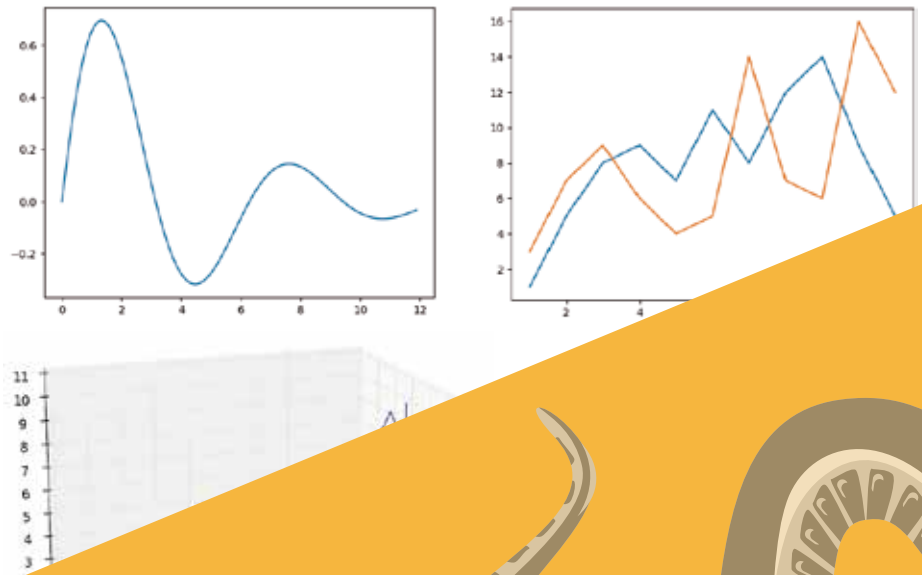


PLOTAGEM DE GRÁFICO DE LINHA USANDO PYTHON

Daniel da Silva Sotão

Abril de 2021

Amanara Plus https://www.projetoamanara.com/plus

QUEM SOMOS?

AMANARA Plus

A iniciativa que chamaremos de "Amanara plus" surge para contribuir com discentes e docentes da graduação em oceanografia e áreas afins. Iremos fazer chover conhecimento relacionados a aquisição e processamento de dados oceanográficos disponíveis gratuitamente na internet. Criaremos tutoriais de utilização de programas e linguagens de programação comumente utilizadas no mercado de trabalho e na academia, como por exemplo: Ocean Data View, Python, Matlab, R e QGIS.

Tentaremos facilitar ao máximo o aprendizado criando tutoriais em português. Mas lembre-se, se você não sabe nada de inglês corra, esta língua já não é um diferencial.

Visite o nosso site. https://www.projetoamanara.com

INTRODUÇÃO

A linguagem python é uma ferramenta muito utilizada no mundo todo. É usada para a construção de aplicações web, aprendizado de máquinas (machine learning), visualização de dados, ciência de dados e até mesmo criação de jogos digitais. Sua versatilidade e facilidade permitem que o programador escreva códigos rápidos e simples de serem compreendidos. Na área da ciência, o python é utilizado para o tratamento de dados maçantes que, para o ser humano, seria quase impossível de manipular.

Esse tutorial tem como objetivo lhe mostrar como plotar um gráfico simples utilizando linguagem python para que você possa, em sua jornada, desenvolver suas próprias visualizações de dados.

UTILIZANDO ARQUIVOS EM FORMATO CSV

Os arquivos CSV (do inglês "Character-separated values" ou "valores separados por um delimitador") servem para armazenar dados tabulares (números e texto) em texto simples. O "texto simples" significa que o arquivo é uma sequência de caracteres puros, sem qualquer informação escondida que o computador tenha que processar.

```
Date,Time,ms,LEVELm,TEMPERATUREoc
2012/10/02,00:00:00,0,4.002,28.60
2012/10/02,00:05:00,0,3.968,28.70
2012/10/02,00:10:00,0,3.990,28.70
2012/10/02,00:15:00,0,3.852,28.70
2012/10/02,00:20:00,0,3.886,28.70
2012/10/02,00:25:00,0,3.814,28.50
```

Os dados de um arquivo CSV possuem essa forma – repare que os parâmetros são divididos por um delimitador (uma vírgula).

