PS2

首先导入 pandas，numpy，matplotlib

```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
%matplotlib inline
```

1. Significant earthquakes since 2150 B.C.

读取.tsv 文件,命名为 Sig_Eqs

```
# 1
# read the tsv file
Sig_Eqs = pd.read_csv('earthquakes-2022-10-19_17-21-49_+0800.tsv', sep='\t')
Sig_Eqs
```

1.1 计算 2150BC 以来地震造成的总死亡人数，print 总数前 20 的国家。

先根据国家分组，计算总和得出每个国家的死亡人数，再按照死亡人数排序，降序，取前 20 行。

```
#1.1
Sig_Eqs.groupby(['Country']).sum().sort_values(['Deaths'],ascending=False)['Deaths'][0:20]
```

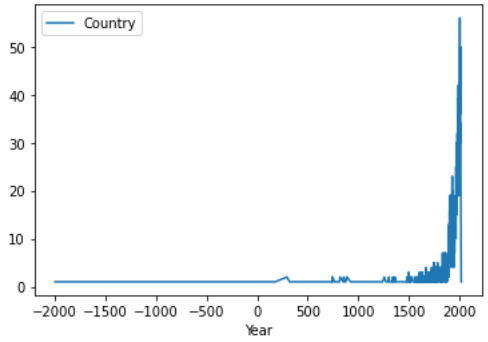
```
Country
CHINA           2075019.0
TURKEY          1134569.0
IRAN            1011446.0
ITALY            498477.0
SYRIA            439224.0
HAITI            323474.0
AZERBAIJAN       317219.0
JAPAN            278142.0
ARMENIA          191890.0
PAKISTAN         145083.0
IRAQ             136200.0
ECUADOR          135479.0
TURKMENISTAN     117412.0
PERU             102219.0
ISRAEL            90388.0
PORTUGAL          83531.0
GREECE            79174.0
CHILE             64276.0
INDIA             63491.0
TAIWAN            57135.0
Name: Deaths, dtype: float64
```

1.2 选取 Ms>3.0，计算每年的地震数量，画图。

先用 loc 筛选出 Ms>3.0 的数据，再根据年份分组并计算（count），country 计算出来的值=地震的数量，最后 plot。

```
# 1.2
Sig_Eqs.loc[Sig_Eqs['Ms']>3.0].groupby(['Year']).count().loc[:,['Country']].plot()
```

```
<AxesSubplot:xlabel='Year'>
```



趋势是震级大于 3.0 的地震的次数是逐年增加的，尤其是 1500 年后上升趋势大幅增加。

原因：地震时由大陆板块的挤压碰撞形成的，增加的原因是地球南北两极的冰雪融化导致地北两极的质量分布在大规模的减少,进而导致地壳在被地下的岩浆所推动,增加了板块运动。

1.3 定义一个函数，print 自 2150BC 给定国家地震的总数，在这个国家有史以来最大的地震发生的日期和位置。
首先，先从原始数据中提取出 country，Ms，location name，year，mo，dy 这几列得到 df。用 unique 把所有国家以列表的形式列出来。

```
df = Sig_Eqs.loc[:,['Country','Ms','Location Name','Year','Mo','Dy']]
df
Country_List = df['Country'].unique()
```

把原始的日期年月日格式化并建立一个新的日期列表（DATE）

```
df['cYear'] = df['Year'].astype(str)
df['cMo'] = df['Mo'].astype(str)
df['cDy'] = df['Dy'].astype(str)
df['cMo'][df['Mo'] < 10] = '0' + df['cMo'].astype(str)
df['cDy'][df['Dy'] < 10] = '0' + df['cDy'].astype(str)
df['DATE'] = df['cYear'] + '/' + df['cMo'] + '/' + df['cDy']
df
```

