostrar o gráfico
plt.show()
```



A função plot

A função plot(..) é usada para plotar os dados em um gráfico. Você pode personalizar o estilo do gráfico adicionando argumentos à função plot(..), como a cor e o estilo da linha.

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

# Personalizar o estilo do gráfico
plt.plot(x, y, color='red', linestyle='dashed')

plt.xlim([0, 10])
plt.ylim([-1, 1])

# Personalizar o título do gráfico
plt.title('Função Senoidal')
# Personalizar os rótulos dos eixos
plt.xlabel('x')
plt.ylabel('y')
# Mostrar o gráfico
plt.show()
```



O numpy fornece várias funções matemáticas

Lista das principais funções matemáticas oferecidas pelo NumPy:

- Funções trigonométricas
- Funções hiperbólicas
- Funções exponenciais
- Funções logarítmicas
- Funções máximo e mínimo
- Funções de estatísticas

Essas funções podem ser usadas para realizar uma ampla variedade de operações matemáticas, incluindo cálculos trigonométricos, exponenciais, logarítmicos, arredondamentos, máximos e mínimos, estatísticas e outras.

Lembrando que tais funções trabalham com vetores do tipo numpy array. Com isso, caso você tenha uma lista de valores é preciso convertê-la para np.array.

Usando uma função matemática do NumPY

```
1 import numpy as np
2
3 x = np.array([0, 10, 20, 30, 40, 45, 50, 60, 70, 90, 100, 110, 120, 130, 140, 150, 160, 170, 180])
4
5 print(f'Valores de x: {x}')
6
7 y = np.sin(x)
8
9 print(f'Valores de y: {y}')
```

```
Valores de x: [  0  10  20  30  40  45  50  60  70  90 100 110 120 130 140 150 160 170
 180]
Valores de y: [ 0.          -0.54402111  0.91294525 -0.98803162  0.74511316  0.85090352
 -0.26237485 -0.30481062  0.77389068  0.89399666 -0.50636564 -0.04424268
  0.58061118 -0.93010595  0.98023966 -0.71487643  0.21942526  0.34664946
 -0.80115264]
```

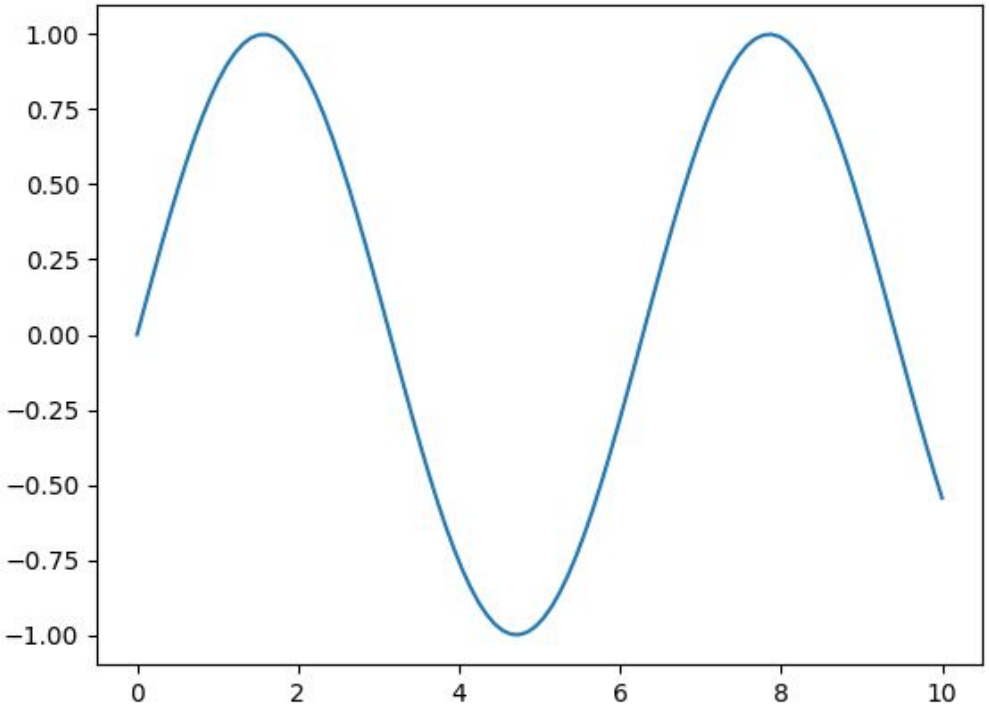
Lista de funções matemáticas suportadas pelo Numpy:
https://numpy.org/doc/stable/reference/routines.math.html

Criando uma classe Grafico

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

class Grafico:
    def __init__(self):
        self.x = None
        self.y = None

    def criar_eixo_x(self, inicio, fim, passo ):
        self.x = np.linspace(inicio, fim, passo)

    def calcular_y(self, funcao):
        self.y = funcao(self.x)

    def configura_limites_dos_eixos(self, limite_inferior_x=None, limite_superior_x=None, limite_inferior_y=None, limite_superior_y=None):
        if ((limite_inferior_x is not None) and (limite_superior_x is not None) and (limite_inferior_y is not None) and (limite_superior_y is not None)
            plt.xlim([limite_inferior_x, limite_superior_x])
            plt.ylim([limite_inferior_y, limite_superior_y])