ACESSANDO E VISUALIZANDO DADOS ARMAZENADOS EM UM ARQUIVO CSV

Para acessar e visualizar um arquivo CSV será necessário usar o módulo csv de Python para processar os dados. Então usaremos o módulo matplotlib para gerar um gráfico com base nos dados contidos no arquivo CSV.

Os arquivos CSV podem ser complicados para os seres humanos lerem, mas são fáceis para os programas processarem e extraírem valores, o que agiliza a operação de análise de dados. Começaremos com um pequeno conjunto de dados oceanográficos formatados em CSV. Nosso arquivo de exemplo, ao qual acompanha este tutorial é proveniente de um sensor que mede temperatura e pressão da coluna de água. Ao baixar o dado, o software do equipamento automaticamente converte os dados de pressão em profundidade da coluna de água acima dele (em metros). Com isso, é possível obter a variação da maré ao longo do tempo. Neste caso específico o sensor obteve um dado a cada 5 minutos durante aproximadamente 5 dias.

Escreva o código a seguir no seu editor de código:

```
a  import csv

   arquivo_endereco = 'C:\\Users\\Documents\\Python\\Data_Analysis\\Dados_Mare.csv'

b  with open(arquivo_endereço) as arquivo:
c      ler = csv.reader(arquivo)
d      linha_cabecalho = next(ler)
e      print(linha_cabecalho)
```

Depois de importar o módulo csv, armazenamos o nome do arquivo que estamos trabalhando em arquivo_endereco. Então abrimos o arquivo e armazenamos o objeto arquivo resultante em arquivo (b). Em seguida, chamamos csv.reader() e lhe passamos o objeto arquivo como argumento a fim de criar um objeto reader associado a esse arquivo (c). Armazenamos o resultado de csv.reader() na variável ler.

O módulo csv contém uma função chamada next(), que devolve a próxima linha do arquivo quando recebe a variável ler. Na listagem anterior, chamamos next() apenas uma vez para obter a primeira linha do arquivo, que contém os cabeçalhos (d). Armazenamos os dados devolvidos em linha_cabecalho. Ao rodar o programa, podemos ver que linha_cabecalho nos devolve uma lista que contém cabeçalhos significativos relacionados a dados oceanográficos, que nos informam quais dados estão armazenados em cada linha:

['Date', 'Time', 'ms', 'LEVELm', 'TEMPERATUREoc']

arquivo_endereco é o nome da variável que contém o arquivo CSV. Note que é preciso colocar o endereço da localização do arquivo CSV. No caso, está contido em 'C:\\Users\\Documents\\Python\\Data_Analysis\\Dados_Mare.csv'

EXIBINDO O CABEÇALHO E SUAS POSIÇÕES

Para facilitar a compreensão dos dados de cabeçalho do arquivo, exiba cada cabeçalho e a sua posição na lista:

```
import csv

arquivo_endereco = 'C:\\Users\\Documents\\Python\\Data_Analysis\\Dados_Mare.csv'

with open(arquivo_endereço) as arquivo:
    ler = csv.reader(arquivo)
    linha_cabecalho = next(ler)
a   # print(linha_cabecalho)


 for indice, coluna_cabecalho in enumerate(linha_cabecalho):
     print(indice, coluna_cabecalho)
```

O laço for possui as variáveis indice e coluna_cabecalho. Usamos enumerate() na lista para obter o índice de cada item, assim como o valor. (Observe que comentamos o print() em 'a' em troca dessa versão mais detalhada.)

Eis a saída mostrando o índice de cada cabeçalho:

```
0 Date
1 Time
2 ms
3 LEVELm
4 TEMPERATUREoc
```

Nessa saída vemos que as datas, horas e temperaturas estão armazenadas nas colunas 0, 1 e 4 respectivamente. Para explorar esses dados, processaremos cada linha do arquivo 'Dados_Mare.-csv' e extrairemos os valores nos índices 0, 1 e 4.

