FIEP SESI SENAI IEL

SENAI

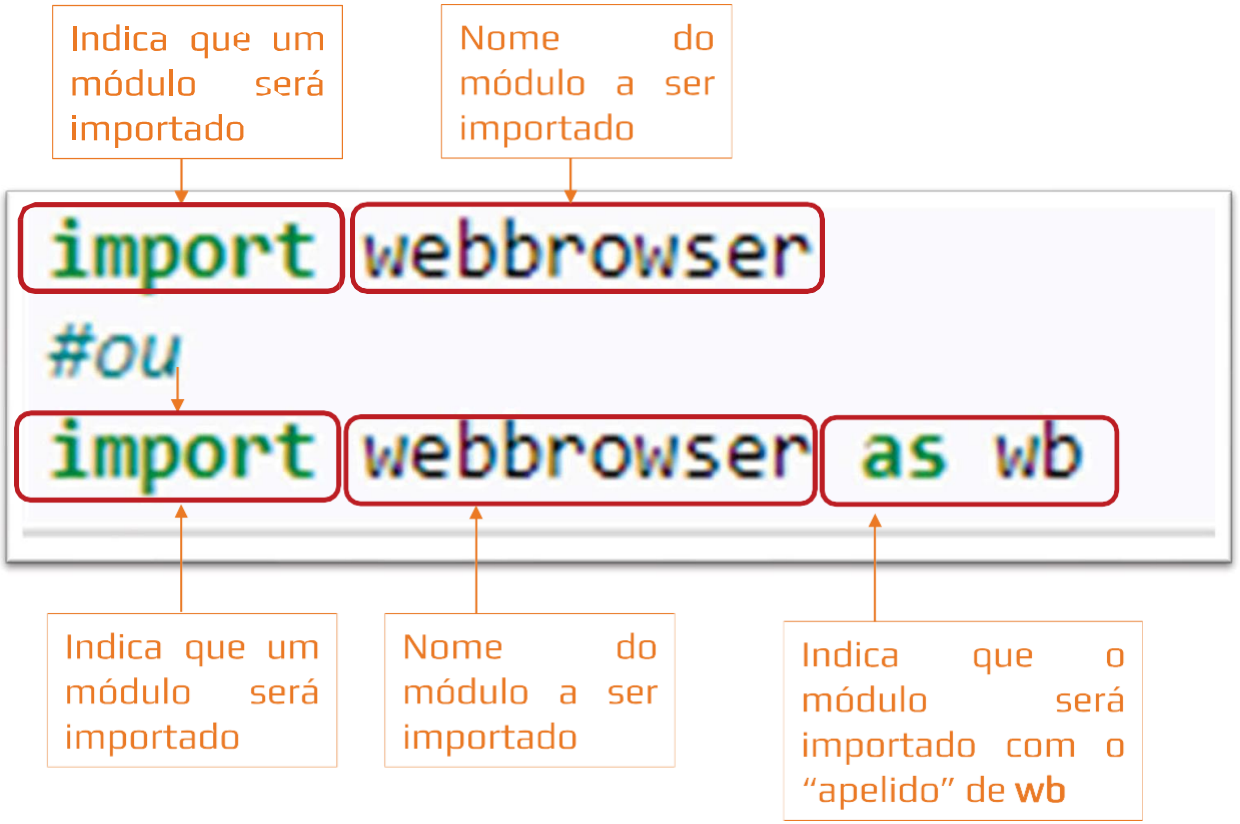
Módulos e Bibliotecas

Até agora só vimos os objetos e métodos que existem dentro do próprio Python, mas uma das principais “armas” dessa linguagem são as bibliotecas.

Bibliotecas são módulos prontos que podem ser importados para nossos projetos. Elas nos permitem executar códigos que levariam muito tempo para executar

Existem centenas de módulos disponíveis, ter todos eles integrados consumiria muita memória.