    def configura_titulo_e_rotulos(self, titulo=None, rotulo_x=None, rotulo_y=None):
        if titulo is not None:
            plt.title(titulo)
        if rotulo_x is not None:
            plt.xlabel(rotulo_x)
        if rotulo_y is not None:
            plt.ylabel(rotulo_y)

    def exibir_gráfico(self):
        plt.plot(self.x, self.y)
        plt.show()
```

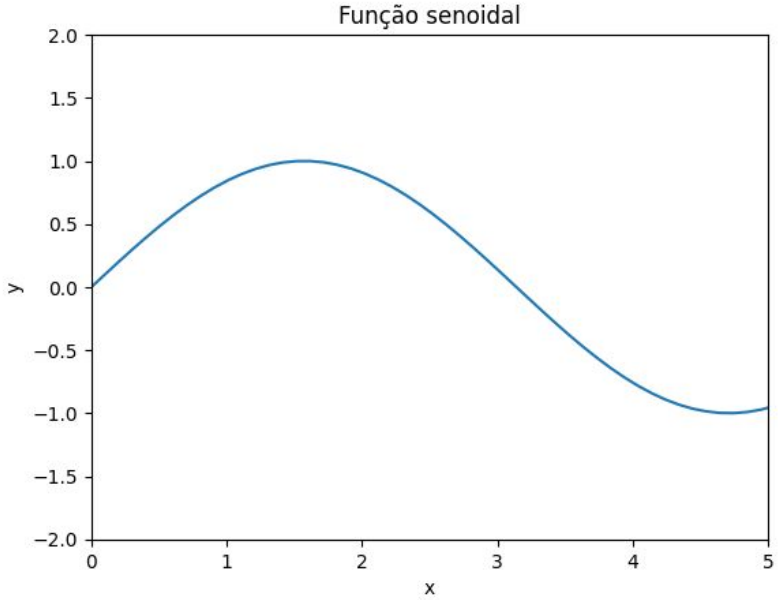
Exemplo de instanciação de um objeto da classe Grafico

```
1 # Exemplo de uso da classe Gráfico
2 grafico_seno = Grafico()
3 grafico_seno.criar_eixo_x(0, 10, 100)
4 # Exemplo de uso da função seno do numpy
5 grafico_seno.calcular_y(np.sin)
6 grafico_seno.exibir_gráfico()
```



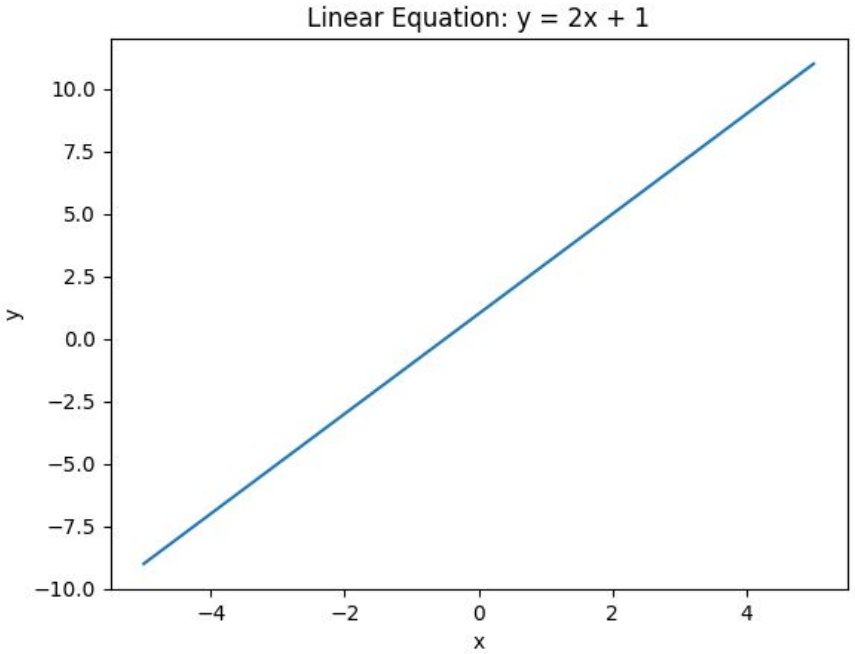
Exemplo de instanciação de um objeto da classe Grafico