EXTRAINDO E LENDO DADOS

Agora que sabemos quais são as colunas de dados que precisaremos, vamos ler alguns desses dados. Em primeiro lugar, vamos ler a temperatura de cada hora e dia:

```
a   lista_temperaturas = []

    for linha in ler:
b       temperatura = float(linha[4])
c       lista_temperaturas.append(temperatura)

d   print(lista_temperaturas)
```

Criamos uma lista vazia chamada lista temperaturas e então percorremos as linhas restantes do arquivo com um laço for. O objeto reader continua a partir de onde parou no arquivo CSV e devolve automaticamente cada linha após a sua posição atual. Como já lemos a linha com o cabeçalho, o laço começará na segunda linha, em que os dados propriamente ditos têm início. A variável temperatura armazena os valores da temperatura contidos no índice 4 em forma de ponto flutuante (float) (b). A cada passagem pelo laço for salvamos os dados do índice 4 – a última coluna – em lista_temperaturas (c).

Ao rodar o programa, será mostrado a lista lista_temperaturas e seus dados armazenados :

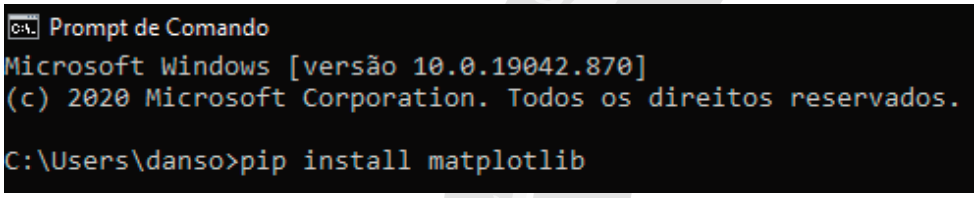
[28.6, 28.7, 28.7, 28.7, 28.7, 28.5, 28.7, 28.5, 28.4, 28.4, 28.4, 28.5, 28.5, 28.5, 28.5, 28.5, 28.5, 28.5, 28.5, 28.5, 28.4, 28.5, 28.5, 28.5, 28.6, 28.6, 28.6, 28.6, 28.6, 28.6, 28.6, 28.6, 28.6, 28.6, 28.6, 28.5, 28.5, 28.6, 28.6, 28.6, 28.6, 28.6, 28.6, 28.6, 28.6, 28.6, 28.5, 28.5, 28.6, 28.5, 28.5, 28.5, 28.5, 28.5 ..... 28.7, 28.7, 28.7, 28.7, 28.7, 28.7, 28.7, 28.7, 28.6]

Agora que você já sabe como visualizar um arquivo CSV, vamos criar listas para os outros parâmetros e visualizar esses dados em um gráfico.

MÃO NA MASSA

INSTALANDO O MATPLOTLIB

Para instalar o matplotlib é bastante simples. No prompt de comando digite “pip install matplotlib” (sem aspas) e pressione Enter.



Replique o código a seguir:

```
    import csv
    from matplotlib import pyplot as plt
    from datetime import datetime

a   local_arquivo = 'C:\\Users\\Documents\\Python\\Data_Analysis\\'
    arquivo_endereco = 'C:\\Users\\Documents\\Python\\Data_Analysis\\Dados_Mare.csv'

    with open(arquivo_endereco) as arquivo:
        ler = csv.reader(arquivo_endereco)
        linha_cabecalho = next(ler)

b       lista_temperaturas, lista_datas, lista_niveis = [], [], []
```

```
        for linha in ler:
c           data_atual = linha[0] + ' ' + linha[1]
            nivel = float(linha[3])
            temperatura = float(linha[4])
d           data = datetime.strptime(data_atual, '%Y/%m/%d %H:%M:%S')
            lista_datas.append(data)
            lista_temperaturas.append(temperatura)
            lista_niveis.append(nivel)

e figura = plt.figure(dpi=128, figsize=(10, 6))
f plt.plot(lista_datas, lista_temperaturas, color='red', linewidth=0.8)
g plt.title("Temperatura entre 04 e 05 de Outubro")
h plt.xlabel('Data e hora', fontsize=16)
i figura.autofmt_xdate()
j plt.ylabel('Temperatura (°C)', fontsize=16)
k plt.tick_params(axis='both', labelsize=8)
l plt.savefig(local_arquivo+'temperatura')

    figura = plt.figure(dpi=128, figsize=(10, 6))
    plt.plot(lista_datas, lista_niveis, color='blue', linewidth=0.8)
    plt.title("Nível do mar entre 04 e 05 de Outubro")
    plt.xlabel('Data e hora', fontsize=16)
    figura.autofmt_xdate()
    plt.ylabel('Nível do mar (m)', fontsize=16)
    plt.tick_params(axis='both', labelsize=8)
    plt.savefig(local_arquivo +'nível')
```

ENTENDENDO O CÓDIGO

No início do código acrescentamos dois novos módulos: matplotlib e datetime.