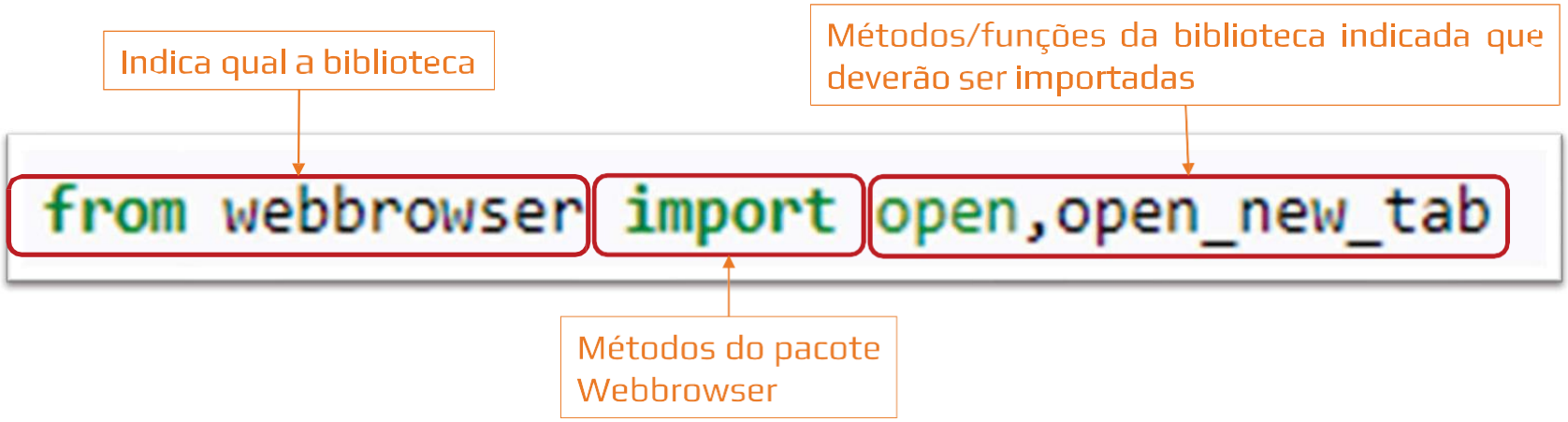
Por isso, sempre que precisarmos, vamos importar esses módulos dentro de nossos projetos usando o comando **import**



Em alguns casos não é interessante importar toda a biblioteca, mas só uma parte dela.

Para isso, usamos a estrutura abaixo:

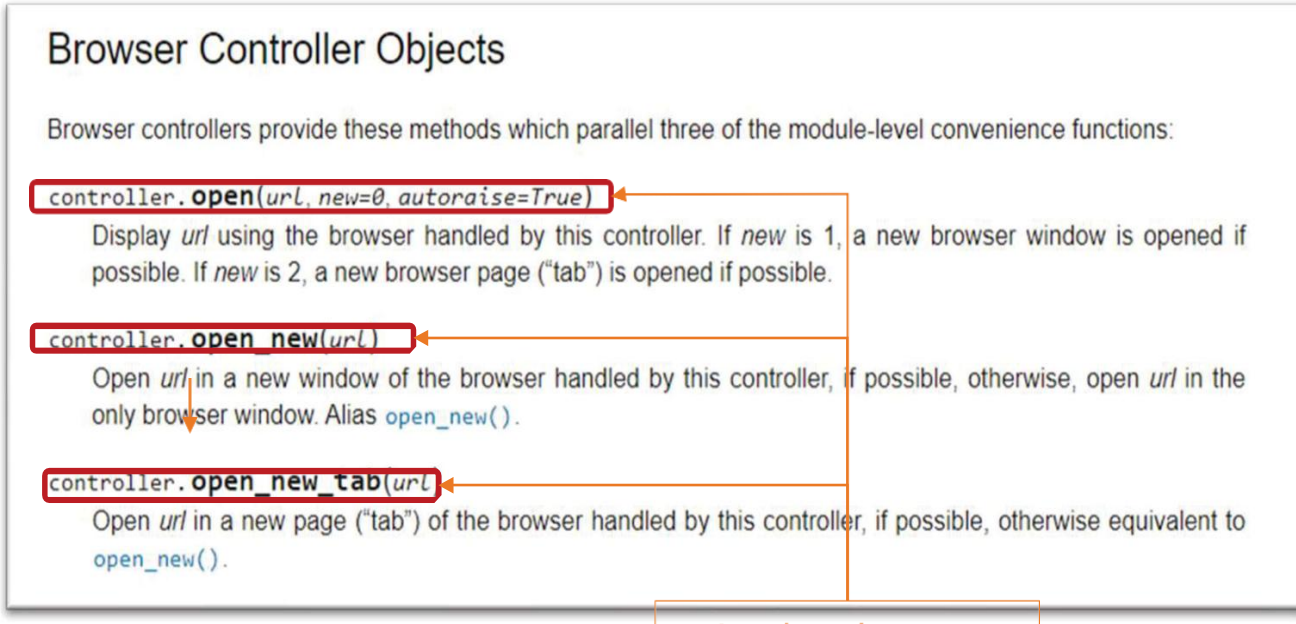
From biblioteca import módulo



Muito provavelmente você está se perguntando:

1)Mas o que faz o Webbrowser?
2)Como saber se o módulo que eu preciso é o Webbrowser?