```
grafico_seno = Grafico()
grafico_seno.criar_eixo_x(0, 10, 100)
grafico_seno.calcular_y(np.sin)  # Exemplo de uso da função seno do numpy
grafico_seno.configura_limites_dos_eixos(limite_inferior_x=0, limite_superior_x=5, limite_inferior_y=-2, limite_superior_y=2)
grafico_seno.configura_titulo_e_rotulos(titulo="Função senoidal", rotulo_x="x", rotulo_y="y")
grafico_seno.exibir_gráfico()
```



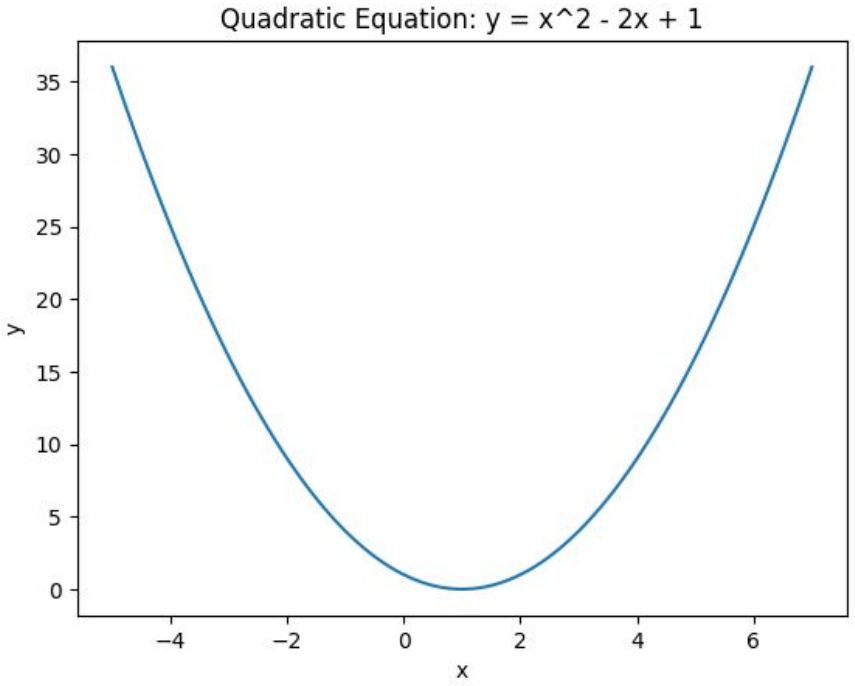
Gráficos 2d mais comuns (Gráfico de uma reta)

```python
# Define the equation of the line
def f(x):
    return 2*x + 1

# Create an array of x values
x = np.linspace(-5, 5, 100)

# Evaluate the equation of the line for each x value
y = f(x)

# Plot the line
plt.plot(x, y)

# Add labels and title
plt.xlabel('x')
plt.ylabel('y')
plt.title('Linear Equation: y = 2x + 1')

# Display the plot
plt.show()
```



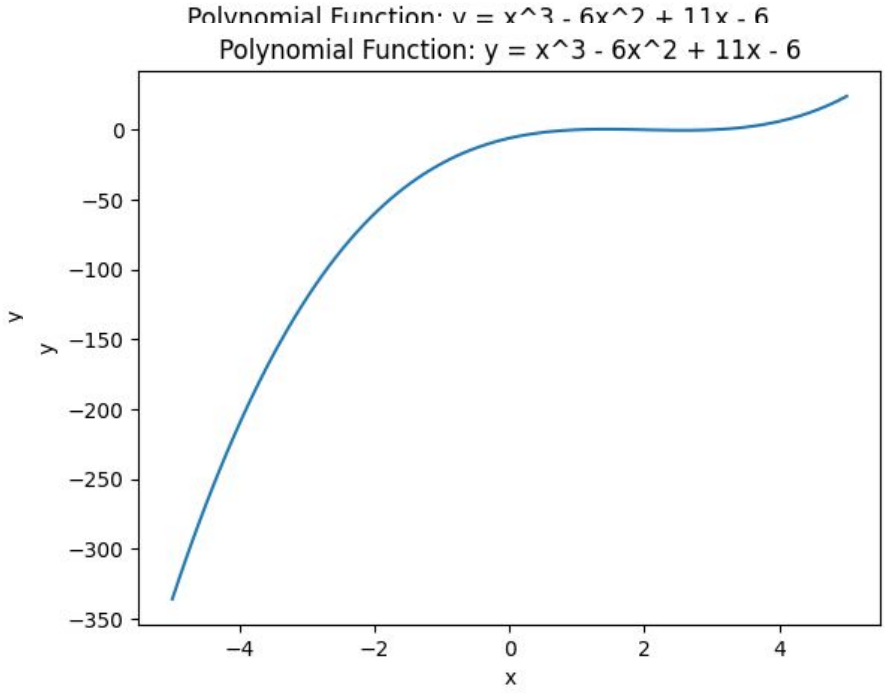
Gráficos 2d mais comuns (Gráfico de uma curva - função do 2o. grau)

