

Gera série temporal de vento a partir do WaveWatch3 a partir de arquivos 'grib'

Carrega Grib file

Precisa usar o pacote 'pygrib'

Não consegui instalar o 'pygrib' com o 'pip', mas consegui usando o 'conda', então para usar tem que rodar o jupyter pelo prompt do miniconda.

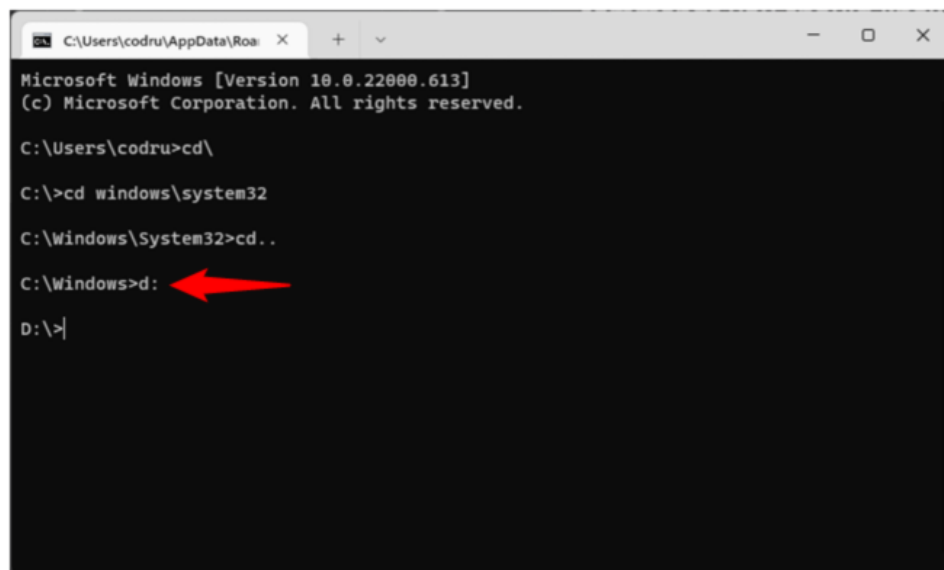
Para mudar de 'drive' (ou disco) tem que usar

2. How to change the drive in CMD (Command Prompt)

To access another drive, type the drive's letter, followed by `:`. For instance, if you wanted to change the drive from `C:` to `D:`, you should type:

```
d:
```

... and then press *Enter* on your keyboard.



```
C:\Users\codru>cd\  
C:\>cd windows\system32  
C:\Windows\System32>cd..  
C:\Windows>d:  
D:\>
```

Os dados do WaveWatch3 foram descarregados no NCEP no formato 'grib'

What is a GRIB file?

GRIB is a **file format for the storage and transport of gridded meteorological data, such as Numerical Weather Prediction model output**. It is designed to be self-describing, compact and portable across computer architectures. The GRIB standard was designed and is maintained by the World Meteorological Organization.

https://weather.gc.ca/grib/what_is_GRIB_e

[What is GRIB - Environment Canada](#)

URL da localização dos arquivos

<https://www.ncei.noaa.gov/data/oceans/ncep/>

Dados globais 'glo_30m'

Para baixar os dados tem que ir clicando e baixando manualmente! Não descobri como baixar por FTP, embora diga-se que dá!

Ajuda!

<https://www.youtube.com/watch?v=yLoudFv3hAY>

<https://www.luisalucchese.com/post/how-to-read-grib2-raster-python/>

```
In [1]: import pygrib
import numpy as np
import matplotlib.pyplot as plt
import os
import datetime
import pickle

import time
```

```
In [2]: path = r'd:\WaveWatch3\Wind\'
fdir = os.listdir(path)
```

a estrutura do Grib tem os resultados em intervalos de 3 horas para o modelo, e primeiro vem a sequencia da componente U, e depois vem a sequencia da componente V, e o número vai mudar dependendo do mes

```
In [3]: # https://www.youtube.com/watch?v=kEhcJP3G0xw

grbs = pygrib.open(path + fdir[0])
grbs.seek(0)

c = 0
for grb in grbs:
    print(grb)
    print(grb.validDate)
    print(grb.date)
    print(grb.time)
    print(grb.name)
    print(grb.units)
    c+=1
    if c == 3:
        break
```

```
1:U component of wind:m s**-1 (instant):regular_ll:surface:level 1:fcst time 0 hrs:f
rom 201508010000
2015-08-01 00:00:00
20150801
0
U component of wind
m s**-1
2:U component of wind:m s**-1 (instant):regular_ll:surface:level 1:fcst time 3 hrs:f
rom 201508010000
2015-08-01 03:00:00
20150801
0
```

```

U component of wind
m s**-1
3:U component of wind:m s**-1 (instant):regular_ll:surface:level 1:fcst time 6 hrs:f
rom 201508010000
2015-08-01 06:00:00
20150801
0
U component of wind
m s**-1