O módulo matplotlib é responsável por criar a visualização dos dados por meio de gráficos, enquanto que o módulo datetime é responsável por transformar uma String de data para um objeto do tipo datetime.

Em "a" criamos uma variável com o endereço para salvarmos os gráficos gerados.

Em "b" criamos mais duas listas vazias além de lista_temperaturas e usaremos a mesma lógica que usamos anteriormente.

Em data_atual (c) nós concatenamos o valor contido em linha[0] (data) com o valor contido em linha[1] (hora) e acrescentamos um espaço em formato de String " ". Fizemos isso para poder transformar essa String única em um objeto datetime.

Vamos adicionar as datas em nosso gráfico para deixá-lo mais útil. A data e hora do arquivo está na segunda linha do arquivo. Os dados serão lidos na forma de String, portanto precisamos de um modo de converter a String data e hora em um objeto que represente essa data. Podemos construir um objeto que represente a data e hora usando o método strptime() do módulo datetime.

Em data (d) usamos o método datetime.strptime(), inserimos a string data_atual e em seguida dizemos o formato que queremos '%Y/%m/%d %H:%M:%S'. A tabela a seguir mostra alguns dos argumentos para formatação da data e hora.

Argumentos	Significado
%A	Nome do dia da semana, por exemplo, Monday
%B	Nome do mês, por exemplo, January
%m	Mês, como um número (de 01 a 12)
%d	Dia do mês, como um número (de 01 a 31)
%Y	Ano com quatro dígitos, por exemplo, 2021
%y	Ano com dois dígitos, por exemplo, 98
%H	Hora, no formato 24 horas (de 00 a 23)
%I	Hora, no formato de 12 horas (de 01 a 12)
%p	AM ou PM
%M	Minutos (de 00 a 59)
%S	Segundos (de 00 a 61)

Após convertermos data em um objeto do tipo datetime, inserimos em lista_datas.

Saindo do laço for, o código passa para a próxima etapa, as configurações para plotar o gráfico. Começando pela linha "e" onde configuramos algumas características do gráfico como a resolução (dpi=128) e o formato da figura (figsize(10, 6)).

Em "f" que começa a criação do gráfico. Usamos o método plot() do módulo matplotlib e passamos os argumentos necessários. O método plot() requer os valores para o eixo X e em seguida para o eixo Y (plot(x, y)). Além disso, passamos uns argumentos a mais como a cor da linha do gráfico (color='red') e a grossura da linha (linewidth=0.8).

Em plt.title() (g) passamos o título do nosso gráfico.

Em plt.xlabel() (h) passamos o título do eixo X. Da mesma forma, em plt.ylabel() (j) passamos o título do eixo Y. Note que passamos alguns argumentos opcionais para estilização dos eixos como color e fontsize.

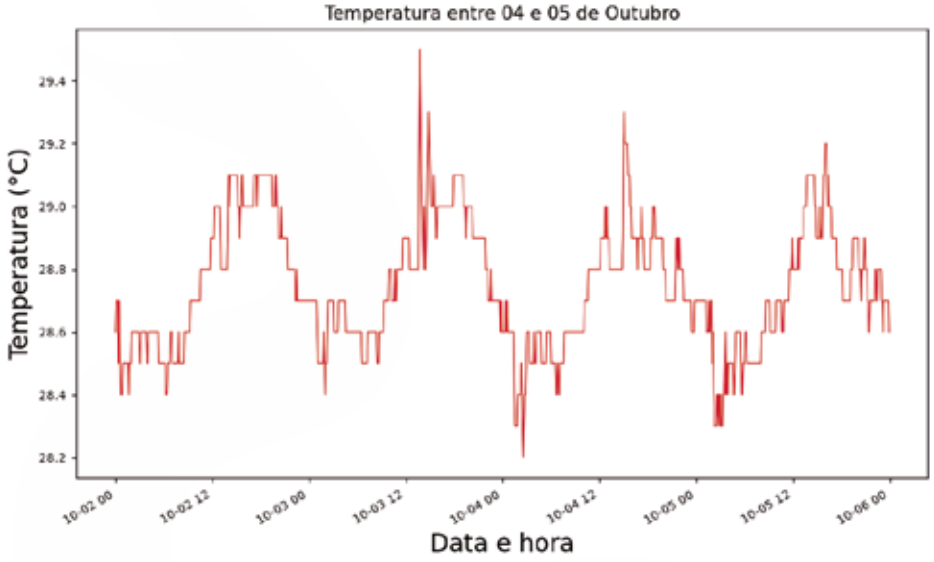
O método figura.autofmt_xdate() (i) faz com que os valores no eixo X fiquem inclinados para evitar eventuais sobreposições.