创建一个空的 dataframe 储存得出的结果。
定义函数 CountEq_LargestEq(i)，i 代表国家，先在 df 中索引出国家 i 的所有行，再索引这个国家地震震级的最大值，加上索引国家，地点名称，震级，日期四列。在 df 中用 count 函数算出 i 国家的地震总数得到 Total_Number 这一列，最后将 df3append 到 re 的结果中。

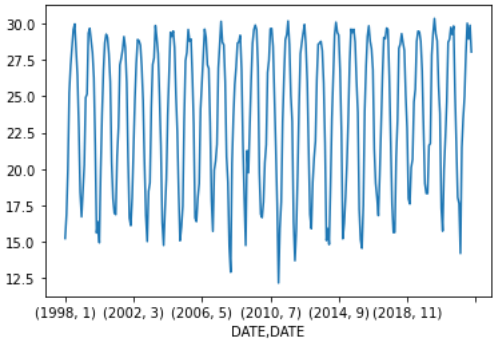
```
re = pd.DataFrame(columns=['Country','Location Name','Ms','DATE'])
re

def CountEq_LargestEq(i):
    df2 = df[df['Country'] == str(i)]
    df3 = df2[df['Ms'] == df2['Ms'].max()][['Country','Location Name','Ms','DATE']]
    df3['Total_Number'] = df[df['Country'] == str(i)]['Country'].count()
    global re
    re = re.append(df3)
```

利用 for 循环在国家的列表中循环，重复循环 CountEq_LargestEq(i)的函数，降序排列，得到每个国家对应的地震总数，最大震级的地点和时间。

2.
首先读取文件。
利用 loc 函数把原文件中的 DATE 和 TMP 的列读取出来，记为 T_data。
利用 split 函数分列。
再将 DATE 这一列以 T 分成两列，前一列是日期，后一列是时间。
将 TMP 这一列以，分成两列，前一列是 TMP,后一列是 CODE。
最后根据年月画图。

```
# 2
T_data = baoan_weather.loc[:,('DATE','TMP')]
T_data['DATE'],T_data['TIME'] = T_data['DATE'].str.split('T',1).str
T_data['TMP'],T_data['CODE'] = T_data['TMP'].str.split(',',1).str
T_data['TMP'] = T_data['TMP'].astype(float)/10
T_data['DATE'] = pd.to_datetime(T_data['DATE'])
T_data2 = T_data.loc[T_data['CODE'] == '1'][['DATE','TMP','TIME']]
T_data2
T_data2.groupby([T_data2['DATE'].dt.year,T_data2['DATE'].dt.month])['TMP'].mean().plot()
```

深圳过去 25 年的月平均气温没有明显的变化。



3.

3.1 用 grouby 函数对 SID 分组，后找出 WMO_WIND 的最大值，并读取 NAME。

```
# 3.1
df.groupby(['SID']).max().sort_values('WMO_WIND', ascending=False)[0:10]['NAME']
```

```
C:\Users\李彦辰\AppData\Local\Temp/ipykernel_26296/2409264014.py:2: FutureWarning: Dropping invalid columns in D
ataFrameGroupBy.max is deprecated. In a future version, a TypeError will be raised. Before calling .max, select
only columns which should be valid for the function.
  df.groupby(['SID']).max().sort_values('WMO_WIND', ascending=False)[0:10]['NAME']
```

```
SID
1997125S08079            RHONDA
2005237N14148             TALIM
2005054S09173             PERCY
2005063S12141            INGRID
2005092S11102    ADELINE:JULIET
2005148N06156             NESAT
2005192N11318             EMILY
2005192N22155           HAITANG
2005230N20144             MAWAR
2005236N23285           KATRINA
Name: NAME, dtype: object
```