```

```

In [4]: # para ver o conteúdo de 1 grib

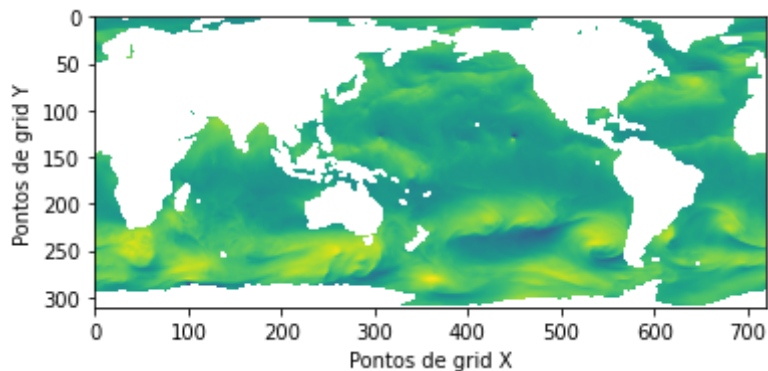
# eu não entendi muito bem esta parte, mas parece que sempre é bom 'puxar' de novo o
grbs.seek(0)
grb=grbs[1]
plt.imshow(grb.values) #valores da componente U do vento
plt.xlabel('Pontos de grid X')
plt.ylabel('Pontos de grid Y')

```

```

Out[4]: Text(0, 0.5, 'Pontos de grid Y')

```



```

In [5]: # para ver o conteúdo de 1 grib
grbs.seek(0)
grb=grbs[1]

# pega os valores de latitude e longitude para todos os pontos da malha (latitude an
lalo1 = grb.latlons() #sai como tupla
lalo2 = np.array(lalo1)

la = lalo2[0,:,:]
lo = lalo2[1,:,:]

extents = [np.min(lo), np.max(lo), np.min(la), np.max(la)]

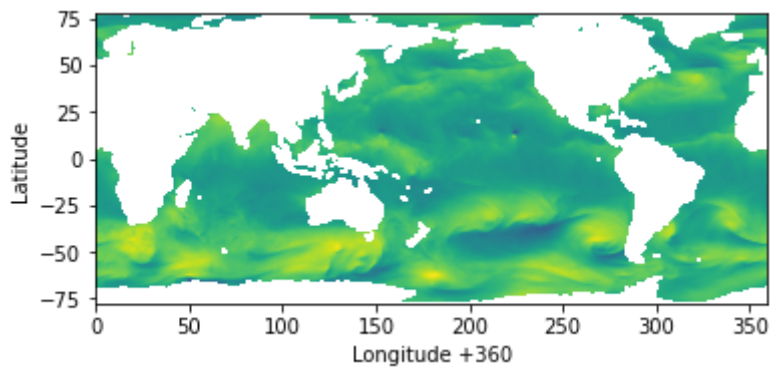
plt.imshow(grb.values, extent=extents)
plt.xlabel('Longitude +360')
plt.ylabel('Latitude')

```

```

Out[5]: Text(0, 0.5, 'Latitude')

```



```
In [6]: print(type(lalo1))
        print(lalo2.shape)
```

```
<class 'tuple'>
(2, 311, 720)
```

