

CC8210 – NCA210

Programação Avançada I

Prof. Reinaldo A. C. Bianchi
Prof. Isaac Jesus da Silva
Prof. Danilo H. Perico

Aula de Hoje

- Introdução à biblioteca matplotlib.
- Referências:
 - <https://matplotlib.org/users/index.html>

Introdução à Biblioteca Gráficas matplotlib

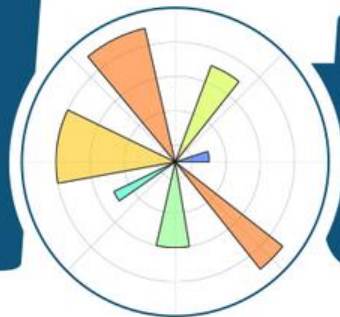
Visualização de Dados

- A visualização de dados consiste na apresentação de informações através de elementos visuais, como gráficos, diagramas, mapas, tabelas, entre outros.
 - Por meio dela, fica muito mais fácil analisar os resultados, auxiliando o processo de identificar padrões e tendências e tomar decisões.
 - Boas visualizações de dados ajudam a descomplicar as informações, transmitindo uma mensagem clara e objetiva.

Benefícios da Visualização de Dados

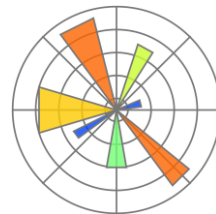
- Com o enorme volume de informações disponíveis, as ferramentas de visualização de dados são cada vez mais fundamentais para as empresas.
- Benefícios:
 - Apresentação mais envolvente / Absorção rápida das informações
 - Facilita a tomada de decisão
 - Auxilia a encontrar conexões importantes para solucionar os problemas da sua empresa

matplotlib





- Matplotlib é uma biblioteca de software para criação de gráficos e visualizações de dados em geral, feita para e da linguagem de programação Python e sua extensão de matemática NumPy.
- Criada pelo biólogo e neurocientista John D. Hunter, a biblioteca possui uma comunidade ativa atuando em seu desenvolvimento.



- É distribuída sob uma licença BSD.
- Oferece uma interface de programação orientada a objetos para incluir gráficos em aplicações usando toolkits de interface gráfica.
- Ele permite que você trace gráficos diferentes, como gráficos de barras, dispersão, histogramas, espectros e muito mais.
- Muitas vezes requer apenas algumas linhas de código.

Usando o Matplotlib pyplot

- É instalado junto com o Anaconda.
- Para usar:

```
import matplotlib.pyplot as plt
```

- **Pyplot:**
 - Permitem a criação de figuras, uma área para exibir o gráfico na figura, desenhar linhas na área do gráfico, decorar o gráfico com rótulos, etc.

Usando o Matplotlib pyplot

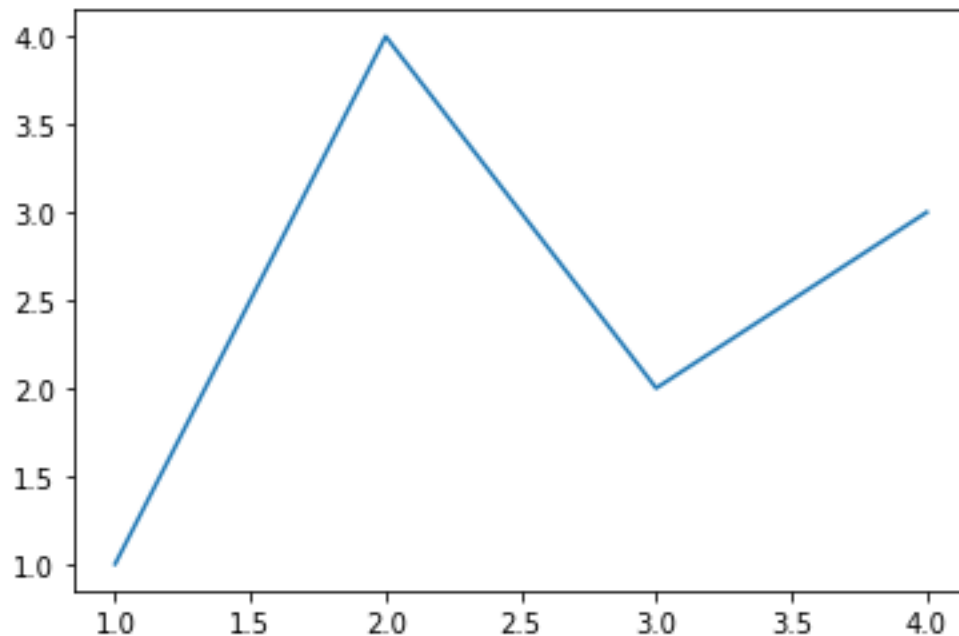
- No pyplot, vários estados são preservados após a chamada de uma função dentro de um contexto, simplificando assim o seu trabalho sobre a figura ou área de desenho atuais.
- Assim, você pode ter uma figura complexa formada por várias áreas de gráficos distintas, e o pyplot mantém para você informações sobre para cada área de desenho.

Criando um gráfico

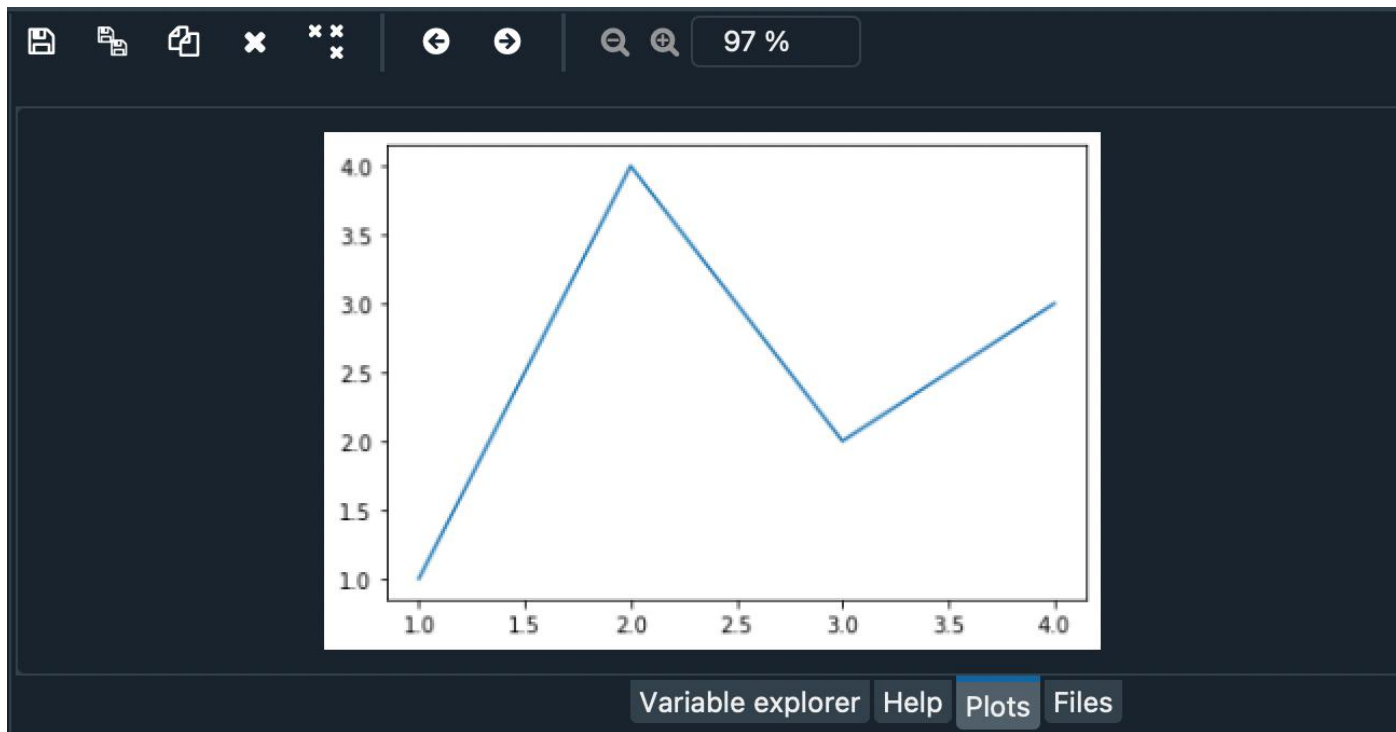
- Podemos criar um objeto gráfico usando a função `subplot()`:

```
import matplotlib.pyplot as plt
fig, ax = plt.subplots()
ax.plot([1, 2, 3, 4], [1, 4, 2, 3])
plt.show()
```

Criando um gráfico



No Spyder os gráficos são apresentados na tab plot



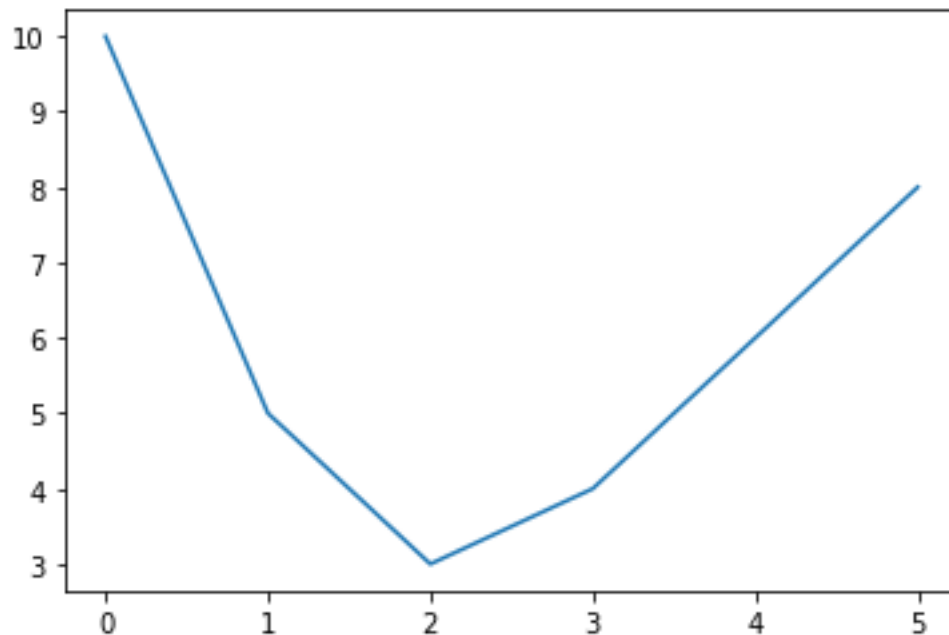
Criando um gráfico

- Podemos criar um objeto gráfico usando a função `plot()`:

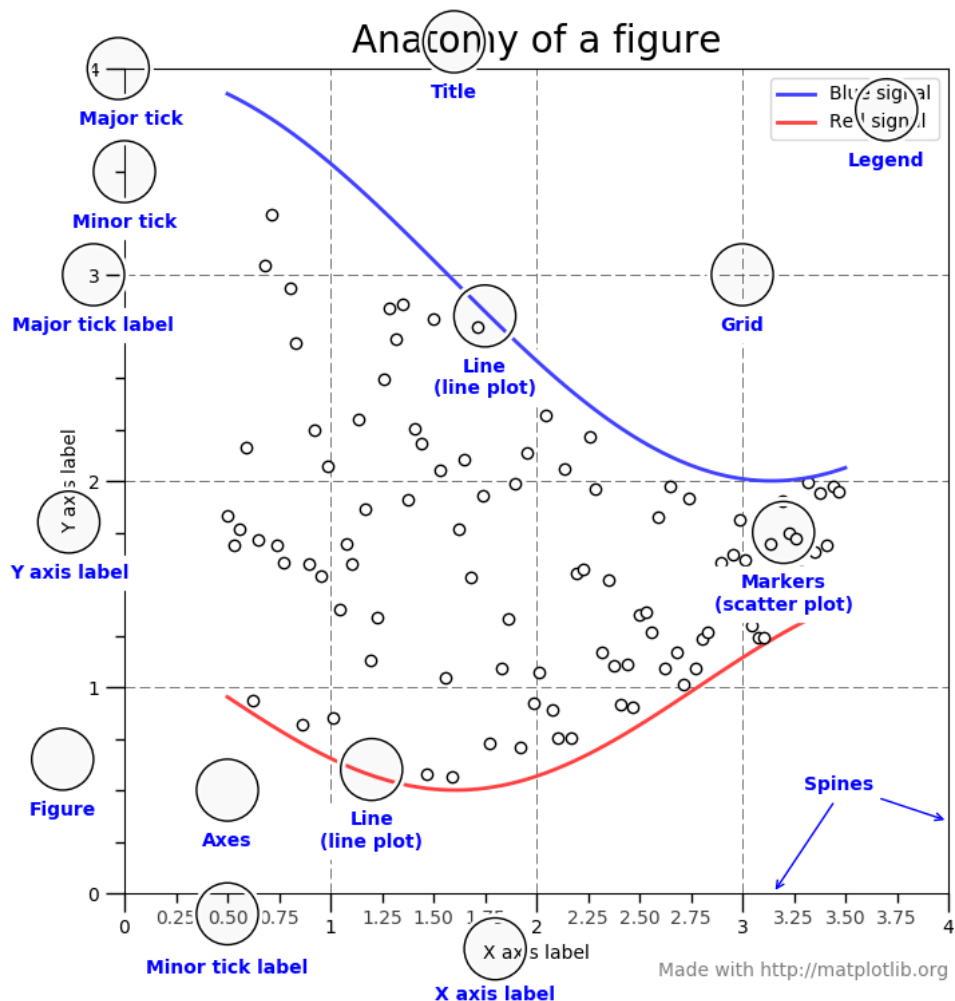
```
import matplotlib.pyplot as plt  
plt.plot( [10,5,3,4,6,8] )  
plt.show()
```

- O método `plot()` executa o desenho nos gráficos "atuais", criando esses gráficos (e sua figura-mãe) se eles ainda não existirem.