```python
# Define the equation of the curve
def f(x):
    return x**2 - 2*x + 1

# Create an array of x values
x = np.linspace(-5, 7, 100)

# Evaluate the equation of the curve for each x value
y = f(x)

# Plot the curve
plt.plot(x, y)

# Add labels and title
plt.xlabel('x')
plt.ylabel('y')
plt.title('Quadratic Equation: y = x^2 - 2x + 1')

# Display the plot
plt.show()
```



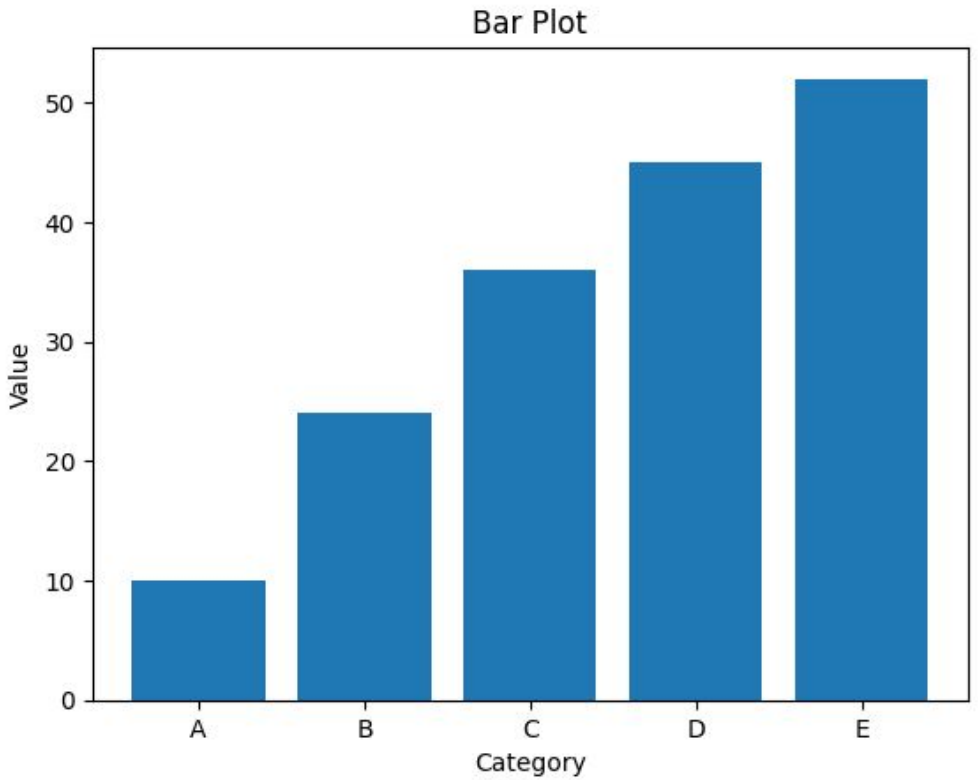
Gráficos 2d mais comuns (Gráfico de uma curva - função do 3o. grau)

```python
# Define the equation of the function
def f(x):
    return x**3 - 6*x**2 + 11*x - 6

# Create an array of x values
x = np.linspace(-5, 5, 100)

# Evaluate the equation of the function for each x value
y = f(x)

# Plot the function
plt.plot(x, y)

# Add labels and title
plt.xlabel('x')
plt.ylabel('y')
plt.title('Polynomial Function: y = x^3 - 6x^2 + 11x - 6')

# Display the plot
plt.show()
```



Gráficos 2d mais comuns (Gráfico de barras)

```
# Create some data
# categorias (A, B, C, D e E)
x = ['A', 'B', 'C', 'D', 'E']

# valores de cada categoria
y = [10, 24, 36, 45, 52]

# Create a bar plot
plt.bar(x, y)

# Add labels and title
plt.xlabel('Category')
plt.ylabel('Value')
plt.title('Bar Plot')

# Display the plot
plt.show()
```

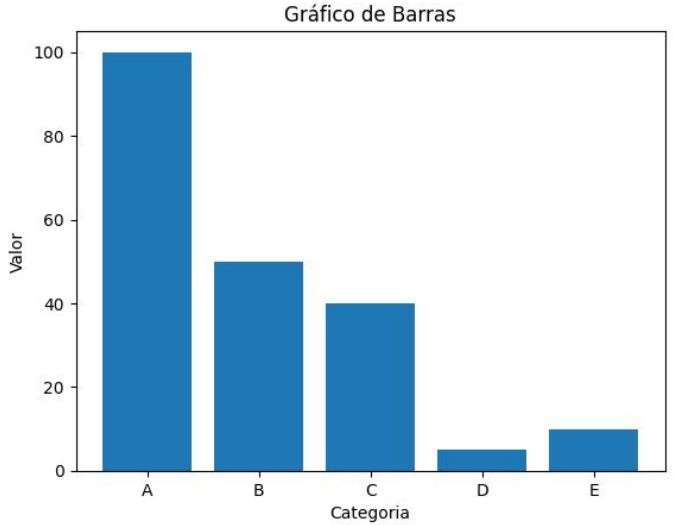


Criando uma classe GraficoBarras

```python
import matplotlib.pyplot as plt

class GraficoBarras:
    def __init__(self):
        self.x = None
        self.y = None

    def criar_grafico(self, x, y):
        self.x = x
        self.y = y

    def configura_titulo_e_rotulos(self, titulo=None, rotulo_x=None, rotulo_y=None):
        if titulo is not None:
            plt.title(titulo)
        if rotulo_x is not None:
            plt.xlabel(rotulo_x)
        if rotulo_y is not None:
            plt.ylabel(rotulo_y)

    def exibir_grafico(self):
        plt.bar(self.x, self.y)
        plt.show()
```

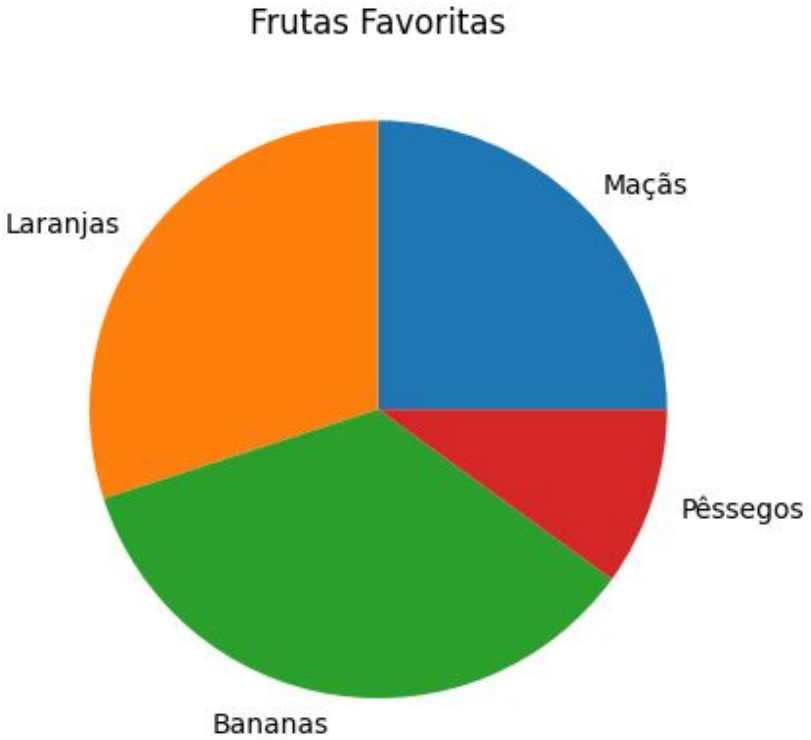
Instanciando uma classe do tipo GraficoBarras

