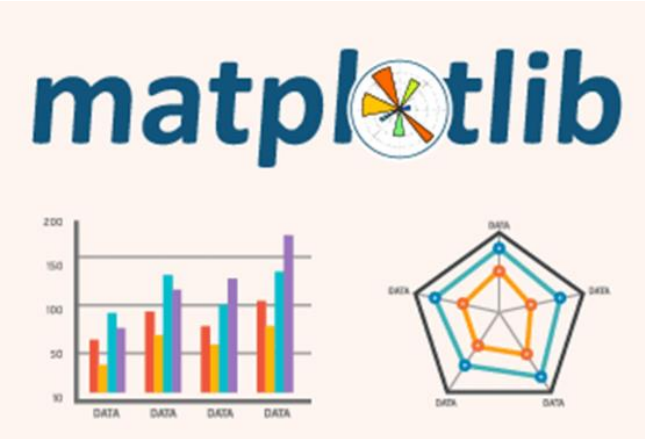
Biblioteca MatPlotLib

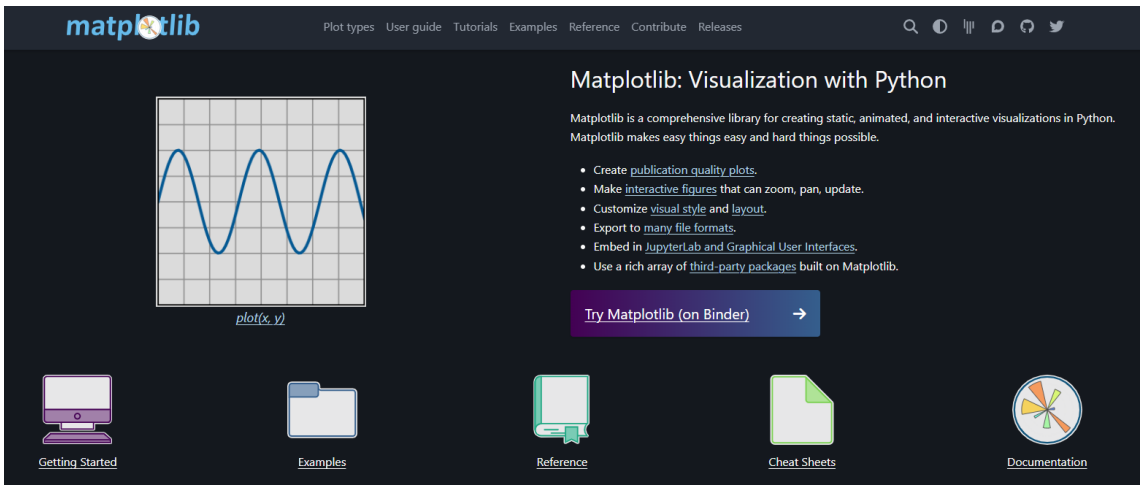
Introdução Matplotlib



Trata se de uma biblioteca **python**, onde podemos criar nossos **gráficos** para a ilustração melhor dos dados!!

Isso mesmo!! Após filtrar e limpar os dados com a biblioteca do **Pandas** juntamente com o **Numpy**, com o **Matplotlib** podemos pegar esses dados finais e gerar gráficos a partir dele!!

E para consultar cada **tipo** de **gráfico** que podemos gerar com o **matplotlib**, segue a sua documentação:



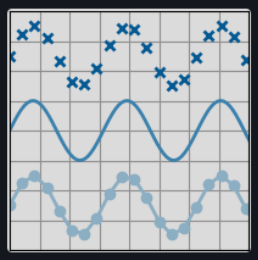
https://matplotlib.org/

Ainda sobre a documentação, podemos ver um tutorial bem simples de como podemos gerar um **gráfico**, como por exemplo um gráfico do tipo **PLOT**

plot(x, y)

Plot y versus x as lines and/or markers.

See `plot`.