Criando um gráfico



Anatomia de um Gráfico



Anatomia de uma figura

- Figura
 - A figura mantém o controle de todos os Gráficos (Axes), um conjunto com os outros objetos (títulos, lendas da figura, etc), e o canvas.
 - Não se preocupe muito com o canvas:
 - É o objeto que realmente faz o desenho para obter o seu gráfico, mas como o usuário é mais ou menos invisível para você.
 - Uma figura pode conter qualquer número de Gráficos (Axes), mas normalmente terá pelo menos um.

Anatomia de uma figura

- Gráficos (Axes ou Plot)
 - É a região da imagem com o espaço de dados (um gráfico).
 - Uma determinada figura pode conter muitos Gráficos, mas um determinado objeto Axes só pode estar em uma Figura.
 - Os Gráficos contêm dois (ou três no caso de objetos do Eixo 3D) eixos de coordenadas, e que cuidam dos limites dos dados.
 - Os limites de dados também podem ser controlados por meio dos métodos `axes.Axes.set_xlim()` e `axes.Axes.set_ylim()`.

Anatomia de uma figura

- Eixos (Axis)
 - Estes são os objetos da linha numérica.
 - Eles cuidam de definir os limites de gráfico e gerar as marcas no eixo (ticks) e rótulos (ticklabels).
 - A localização das marcas é determinada por um objeto Localizador e as grades são formatadas por um Formatter.
 - A combinação do localizador correto e do formatter dá um controle muito fino sobre os locais de marca e rótulos.

IMPORTANTE – Tradução meio estranha

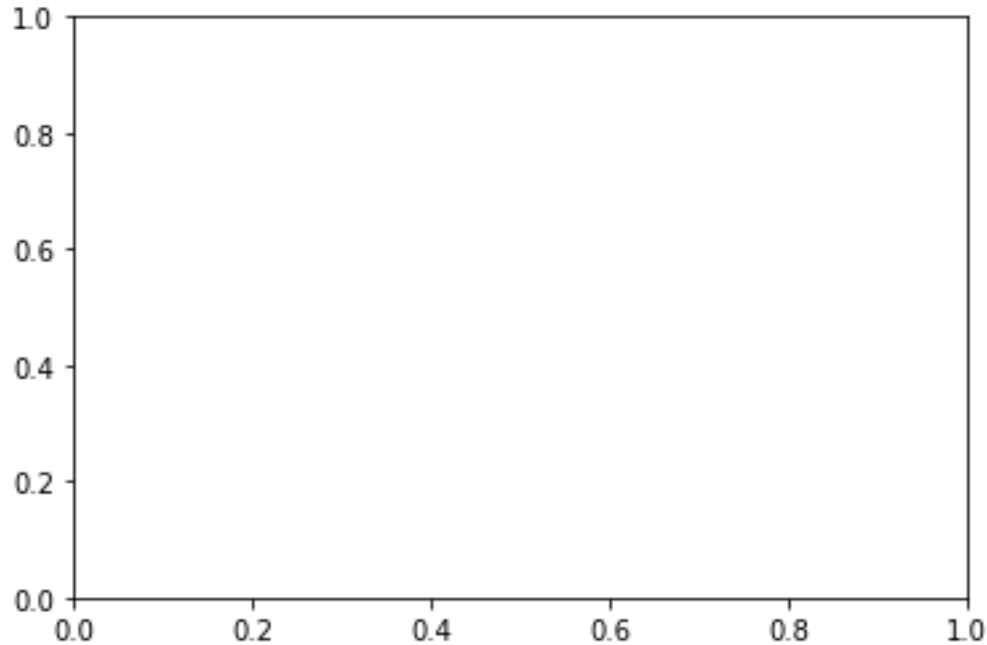
- Cuidado com a diferença entre AXES x AXIS
- AXES:
 - Gráfico
 - Plot
- AXIS:
 - Eixo das coordenadas do gráfico
 - Abscissa, ordenada

Criando uma figura

- A maneira mais fácil de se criar uma figura é usando a função `figure()`:

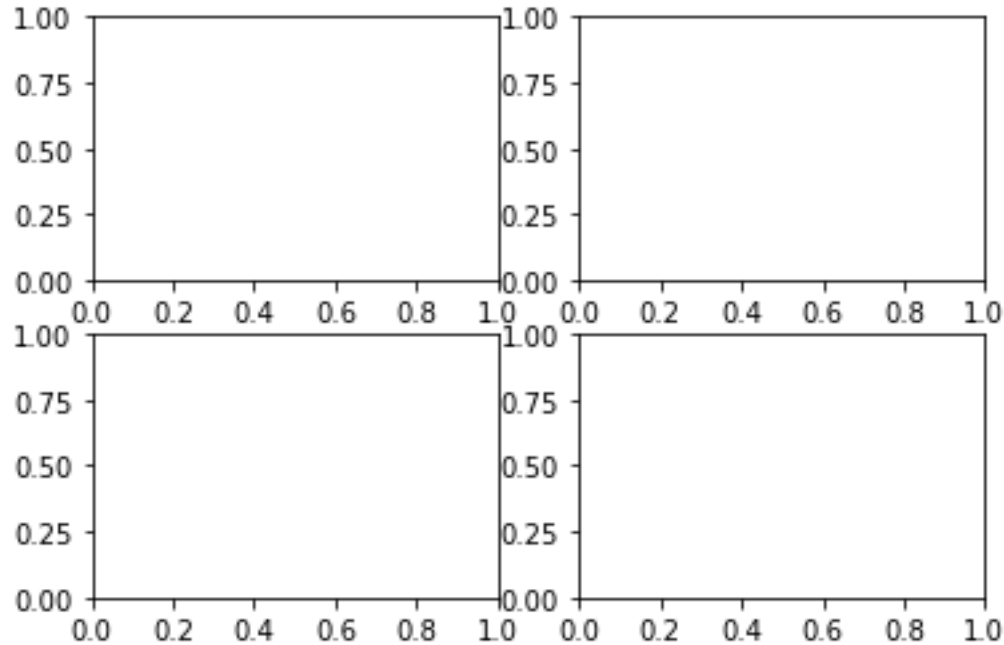
```
import matplotlib.pyplot as plt
fig = plt.figure()    # an empty figure
fig, ax = plt.subplots() # single Axes
fig, axs = plt.subplots(2, 2)
# a figure with a 2x2 grid of Axes
```

Criando uma figura com um Gráfico (Axes)



```
fig, ax = plt.subplots()
```

Criando uma figura com 2 x 2 Gráficos (Axes)



```
fig, axs = plt.subplots(2,2)
```

Anatomia de uma figura

- Artistas (Artists)
 - Basicamente tudo o que você pode ver na figura é um artista.
 - Isso inclui os objetos Figura, Gráficos e Eixo, objetos de texto, objetos Line2D...
 - Quando a figura é renderizada, todos os artistas são atraídos para a tela.
 - A maioria dos artistas está ligada a um gráfico:
 - Um artista não pode ser compartilhado por múltiplos gráficos, ou movido de um para outro.

Títulos e rótulos

- Podemos definir os títulos e rótulos dos eixos usando os métodos:
 - Um Título é definido via `set_title()`
 - Uma etiqueta (ou rótulo) para o eixo X é definida via `set_xlabel()`
 - Uma etiqueta (ou rótulo) para o eixo Y é definida via `set_ylabel()`

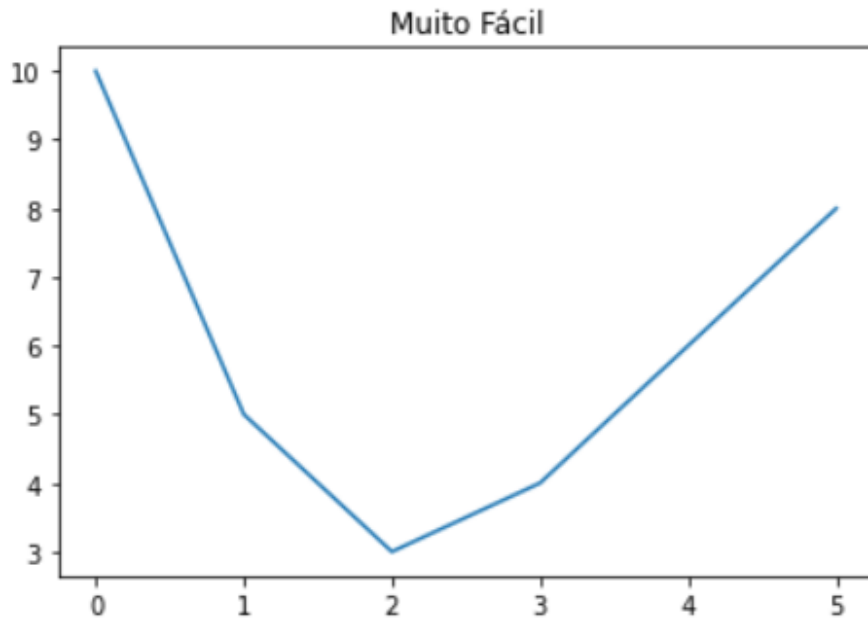
Criando um gráfico com título

- Podemos criar um título para a figura usando a função `title()`:

```
import matplotlib.pyplot as plt  
plt.plot( [10,5,3,4,6,8] )  
plt.title("Muito Fácil")
```

Criando um gráfico com título

```
import matplotlib.pyplot as plt  
plt.plot( [10,5,3,4,6,8] )  
plt.title("Muito Fácil") |  
plt.show()
```



Criando um gráfico com título

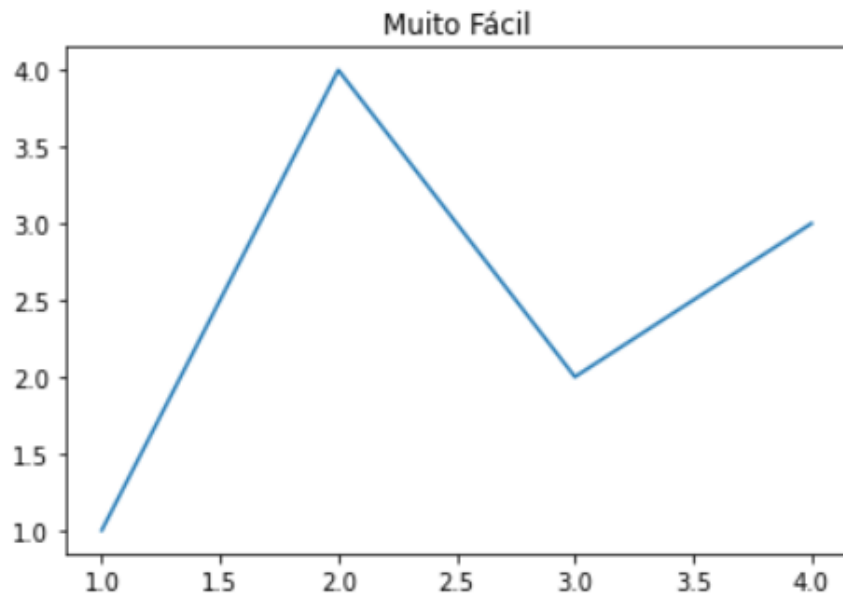
- Podemos criar um objeto gráfico usando a função `set_title()`:

```
import matplotlib.pyplot as plt
fig, ax = plt.subplots()
ax.plot([1, 2, 3, 4], [1, 4, 2, 3])
ax.set_title("Muito Fácil")
```

Criando um gráfico com título

```
import matplotlib.pyplot as plt
fig, ax = plt.subplots()
ax.plot([1, 2, 3, 4], [1, 4, 2, 3])
ax.set_title("Muito Fácil")
```

```
Text(0.5, 1.0, 'Muito Fácil')
```



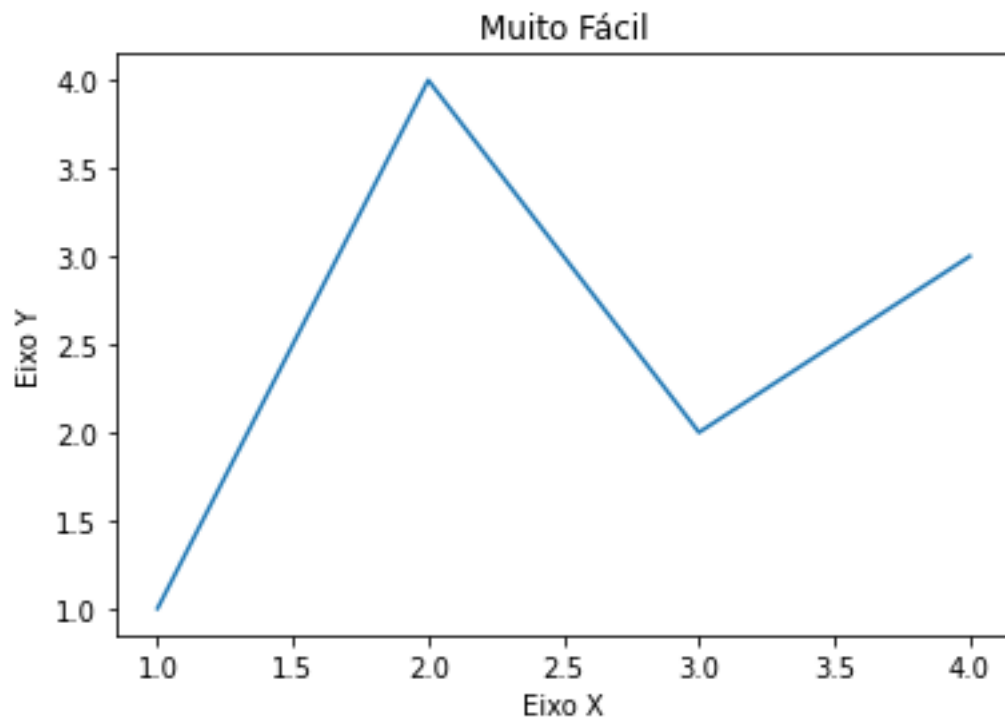
Criando um gráfico com título e rótulos

- Podemos criar um objeto gráfico usando a função `set_title()`, `set_xlabel()` e `set_ylabel()`:

```
import matplotlib.pyplot as plt
fig, ax = plt.subplots()
ax.plot([1, 2, 3, 4], [1, 4, 2, 3])
ax.set_title("Muito Fácil")
ax.set_xlabel ("Eixo X")
ax.set_ylabel ("Eixo Y")
```

Criando um gráfico com título

```
import matplotlib.pyplot as plt
fig, ax = plt.subplots()
ax.plot([1, 2, 3, 4], [1, 4, 2, 3])
ax.set_title("Muito Fácil")
ax.set_xlabel ("Eixo X")
ax.set_ylabel ("Eixo Y")
```



Modificando a linha...