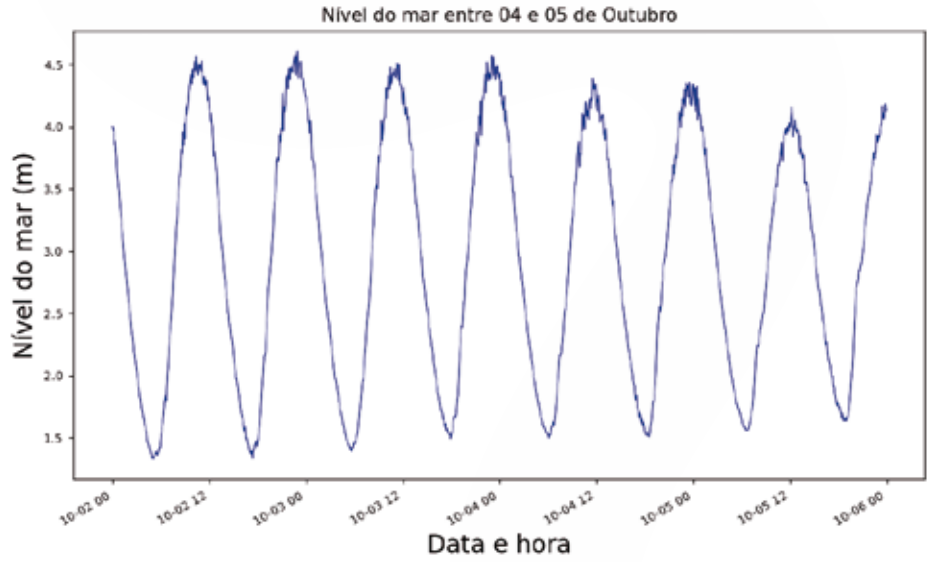
Em "k" o método plt.tick_params() é responsável para estilizar os valores contidos nos eixos X e Y. O argumento axis diz para python em qual eixo deve ser feito a mudança (no caso, escolhemos "both" para ambos os eixos). O argumento labelsize informa o tamanho da fonte do valor nos eixos.

Por fim, pedimos para Python executar a criação do gráfico por meio do método plt.savefig() (l) no qual salvamos o arquivo em um local específico. A variável location no início do código é o endereço escolhido para salvar o gráfico. Ao passarmos a variável location em savefig() repare que concatenamos com outra palavra. Essa palavra vai ser o nome do arquivo (nesse caso, como estamos configurando o gráfico de temperatura, colocamos o nome do arquivo do mesmo jeito).

Depois desse primeiro bloco, criamos mais outro bloco com a mesma lógica, porém, apenas trocamos a variável (lista_niveis) e nome do gráfico.

Ao executar o código, seus gráficos serão salvos em formato PNG e estarão prontos para serem visualizados.





Sinta-se à vontade para modificar o código e aprimorar ele, pois esse código é apenas para fins didáticos e não está escrito do jeito mais aprimorado possível.

Se você tiver alguma dúvida ou queira contribuir para a iniciativa Amanara entre em contato conosco.

Acesse: https://www.projetoamanara.com/contato

Também não deixe de conferir outros tutoriais disponíveis em nosso site!

Acesse: https://www.projetoamanara.com/plus

Caso queira aprender mais sobre a linguagem de programação Python para usar na sua carreira como oceanógrafo(a), aqui estão alguns links de vídeo aulas no youtube e referências bibliográficas.

Curso Em Vídeo - Python: https://www.youtube.com/watch?v=S9uPNppGsGo&list=PLvE-ZAFRgX8hnECDn1v9HNTI71veL3oW0&index=2&ab_channel=CursoemV%C3%ADdeo