```python
# Exemplo de uso da classe GraficoBarras
grafico = GraficoBarras()
categorias = ['A', 'B', 'C', 'D', 'E']
valores = [100, 50, 40, 5, 10]
grafico.criar_grafico(categorias, valores)
grafico.configura_titulo_e_rotulos(titulo='Gráfico de Barras', rotulo_x='Categoria', rotulo_y='Valor')
grafico.exibir_grafico()
```



Gráficos 2d mais comuns (Gráfico de Pizza - Pie chart)

```
# Dados de exemplo
labels = ['Maçãs', 'Laranjas', 'Bananas', 'Pêssegos']
valores = [25, 30, 35, 10]

# Criar o gráfico de pizza
plt.pie(valores, labels=labels)

# Adicionar um título
plt.title('Frutas Favoritas')

# Mostrar o gráfico
plt.show()
```

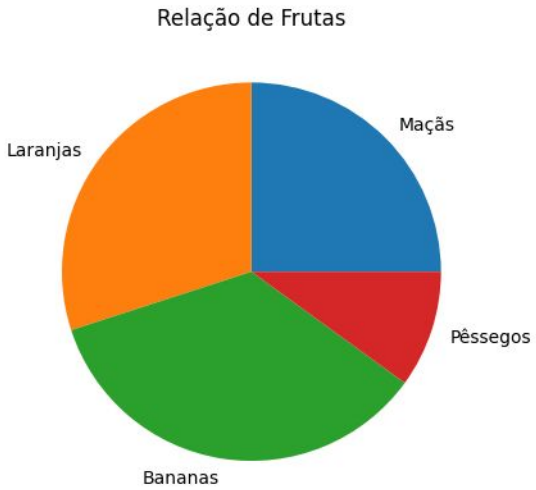


Criando uma classe GraficoPizza

```python
import matplotlib.pyplot as plt

class GraficoPizza:
    def __init__(self):
        self.labels = None
        self.valores = None

    def criar_grafico(self, labels, valores):
        self.labels = labels
        self.valores = valores

    def configura_titulo(self, titulo=None):
      if titulo is not None:
        plt.title(titulo)

    def exibir_grafico(self):
        plt.pie(self.valores, labels=self.labels)
        plt.show()
```

Instanciando uma classe do tipo GraficoPizza

```
1 # Exemplo de uso da classe GraficoPizza
2 grafico_pizza = GraficoPizza()
3 frutas = ['Maçãs', 'Laranjas', 'Bananas', 'Pêssegos']
4 quantidades = [25, 30, 35, 10]
5 grafico_pizza.criar_grafico(frutas, quantidades)
6 grafico_pizza.configura_titulo(titulo='Relação de Frutas')
7 grafico_pizza.exibir_grafico()
```



Gráficos de dispersão (Scatter Plot)

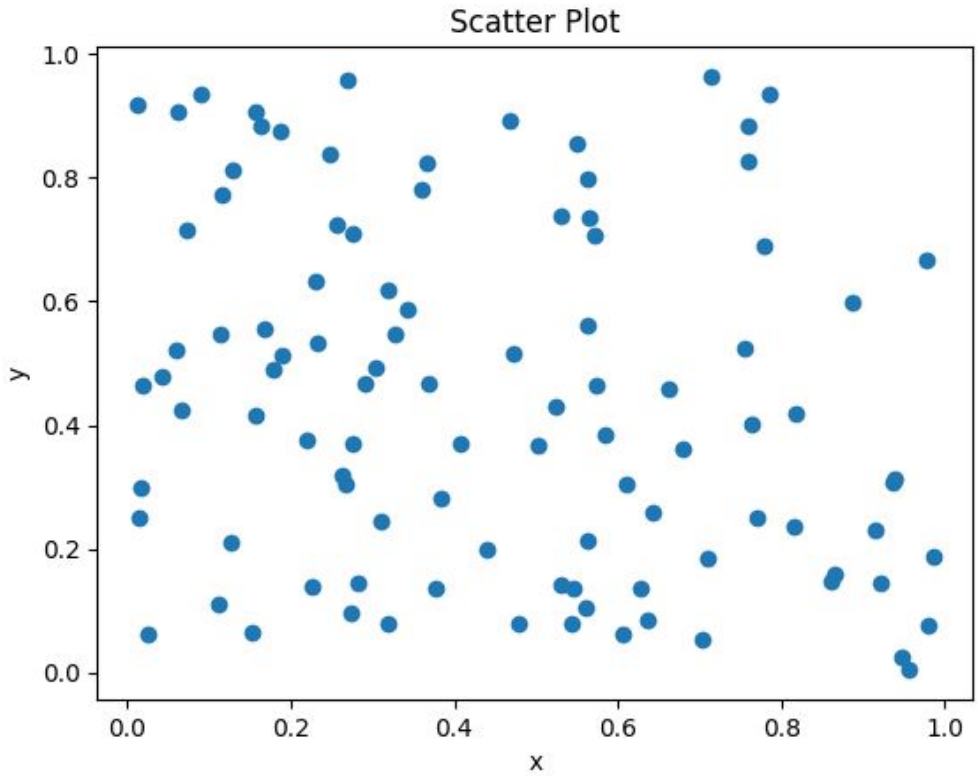
O gráfico de dispersão é uma ferramenta gráfica amplamente utilizada em estatística e outras áreas do conhecimento para visualizar a relação entre duas variáveis quantitativas. Ao plotar os pontos no gráfico, é possível detectar padrões e tendências nos dados e identificar possíveis correlações entre as variáveis.

https://en.wikipedia.org/wiki/Scatter_plot

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

# Create some data
x = np.random.rand(100)
y = np.random.rand(100)

# Create a scatter plot
plt.scatter(x, y)

# Add labels and title
plt.xlabel('x')
plt.ylabel('y')
plt.title('Scatter Plot')

# Display the plot
plt.show()
```