Há milhares de bibliotecas no Python, é impossível apresentar todas. Essa, por exemplo, nos ajuda a integrar Python com Web, tendo funções como a de abrir páginas no navegador, por exemplo.



Métodos do pacote Webbrowser

Aqui, vamos apresentar as bibliotecas principais/mais famosas.

Sintam-se livres para pesquisar outras, contem com a comunidade global do Python e conosco para dar apoio :)

Vamos falar de um módulo bem recorrente em Python, o **time**

O módulo time nos ajuda a trabalhar com tempos e datas, mas não é o único que tem essa funcionalidade.

Vamos falar apenas dos métodos principais dessa função, pode pesquisar mais a fundo no link abaixo, se quiser: https://docs.python.org/3/library/time.html

```
1 import time
```

Vamos ver os principais métodos da biblioteca **time** :

- **EPOCH** – É o marco zero do Python. 01/01/1970;
- **time ()** – Nos retorna a diferença entre o tempo de hoje até a EPOCH em segundos
- **ctime()** – retorna uma **string** com a data no modelo UTC. UTC é um formato padrão utilizado pelo Python;
- **sleep()** – faz com que o python aguarde um tempo dado (em segundos) para executar a próxima linha de código;
- **gmtime()** – retorna as informações de data de forma detalhada;

Perceba que para usarmos os métodos sempre chamamos primeiramente a biblioteca.

Ou seja **time**.método

```
1 segundos_hoje = time.time()
2 print(segundos_hoje)
```

```
1602013004.6264832
```

```
1 #para esperar 5 segundos fazemos:
2 print('Começando')
3 time.sleep(5)
4 print('Rodou 5 segundos após')
```

```
Começando
Rodou 5 segundos após
```

```
1 data_hoje = time.ctime()
2 #ou entao data_hoje = time.ctime(time())
3 print(data_hoje)
```

```
Tue Oct  6 16:37:36 2020
```

Métodos

```
1 data_atual = time.gmtime()
2 print(data_atual)
```

```
time.struct_time(tm_year=2020, tm_mon=10, tm_mday=6, tm_hour=19, tm_min=42, tm_sec=38, tm_wday=1, tm_yday=280, tm_isdst=0)
```

Vamos para outra biblioteca, dessa vez para criação de gráficos. Uma das bibliotecas mais famosas é o **matplotlib.** Caso tenha mais interesse sobre a biblioteca, aqui está o link da documentação da biblioteca:

https://matplotlib.org/stable/contents.html#

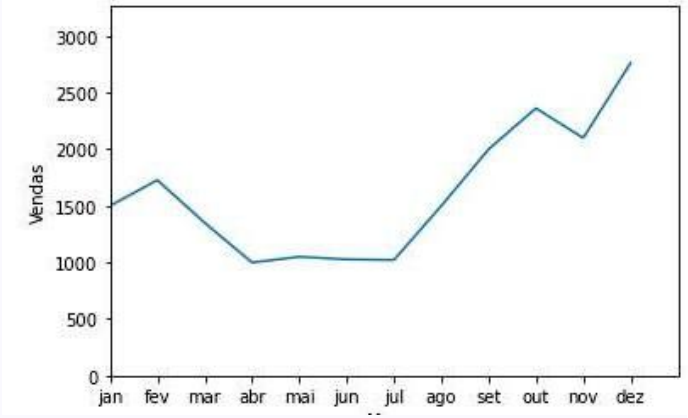
Vamos importar o **matplotlib** como **plt**. Para criarmos um gráfico basta usarmos os métodos abaixo:

- **.plot(eixox,eixoy)** – indica quais são os dados que formarão os gráfico;
- **.ylabel()** – Define o rótulo de dados do eixo **Y**;
- **.xlabel()** – Define o rótulo de dados do eixo **X;**
- **.axis()** – Define mínimo e máximo dos eixos X e Y. Nessa ordem.
- **.show()** – Plota o

gráfico

```
1  vendas_meses = [1500, 1727, 1350, 999, 1050, 1027, 1022, 1500, 2000, 2362, 2100, 2762]
2  meses = ['jan', 'fev', 'mar', 'abr', 'mai', 'jun', 'jul', 'ago', 'set', 'out', 'nov', 'dez']
3
4  #plotar o gráfico da forma mais simples
5  import matplotlib.pyplot as plt
6
7  plt.plot(meses, vendas_meses)
8  plt.ylabel('Vendas')
9  plt.xlabel('Meses')
10 plt.axis([0, 12, 0, max(vendas_meses)+500])
11 plt.show()
```

Métodos



tag

Vamos entender um pouco melhor como são usados este métodos.