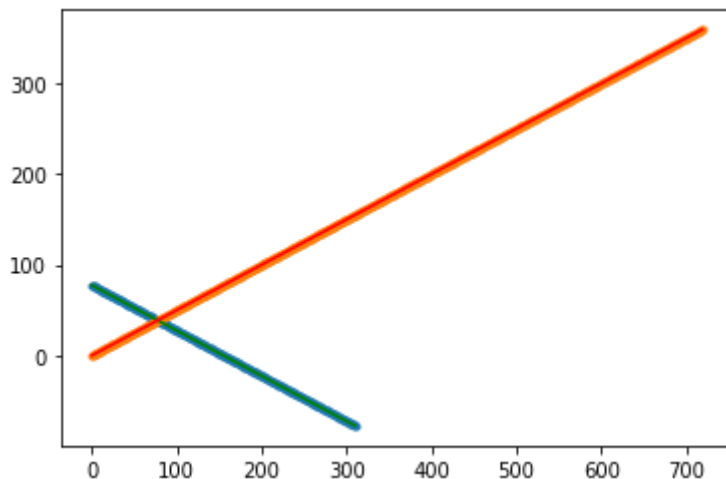
```
In [7]: # como variam os valores de latitude e longitude
```

```
latitudes = lalo2[0, :, :]
longitudes = lalo2[1, :, :]

lats = latitudes[:,0]
lons = longitudes[0,:]

plt.plot(latitudes[:,0], 'g')
plt.plot(longitudes[0,:], 'r')
plt.plot(lats, '.', color='tab:blue', zorder=0)
plt.plot(lons, '.', color='tab:orange', zorder=0)
```

```
Out[7]: [<matplotlib.lines.Line2D at 0x20a13d6a220>]
```



```
In [7]: latitudes.shape
```

```
Out[7]: (311, 720)
```

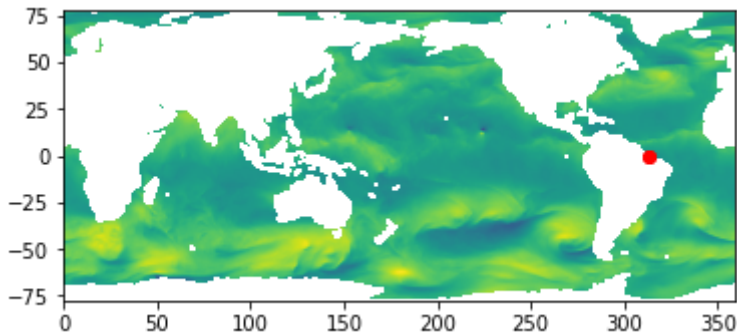
```
In [8]: # posição do ADCP e da estação meteorológica (Bragança, PA)
adcp_x = -46.564 + 360
adcp_y = -0.672

met_x = -46.6037 + 360
met_y = -0.8301
```

Posição do ADCP e estação meterológica dentro da malha do modelo

In [9]:

```
extends = [lons[0], lons[-1], lats[-1], lats[0]]
plt.imshow(grb.values, extent=extends)
plt.plot(adcp_x, adcp_y, 'ro')
plt.plot(met_x, met_y, 'ro')
plt.show()
```



Para achar o ponto mais próximo do fundeio, tentativa e erro!

Mudando os valores de 'linha' e 'coluna' e mudando os valores do 'xlim' e 'ylim'

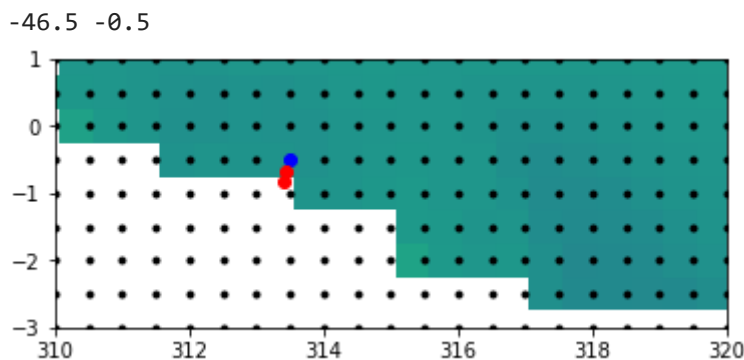
In [10]:

```
linha = 156
coluna = 627

extends = [lons[0], lons[-1], lats[-1], lats[0]]
plt.imshow(grb.values, extent=extends)
plt.plot(longitudes, latitudes, 'k.')
plt.plot(longitudes[linha, coluna], latitudes[linha, coluna], 'bo')
plt.plot(adcp_x, adcp_y, 'ro')
plt.plot(met_x, met_y, 'ro')
plt.xlim(310, 320)
plt.ylim(-3, 1)

print(longitudes[linha, coluna]-360, latitudes[linha, coluna])

plt.show()
```



Para gerar a série temporal para a o ponto selecionado

Demora! pois carrega cada arquivo para selecionar os dados da estação de interesse!

