1. Significant earthquakes since 2150 B.C.

In [345]:
```
import pandas as pd
import numpy as np
import xarray as xr
from matplotlib import pyplot as plt
%matplotlib inline

# Load the data
Sig_Eqs = pd.read_csv('earthquakes-2023-11-05_22-09-37_+0800.tsv',sep='\t')
Sig_Eqs
```

Out[345]:

	Search Parameters	Id	Year	Mo	Dy	Hr	Mn	Sec	Tsu	Vol	...	Total Missing	Total Missing Description	Total Injuries	Total Injuries Description	Total Damage ($Mil)	Total Damage Description	Total Houses Destroyed	Total Houses Destroyed Description	Da
0	[]	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
1	NaN	1.0	-2150.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
2	NaN	2.0	-2000.0	NaN	NaN	NaN	NaN	NaN	1.0	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
3	NaN	3.0	-2000.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	1.0	NaN	1.0	
4	NaN	5877.0	-1610.0	NaN	NaN	NaN	NaN	NaN	3.0	1351.0	...	NaN	NaN	NaN	NaN	NaN	3.0	NaN	NaN	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
6394	NaN	10708.0	2023.0	10.0	7.0	6.0	41.0	3.0	NaN	NaN	...	NaN	NaN	1950.0	4.0	NaN	3.0	2862.0	4.0	
6395	NaN	10711.0	2023.0	10.0	7.0	8.0	40.0	13.0	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	2.0	200.0	3.0	
6396	NaN	10709.0	2023.0	10.0	8.0	20.0	25.0	23.0	5891.0	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
6397	NaN	10710.0	2023.0	10.0	11.0	0.0	41.0	56.0	NaN	NaN	...	NaN	NaN	164.0	3.0	NaN	2.0	NaN	NaN	
6398	NaN	10712.0	2023.0	10.0	15.0	3.0	36.0	0.0	NaN	NaN	...	NaN	NaN	153.0	3.0	NaN	2.0	NaN	2.0	

6399 rows × 49 columns

1.1

In [214]:
```
# Group the data by country, calculate total deaths in all country
# and sort them in descending order
sig_deaths10 = Sig_Eqs.groupby(['Country']).sum('Total Deaths').sort_values("Total Deaths",ascending=False).head(10)
print("The top ten countries along with the total number of deaths: ")
sig_deaths10['Total Deaths']
```

```
The top ten countries along with the total number of deaths:
```

Out[214]:
```
Country
CHINA          2041929.0
TURKEY          995648.0
IRAN            758650.0
SYRIA           437700.0
ITALY           422679.0
JAPAN           356083.0
HAITI           323776.0
AZERBAIJAN      310119.0
INDONESIA       282819.0
ARMENIA         189000.0
Name: Total Deaths, dtype: float64
```