- Podemos mudar o tipo da linha de dados passando parâmetros para a função `plot()`.

```
x = [1, 2, 3, 4]
```

```
y = [1, 4, 2, 3]
```

```
fig, ax = plt.subplots()
```

```
ax.plot(x, y, color='r', linestyle='--', linewidth=10)
```


Tipos de linhas

line styles

'.'

'-.' -.-.-.-.-

'--' ---

'_' _____

Tipos de linhas

solid
'solid'

dotted
'dotted'

dashed
'dashed'

dashdot
'dashdot'

loosely dotted
(0, (1, 10))

dotted
(0, (1, 1))

densely dotted
(0, (1, 1))

loosely dashed
(0, (5, 10))

dashed
(0, (5, 5))

densely dashed
(0, (5, 1))

loosely dashdotted
(0, (3, 10, 1, 10))

dashdotted
(0, (3, 5, 1, 5))

densely dashdotted
(0, (3, 1, 1, 1))

dashdotdotted
(0, (3, 5, 1, 5, 1, 5))

loosely dashdotdotted
(0, (3, 10, 1, 10, 1, 10))

densely dashdotdotted
(0, (3, 1, 1, 1, 1, 1))

Cores

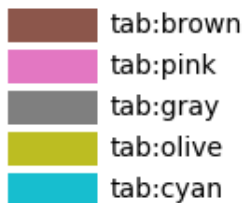
- Cores Básicas:

Base Colors



- Paleta de cores

Tableau Palette



Cores

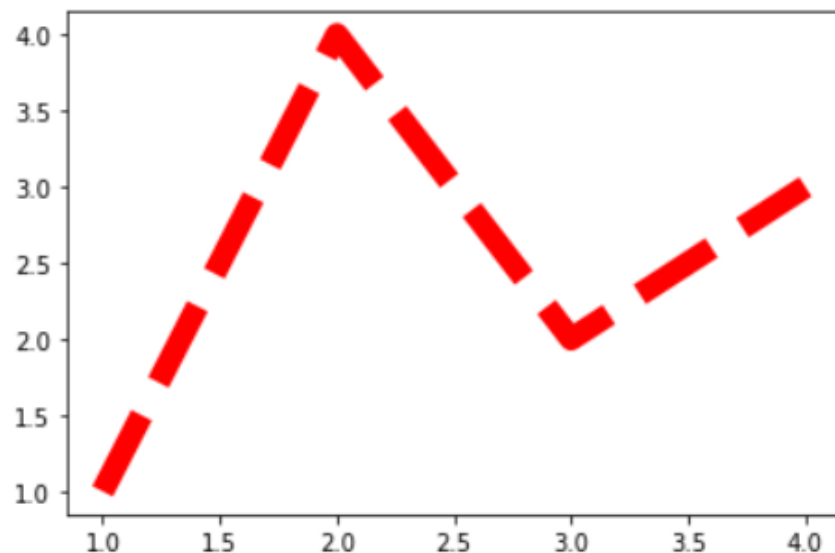
CSS Colors

black	bisque	forestgreen	slategrey
dimgray	darkorange	limegreen	lightsteelblue
dimgrey	burlywood	darkgreen	cornflowerblue
gray	antiquewhite	green	royalblue
grey	tan	lime	ghostwhite
darkgray	navajowhite	seagreen	lavender
darkgrey	blanchedalmond	mediumseagreen	midnightblue
silver	papayawhip	springgreen	navy
lightgray	moccasin	mintcream	darkblue
lightgrey	orange	mediumspringgreen	mediumblue
gainsboro	wheat	mediumaquamarine	blue
whitesmoke	oldlace	aquamarine	slateblue
white	floralwhite	turquoise	darkslateblue
snow	darkgoldenrod	lightseagreen	mediumslateblue
rosybrown	goldenrod	mediumturquoise	mediumpurple
lightcoral	cornsilk	azure	rebeccapurple
indianred	gold	lightcyan	blueviolet
brown	lemonchiffon	paleturquoise	indigo
firebrick	khaki	darkslategray	darkorchid
maroon	palegoldenrod	darkslategrey	darkviolet
darkred	darkkhaki	teal	mediumorchid
red	ivory	darkcyan	thistle
mistyrose	beige	aqua	plum
salmon	lightyellow	cyan	violet
tomato	lightgoldenrodyellow	darkturquoise	purple
darksalmon	olive	cadetblue	darkmagenta
coral	yellow	powderblue	fuchsia
orangered	olivedrab	lightblue	magenta
lightsalmon	yellowgreen	deepskyblue	orchid
sienna	darkolivegreen	skyblue	mediumvioletred
seashell	greenyellow	lightskyblue	deeppink
chocolate	chartreuse	steelblue	hotpink
saddlebrown	lawngreen	aliceblue	lavenderblush
sandybrown	honeydew	dodgerblue	palevioletred
peachpuff	darkseagreen	lightslategray	crimson
peru	palegreen	lightslategrey	pink
linen	lightgreen	slategray	lightpink

Mudando a linha

```
x = [1, 2, 3, 4]
y = [1, 4, 2, 3]
fig, ax = plt.subplots()
ax.plot(x, y, color='r', linestyle='--', linewidth=10)
```

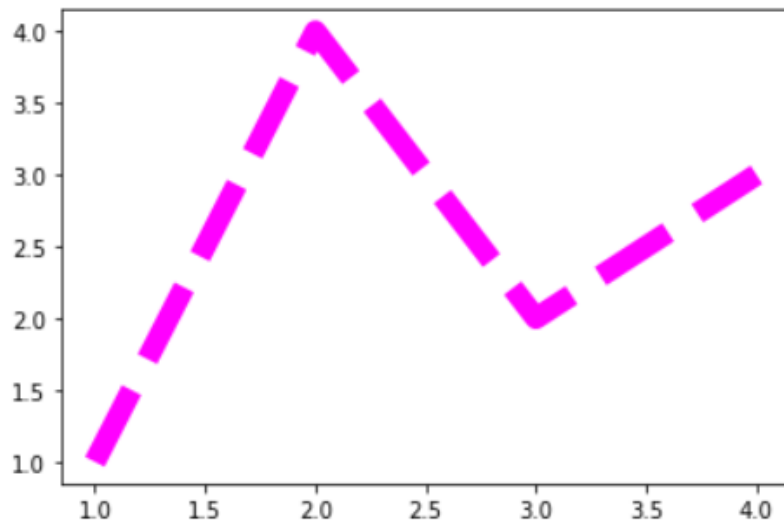
[<matplotlib.lines.Line2D at 0x17d964ec5b0>]



Mudando a linha color='fuchsia'

```
x = [1, 2, 3, 4]
y = [1, 4, 2, 3]
fig, ax = plt.subplots()
ax.plot(x, y, color='fuchsia', linestyle='--', linewidth=10)
```

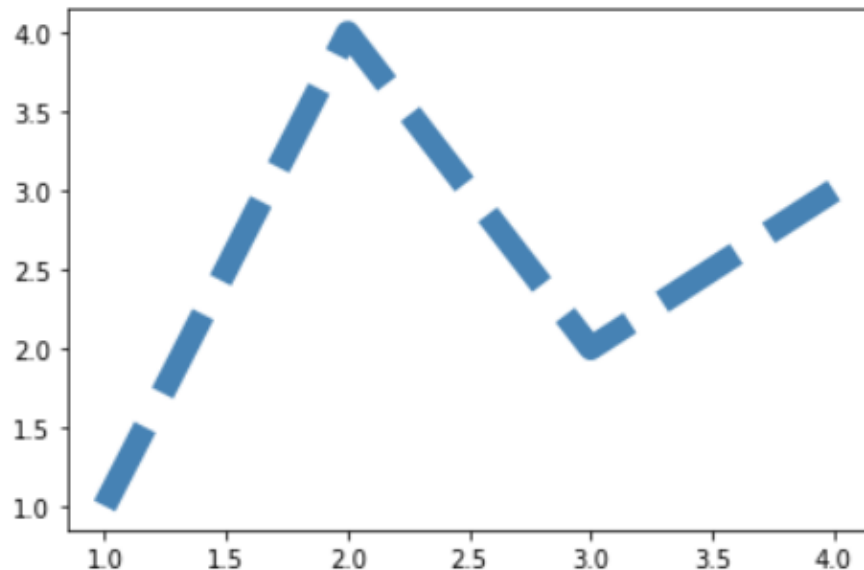
[<matplotlib.lines.Line2D at 0x17d96542730>]



Mudando a linha color= 'steelblue'

```
x = [1, 2, 3, 4]
y = [1, 4, 2, 3]
fig, ax = plt.subplots()
ax.plot(x, y, color='steelblue', linestyle='--', linewidth=10)
```

[<matplotlib.lines.Line2D at 0x17d9659a820>]



Adicionando um marcador na linha...

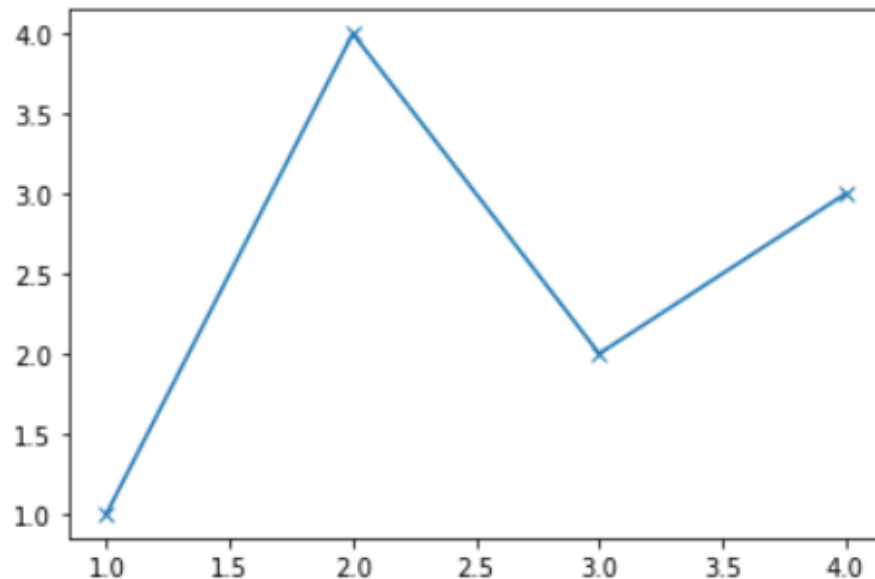
- Podemos usar um marcador na linha de dados passando parâmetros para a função `plot()`.

```
x = [1, 2, 3, 4]
y = [1, 4, 2, 3]
fig, ax = plt.subplots()
ax.plot(x, y, marker = 'x')
```


Marcadores nos pontos: `marker = 'x'`

```
x = [1, 2, 3, 4]
y = [1, 4, 2, 3]
fig, ax = plt.subplots()
ax.plot(x, y, marker = 'x')
```

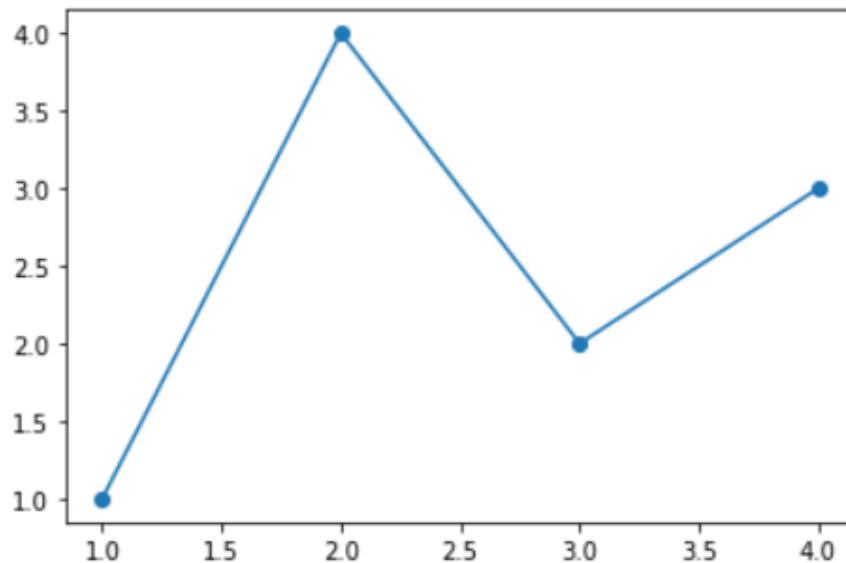
[<matplotlib.lines.Line2D at 0x17d965f2820>]



Marcadores nos pontos: `marker = 'o'`














































```
x = [1, 2, 3, 4]
y = [1, 4, 2, 3]
fig, ax = plt.subplots()
ax.plot(x, y, marker = 'o')
```

[<matplotlib.lines.Line2D at 0x17d966488b0>]

























Marcadores

Filled markers

'o'				'p'			
'v'				'*'			
'^'				'h'			
'<'				'H'			
'>'				'D'			
'8'				'd'			
's'				'P'			
				'X'			

Marcadores

Un-filled markers

'.'		1	
','		2	
'1'		3	
'2'		4	
'3'		5	
'4'		6	
'+'		7	
'x'		8	
' '		9	
'_'		10	
0		11	

Retirando a linha...