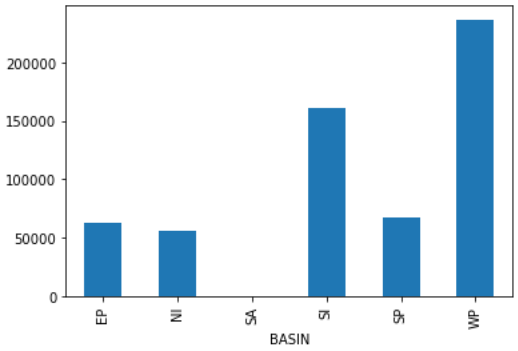
？ 3.2 这一题有问题，不知道代码哪里出错，读取出来的所有 WMO_WIND 的值均为 95，plot 后显示 no numeric data to plot。

```
# 3.2 ???
plot_df = df.groupby('SID').agg({'WMO_WIND':'max'}).sort_values(by='WMO_WIND',ascending=False)[0:20]
plot_df
#plot_df.plot(kind='bar')
```

3.3 用 grouby 函数对 BASIN 分组，对 NUMBER count，最后 plot。

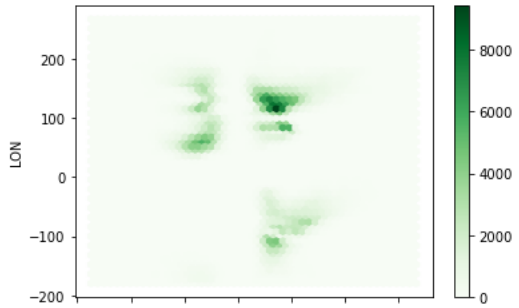
```
# 3.3
df.groupby('BASIN')['NUMBER'].count().plot(kind='bar')
```

```
<AxesSubplot:xlabel='BASIN'>
```



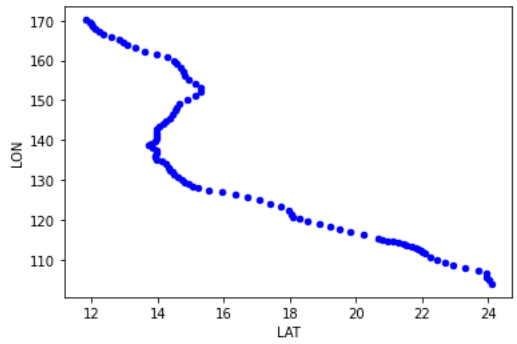
3.4 用 plot.hexbin，x 为 LAT，y 为 LON

```
: # 3.4
  df.plot.hexbin(x='LAT', y='LON', gridsize=50,cmap='Greens')
: <AxesSubplot:xlabel='LAT', ylabel='LON'>
```



3.5 找出符合 SEASON 是 2018 和 NAME == MANGKHUT 的值，plot.scatter

```
]: # 3.5
  df[(df['SEASON'] == 2018) & (df['NAME'] == 'MANGKHUT')].plot.scatter(x='LAT',y='LON',c='Blue')
]: <AxesSubplot:xlabel='LAT', ylabel='LON'>
```

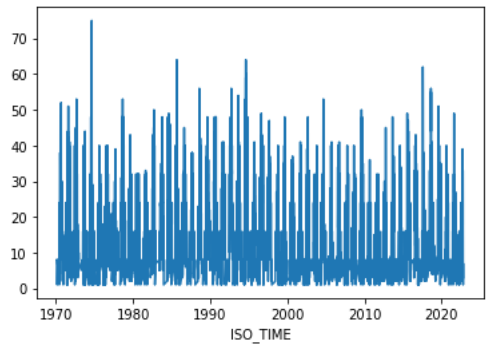


3.6 df2 是新的 dataframe，包含 SANSON 大于等于 1970 之后的，和 BASIN 是 WP 和 EP 的所有数据

```
# 3.6
df2 = df.loc[(df['SEASON'] >= 1970) & (df['BASIN'].isin(['WP','EP']))]
df2.head()
```

3.7 用 ISO_TIME 分组，计算 NUMBER 的 count，再 plot

```
: # 3.7
  df2.groupby([df2['ISO_TIME'].dt.date]).count()['NUMBER'].plot()
: <AxesSubplot:xlabel='ISO_TIME'>
```