1.2

In [215]:
```
Sig_Eqs[Sig_Eqs['Mag']>6.0]['Id'].count()
```

Out[215]:
```
2946
```

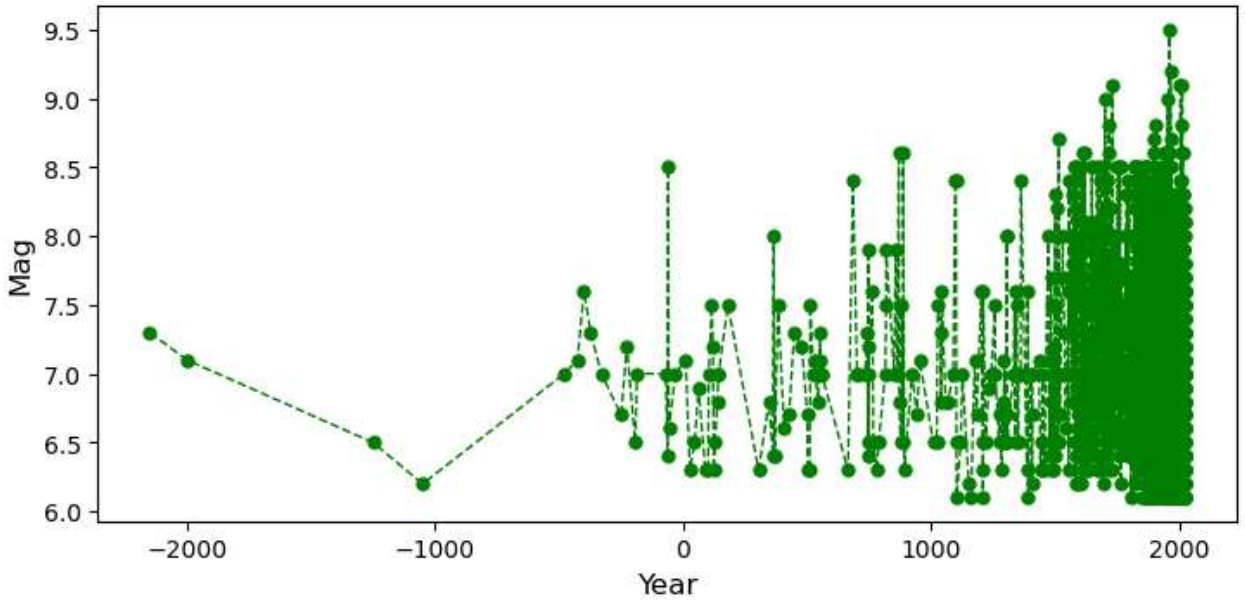
In [216]:
```
sig_eqs_ex6 = Sig_Eqs[Sig_Eqs['Mag']>6.0][['Year','Mag']]
sig_eqs_ex6
```

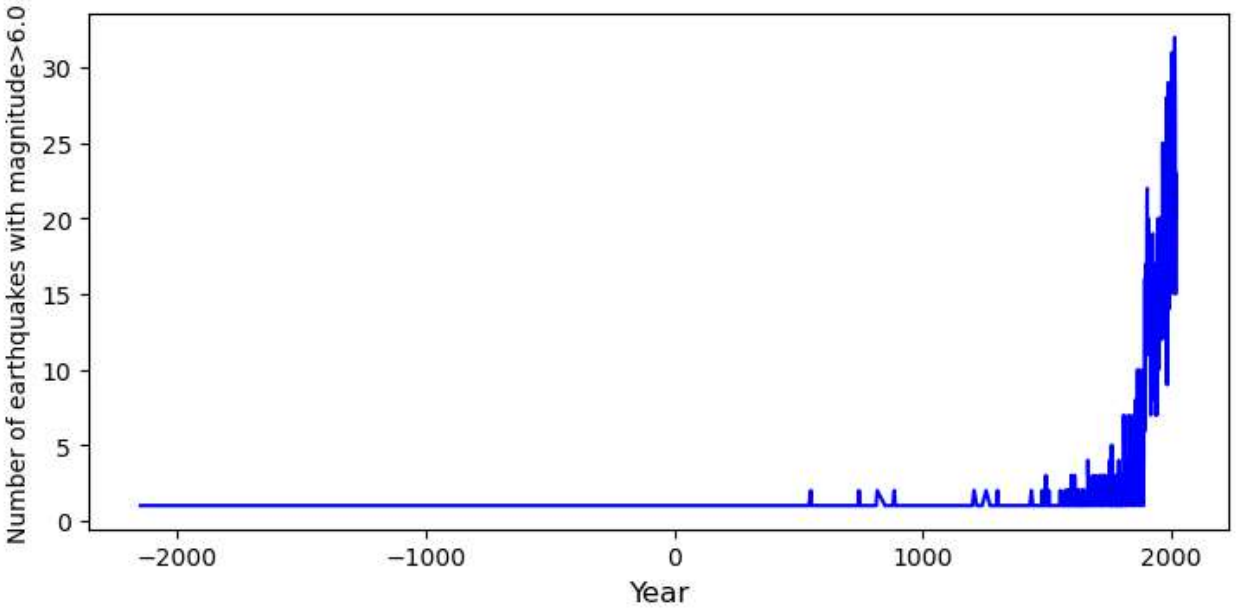
Out[216]:

	Year	Mag
1	-2150.0	7.3
3	-2000.0	7.1
8	-1250.0	6.5
9	-1050.0	6.2
15	-479.0	7.0
...	...	...
6393	2023.0	6.1
6394	2023.0	6.3
6395	2023.0	6.9
6397	2023.0	6.3
6398	2023.0	6.3