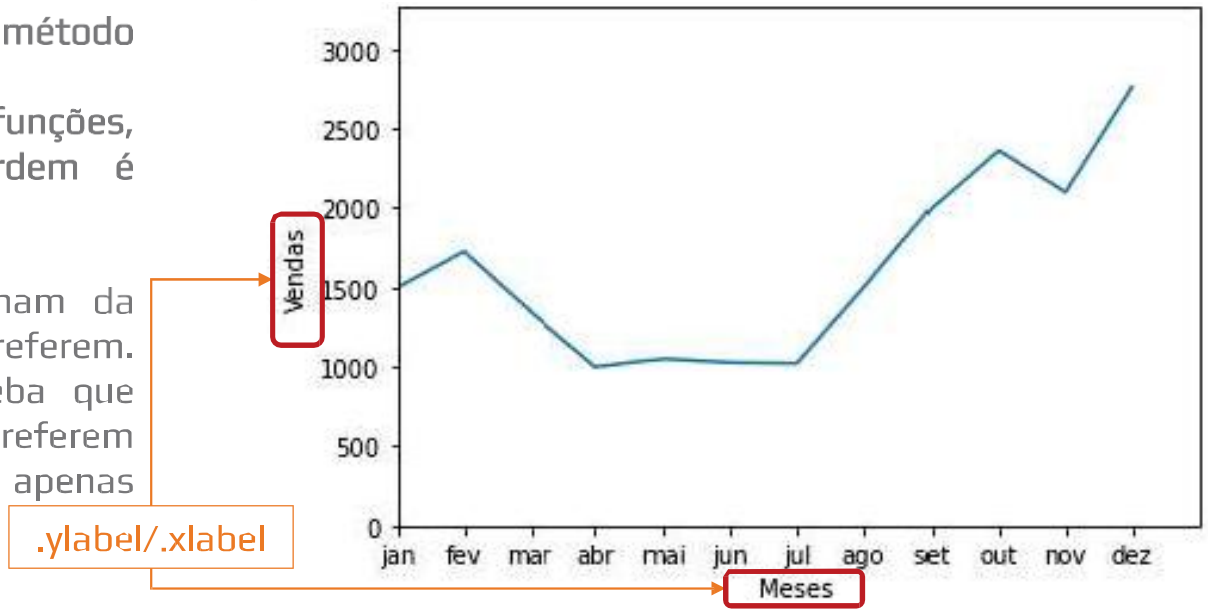
O método plot. Ao usarmos a ordem (meses, vendas_meses) , definimos que:

- **meses** estará no **eixo X;**
- **venda_meses** no **eixo y;**

Isso ocorre pois os argumentos deste método estabelece essa ordem.
Assim como vimos no módulo de funções, precisamos sempre lembrar que a ordem é importante.

Já os métodos **.ylabel** e **.xlabel** funcionam da mesma forma, só alterando o eixo a que se referem. Respectivamente eixo y e eixo x. Perceba que **'Vendas' e 'Meses', são STRINGS** e não se referem as variáveis meses e vendas_meses. São apenas rótulos do gráfico.

```
7  plt.plot(meses, vendas_meses)
8  plt.ylabel('Vendas')
9  plt.xlabel('Meses')
10 plt.axis([0, 12, 0, max(vendas_meses)+500])
11 plt.show()
```



O próximo método utilizado é o **.axis().** Este método nos permite definir os valores mínimos e máximo dos eixos x e y.