Para executar tudo, tem que tirar o 'break' do final do LOOP

```
In [12]: # indices do ponto mais próximo do ADCP
linha = 156
coluna = 627

u = []
v = []
t = []
conta = 0
c = 0
time_start = time.time()
for f in fdir:
    print('Processando ', f)
    grbs = pygrib.open(path + f)
    grbs.seek(0)

    t1 = []
    u1 = []
    v1 = []

    for grb in grbs:
        c += 1
        if 'U' in grb.name:
            t1.append(grb.validDate)
            u1.append(grb.values[linha, coluna])
        elif 'V' in grb.name:
            v1.append(grb.values[linha, coluna])

    t = t + t1
    u = u + u1
    v = v + v1

    conta += 1
    if conta == 1:
        break

time_elapsed = time.time() - time_start
print(time_elapsed)
```

```
Processando multi_1.glo_30m.wind.201508.grb2
32.370511054992676
```

```
In [13]: c
```

```
Out[13]: 498
```

```
In [45]: plt.plot(t, u)
plt.plot(t, v)
```

```
-----
NameError                                Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_5532\928130233.py in <module>
----> 1 plt.plot(t, u)
      2 plt.plot(t, v)
```

NameError: name 't' is not defined

In [15]:

```
print(t[0:3])
```

```
[datetime.datetime(2015, 8, 1, 0, 0), datetime.datetime(2015, 8, 1, 3, 0), datetime.datetime(2015, 8, 1, 6, 0)]
```

In [16]:

```
# j = [t, u, v]
# with open('WW3_vento_bragranca20152017.pkl', 'wb') as io:
#     pickle.dump(j, io)
```

In []:

Complemento!

Rearranja os dados para apresentar as longitudes para -180:180

In [59]:

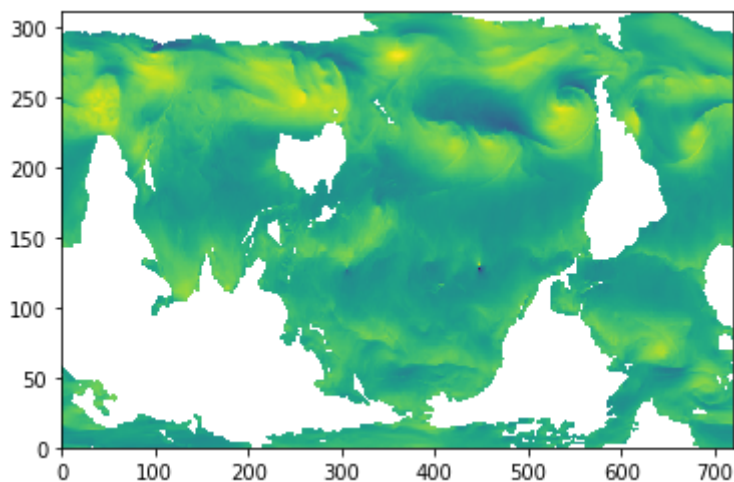
```
# pega os valores do grib e transforma em um array, mas os espaços em branco do gri
pega_values = np.array(grb.values)
print(pega_values.shape)

pega_values[pega_values==9999.0] = np.nan

plt.pcolor(pega_values)
```

(311, 720)

Out[59]: <matplotlib.collections.PolyCollection at 0x20aa390dc40>



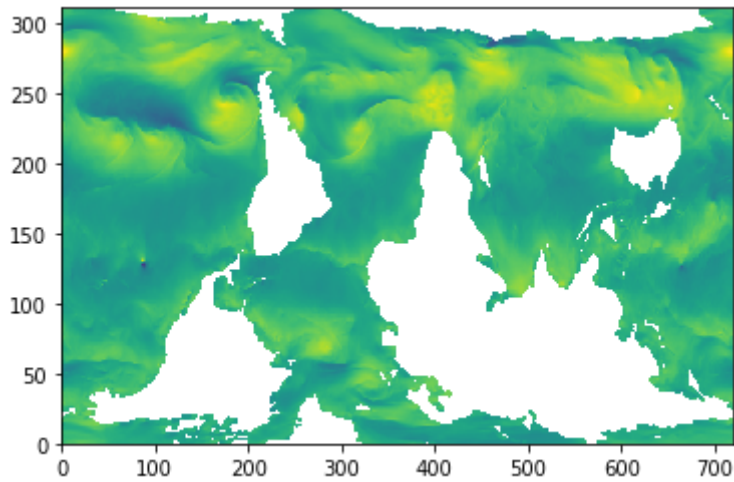
In [64]:

```
# para fazer o mapa com configuração -180:180, tem que rearranjar a matri, passando
# e fazer isso para a matriz de longitudes também

pv = np.hstack((pega_values[:,361:], pega_values[:,361]))

plt.pcolor(pv)
```