- Podemos retirar a linha de dados passando parâmetros para a função `plot()`.

```
x = [1, 2, 3, 4]
```

```
y = [1, 4, 2, 3]
```

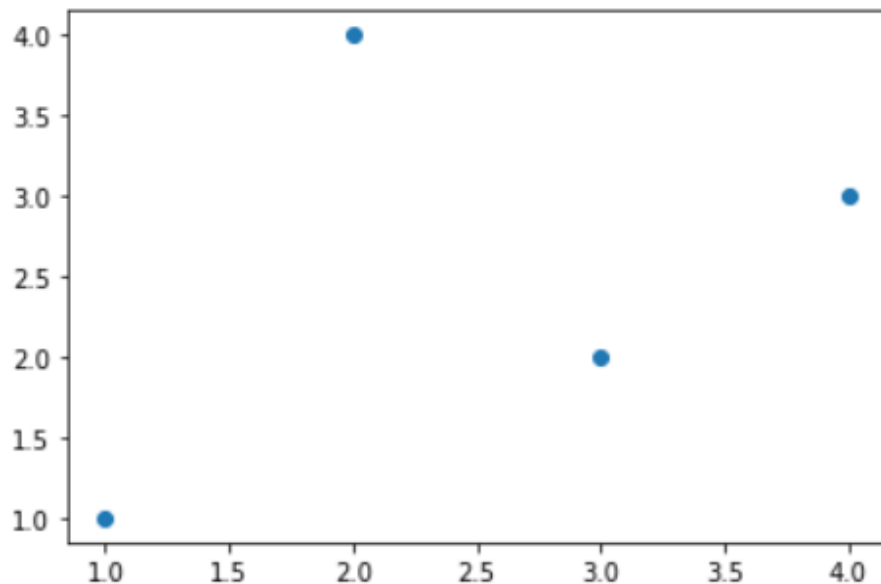
```
fig, ax = plt.subplots()
```

```
ax.plot(x, y, marker = 'o', linestyle = 'None')
```

Marcadores nos pontos: `linestyle = 'None'`

```
x = [1, 2, 3, 4]
y = [1, 4, 2, 3]
fig, ax = plt.subplots()
ax.plot(x, y, marker = 'o', linestyle = 'None')
```

[<matplotlib.lines.Line2D at 0x17d967460d0>]



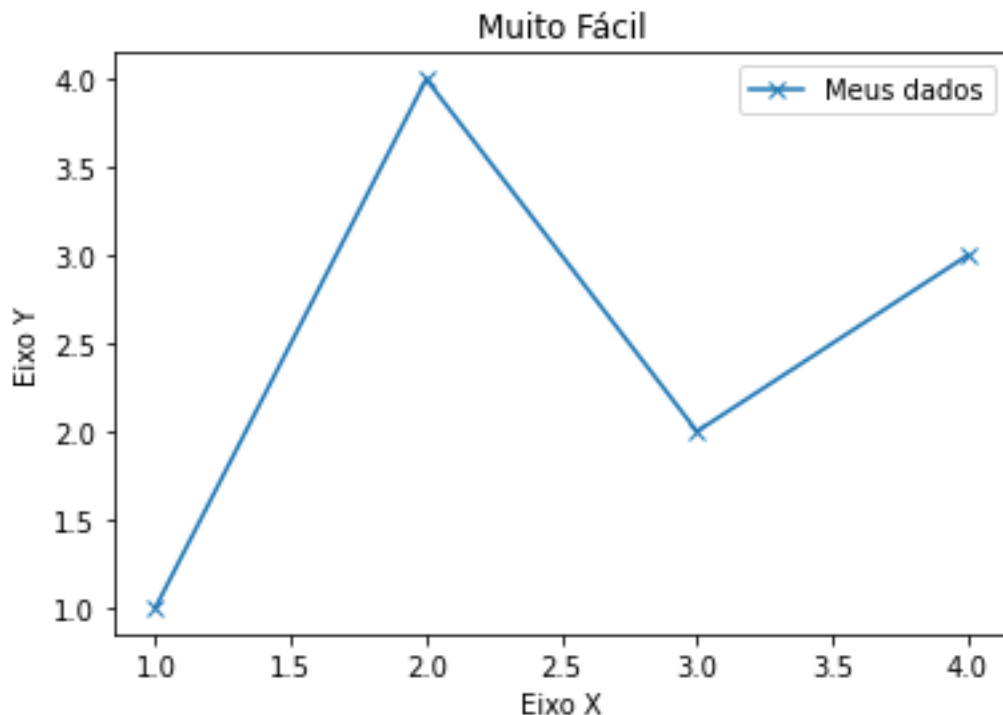
Atribuindo o nome dos eixos

- Podemos mudar o espaçamento dos dados nos eixos usando a função `set_xlabel()` e `set_ylabel()`:

```
x = [1, 2, 3, 4]
y = [1, 4, 2, 3]
fig, ax = plt.subplots()
ax.plot(x, y, marker = 'x')
ax.legend(['Meus dados'])
ax.set_title("Muito Fácil")
ax.set_xlabel ("Eixo X")
ax.set_ylabel ("Eixo Y")
```

Atribuindo o nome dos eixos

```
x = [1, 2, 3, 4]
y = [1, 4, 2, 3]
fig, ax = plt.subplots()
ax.plot(x, y, marker = 'x')
ax.legend(['Meus dados'])
ax.set_title("Muito Fácil")
ax.set_xlabel ("Eixo X")
ax.set_ylabel ("Eixo Y")
```



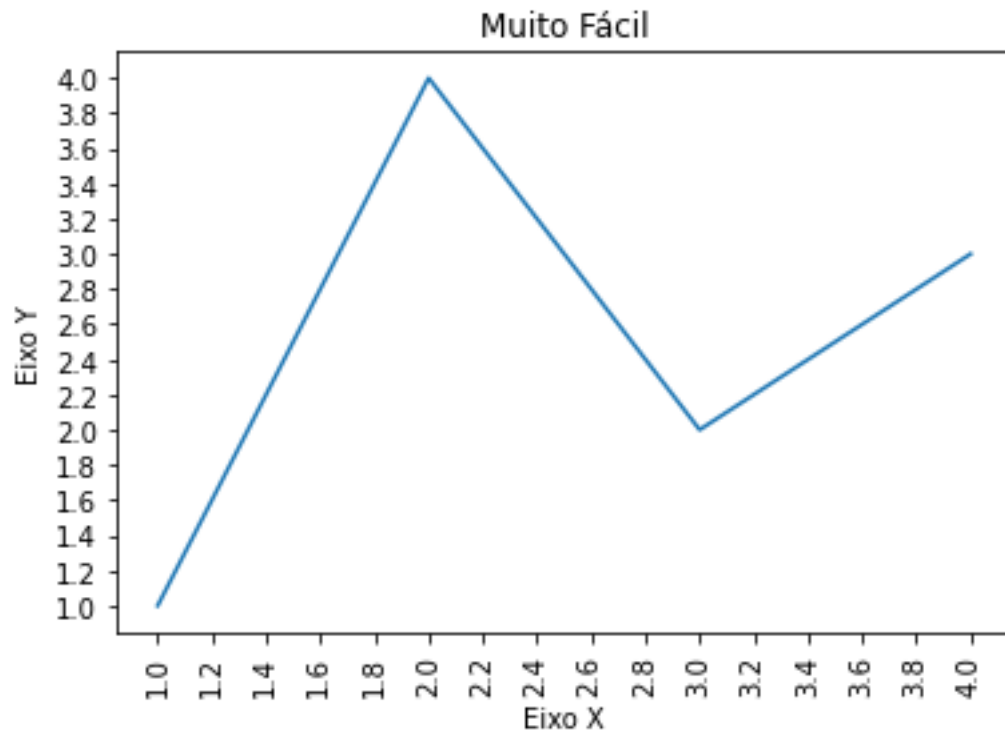
Mudando o espaçamento dos ticks

- Podemos mudar o espaçamento dos dados nos eixos usando a função `xticks()` e `yticks()`:

```
plt.xticks(np.arange(min(x), max(x)+0.1, 0.2), rotation=90)
plt.yticks(np.arange(min(y), max(y)+0.1, 0.2))
```

Mudando o espaçamento dos ticks

```
x = [1, 2, 3, 4]
y = [1, 4, 2, 3]
fig, ax = plt.subplots()
ax.plot(x, y, marker = 'x')
ax.legend(['Meus dados'])
ax.set_title("Muito Fácil")
ax.set_xlabel ("Eixo X")
ax.set_ylabel ("Eixo Y")
plt.xticks(np.arange(min(x), max(x)+0.1, 0.2), rotation=90)
plt.yticks(np.arange(min(y), max(y)+0.1, 0.2))
```



Marcando a grade

- Podemos criar uma grade para a figura com usando a função `grid()`:

```
ax.grid(color='r', linestyle='-', linewidth=1)
```

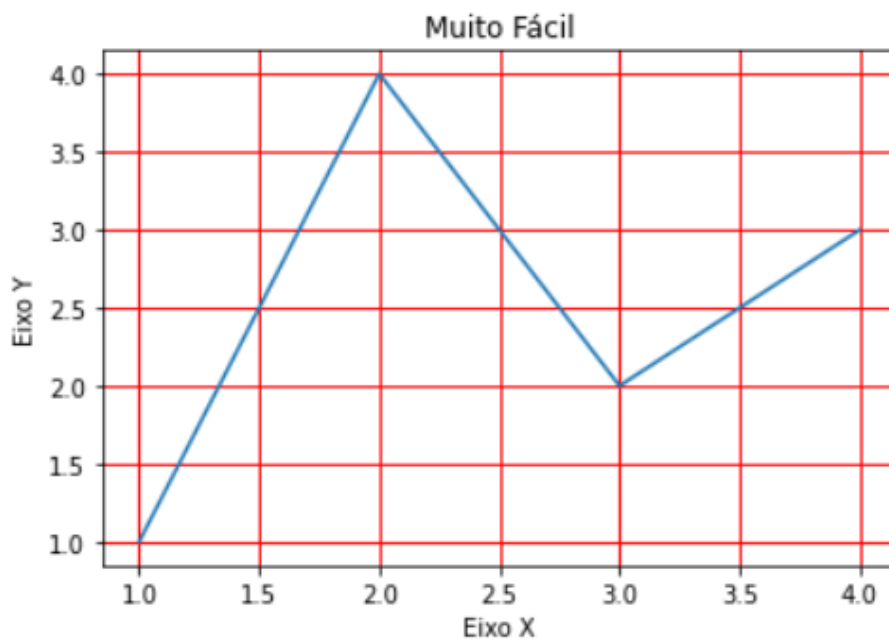
```
ax.grid(color='g', linestyle='--', linewidth=1)
```

```
ax.grid(color='b', linestyle='-.', linewidth=1)
```

```
ax.grid(color='y', linestyle=':', linewidth=1)
```

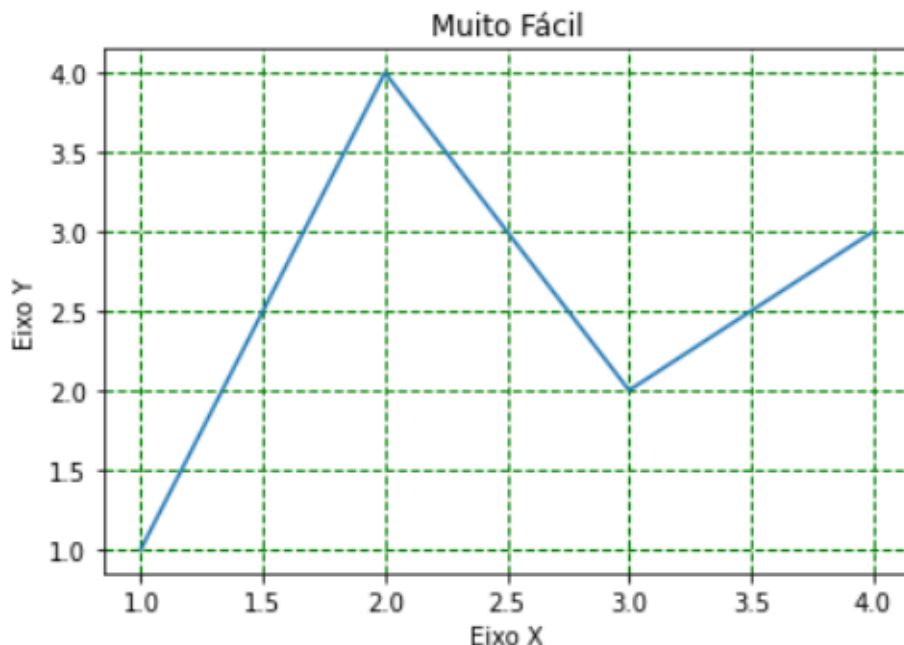
Marcando a grade

```
fig, ax = plt.subplots()
ax.plot([1, 2, 3, 4], [1, 4, 2, 3])
ax.set_title("Muito Fácil")
ax.set_xlabel ("Eixo X")
ax.set_ylabel ("Eixo Y")
ax.grid(color='r', linestyle='-', linewidth=1)
```



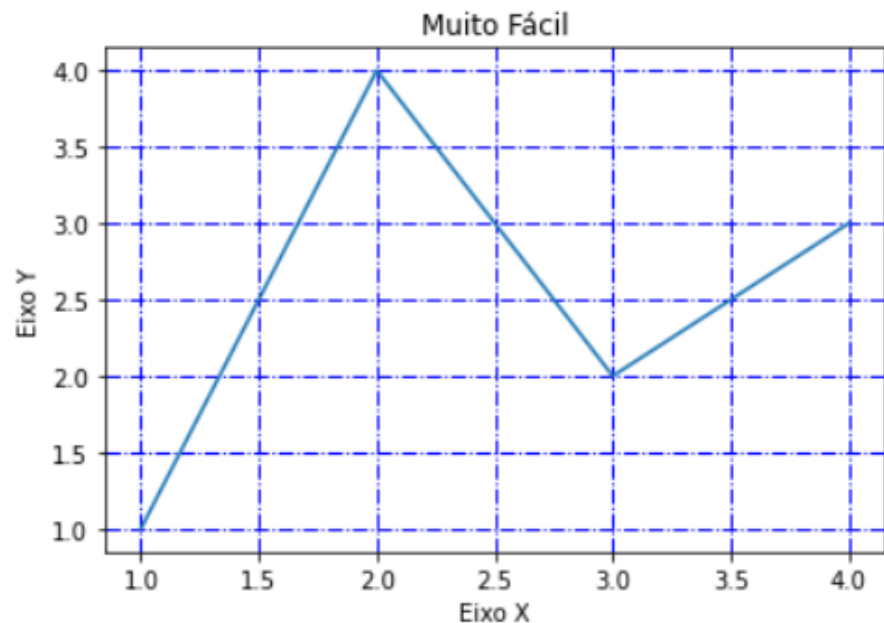
Marcando a grade

```
fig, ax = plt.subplots()
ax.plot([1, 2, 3, 4], [1, 4, 2, 3])
ax.set_title("Muito Fácil")
ax.set_xlabel ("Eixo X")
ax.set_ylabel ("Eixo Y")
ax.grid(color='g', linestyle='--', linewidth=1)
```