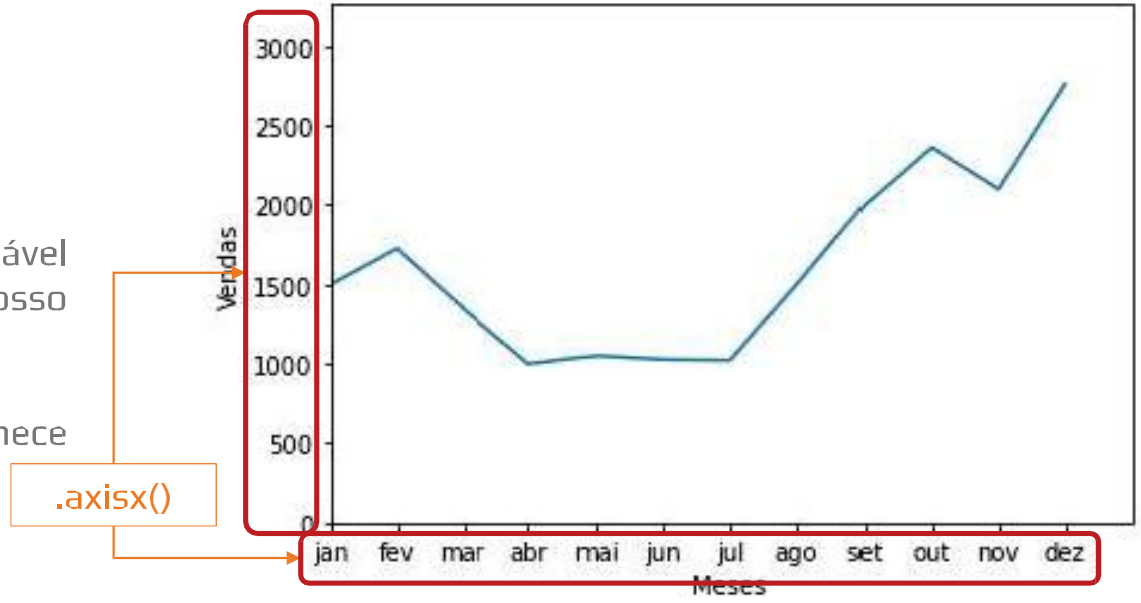
IMPORTANTE! Assim, como vimos no módulo de definição de funções, este método segue uma ordem específica.

1) Min eixo x;
2) Max eixo x;
3) Min eixo y;
4) Max eixo y.

Perceba que usamos a própria variável vendas_meses como parâmetro máximo do nosso eixo y ao fazermos **max(vendas_meses)+500.**

Por fim, usamos o método **show()** que nos fornece o gráfico definido por nós.

```
7  plt.plot(meses, vendas_meses)
8  plt.ylabel('Vendas')
9  plt.xlabel('Meses')
10 plt.axis([0, 12, 0, max(vendas_meses)+500])
11 plt.show()
```



Outro método muito comum, principalmente quando estamos tratando dados estatisticamente, é o numpy. Segue abaixo o link da documentação: https://numpy.org/doc/stable/user/tutorials_index.html

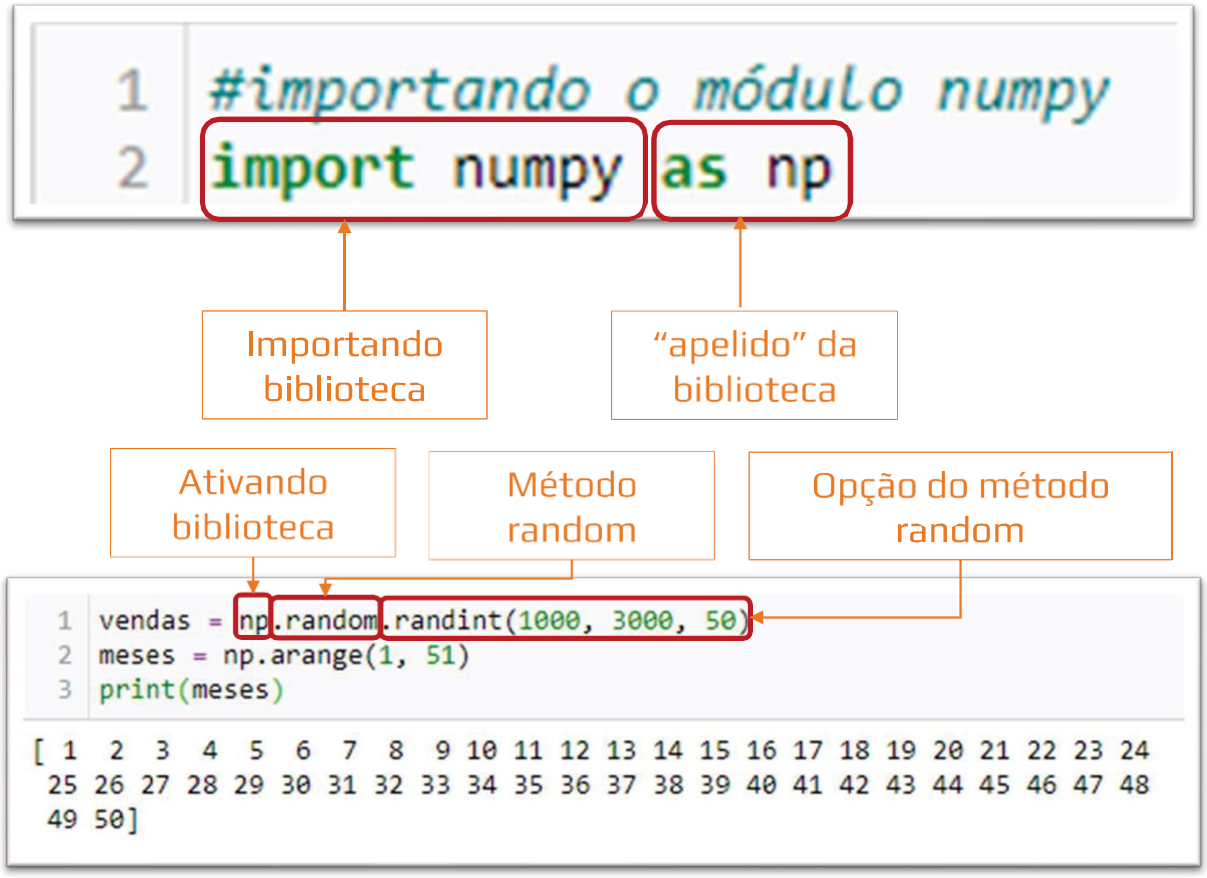
Vamos usar o Numpy juntamente do Matplotlib que vimos anteriormente.