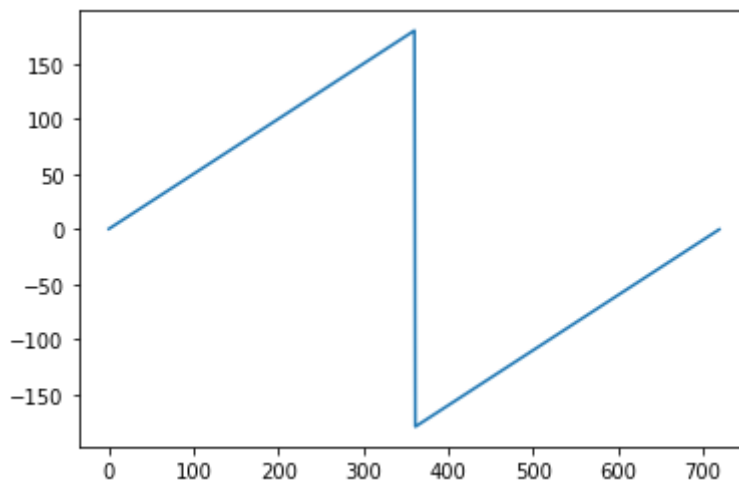
Out[64]: <matplotlib.collections.PolyCollection at 0x20ab8edc5b0>



```
In [53]: # corrige os valores de longitude do trecho >180 para ficarem -180:0
lons2 = np.copy(lons)
for i,x in enumerate(lons2):
    if x > 180:
        lons2[i] = lons2[i] - 360

plt.plot(lons2)
```

Out[53]: [



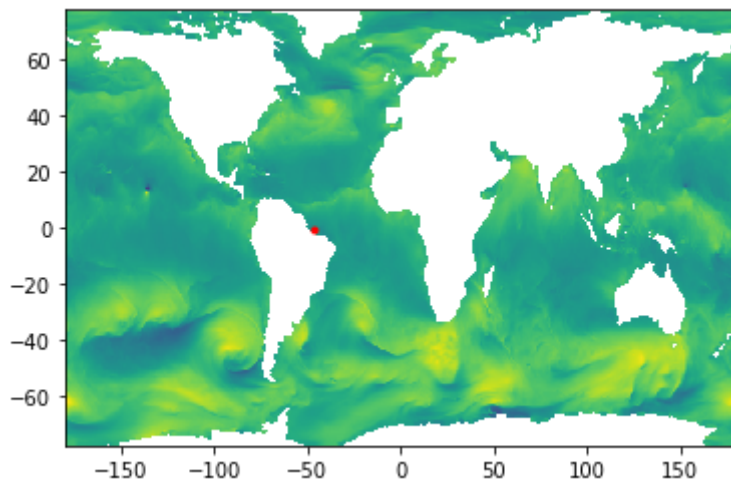
```
In [66]: # posição do ADCP e da estação meteorológica (Bragança, PA)
adcp_x = -46.564
adcp_y = -0.672

met_x = -46.6037
met_y = -0.8301
```

```
In [67]: xx, yy = np.meshgrid(lons2, lats)
xx2 = np.hstack((xx[:,361:], xx[:,361]))

plt.pcolor(xx2, yy, pv)
plt.plot(adcp_x, adcp_y, 'r.'))
```

Out[67]: [



```
In [72]: xx, yy = np.meshgrid(lons2, lats)
xx2 = np.hstack((xx[:,361:], xx[:, :361]))

plt.pcolor(xx2, yy, pv)
plt.plot(adcp_x, adcp_y, 'r.')
plt.xlim(-50, -45)
plt.ylim(-3, 1)
```

Out[72]: (-3.0, 1.0)