2946 rows × 2 columns

In [217]:
```
# Given data in the x and y directions
x=sig_eqs_ex6["Year"]
y=sig_eqs_ex6["Mag"]

# Count the number of earthquakes with magnitude larger than 6.0 each year
eq_count=Sig_Eqs[Sig_Eqs['Mag']>6.0].groupby('Year').size()

# Plot figure1()
plt.figure(figsize=(8.5, 8.5))
plt.subplot(2,1,1)
plt.plot(x,y,'go--',linewidth=1,markersize=5)
plt.xlabel("Year",size=12)
plt.ylabel("Mag",size=12)
plt.show()

# Plot figure2()
plt.figure(figsize=(8.5, 8.5))
plt.subplot(2,1,2)
plt.plot(eq_count.index, eq_count.values,'b-')     # Plot the time series
plt.xlabel('Year',size=12)
plt.ylabel('Number of earthquakes with magnitude>6.0',size=10)

plt.show()
```





As we can see from the plots, there is a clear upward trend in the number of earthquakes with magnitude larger than 6.0 since the 1950s. In recent years, major earthquakes (earthquake magnitude reaching 6.0 and above) have occurred significantly more and more frequently. This could be due to a combination of factors, including better detection methods, increased population density, and changes in the Earth's crust.

1.3

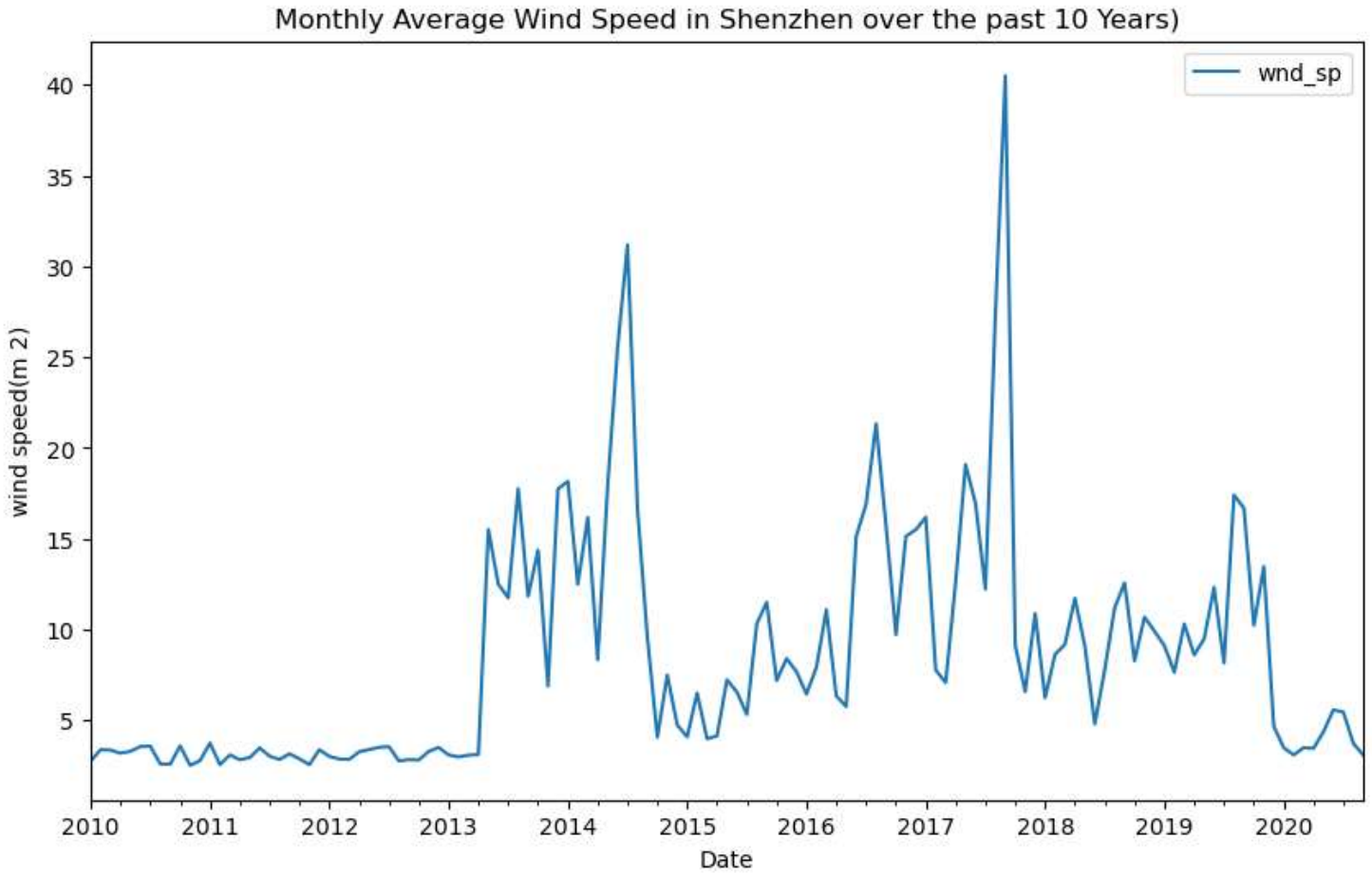
In [453]:
```
def CountEq_LargestEq(COUNTRY,Sig_Eqs):
    country_eqs = Sig_Eqs[Sig_Eqs['Country'] == COUNTRY]
    total_eqs = len(country_eqs)    #Calculate the total number of earthquakes in given country
    if total_eqs == 0:
        return total_eqs, None
    else:
        #Find the date of the largest earthquake in given country
        largest_eq_date = country_eqs[['Year','Mo','Dy','Mag']].sort_values('Mag',ascending=False).head(1)
    return total_eqs, largest_eq_date


CountEq_LargestEq('CHINA',Sig_Eqs)

# Apply function to each country
results = Sig_Eqs.groupby('Country').apply(lambda x: CountEq_LargestEq(x['Country'].iloc[0], x))

# Sort results in descending order
results.sort_values(by='total_eqs', ascending=False, inplace=True)

print(results)
```