Para importar o Numpy vamos usar o comando abaixo:

Import numpy as np

Aqui vamos replicar o exemplo anterior, mas utilizando o numpy para gerar números aleatórios de vendas (linha 1).

Já na linha 2 vamos usar o método **.arange()** para criação de 50 meses que coreesponderão as vendas geradas aleatoriamente.



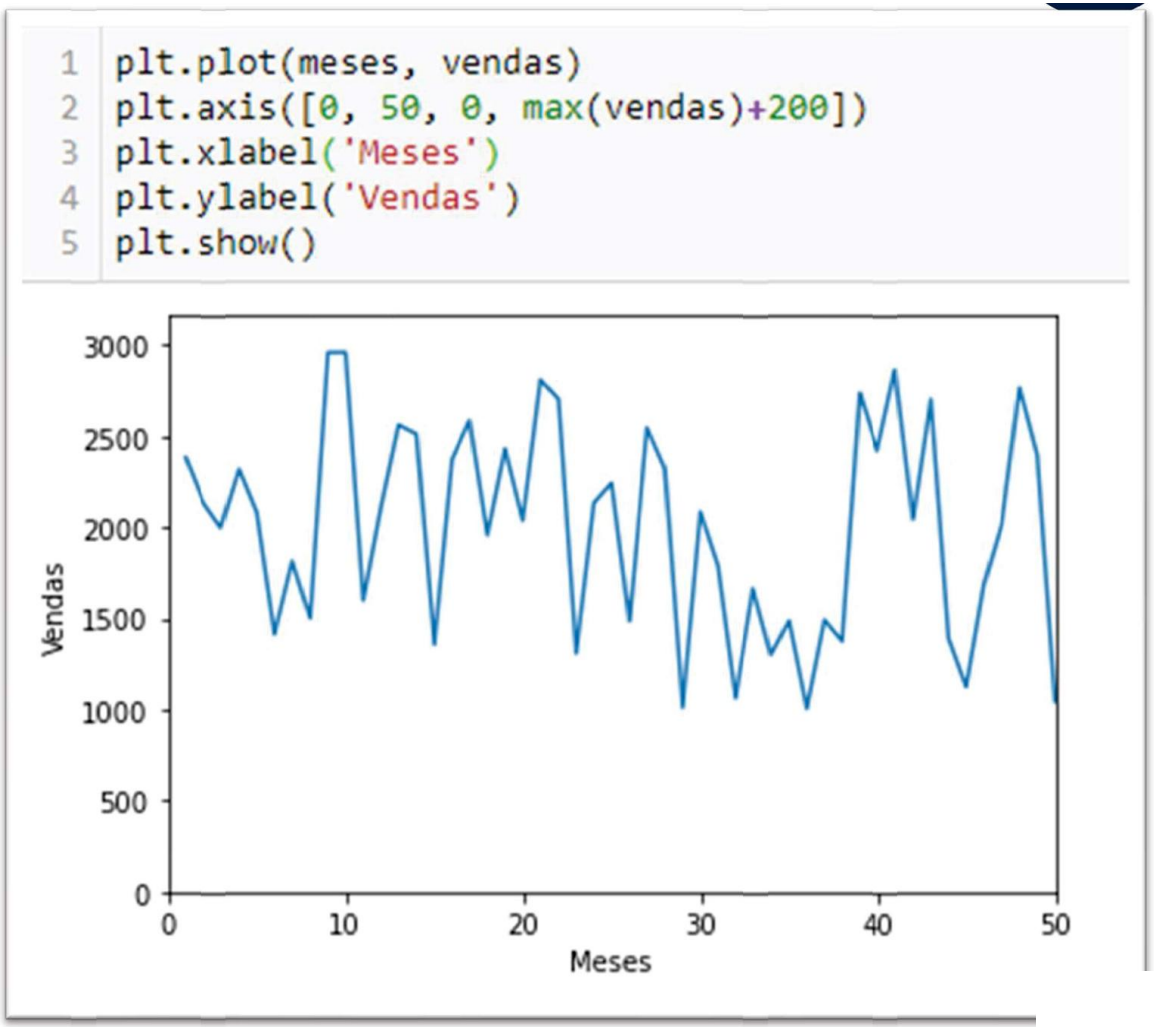
Vamos agora usar outras funcionalidades da nossa biblioteca matplotlib.