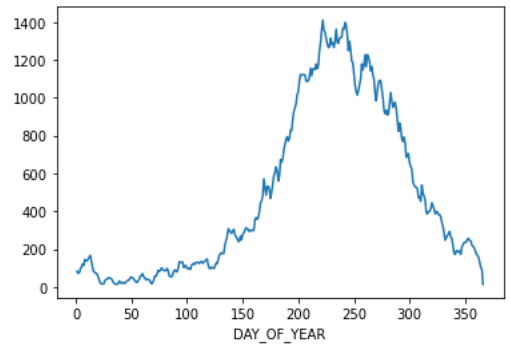
3.8 用 day_of_year 单列一列出来，这一列是指所有年的同一天都是一个数值，再对这一列进行分组，即以天分组，再用 size plot

本题石绍提出了问题，因为用 size 是包含了空值，要求平均。但是我认为用 size 算出来的趋势和平均后的趋势是相同的。

```
# 3.8
df2['DAY_OF_YEAR'] = df2['ISO_TIME'].dt.day_of_year
df2.groupby(['DAY_OF_YEAR']).size().plot()
```

```
C:\Users\李彦辰\AppData\Local\Temp/ipykernel_26296/2335276101.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df2['DAY_OF_YEAR'] = df2['ISO_TIME'].dt.day_of_year
```

```
<AxesSubplot:xlabel='DAY_OF_YEAR'>
```

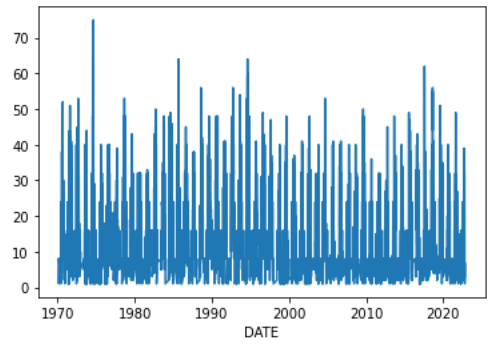


3.9 先对 DATE 进行分组，再用 size plot

```
# 3.9
df2['DATE'] = df2['ISO_TIME'].dt.date
df2.groupby(['DATE']).size().plot()
```

```
C:\Users\李彦辰\AppData\Local\Temp/ipykernel_26296/3322063674.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df2['DATE'] = df2['ISO_TIME'].dt.date
```

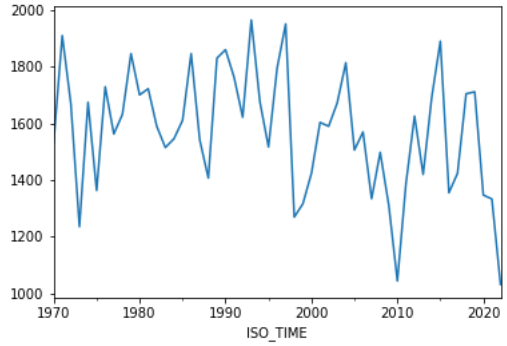
```
<AxesSubplot:xlabel='DATE'>
```



3.10 以年为单位，用 resample 重新处理

```
# 3.10
daily_counts = df2.groupby(["ISO_TIME"]).size()
daily_counts.resample('Y').size().plot()
```

```
<AxesSubplot:xlabel='ISO_TIME'>
```



和石绍、王子珂讨论了 3.8-3.10

4.