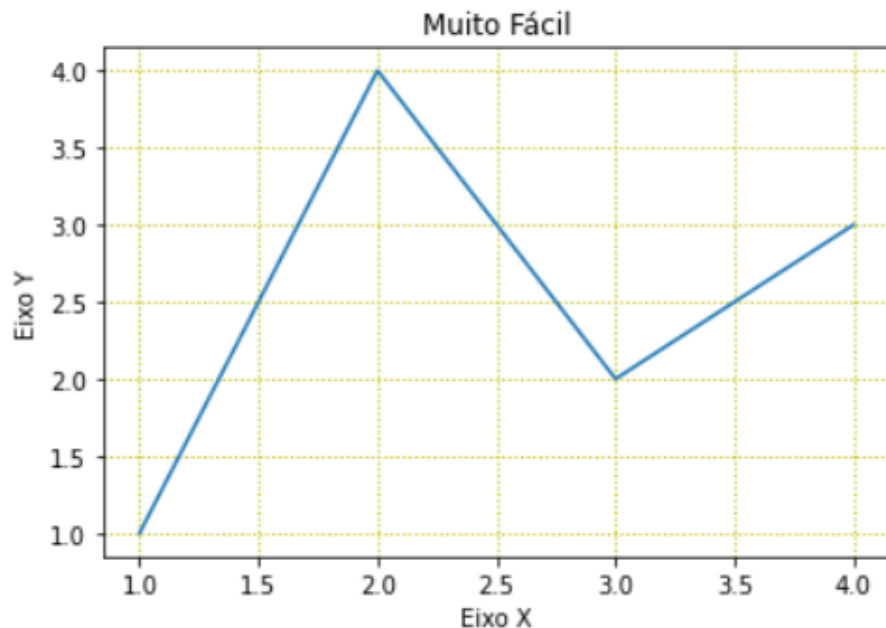
Marcando a grade

```
fig, ax = plt.subplots()
ax.plot([1, 2, 3, 4], [1, 4, 2, 3])
ax.set_title("Muito Fácil")
ax.set_xlabel ("Eixo X")
ax.set_ylabel ("Eixo Y")
ax.grid(color='b', linestyle='-.', linewidth=1)
```



Marcando a grade

```
fig, ax = plt.subplots()
ax.plot([1, 2, 3, 4], [1, 4, 2, 3])
ax.set_title("Muito Fácil")
ax.set_xlabel ("Eixo X")
ax.set_ylabel ("Eixo Y")
ax.grid(color='y', linestyle=':', linewidth=1)
```



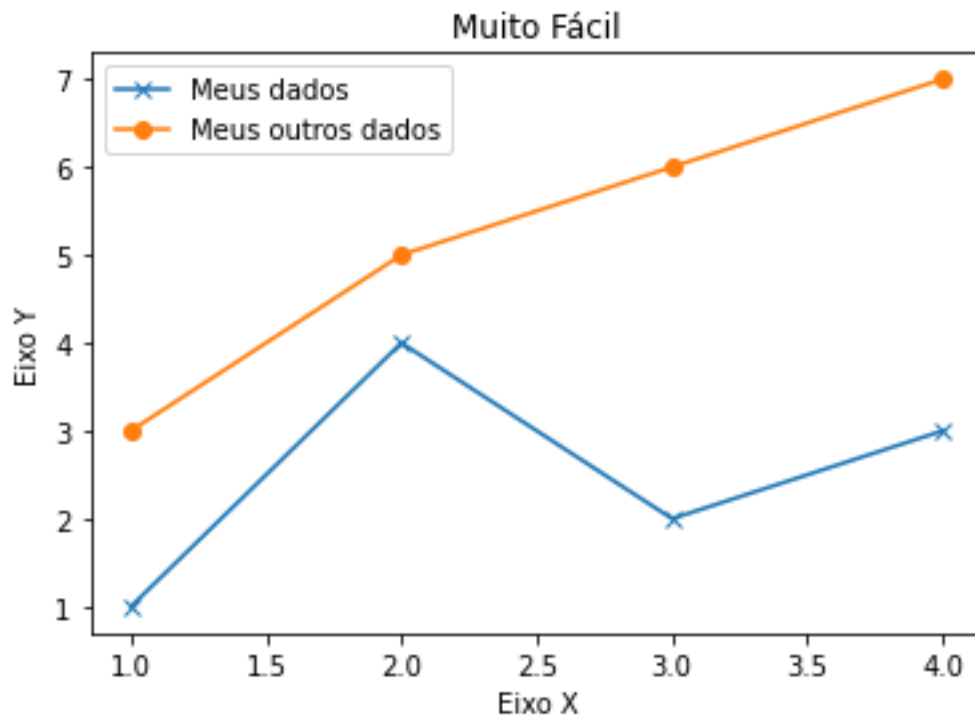
Mais de um conjunto de dados

- É possível imprimir mais de um conjunto de dados, várias linhas, no mesmo gráfico...

```
x = [1, 2, 3, 4]
y = [1, 4, 2, 3]
z = [3, 5, 6, 7]

fig, ax = plt.subplots()
ax.plot(x, y, marker = 'x')
ax.plot(x, z, marker = 'o')
ax.legend(['Meus dados', 'Meus outros dados'])
ax.set_title("Muito Fácil")
ax.set_xlabel ("Eixo X")
ax.set_ylabel ("Eixo Y")
```


Mais de um conjunto de dados

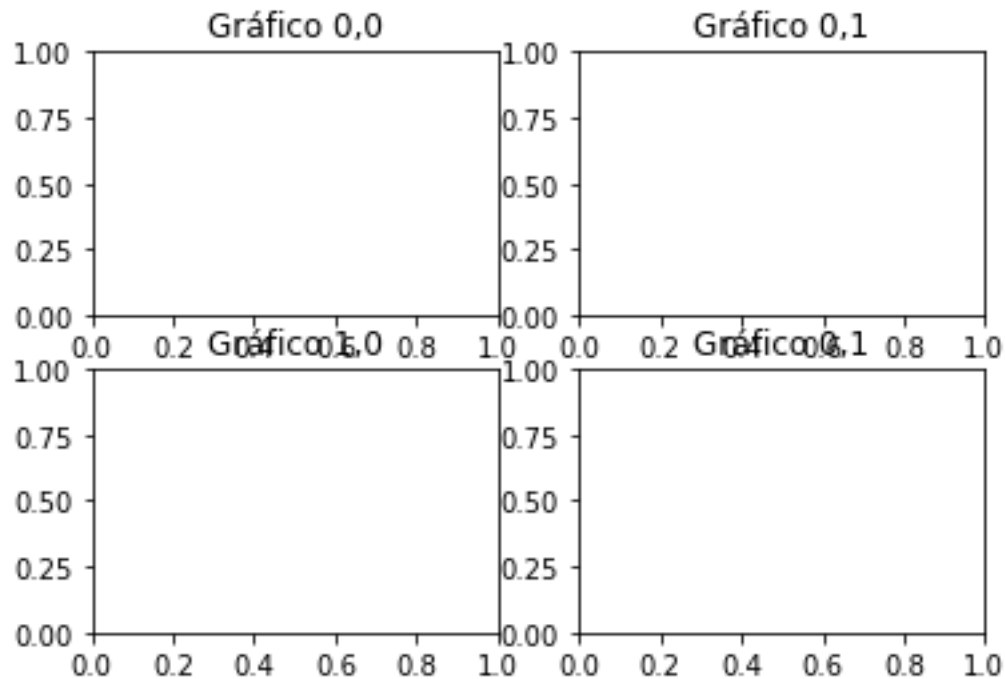


Criando sub-gráficos com títulos

- Podemos criar um título para a figura com subplots usando a função `set_title()`:

```
import matplotlib.pyplot as plt
fig, axs = plt.subplots(2,2)
axs[0,0].set_title("Gráfico 0,0")
axs[0,1].set_title("Gráfico 0,1")
axs[1,0].set_title("Gráfico 1,0")
axs[1,1].set_title("Gráfico 0,1")
```

Criando sub-gráficos com títulos ☹

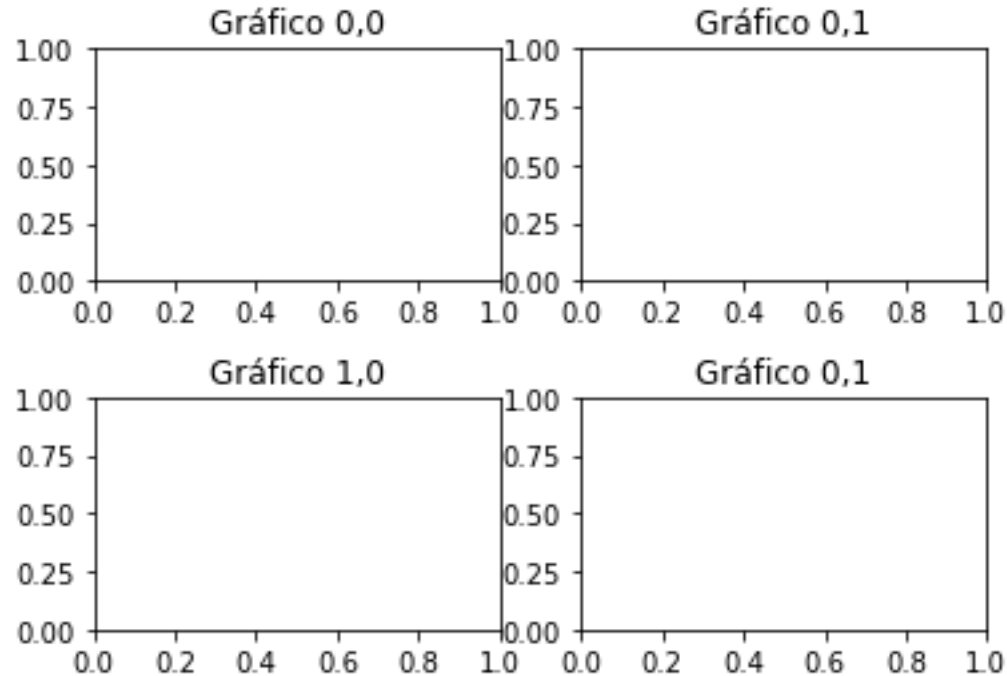


Criando sub-gráficos com títulos e espaço

- Para ajustar o espaço podemos usar a função `subplot_adjust()`:

```
import matplotlib.pyplot as plt
fig, axs = plt.subplots(2,2)
plt.subplots_adjust( hspace=0.5 )
axs[0,0].set_title("Gráfico 0,0")
axs[0,1].set_title("Gráfico 0,1")
axs[1,0].set_title("Gráfico 1,0")
axs[1,1].set_title("Gráfico 0,1")
```

Criando sub-gráficos com títulos e espaço



subplots_adjust

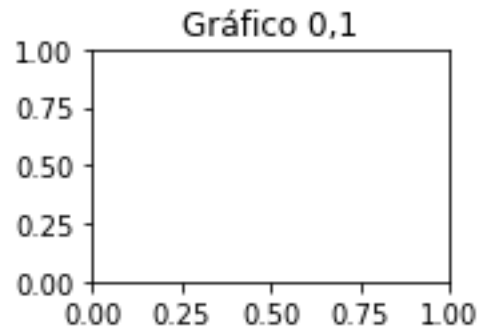
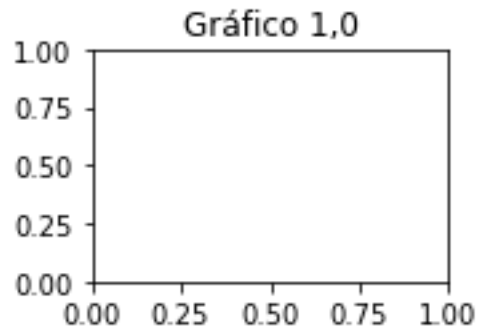
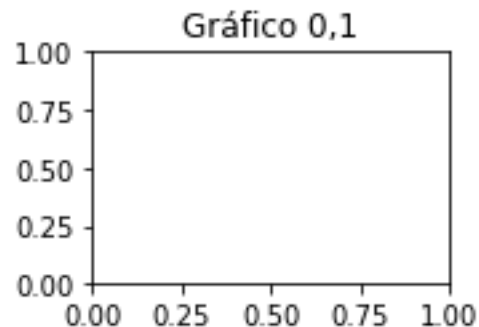
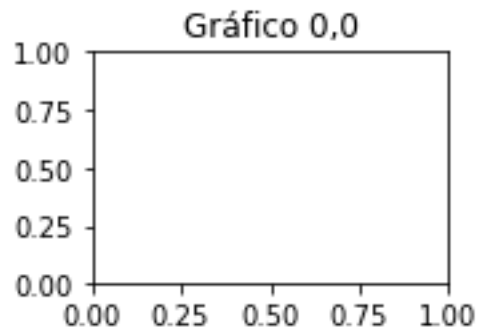
- `subplots_adjust(left=None, bottom=None, right=None, top=None, wspace=None, hspace=None)`
 - `left` = the left side of the subplots of the figure
 - `right` = the right side of the subplots of the figure
 - `bottom` = the bottom of the subplots of the figure
 - `top` = the top of the subplots of the figure
 - `wspace` = the amount of width for blank space between subplots
 - `hspace` = the amount of height for white space between subplots