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

plt.style.use('_mpl-gallery')

# make data
x = np.linspace(0, 10, 100)
y = 4 + 1 * np.sin(2 * x)
x2 = np.linspace(0, 10, 25)
y2 = 4 + 1 * np.sin(2 * x2)

# plot
fig, ax = plt.subplots()

ax.plot(x2, y2 + 2.5, 'x', markeredgewidth=2)
ax.plot(x, y, linewidth=2.0)
ax.plot(x2, y2 - 2.5, 'o-', linewidth=2)

ax.set(xlim=(0, 8), xticks=np.arange(1, 8),
       ylim=(0, 8), yticks=np.arange(1, 8))

plt.show()
```

Criando o primeiro gráfico

Primeiramente teremos que importar **todas** as **bibliotecas** necessárias, além do **MatPlotLib**, importaremos também o **Pandas** e claro o **Numpy**, até porque o **MatPlotLib** trabalha diretamente com arrays numéricos, e como já sabemos, os arrays do **numpy** são mais rápidos e eficientes!!

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

✓ 0.0s

Após realizar a importação das **bibliotecas**, vamos já importar o nosso arquivo de dados **data.csv** pela biblioteca **Pandas**!

```
df = pd.read_csv('data.csv')
df
```

✓ 0.0s Open 'df' in Data Wrangler

	# Year	# Australia	# Canada	# France	# Germany	# Italy
0	1990	*Missing value*	1.87	3.63	2.65	4.59
1	1991	1.96	1.92	3.45	2.9	4.5
2	1992	1.89	1.73	3.56	3.27	4.53
3	1993	1.73	1.57	3.41	3.07	3.68
4	1994	1.84	1.45	3.59	3.52	3.7
5	1995	1.95	1.53	4.26	3.96	4.0
6	1996	2.12	1.61	4.41	3.94	4.39
7	1997	2.05	1.62	4.0	3.53	4.07
8	1998	1.63	1.38	3.87	3.34	3.84
9	1999	1.72	1.52	3.85	3.42	3.87

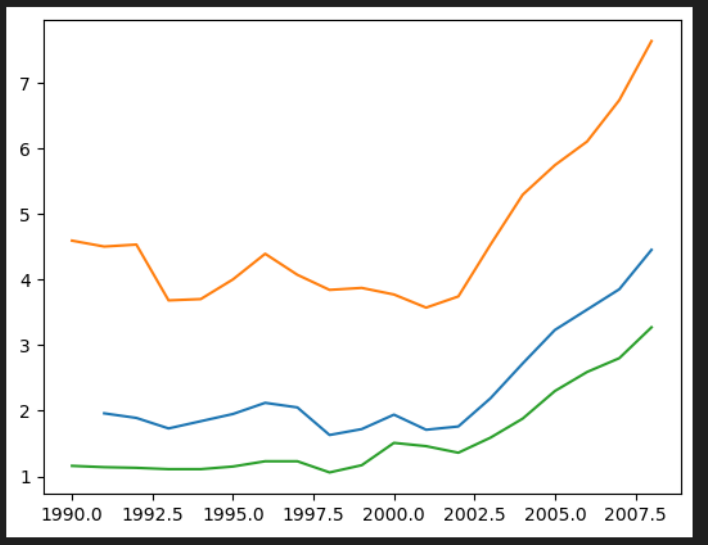
19 rows x 11 cols 10 per page Page 1 of 2

Agora sim vamos criar o nosso gráfico, onde será um gráfico bem simples do tipo **PLOT**, e para criar esse gráfico utilizamos a função **plot()** definindo os **eixos x** e **y**, onde no **eixo x** terá as informações referente aos **anos**, já no **eixo y** terá as informações do número de vendas referente para cada **país** realizado, que nesse caso será (**Australia**, **Italy**, **USA**)!

E após definir os eixos x e y pela função **plot()**, chamaremos a função **show()** para que seja exibido a construção desse **gráfico plot**!