Out[453]:
```
(620,
          Year   Mo    Dy  Mag
 982  1668.0  7.0  25.0  8.5)
```

2. Wind speed in Shenzhen during the past 10 years

In [505]:
```
# Read the .csv file
sz_df = pd.read_csv('2281305.csv')

#Separate the WND columns with commas to form multiple columns of wind related data.
sz_df_wnd = sz_df['WND'].str.split(',',expand=True)
sz_df_wnd.columns = ['agl','aglq','wnd_ty','wnd_sp','wndq']
sz_df = pd.concat([sz_df,sz_df_wnd], axis=1)
sz_df = sz_df.drop('WND',axis=1)
sz_df['wnd_sp'] = sz_df['wnd_sp'].str.slice(0, 5).astype(float)

# Filter the data
sz_df = sz_df.loc[sz_df_wnd['wnd_sp'] != 9999]

# Convert date to datetime
sz_df['DATE'] = pd.to_datetime(sz_df['DATE'], format="%Y-%m-%dT%H:%M:%S")
sz_df=sz_df[['DATE','agl','aglq','wnd_ty','wnd_sp','wndq']]
sz_df['YYYY-MM']=sz_df['DATE'].dt.to_period('M')

# Calculate monthly average wind speed
wnd_mon=sz_df[['YYYY-MM','wnd_sp']].groupby(['YYYY-MM']).mean() /10
wnd_mon

# Plot the data
plt.figure(figsize=(10, 6))
wnd_mon['wnd_sp'].plot()
plt.xlabel('Date')
plt.ylabel('wind speed(m$^2$)')
plt.title('Monthly Average Wind Speed in Shenzhen over the past 10 Years)')
plt.legend()
plt.show()
```

```
C:\Users\Administrator\AppData\Local\Temp\ipykernel_15676\2302612468.py:2: DtypeWarning: Columns (4,8,9,12,15,21,22,24,26,31,33,34) have mixed types. Specify dtype option on import or set low_memory=False.
  sz_df = pd.read_csv('2281305.csv')
```



3. Explore a data set

Based on the Intergrated Surface Dataset(Global) provided by the National Centers for Environmental Information (NCEI), I downloaded the comprehensive data of Beijing from January 1, 2022 to December 31, 2022, and extracted the air temperature data for analysis.