Criando sub-gráficos com títulos e espaço

- Para ajustar o espaço podemos usar a função `subplot_adjust()`:

```
import matplotlib.pyplot as plt
fig, axs = plt.subplots(2,2)
plt.subplots_adjust(hspace=0.5, wspace=0.5)
axs[0,0].set_title("Gráfico 0,0")
axs[0,1].set_title("Gráfico 0,1")
axs[1,0].set_title("Gráfico 1,0")
axs[1,1].set_title("Gráfico 0,1")
```

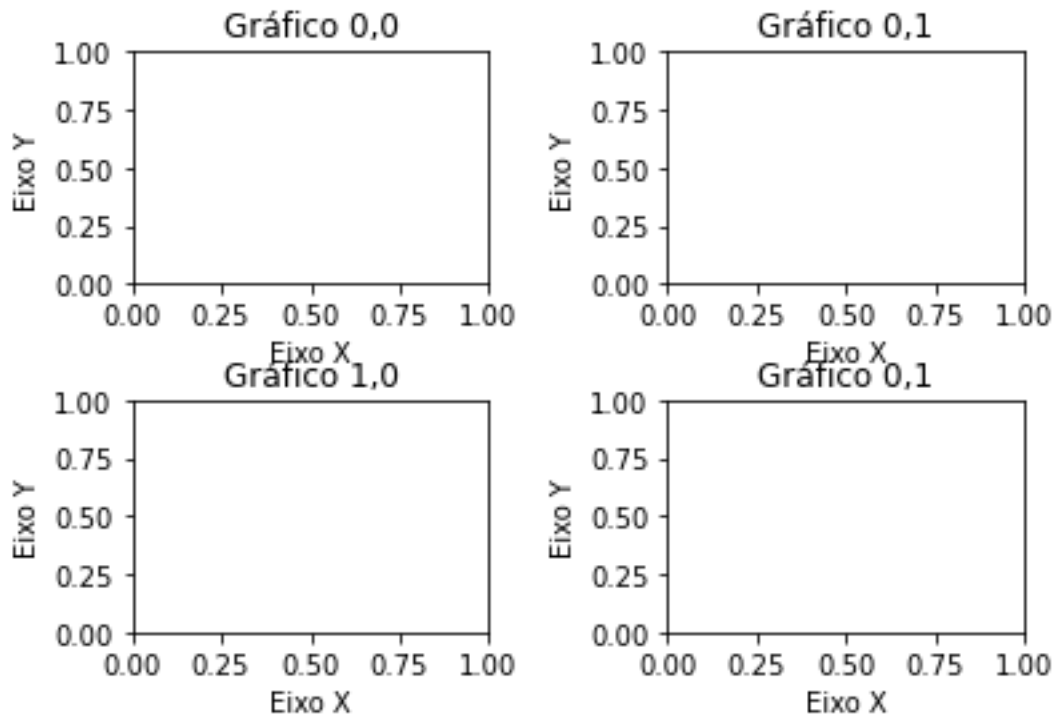
Criando sub-gráficos com títulos e espaço



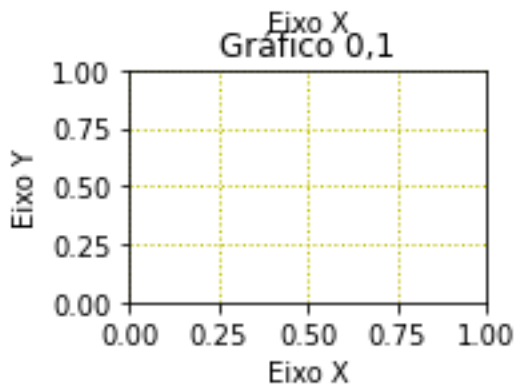
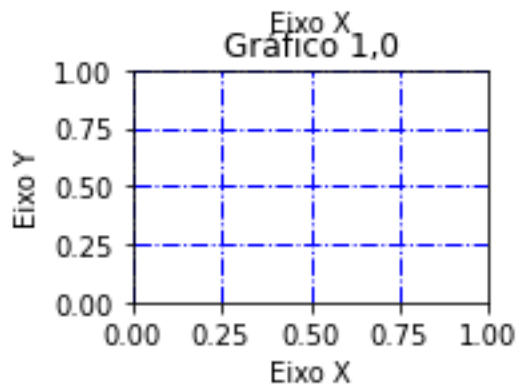
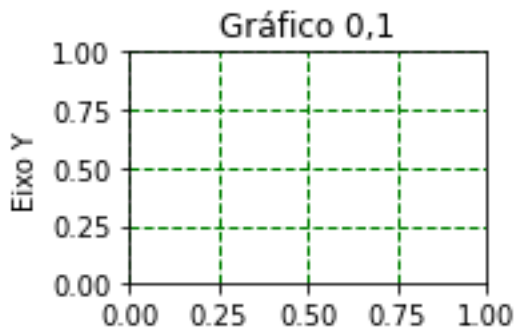
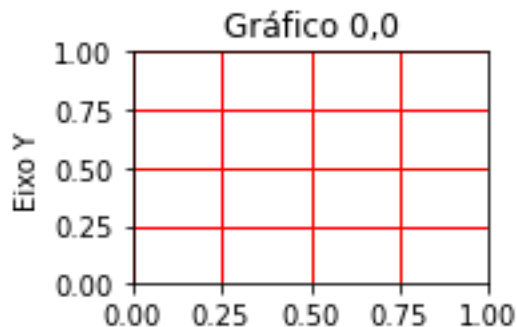
Criando sub-gráficos com títulos e rótulos e espaço

```
import matplotlib.pyplot as plt
fig, axs = plt.subplots(2,2)
plt.subplots_adjust( hspace=0.5 , wspace = 0.5)
axs[0,0].set_title("Gráfico 0,0")
axs[0,0].set_xlabel ("Eixo X")
axs[0,0].set_ylabel ("Eixo Y")
axs[0,1].set_title("Gráfico 0,1")
axs[0,1].set_xlabel ("Eixo X")
axs[0,1].set_ylabel ("Eixo Y")
axs[1,0].set_title("Gráfico 1,0")
axs[1,0].set_xlabel ("Eixo X")
axs[1,0].set_ylabel ("Eixo Y")
axs[1,1].set_title("Gráfico 0,1")
axs[1,1].set_xlabel ("Eixo X")
axs[1,1].set_ylabel ("Eixo Y")
```

Criando sub-gráficos com títulos e rótulos e espaço



Criando sub-gráficos com títulos e rótulos e espaço e grade



Mais detalhes, veja o help...

- Cada função tem outras dezenas de parâmetros...
- Por exemplo, ver:
 - https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.grid.html
 - https://matplotlib.org/3.1.0/api/_as_gen/matplotlib.lines.Line2D.html#matplotlib.lines.Line2D.set_linestyle

Tipos de Gráficos

Tipos de gráficos

- A biblioteca matplotlib possui muitos tipos diferentes de gráficos:
 - Linha
 - Barra
 - Pizza
 - Histograma
 - Scatter plot
 - 3D plot
 - Image plot
 - Contour plot
 - Polar plot

Tipos de gráficos

- A biblioteca matplotlib possui muitos tipos diferentes de gráficos:

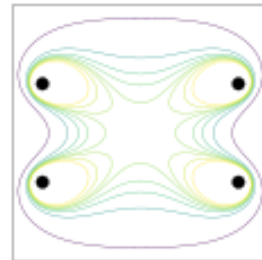
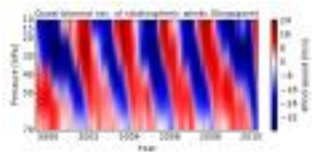
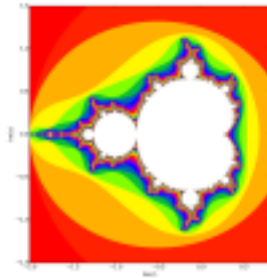
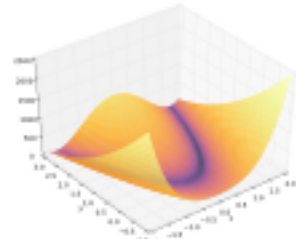
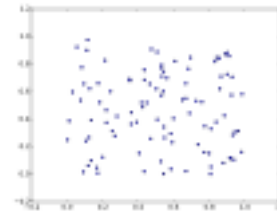
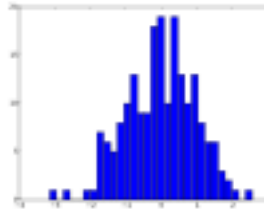
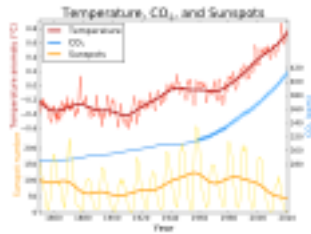


Gráfico de Linha

- Já vimos...

Gráfico de barras

- `matplotlib.pyplot.bar`
 - The bars are positioned at x with the given *alignment*. Their dimensions are given by *width* and *height*. The vertical baseline is *bottom*(default 0).
 - Each of x , *height*, *width*, and *bottom* may either be a scalar applying to all bars, or it may be a sequence of length N providing a separate value for each bar.

Pizza

- `matplotlib.pyplot.pie`
 - Make a pie chart of array x .
 - The fractional area of each wedge is given by $x/\text{sum}(x)$.
 - If $\text{sum}(x) < 1$, then the values of x give the fractional area directly and the array will not be normalized.
 - The resulting pie will have an empty wedge of size $1 - \text{sum}(x)$.
 - The wedges are plotted counterclockwise, by default starting from the x-axis.

Espalhamento (Scatter plot)

- `matplotlib.pyplot.scatter`
 - A scatter plot of y vs. x with varying marker size and/or color.

Histograma

- `matplotlib.pyplot.hist`
 - Compute and draw the histogram of x . The return value is a tuple $(n, bins, patches)$ or $([n0, n1, \dots], bins, [patches0, patches1, \dots])$ if the input contains multiple data. See the documentation of the *weights* parameter to draw a histogram of already-binned data.
 - Multiple data can be provided via x as a list of datasets of potentially different length $([x0, x1, \dots])$, or as a 2-D ndarray in which each column is a dataset. Note that the ndarray form is transposed relative to the list form.

3D

- `mpl_toolkits.mplot3d`
 - Aqui é um enrosco, por que tem todo um toolkit para desenhar gráficos em 3D.
 - Seria outra aula...

Contorno

- `matplotlib.pyplot.contour`
 - `contour` and `contourf` draw contour lines and filled contours, respectively.
 - Except as noted, function signatures and return values are the same for both versions.

Boxplot

- `matplotlib.pyplot.boxplot`
 - Make a box and whisker plot for each column of `x` or each vector in sequence `x`.
 - The box extends from the lower to upper quartile values of the data, with a line at the median.
 - The whiskers extend from the box to show the range of the data.
 - Flier points are those past the end of the whiskers.

Polar

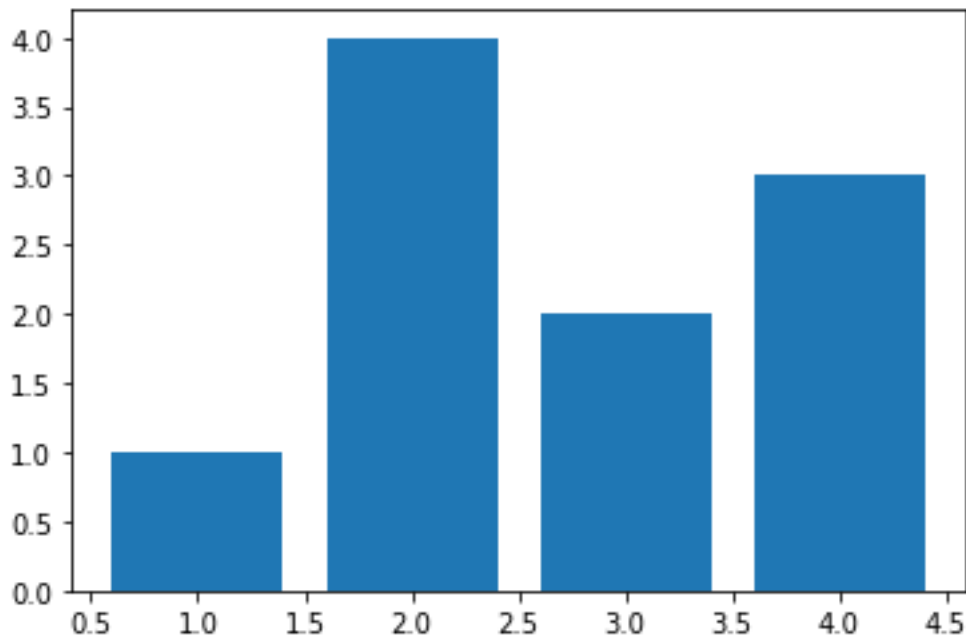
- `matplotlib.pyplot.polar`
 - Make a polar plot.
 - `polar(theta, r, **kwargs)`
 - Multiple *theta*, *r* arguments are supported

Exemplos

- Todos daqui:
- <https://matplotlib.org/3.3.2/gallery/index.html>

Bar chart

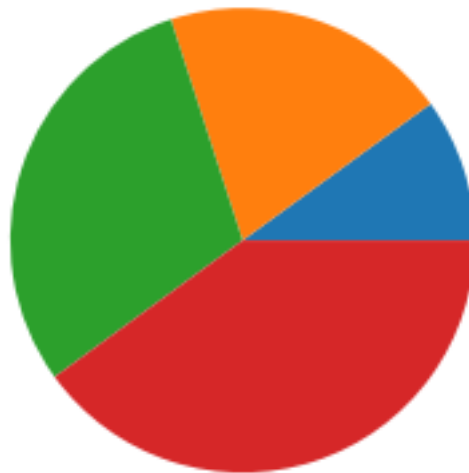
```
x = [1, 2, 3, 4]
y = [1, 4, 2, 3]
fig, ax =
plt.subplots()
ax.bar(x,y)
plt.show()
```