```
plt.plot(df_gas['Year'], df_gas['Australia'])
plt.plot(df_gas['Year'], df_gas['Italy'])
plt.plot(df_gas['Year'], df_gas['USA'])

plt.show()
```

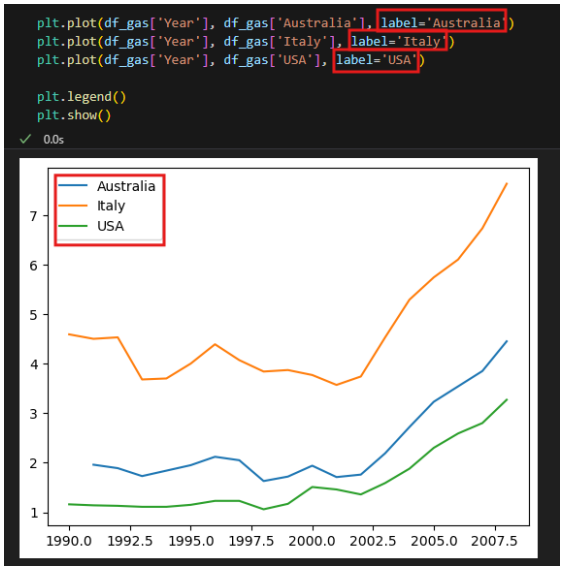
✓ 0.0s



Adicionando legendas e ticks

Porém como podemos notar, precisamos de mais detalhes para que o gráfico fique mais entendível, como por exemplo as **legendas** indicando o que seria cada traçado!

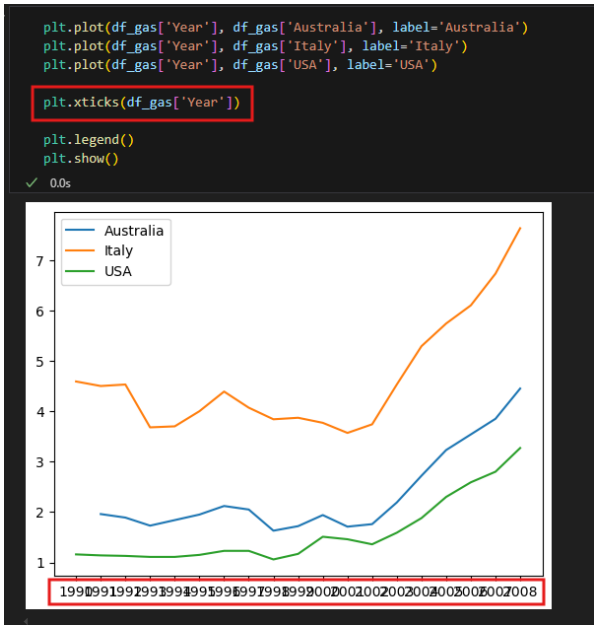
E para definir as legendas no gráfico é muito simples, basta definir uma **label** para cada **PLOT** do gráfico, e consequentemente chamar a função **legend()**, para que seja exibido na legenda as **labels** definidas para cada **plot** do gráfico!



Além da legenda, podemos melhorar a exibição dos números referente aos eixos x ou y, conhecido como **TICKS**!!

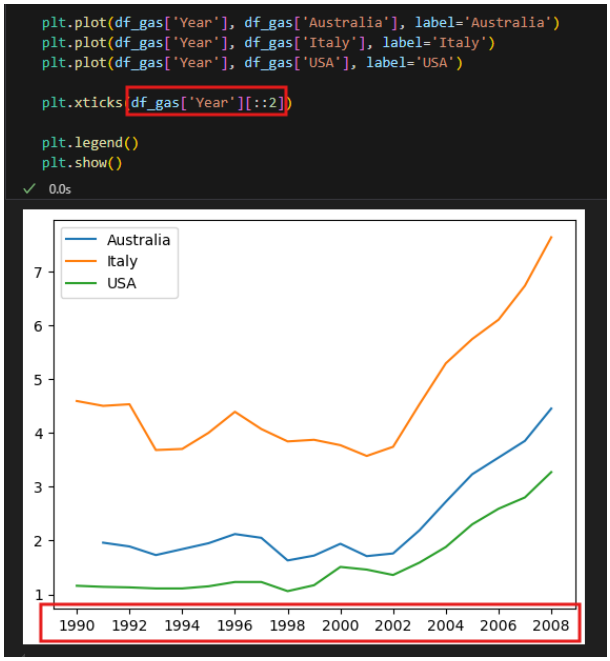
E no nosso caso iremos apenas alterar a exibição dos valores do **eixo x**, pois o padrão ficou meio confuso, pois se trata de números referente a **anos** e não possui **valores decimais**!!

Dito isso, vamos usar a função **xticks()** onde definimos o **array** dos valores que queremos que seja exibido no **eixo x**, que seria a própria listagem dos valores referente aos anos!! Dessa forma forcamos para que não seja mais exibido os valores resumidos por padrão, como está sendo feito anteriormente!!