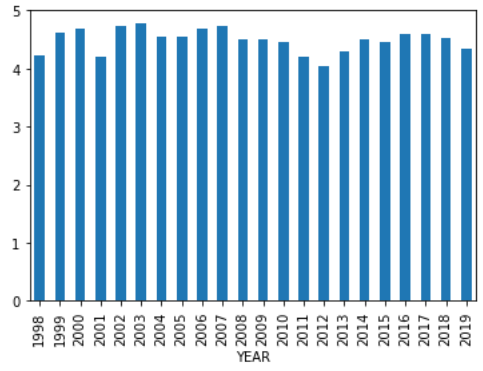
这个文件包括 USW00094724 站点的气候数据。包括 50 个气候变量，这些变量是根据全球历史气候网络每日的数据计算得出的结果。

4.1 首先是提取 STATION,DATE,AWND，去掉 AWND 当中的空值。AWND 表示月平均风速。

```
# 4.1
df1 = df.loc[:,('STATION','DATE','AWND')]
AWND_df = df1.dropna()
AWND_df['YEAR'],AWND_df['MONTH'] = AWND_df['DATE'].str.split('-',1).str
AWND_df.reset_index(drop=True, inplace=True)
AWND_df
```

4.2 计算月平均风速的，plot 为柱状图，发现风速大概稳定在 4.5 左右，2012 和 2001 年较低，其他没有明显的变化趋势。

```
<AxesSubplot:xlabel='YEAR'>
```

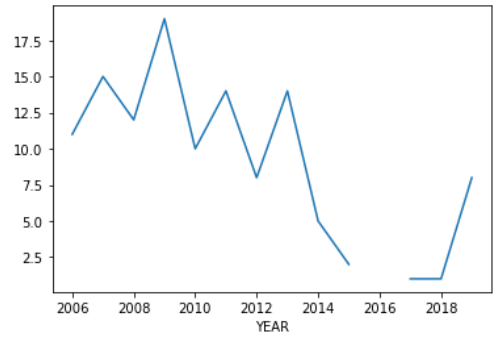


4.3DYHF 代表一个月当中 heavy fog 的总天数。计算一年当中 DYHF 的最大值，plot 柱状图，发现 2009 年是最大，此后整体的趋势是下降，16 年没有值是由于原始数据的缺失导致（下载的解释文件中有说明确实 16 年的数据）。

```
#4.3
df2 = df.loc[:,('STATION','CDSD','CLDD','DATE','DYFG','DYHF','DYTS','EMNT','EMXT')]
df2['YEAR'],df2['MONTH'] = df2['DATE'].str.split('-',1).str
df2
df2.groupby('YEAR')['DYHF'].max().plot()
```

```
C:\Users\李彦辰\AppData\Local\Temp/ipykernel_26296/3833181037.py:3: FutureWarning: Columnar iteration over chara
cters will be deprecated in future releases.
  df2['YEAR'],df2['MONTH'] = df2['DATE'].str.split('-',1).str
```

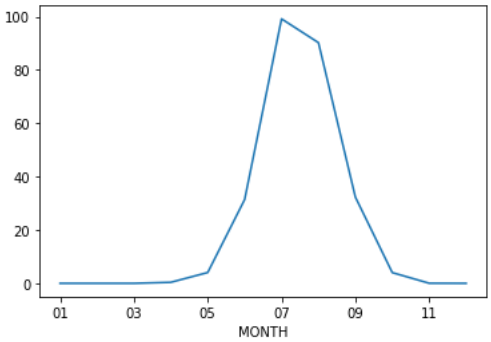
```
<AxesSubplot:xlabel='YEAR'>
```



4.4CLDD 表示降温的天数，输出降温天数的前 10 名，发现这些降温大部分都发生在 7 月和 8 月。以月分组 plot 降温天数的平均值来验证观察到的结果发现结果一致，在 7.8 月降温最多。

```
#4.4
df2.sort_values(['CLDD'],ascending=False)[0:10]
df2.groupby('MONTH')['CLDD'].mean().plot()
```

```
<AxesSubplot:xlabel='MONTH'>
```



4.5EMNT 表示月温度的最小值，输出 EMNT 的前 10 名，发现在 7 月和 8 月也是出现了峰值。与上一题的结论相符合。

```
#4.5
df2.sort_values(['EMNT'],ascending=False)[0:10]
df2.groupby('MONTH')['EMNT'].mean().plot()
```

```
<AxesSubplot:xlabel='MONTH'>
```