Pie chart

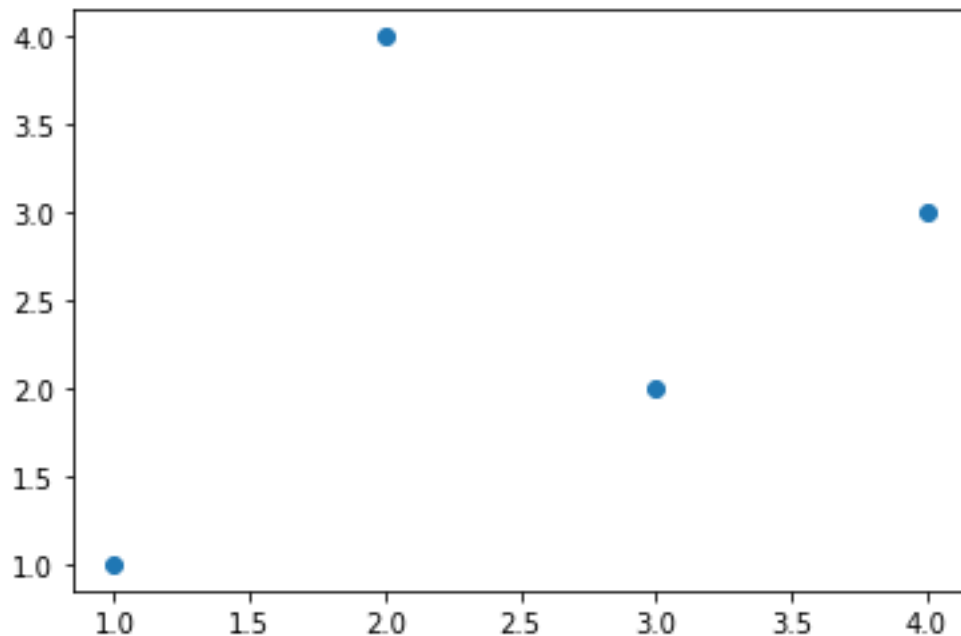
```
x = [1, 2, 3, 4]
```

```
fig, ax =  
plt.subplots()  
ax.pie(x)  
plt.show()
```



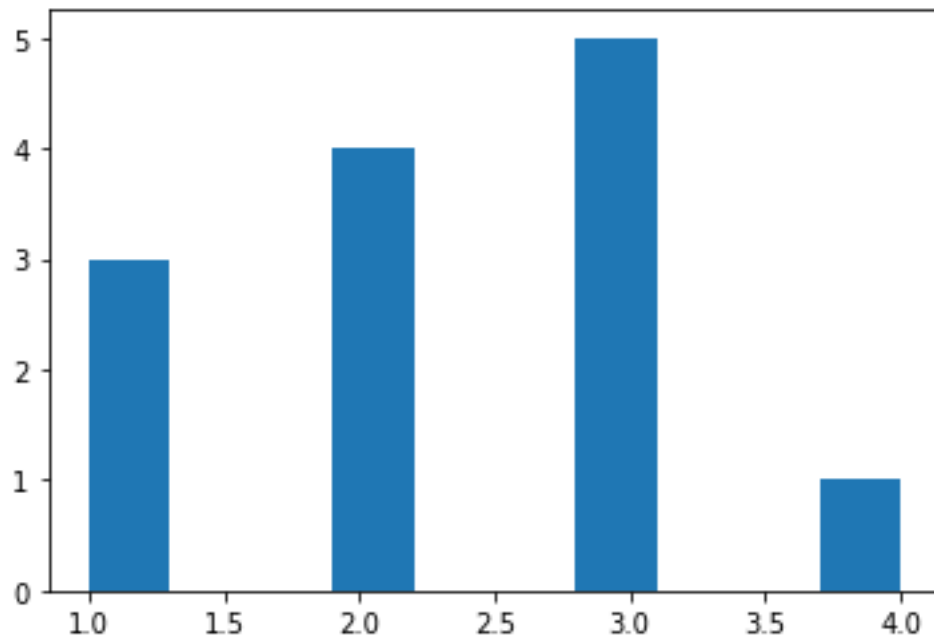
Scatter chart

```
x = [1, 2, 3, 4]
y = [1, 4, 2, 3]
fig, ax =
plt.subplots()
ax.scatter(x,y)
plt.show()
```



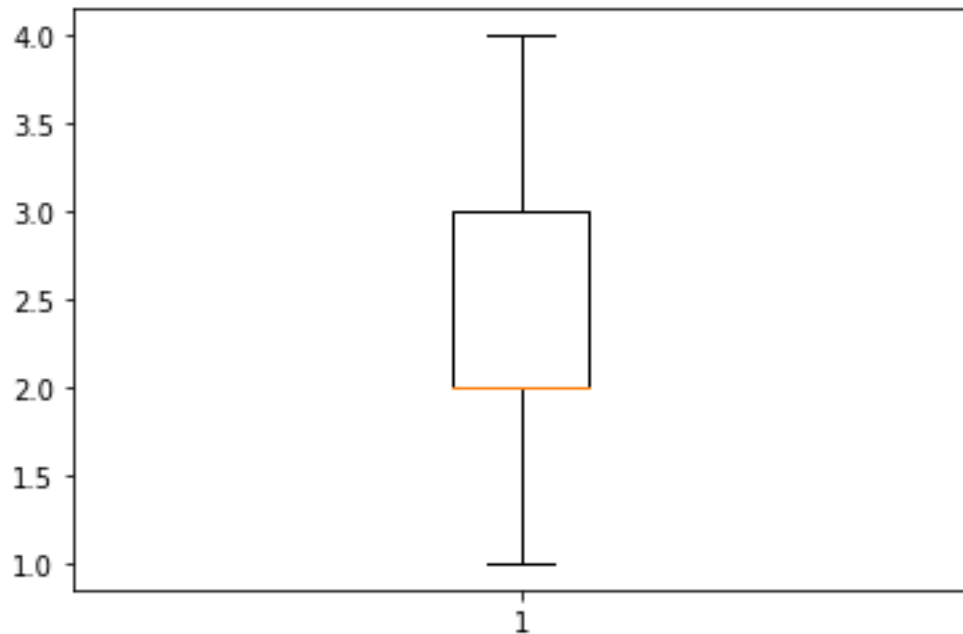
Histograma chart

```
x = [1, 2, 3, 4, 3, 2,  
3, 2, 1, 3, 3, 2, 1]  
fig, ax =  
plt.subplots()  
ax.hist(x)  
plt.show()
```



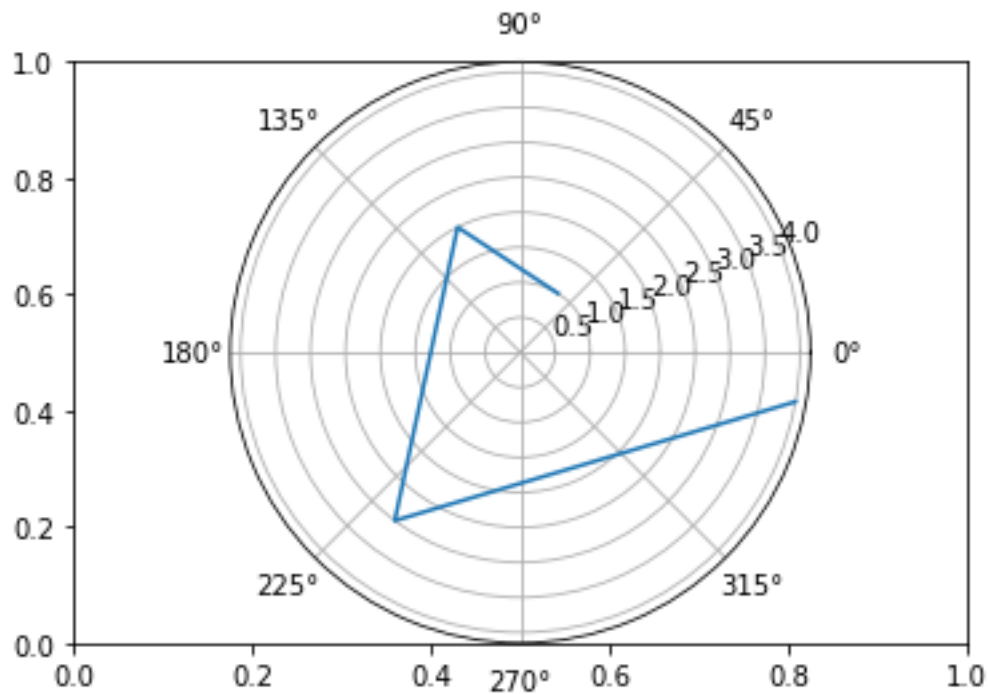
Boxplot

```
x = [1, 2, 3, 4, 3, 2,  
3, 2, 1, 3, 3, 2, 1]  
  
fig, ax =  
plt.subplots()  
ax.boxplot(x)  
plt.show()
```



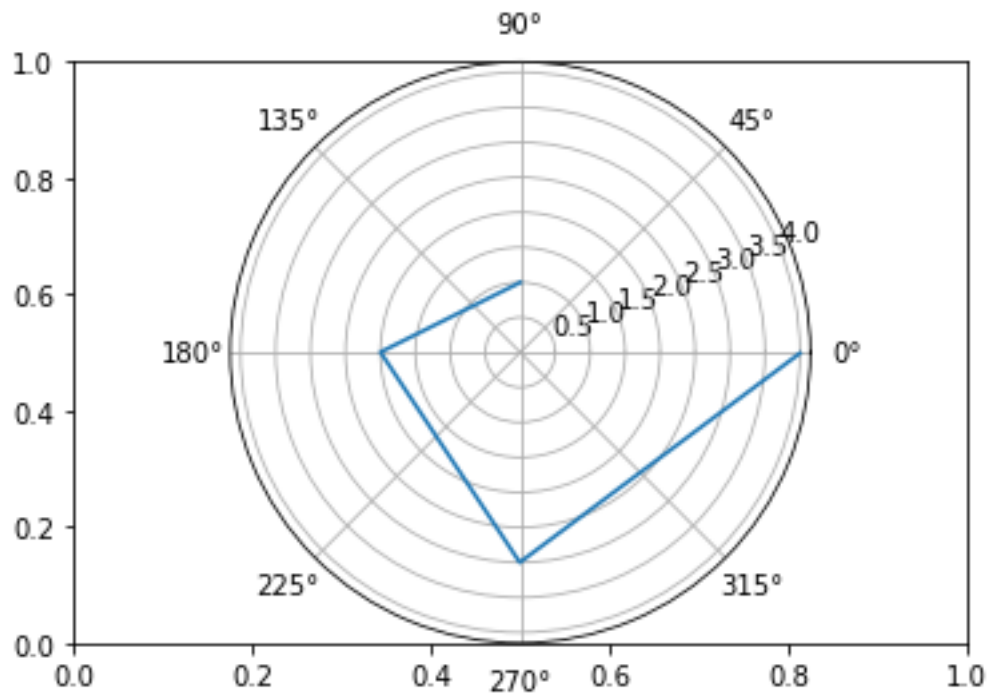
Polar plot

```
r = [1, 2, 3, 4]
t = [90, 180, 270, 360]
fig, ax =
plt.subplots()
plt.polar(t,r)
plt.show()
```



Polar plot

```
r = [1, 2, 3, 4]
t = [1.57, 3.14, 4.71,
     6.28]
fig, ax =
plt.subplots()
plt.polar(t,r)
plt.show()
```



Matplotlib + Pandas

Exemplo: leitura de dados cvs

```
df = pd.read_csv("dados_apartamentos.csv")
```

- O DataFrame tem muitas linhas de dados:
 - Se for muito grande, o Python exibe o início e o final, separados por ...
- Para visualizar sucintamente as primeiras linhas de um DataFrame existe o método `.head()`
- Similarmente o `.tail()` exibe por padrão as últimas 5 linhas do DataFrame.

Arquivo dados_apartamentos.csv

```
condominio,quartos,suites,vagas,area,bairro,preco,pm2
350,1,0.0,1.0,21,Botafogo,340000,16190.48
800,1,0.0,1.0,64,Botafogo,770000,12031.25
674,1,0.0,1.0,61,Botafogo,600000,9836.07
700,1,1.0,1.0,70,Botafogo,700000,10000.0
440,1,0.0,1.0,44,Botafogo,515000,11704.55
917,1,1.0,1.0,60,Botafogo,630000,10500.0
850,1,1.0,1.0,65,Botafogo,740000,11384.62
350,1,1.0,1.0,43,Botafogo,570000,13255.81
440,1,1.0,1.0,26,Botafogo,430000,16538.46
510,1,1.0,1.0,42,Botafogo,500000,11904.76
200,1,0.0,1.0,35,Botafogo,500000,14285.71
552,1,1.0,1.0,67,Botafogo,790000,11791.04
495,1,1.0,1.0,54,Botafogo,515000,9537.04
340,1,1.0,1.0,40,Botafogo,410000,10250.0
800,1,1.0,1.0,60,Botafogo,625000,10416.67
530,1,0.0,1.0,40,Botafogo,360000,9000.0
500,1,0.0,1.0,47,Botafogo,670000,14255.32
```

Saída do df

condominio	quartos	suites	vagas	area	bairro	preco	pm2	
0	350	1	0.0	1.0	21	Botafogo	340000	16190.48
1	800	1	0.0	1.0	64	Botafogo	770000	12031.25
2	674	1	0.0	1.0	61	Botafogo	600000	9836.07
3	700	1	1.0	1.0	70	Botafogo	700000	10000.00
4	440	1	0.0	1.0	44	Botafogo	515000	11704.55
...
1992	1080	3	1.0	1.0	80	Tijuca	680000	8500.00
1993	750	3	0.0	1.0	82	Tijuca	650000	7926.83
1994	700	3	1.0	1.0	100	Tijuca	629900	6299.00
1995	1850	3	1.0	2.0	166	Tijuca	1600000	9638.55
1996	800	3	1.0	1.0	107	Tijuca	540000	5046.73

[1997 rows x 8 columns]