Porém veja que pelo tamanho do gráfico, os números do eixo x ficaram muito grudados!! E para ajustar isso, podemos definir um tipo de "**step**", para que seja pulado um intervalo de valores, e que nesse caso definiremos um intervalo de 2 valores!!

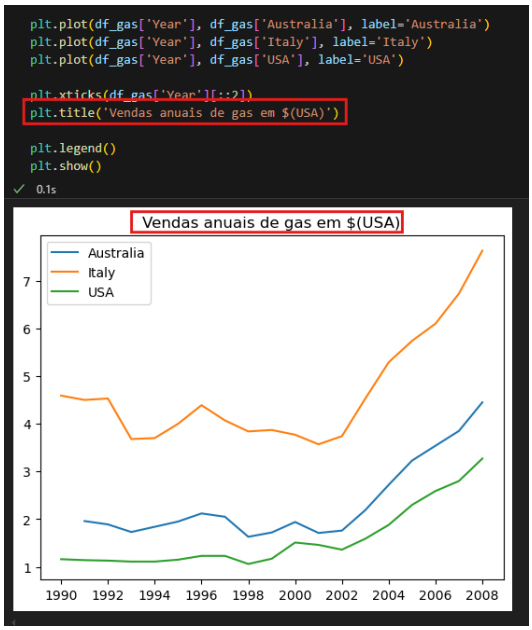
E para isso utilizaremos o recuso de **slice** nosso **array** do **eixo x**!!



Formatando o gráfico com label e cores

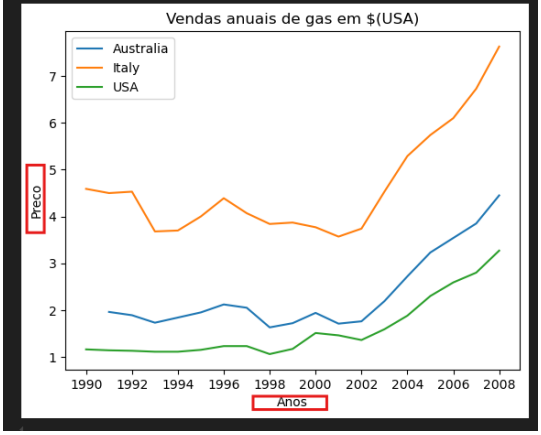
Podemos melhorar ainda mais a visualização do nosso gráfico. Por exemplo, podemos notar que não temos ideia o que se tratam esses valores do **eixo x** ou do **eixo y**!!

Dito isso, vamos realizar alguns ajustes, e para começar vamos colocar um título para o nosso gráfico, e saber o que se trata esse gráfico em si! E para isso, utilizaremos a função **title()**!!



E como foi dito antes, não temos ideia o que se tratam esses valores do **eixo x** ou do **eixo y**!! Sabendo isso vamos colocar labels em nossos eixos, logo usaremos as funções **xlabel()** e **ylabel()**.

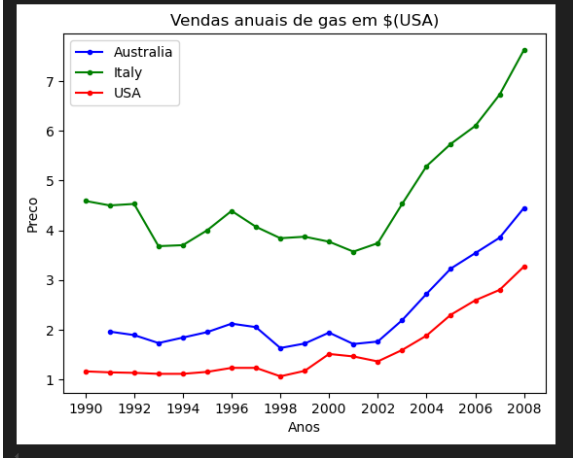
```
plt.plot(df_gas['Year'], df_gas['Australia'], label='Australia')
plt.plot(df_gas['Year'], df_gas['Italy'], label='Italy')
plt.plot(df_gas['Year'], df_gas['USA'], label='USA')