Gráficos de dispersão (Scatter Plot)

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

# Create some data
x = np.random.rand(100)
y = np.random.rand(100)

# Create a scatter plot
plt.scatter(x, y)

# Add labels and title
plt.xlabel('x')
plt.ylabel('y')
plt.title('Scatter Plot')

# Display the plot
plt.show()
```



Gráficos de Histograma

Um histograma é um tipo de gráfico que representa a distribuição de frequências de dados. Ele é composto por barras verticais, cada uma representando a frequência de um intervalo de dados.

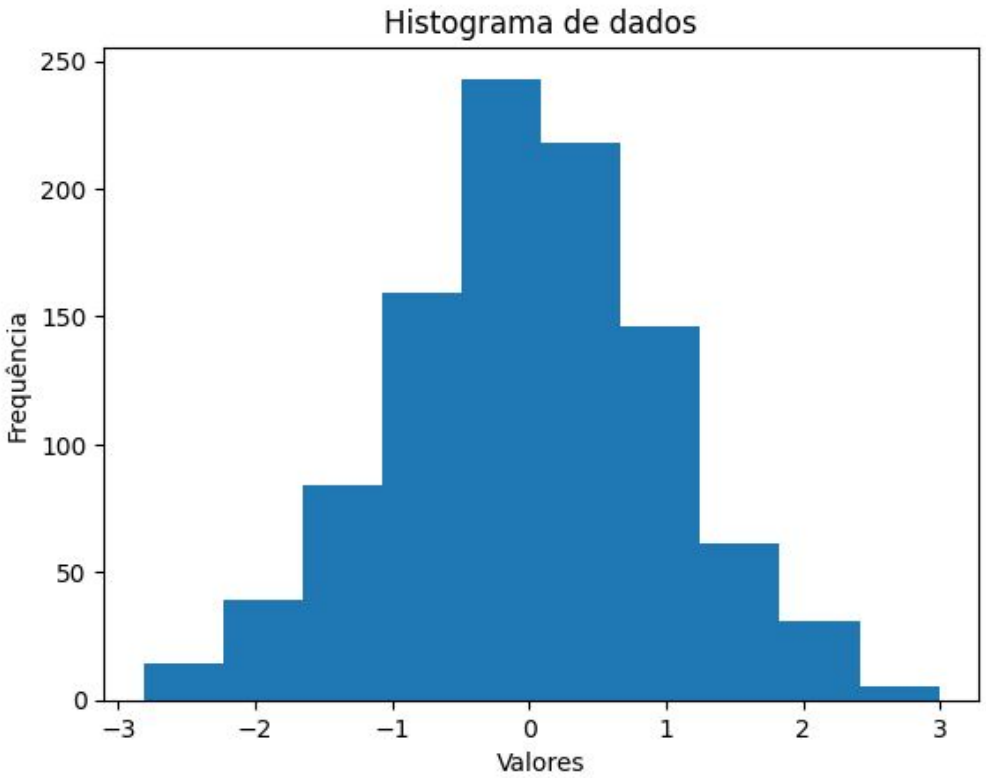
O eixo horizontal do histograma representa os valores dos dados, e o eixo vertical representa a frequência dos dados. A altura de cada barra representa a frequência com que os dados ocorrem dentro do intervalo representado pela barra.

https://en.wikipedia.org/wiki/Histogram

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

# Gerar dados de exemplo
dados = np.random.randn(1000)

# Criar o histograma
plt.hist(dados, bins=10)

# Adicionar rótulos e título
plt.xlabel('Valores')
plt.ylabel('Frequência')
plt.title('Histograma de dados')

# Mostrar o histograma
plt.show()
```