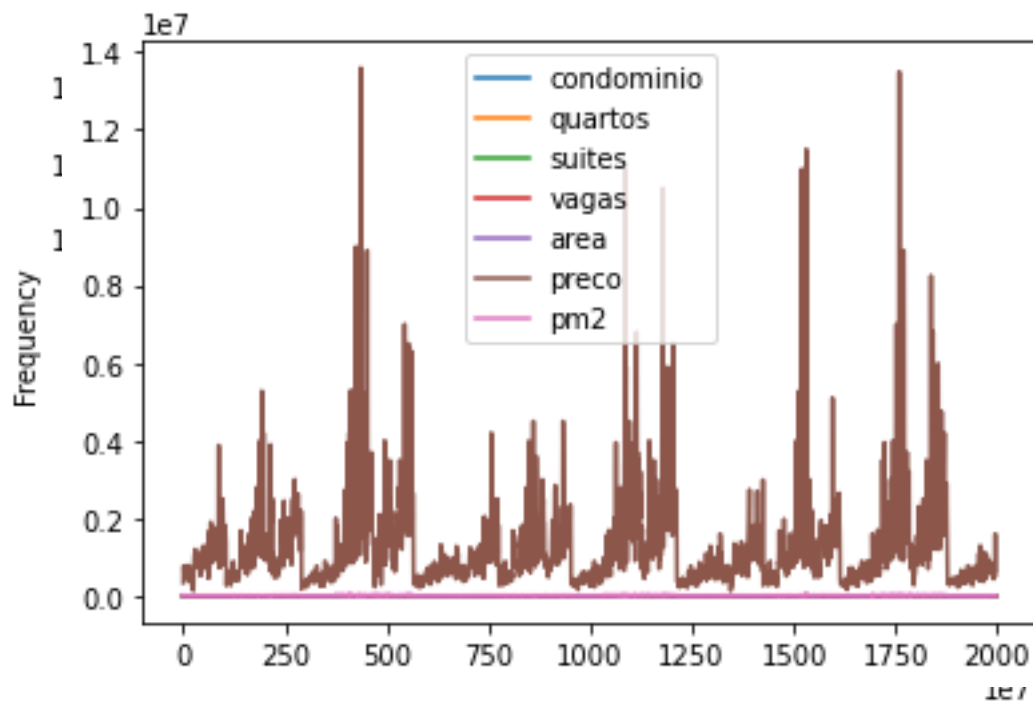
Visualização de Dados

- Os métodos de visualização do pandas são construídos com base no matplotlib para exploração rápida dos dados.
- Para se ter mais liberdade no conteúdo e possibilidades de visualização se recomenda usar diretamente o matplotlib ou ainda, para visualização estatística, o seaborn.

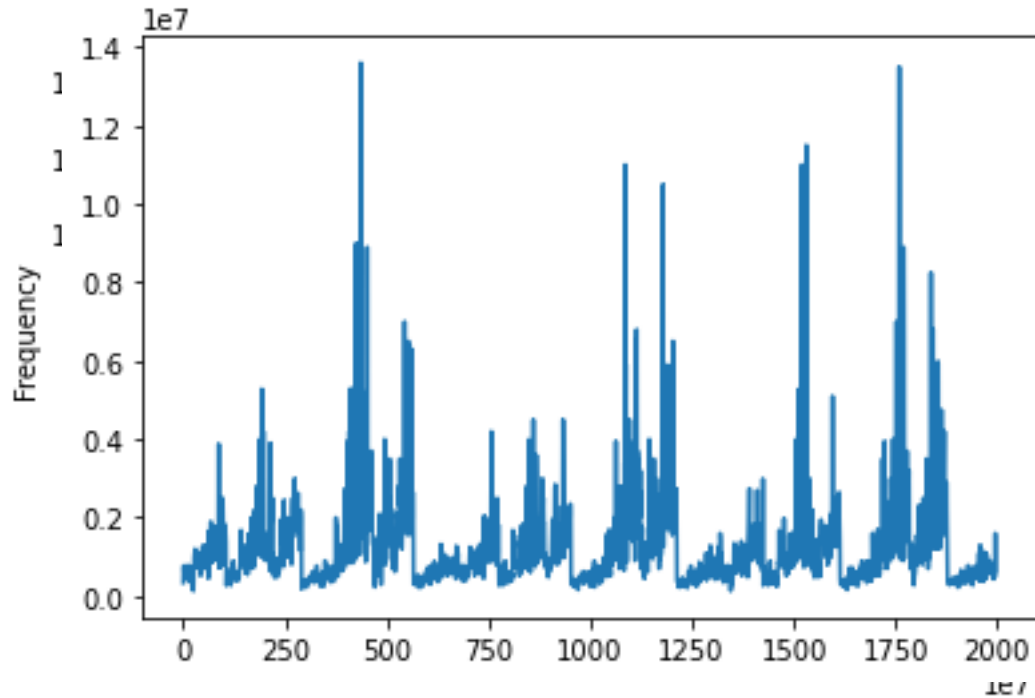
Visualização de Dados

- Tanto Series como DataFrame possuem um método `.plot()` que também pode ser encadeado para gerar visualização de diversos tipos, como histograma, área, pizza, dispersão, entre outros.
- Metodos `.hist()`, `.area()`, `.pie()` e `.scatter()`

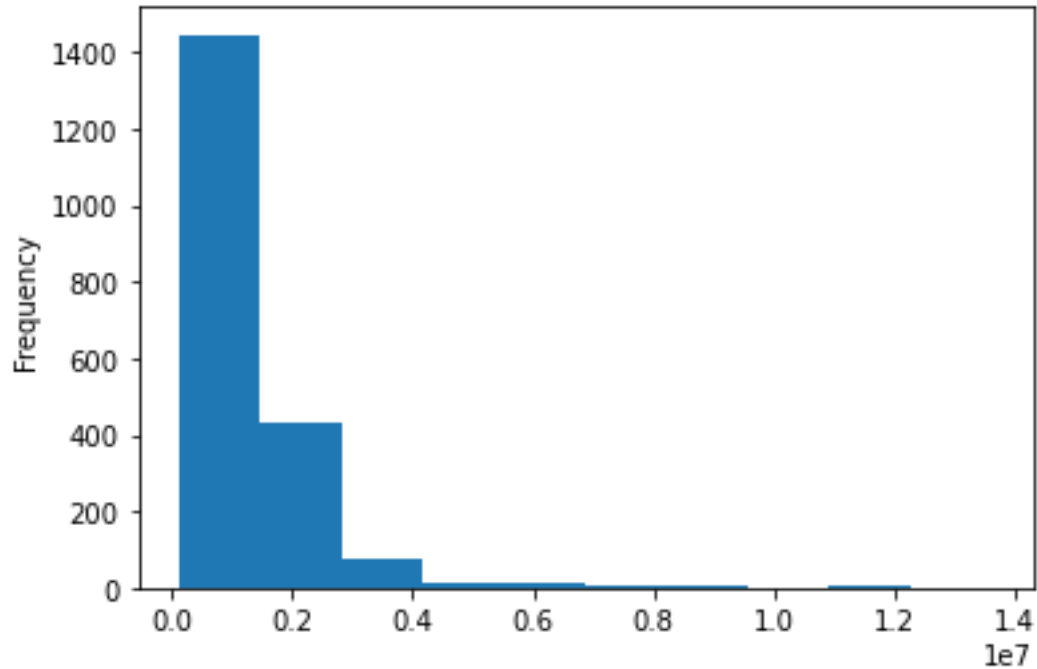
df.plot()



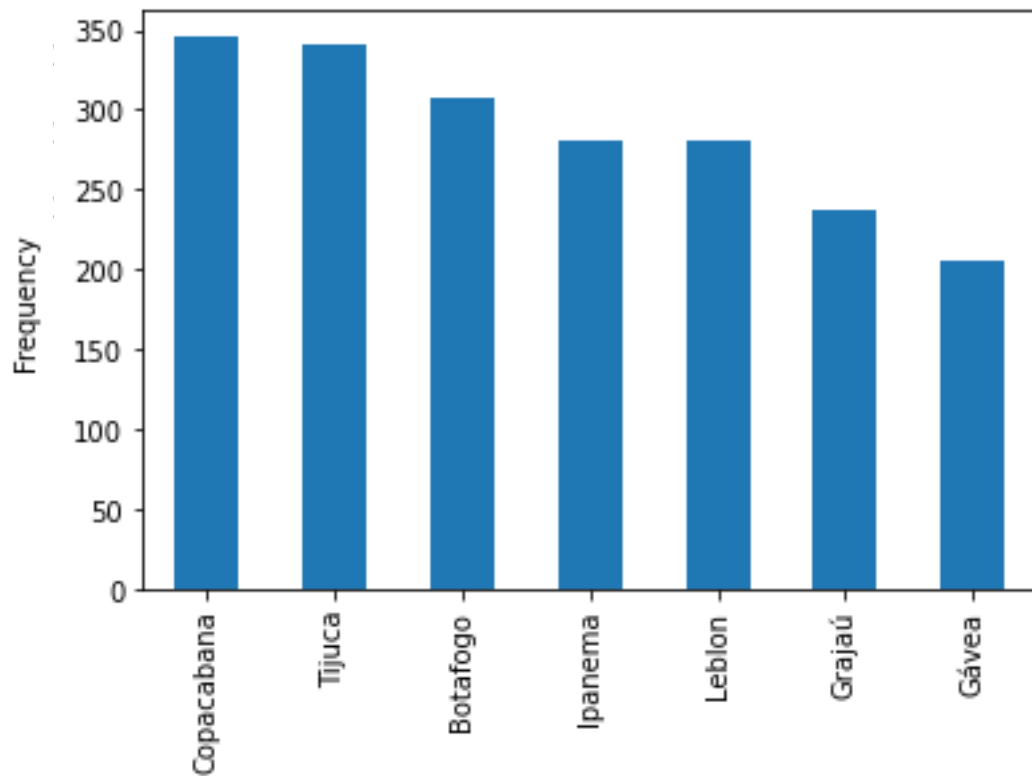
```
df["preco"].plot()
```



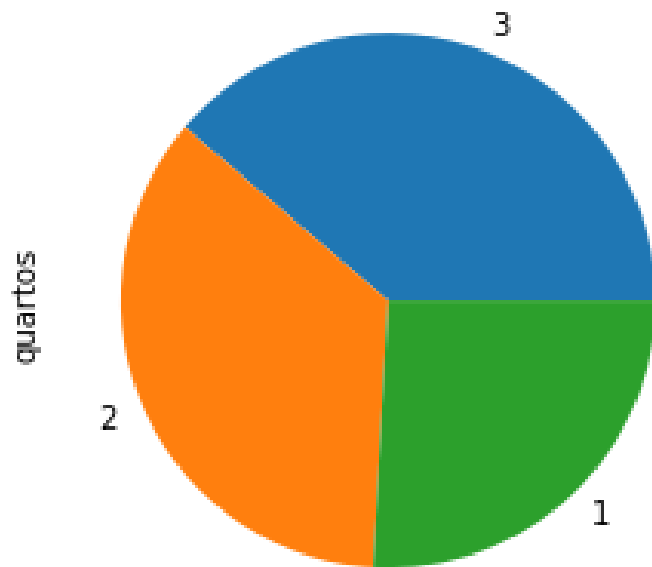

```
df["preco"].plot.hist()
```



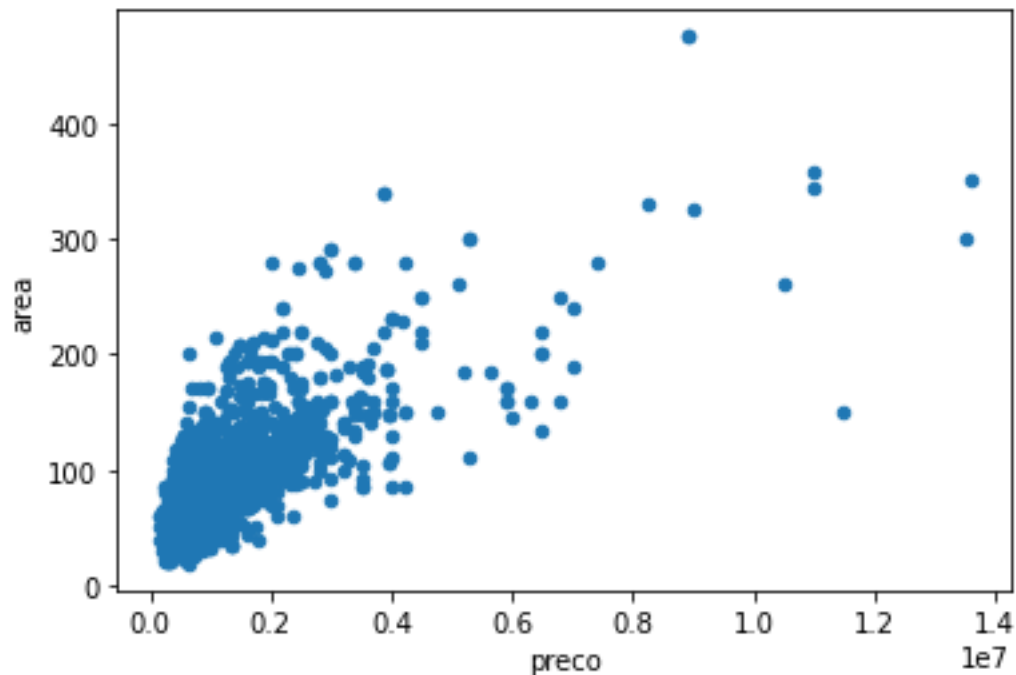
```
df["bairro"].value_counts().plot.bar()
```



```
df["quartos"].value_counts().plot.pie()
```



```
df.plot.scatter(x='preco', y='area')
```



Conclusão

Conclusão

- Foi realizada uma breve apresentação do Matplotlib.
- Elementos principais:
 - Plot, Axes, Axis,
- Quando usado junto com o numpy, sklearn e pandas, cobrem quase todos as necessidades do cientista de dados moderno...

Conclusão

- Nesta aula optou-se por usar os comandos baseados no Axes:

```
fig, ax = plt.subplots()  
ax.plot([1, 2, 3, 4], [1, 4, 2, 3])
```

- Mas quase tudo poderia ser feito usando diretamente o plot:

```
plt.plot([1, 2, 3, 4], [1, 4, 2, 3])
```

Fim