plt.xticks(df_gas['Year'][::2])

plt.title('Vendas anuais de gas em $(USA)')
plt.xlabel('Anos')
plt.ylabel('Preco')

plt.legend()
plt.show()
```



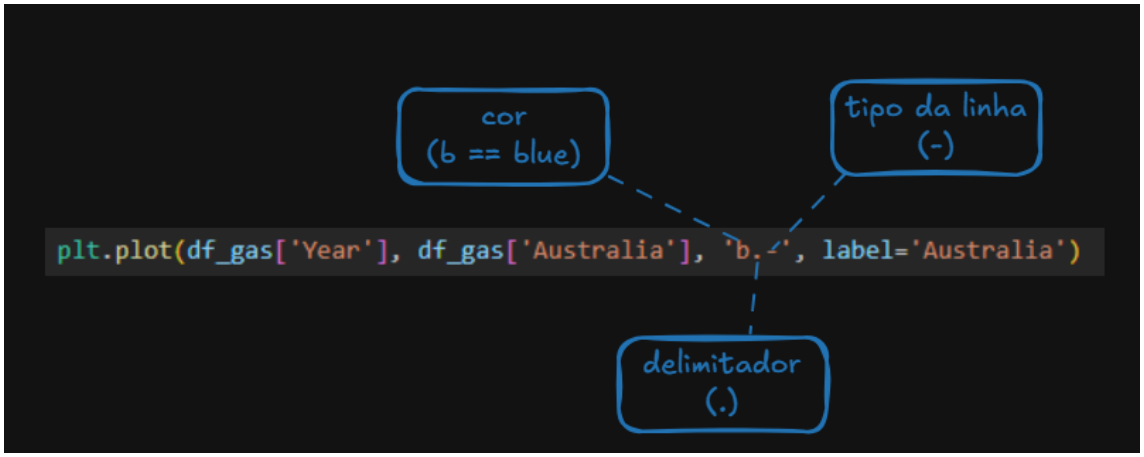
Bem melhor!! E para melhorar ainda mais essa formatação do nosso gráfico, podemos também **trocar as cores** e até mesmo o **tipo** das **linhas trançadas** de cada **plot** do gráfico!!

E para realizarmos isso da maneira mais simples, é através de mais um parâmetro da nossa função **plot()**, como podemos ver a seguir:

```
plt.plot(df_gas['Year'], df_gas['Australia'], 'b.-', label='Australia')
plt.plot(df_gas['Year'], df_gas['Italy'], 'g.-', label='Italy')
plt.plot(df_gas['Year'], df_gas['USA'], 'r.-', label='USA')

plt.xticks(df_gas['Year'][::2])

plt.title('Vendas anuais de gas em $(USA)')
plt.xlabel('Anos')
plt.ylabel('Preco')