(https://www.ncei.noaa.gov/access/search/data-search/global-hourly?dataTypes=AA1&bbox=41.059,115.417,39.442,117.508&pageNum=1&startDate=2022-01-01T00:00:00&endDate=2022-12-31T23:59:59 (https://www.ncei.noaa.gov/access/search/data-search/global-hourly?dataTypes=AA1&bbox=41.059,115.417,39.442,117.508&pageNum=1&startDate=2022-01-01T00:00:00&endDate=2022-12-31T23:59:59))

3.1 Load and filter data

In [511]:
```
# Load file
bj_df = pd.read_csv('54511099999.csv')

#Separate the TMP columns with commas to form multiple columns of air temperature related data.
bj_tmp = bj_df['TMP'].str.split(',',expand=True)
bj_tmp.columns = ['tmp','tmpq']
bj_df = pd.concat([bj_df,bj_tmp], axis=1)
bj_df = bj_df.drop('TMP',axis=1)
bj_df['tmp'] = bj_df['tmp'].str.slice(0, 5).astype(float)

# Filter the data
bj_df = bj_df.loc[bj_df['tmp'] != 9999]
bj_df
```

```
C:\Users\Administrator\AppData\Local\Temp\ipykernel_15676\2756867226.py:2: DtypeWarning: Columns (19,25,28) have mixed types. Specify dtype option on impor
t or set low_memory=False.
  bj_df = pd.read_csv('54511099999.csv')
```

Out[511]:

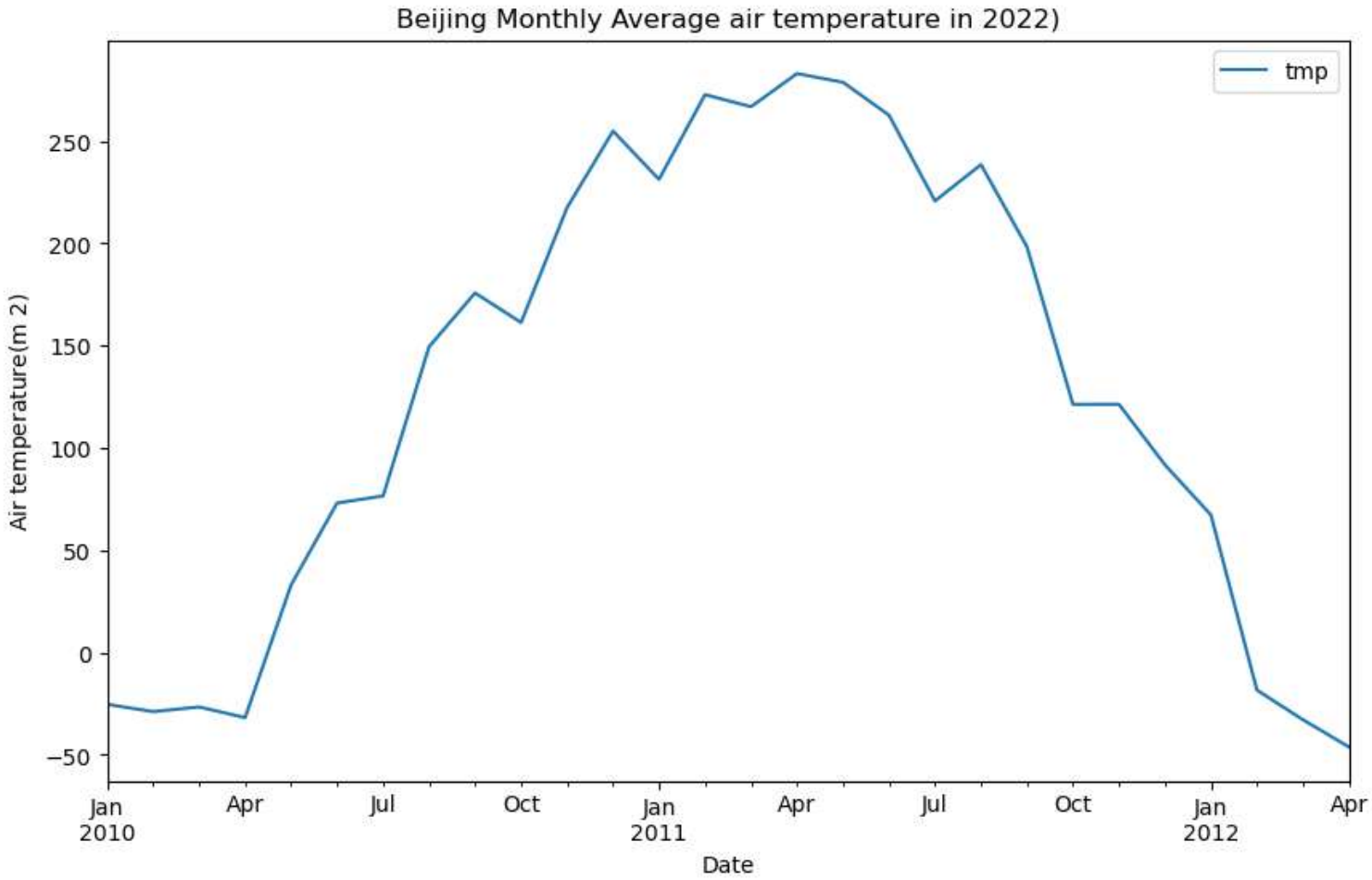
	STATION	DATE	SOURCE	LATITUDE	LONGITUDE	ELEVATION	NAME	REPORT_TYPE	CALL_SIGN	QUALITY_CONTROL	...	MA1	
0	54511099999	2022-01-01T00:00:00	4	40.080111	116.584556	35.35	BEIJING CAPITAL INTERNATIONAL AIRPORT, CH	FM-12	99999	V020	...	99999,9,10211,1	7,1,003,
1	54511099999	2022-01-01T00:00:00	4	40.080111	116.584556	35.35	BEIJING CAPITAL INTERNATIONAL AIRPORT, CH	FM-15	99999	V020	...	10250,1,99999,9	
2	54511099999	2022-01-01T00:30:00	4	40.080111	116.584556	35.35	BEIJING CAPITAL INTERNATIONAL AIRPORT, CH	FM-15	99999	V020	...	10250,1,99999,9	
3	54511099999	2022-01-01T01:00:00	4	40.080111	116.584556	35.35	BEIJING CAPITAL INTERNATIONAL AIRPORT, CH	FM-15	99999	V020	...	10260,1,99999,9	
4	54511099999	2022-01-01T01:30:00	4	40.080111	116.584556	35.35	BEIJING CAPITAL INTERNATIONAL AIRPORT, CH	FM-15	99999	V020	...	10260,1,99999,9	
...	...	...	...	...	...	...	...	...	...	...	...	...	
20242	54511099999	2022-12-31T21:30:00	4	40.080111	116.584556	35.35	BEIJING CAPITAL INTERNATIONAL AIRPORT, CH	FM-15	99999	V020	...	10350,1,99999,9	
20243	54511099999	2022-12-31T22:00:00	4	40.080111	116.584556	35.35	BEIJING CAPITAL INTERNATIONAL AIRPORT, CH	FM-15	99999	V020	...	10350,1,99999,9	
20244	54511099999	2022-12-31T22:30:00	4	40.080111	116.584556	35.35	BEIJING CAPITAL INTERNATIONAL AIRPORT, CH	FM-15	99999	V020	...	10350,1,99999,9	
20245	54511099999	2022-12-31T23:00:00	4	40.080111	116.584556	35.35	BEIJING CAPITAL INTERNATIONAL AIRPORT, CH	FM-15	99999	V020	...	10360,1,99999,9	

	STATION	DATE	SOURCE	LATITUDE	LONGITUDE	ELEVATION	NAME	REPORT_TYPE	CALL_SIGN	QUALITY_CONTROL	...	MA1
20246	54511099999	2022-12-31T23:30:00	4	40.080111	116.584556	35.35	BEIJING CAPITAL INTERNATIONAL AIRPORT, CH	FM-15	99999	V020	...	10360,1,99999,9