Gráficos de Histograma

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

# Gerar dados de exemplo
dados = np.random.randn(1000)

# Criar o histograma
plt.hist(dados, bins=10)

# Adicionar rótulos e título
plt.xlabel('Valores')
plt.ylabel('Frequência')
plt.title('Histograma de dados')

# Mostrar o histograma
plt.show()
```



Gráficos de Boxsplot

Um gráfico de boxplot, também conhecido como diagrama de caixa, é um tipo de gráfico que representa a distribuição de dados. Ele é composto por uma caixa, dois traços horizontais e dois pontos.

A caixa representa o intervalo interquartílico, que é a diferença entre o primeiro e o terceiro quartil. Os dois traços horizontais representam a mediana e a média dos dados. Os dois pontos representam os valores máximos e mínimos dos dados.

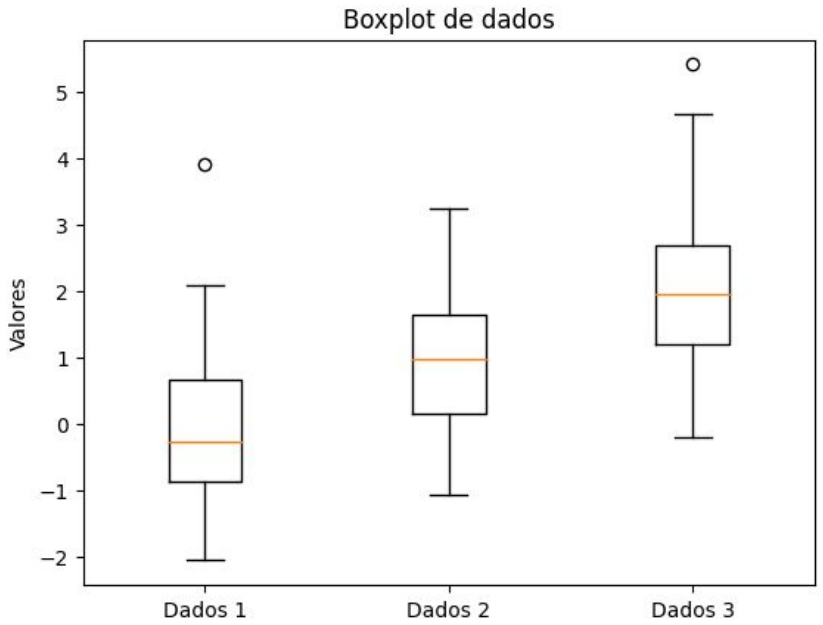
Os gráficos de boxplot são usados para visualizar a distribuição de dados, e podem ser usados para comparar a distribuição de dados de diferentes conjuntos de dados.

https://en.wikipedia.org/wiki/Box_plot

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

# Gerar dados de exemplo
dados1 = np.random.normal(0, 1, 100)
dados2 = np.random.normal(1, 1, 100)
dados3 = np.random.normal(2, 1, 100)

# Colocar dados em uma lista
dados = [dados1, dados2, dados3]

# Criar o boxplot
plt.boxplot(dados)

# Adicionar rótulos e título
plt.xticks([1, 2, 3], ['Dados 1', 'Dados 2', 'Dados 3'])
plt.ylabel('Valores')
plt.title('Boxplot de dados')

# Mostrar o boxplot
plt.show()
```

Gráficos de Boxsplot

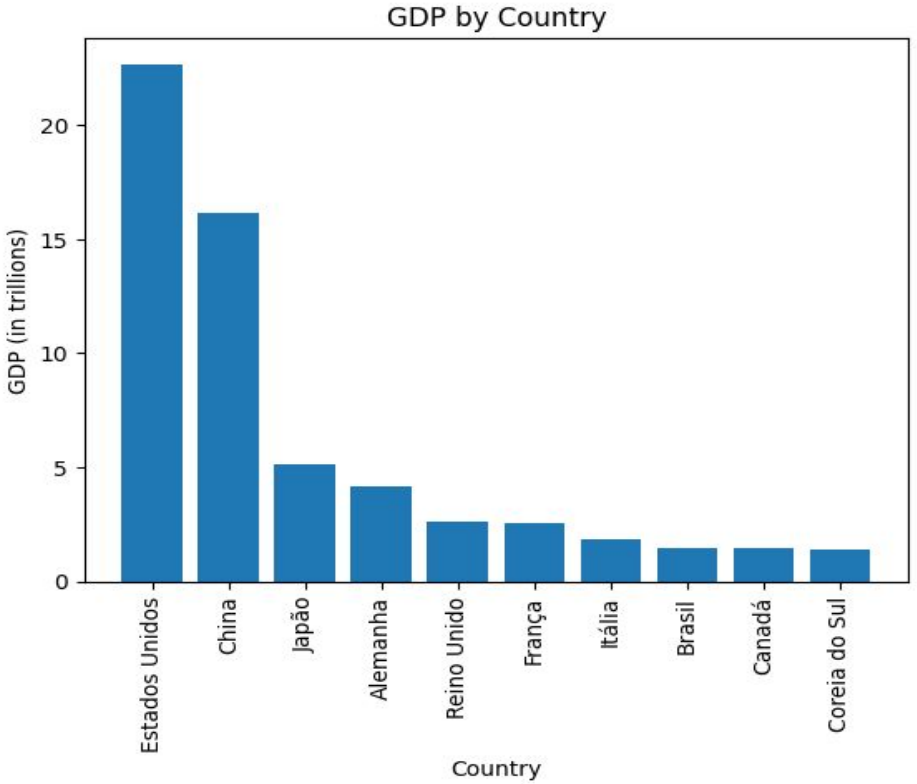
```python
import matplotlib.pyplot as plt
import numpy as np

# Gerar dados de exemplo
dados1 = np.random.normal(0, 1, 100)
dados2 = np.random.normal(1, 1, 100)
dados3 = np.random.normal(2, 1, 100)

# Colocar dados em uma lista
dados = [dados1, dados2, dados3]

# Criar o boxplot
plt.boxplot(dados)

# Adicionar rótulos e título
plt.xticks([1, 2, 3], ['Dados 1', 'Dados 2', 'Dados 3'])
plt.ylabel('Valores')
plt.title('Boxplot de dados')

# Mostrar o boxplot
plt.show()
```