Usando a mesma linha de código do gráfico anterior temos este gráfico apresentado ao lado.

No entanto, conforme falamos, nossa biblioteca tem dezenas, centenas de funcionalidades que podemos usar para melhor a apresentação do gráfico.

É isso que vamos fazer nas próximas páginas.

```
1 plt.plot(meses, vendas)
2 plt.axis([0, 50, 0, max(vendas)+200])
3 plt.xlabel('Meses')
4 plt.ylabel('Vendas')
5 plt.show()
```



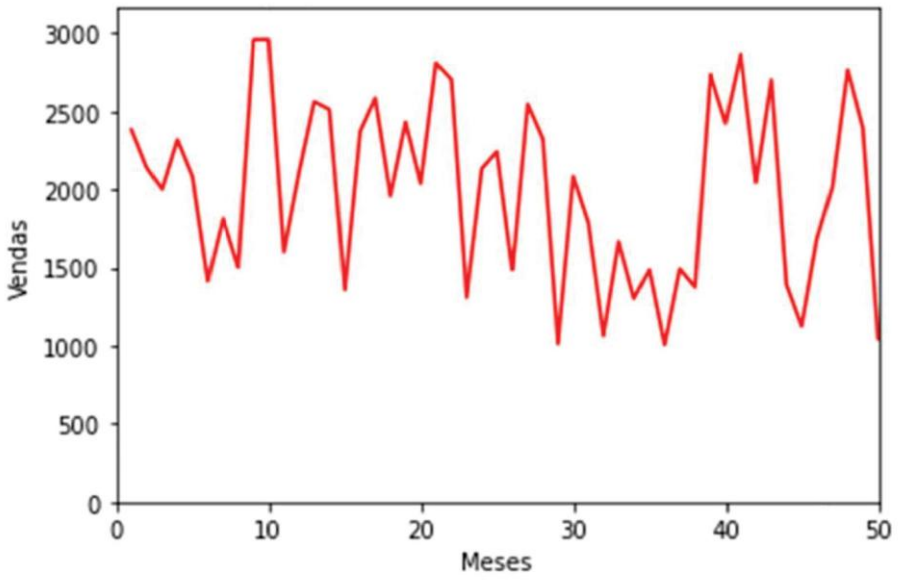
Começando pelo método **.plot()**, vamos alterar a cor da linha do gráfico usando a **keyword color =** .

Perceba que apenas acrescentando uma keyword nos argumentos podemos alterar a cor .

Lembre-se!! Ao fazermos isso, estamos alterando a cor padrão (azul). Muito importante também que só foi possível usar a palavra chave **RESPEITANDO** que este argumento só foi utilizado **APóS a definição dos eixos X e Y**.

```
1 #mudando a linha para apenas os marcadores
2 plt.plot(meses, vendas, color='red')
3 plt.axis([0, 50, 0, max(vendas)+200])
4 plt.xlabel('Meses')
5 plt.ylabel('Vendas')
6 plt.show()
```

Argumento keyword (kwarg)



Aqui são inúmeras as possibilidades de alteração.

Talvez você se pergunte:

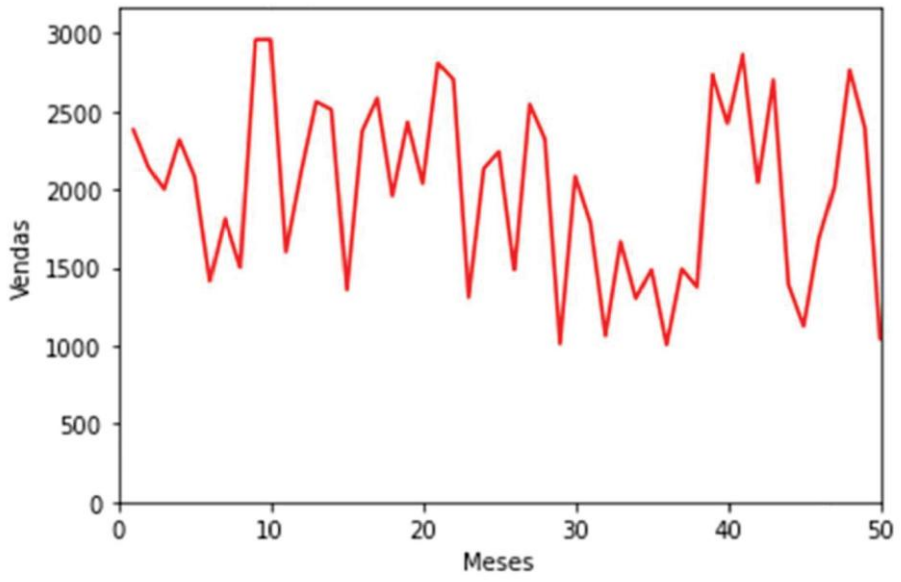
Preciso decorar isso tudo??

A resposta é NÃO! As documentações existem para serem consultadas. Obviamente quanto mais você exercitar, mais você saberá executar sem pesquisar mas **FAZ PARTE** do processo usar as documentações durante a programação.

Seja ela em Python ou qualquer linguagem.

```
1 #mudando a linha para apenas os marcadores
2 plt.plot(meses, vendas, color='red')
3 plt.axis([0, 50, 0, max(vendas)+200])
4 plt.xlabel('Meses')
5 plt.ylabel('Vendas')
6 plt.show()
```

Argumento keyword (kwarg)



Como temos diversos pacotes existentes, pode ser que durante uma pesquisa, um pacote que você achou interessante não está instalado no seu computador.

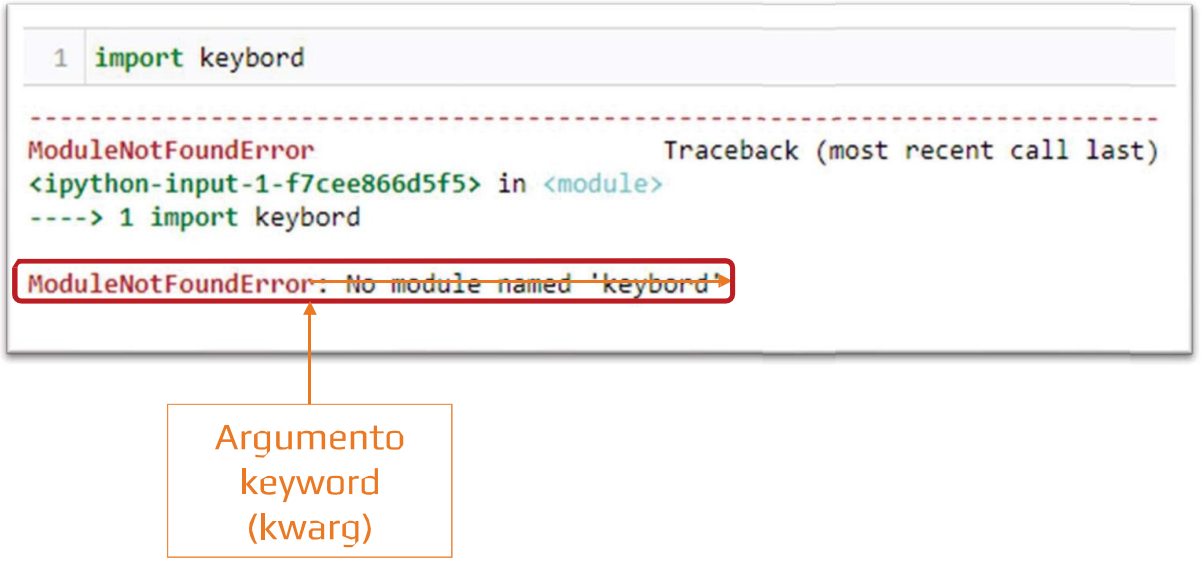
Como saber que a biblioteca não está instalada.

Vamos pegar o exemplo ao lado.

Ao tentar importar a biblioteca keyboard, recebi este erro ModuleNotFoundError.

Se você digitou corretamente o nome da biblioteca, isso significa que não temos isso instalado.

Para resolver esse problema, vamos uma um instalador que já existe "embutido" dentro do Python, o PIP.



Usando o **PIP,** é possível, instalar, desinstala pacotes.

Caso, você queira saber se um pacote está instalado, basta usar o comando abaixo em (esse comando é usado no prompt de comando)

pip freeze

Caso não encontre na lista o pacote desejado, use o comando abaixo para instalar:

pip install nome da biblioteca