plt.legend()
plt.show()
```



E para entender melhor como definimos as **cores**, o **delimitador** e o **tipo de linha**, segue a imagem a seguir com a explicação:

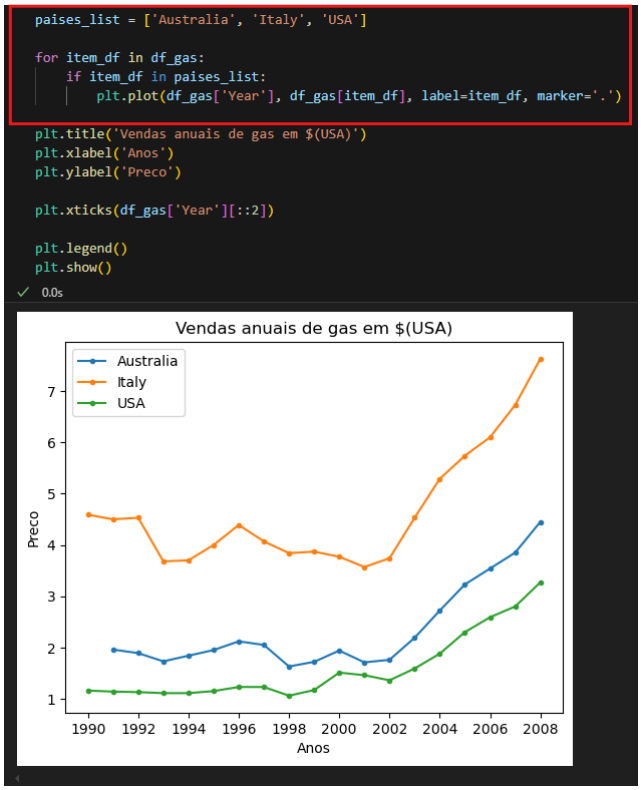


Trabalhando com o for loop

Vimos como trabalhar com cada **plot individualmente**, mas em muitas ocasiões isso não será possível!!

Pois precisaremos gerar o nosso gráfico a partir de uma consulta de dados, podendo gerar uma **lista de plots** dinamicamente!!

Dito isso, a utilização de **for loop**, ou até mesmo uma outra estrutura de laco de repetições, será de extrema importância!! Pois poderemos gerar os nossos **plots dinamicamente** de acordo com a listagem de dados obtida em nossos filtros, como podemos ver a seguir!!



Veja que definimos uma variável **paises_list**, onde definimos quais são os países que serão utilizados como **plots** em nosso gráfico!

Além disso utilizamos o **for loop** para que possamos interagir e definir os **plots dinamicamente**, passando os eixos correspondentes, onde o **eixo x** sempre será o mesmo **"ANO"** e o **eixo y** referente a coluna do **DataFrame** percorrido no **for loop**, que no caso será os valores de cada País em si!

Lembrando que temos várias formas de chegar nesse resultado, inclusive ao invés de iterar no **for loop** utilizando apenas o **nome da coluna**, podemos interagir também com o próprio **objeto Series** do **DataFrame**, utilizando a função **items()** do objeto **DataFrame**, ficando dessa forma:

```
paises_list = ['Australia', 'Italy', 'USA']

for column_gas, series_gas in df_gas.items():
    if column_gas in paises_list:
        plt.plot(df_gas['Year'], series_gas, label=column_gas, marker='.')
```

Salvando o gráfico gerado

Após enfim gerar o nosso gráfico do **MatPlotLib**, podemos **salvar/exportar** o gráfico em uma **imagem** por exemplo, para apresentar esse gráfico gerado para o usuário final!!

E para isso podemos consultar na própria **documentação** qual função podemos utilizar para isso, e iremos reparar que a função a ser utilizada será o **savefig()**, como podemos ver a seguir, onde salvaremos o gráfico como imagem no arquivo "**result_grafico.png**":

```
plt.savefig('result_grafico.png')
✓ 0.1s
```

Result_grafico.png

Porém ao abrir o arquivo da imagem veremos que o gráfico estará em branco!! E isso acontece porque ao utilizar a função **show()**, onde exibimos o gráfico em tela, o **MatPlotLib** acaba zerando esse gráfico!! Por isso que a imagem foi gerada em branco!!

Sabendo disso, vamos remover a função **show()** do nosso código, e após rodar novamente veremos a imagem gerando de fato o nosso gráfico como esperamos!!

```python
paises_list = ['Australia', 'Italy', 'USA']

for column_gas, series_gas in df_gas.items():
    if column_gas in paises_list:
        plt.plot(df_gas['Year'], series_gas, label=column_gas, marker='.')

plt.title('Vendas anuais de gas em $(USA)')
plt.xlabel('Anos')
plt.ylabel('Preco')

plt.xticks(df_gas['Year'][::2])

plt.legend()
# plt.show()

plt.savefig('result_grafico.png')
```

Result_grafico.png