Usando dados de arquivos .csv

```
1 import pandas as pd
2
3 gdp_by_country = 'https://raw.githubusercontent.com/armandossrecife/teste/main/data_country.csv'
4
5 # Load the data from the .csv file into a pandas DataFrame
6 df = pd.read_csv(gdp_by_country)
7
8 # Create a bar plot
9 plt.bar(df['Country'], df['GDP(trillions)'])
10
11 # rotate the x-axis labels by 90 degrees
12 plt.xticks(rotation=90)
13
14 # Add labels and a title to the plot
15 plt.xlabel('Country')
16 plt.ylabel('GDP (in trillions)')
17 plt.title('GDP by Country')
18
19 # Show the plot
20 plt.show()
```

Usando dados de arquivos .csv

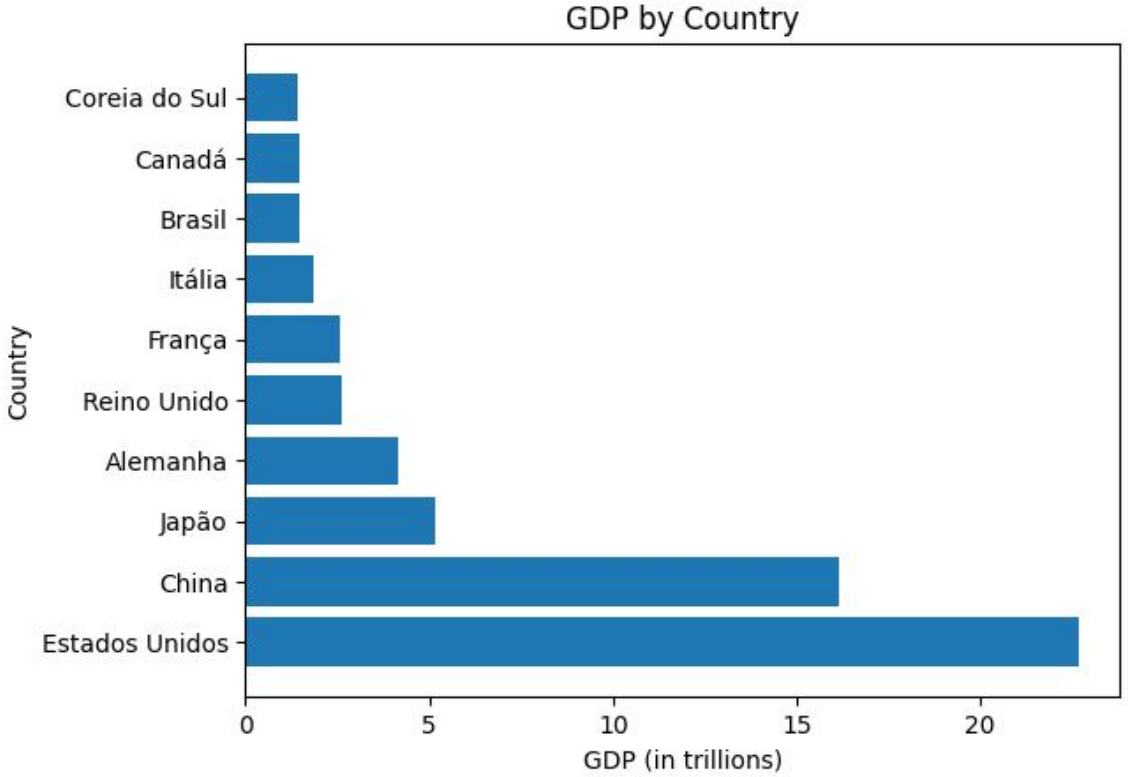


	Country	GDP(trillions)
0	Estados Unidos	22.67
1	China	16.15
2	Japão	5.15
3	Alemanha	4.17
4	Reino Unido	2.62
5	França	2.58
6	Itália	1.85
7	Brasil	1.48
8	Canadá	1.46
9	Coreia do Sul	1.41

Usando dados de arquivos .csv (Gráfico de Barras Horizontal)

```
import pandas as pd

gdp_by_country = 'https://raw.githubusercontent.com/armandossrecife/teste/main/data_country.csv'

# Load the data from the .csv file into a pandas DataFrame
df = pd.read_csv(gdp_by_country)

# Create a bar plot
plt.barh(df['Country'], df['GDP(trillions)'])

# Add labels and a title to the plot
plt.xlabel('GDP (in trillions)')
plt.ylabel('Country')

plt.title('GDP by Country')

# Show the plot
plt.show()
```

Usando dados de arquivos .csv (Gráfico de Barras Horizontal)



	Country	GDP(trillions)
0	Estados Unidos	22.67
1	China	16.15
2	Japão	5.15
3	Alemanha	4.17
4	Reino Unido	2.62
5	França	2.58
6	Itália	1.85
7	Brasil	1.48
8	Canadá	1.46
9	Coreia do Sul	1.41