20242 rows × 40 columns

3.2 Plot monthly averaged air temperature as a function of the observation time

In [510]:
```
# Calculate the monthly average temperature in Beijing based on time series
# Convert date to datetime
bj_df['DATE'] = pd.to_datetime(bj_df['DATE'], format="%Y-%m-%dT%H:%M:%S")
bj_df[["DATE","tmp","tmpq"]]
bj_df['YYYY-MM']=bj_df['DATE'].dt.to_period('M')
# Calculate monthly average air temperature
tmp_mon=bj_df[['YYYY-MM','tmp']].groupby(['YYYY-MM']).mean()
tmp_mon

# Plot the data
plt.figure(figsize=(10, 6))
tmp_mon['tmp'].plot()
plt.xlabel('Date')
plt.ylabel('Air temperature(m$^2$)')
plt.title('Beijing Monthly Average air temperature in 2022)')
plt.legend()
plt.show()
bj_df['DATE']
```



Out[510]:
```
0        2010-01-02 00:00:00
1        2010-01-02 01:00:00
2        2010-01-02 02:00:00
3        2010-01-02 03:00:00
4        2010-01-02 04:00:00
                 ...
20242    2012-04-27 10:00:00
20243    2012-04-27 11:00:00
20244    2012-04-27 12:00:00
20245    2012-04-27 13:00:00
20246    2012-04-27 14:00:00
Name: DATE, Length: 20242, dtype: datetime64[ns]
```

In [512]:
```
# Load file
bj_df = pd.read_csv('54511099999.csv')

#Separate the TMP columns with commas to form multiple columns of air temperature related data.
bj_tmp = bj_df['TMP'].str.split(',',expand=True)
bj_tmp.columns = ['tmp','tmpq']
bj_df = pd.concat([bj_df,bj_tmp], axis=1)
bj_df = bj_df.drop('TMP',axis=1)
bj_df['tmp'] = bj_df['tmp'].str.slice(0, 5).astype(float)

# Filter the data
bj_df = bj_df.loc[bj_df['tmp'] != 9999]

# Convert date to datetime
bj_df['DATE'] = pd.to_datetime(bj_df['DATE'], format="%Y-%m-%dT%H:%M:%S")
bj_df=bj_df[['DATE','tmp','tmpq']]

# Set date as index
bj_df
```

```
C:\Users\Administrator\AppData\Local\Temp\ipykernel_15676\637482396.py:2: DtypeWarning: Columns (19,25,28) have mixed types. Specify dtype option on import or set low_memory=False.
  bj_df = pd.read_csv('54511099999.csv')
```

Out[512]:

	DATE	tmp	tmpq
0	2022-01-01 00:00:00	-64.0	1
1	2022-01-01 00:00:00	-80.0	1
2	2022-01-01 00:30:00	-50.0	1
3	2022-01-01 01:00:00	-20.0	1
4	2022-01-01 01:30:00	-20.0	1
...	...	...	...
20242	2022-12-31 21:30:00	-70.0	1
20243	2022-12-31 22:00:00	-110.0	1
20244	2022-12-31 22:30:00	-110.0	1
20245	2022-12-31 23:00:00	-120.0	1
20246	2022-12-31 23:30:00	-90.0	1

20242 rows × 3 columns

3.3 Statistical characteristics of air temperature data

In [525]:
```
# 1 Calculate the average annual air temperature in Beijing
ann_avg=bj_df['tmp'].mean()/10
ann_avg
```

Out[525]: 12.923396897539769

In [524]:
```
# 2 Calculate the annual maximum air temperature in Beijing
max_avg=bj_df['tmp'].max()/10
max_avg
```

Out[524]: 38.4

In [526]:
```
# 3 Calculate the daily minimum air temperature in Beijing
min_avg=bj_df['tmp'].min()/10
min_avg
```

Out[526]: -16.0

In [530]:
```
# 4 Calculate the variance of annual average temperature in Beijing
bj_cels=bj_df['tmp']/10
var_avg=bj_cels.var()
var_avg
```

Out[530]: 154.64201666070753

In [531]:
```
# 5 Calculate the number of days with a temperature above 35 degrees
bj_df.loc[bj_df['tmp']>350].count()
```

Out[531]:
```
DATE    116
tmp     116
tmpq    116
dtype: int64
```