# Asg 19.7

# Dates and Times in Pandas

# (Coding)

**Files needed for this assignment:**

ufo.csv

In [3]:
```
# set up notebook to display multiple output in one cell

from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"

print('The notebook is set up to display multiple output in one cell.')
```

```
The notebook is set up to display multiple output in one cell.
```

In [4]:
```
# conventional way to import pandas and numpy
import pandas as pd
import numpy as np
from dateutil import parser
from datetime import datetime
from pandas.tseries.offsets import BDay
```

# PART ONE

<div class="alert alert-block alert-info"

# QUESTIONS 1-3:

### DATES AND TIMES IN PANDAS (TIMESTAMP & DATETIME INDEX)

**Question 1:**

Create individual timestamps by parsing the following formatted string dates:

- "1st of January, 2022"
- "February 14th, 2022"
- "4th of July, 2022"

In [5]:
```
date1 = parser.parse("1st of January, 2022")
date3 = parser.parse('4th of July, 2022')
date2 = parser.parse('February 14th, 2022')
print(date1,date2,date3)
```

```
2022-01-01 00:00:00 2022-02-14 00:00:00 2022-07-04 00:00:00
```

**Question 2:**

Use format codes to output the day of the week for the dates given in Question 1 above.

In [6]:
```
print(date1.strftime('%A'),date2.strftime('%A'),date3.strftime('%A'))
```

```
Saturday Monday Monday
```

**Question 3:**

a. Use a NumPy-style vectorized operation to convert the date_3 Timestamp object created in Question 1 into a DateTimeIndex object that contains the date_3 date and the next 14 days.
b. Use a NumPy-style vectorized operation to convert the date_2 Timestamp object created in Question 1 into a DateTimeIndex object that contains the date_2 date and the next 10 hours.
c. Use a NumPy-style vectorized operation to convert the date_2 Timestamp object created in Question 1 into a DateTimeIndex object that contains the date_3 date and the next 17 seconds.

In [7]:
```
print(date3 + pd.to_timedelta(np.arange(12), 'D'))
print(date3 + pd.to_timedelta(np.arange(10), 'H'))
print(date3 + pd.to_timedelta(np.arange(17), 'S'))
```

```
DatetimeIndex(['2022-07-04', '2022-07-05', '2022-07-06', '2022-07-07',
               '2022-07-08', '2022-07-09', '2022-07-10', '2022-07-11',
               '2022-07-12', '2022-07-13', '2022-07-14', '2022-07-15'],
              dtype='datetime64[ns]', freq=None)
DatetimeIndex(['2022-07-04 00:00:00', '2022-07-04 01:00:00',
               '2022-07-04 02:00:00', '2022-07-04 03:00:00',
               '2022-07-04 04:00:00', '2022-07-04 05:00:00',
               '2022-07-04 06:00:00', '2022-07-04 07:00:00',
               '2022-07-04 08:00:00', '2022-07-04 09:00:00'],
              dtype='datetime64[ns]', freq=None)
DatetimeIndex(['2022-07-04 00:00:00', '2022-07-04 00:00:01',
               '2022-07-04 00:00:02', '2022-07-04 00:00:03',
               '2022-07-04 00:00:04', '2022-07-04 00:00:05',
               '2022-07-04 00:00:06', '2022-07-04 00:00:07',
               '2022-07-04 00:00:08', '2022-07-04 00:00:09',
               '2022-07-04 00:00:10', '2022-07-04 00:00:11',
               '2022-07-04 00:00:12', '2022-07-04 00:00:13',
               '2022-07-04 00:00:14', '2022-07-04 00:00:15',
               '2022-07-04 00:00:16'],
              dtype='datetime64[ns]', freq=None)
```

# PART TWO

<div class="alert alert-block alert-info"

# QUESTIONS 4-5:

### PANDAS TIME SERIES: INDEXING BY TIME

**Question 4:**

a. Create a DateTimeIndex named "index" using the following dates:

- '2021-01-1'
- '2021-05-05'
- '2021-07-4'
- '2022-01-1'
- '2022-05-05'
- '2022-07-4'

b. Create a Pandas Series named "strings" using the list below. Use the DateTimeIndex created in Part (a) as the index for your Series.

our_list = ['New Year\'s Day This Year', 'Cinco de Mayo This Year', '4th of July This Year', 'New Year\'s Day Next Year', 'Cinco de Mayo Next Year','4th of July Next Year']

In [8]:
```
our_list = ['New Year\'s Day This Year', 'Cinco de Mayo This Year', '4th of July T
index = pd.DatetimeIndex(['2021-01-1',
'2021-05-05',
'2021-07-4',
'2022-01-1',
'2022-05-05',
'2022-07-4'])
strings = pd.Series(our_list, index=index)
strings
```

Out[8]:
```
2021-01-01    New Year's Day This Year
2021-05-05     Cinco de Mayo This Year
2021-07-04       4th of July This Year
2022-01-01    New Year's Day Next Year
2022-05-05     Cinco de Mayo Next Year
2022-07-04       4th of July Next Year
dtype: object
```

**Question 5:**

a. Pass the "strings" Series the appropriate index values to return the following elements of the "strings" Series:

- 'Cinco de Mayo This Year'
- '4th of July This Year',
- "New Year's Day Next Year",

b. Pass the "strings" Series the appropriate index values to return the all of the elements of the "strings" Series from 2021.

In [9]:
```
print(strings['2021-05-05'])
print(strings['2021-07-04'])
print(strings['2021-01-01'])
print(strings[:])
```

```
Cinco de Mayo This Year
4th of July This Year
New Year's Day This Year
2021-01-01    New Year's Day This Year
2021-05-05     Cinco de Mayo This Year
2021-07-04       4th of July This Year
2022-01-01    New Year's Day Next Year
2022-05-05     Cinco de Mayo Next Year
2022-07-04       4th of July Next Year
dtype: object
```

# PART THREE

<div class="alert alert-block alert-info"

# QUESTIONS 6-10:

### PANDAS TIME SERIES DATA STRUCTURES

**Question 6:**

Use the pd.to_datetime() function to create individual timestamps for the following formatted string dates:

- "19th of March, 2016" ... name this timestamp date_1
- "April 3rd, 2019" ... name this timestamp date_2
- "November 18th, 2020" ... name this timestamp date_3
- "21st of December, 2021" ... name this timestamp date_4

In [10]:
```
date_1 = pd.to_datetime("19th of March, 2016")
date_2 = pd.to_datetime("April 3rd, 2019")
date_3 = pd.to_datetime("November 18th, 2020")
date_4 = pd.to_datetime("21st of December, 2021")
print(date_1,date_2,date_3,date_4)
```

```
2016-03-19 00:00:00 2019-04-03 00:00:00 2020-11-18 00:00:00 2021-12-21 00:00:00
```

**Question 7:**

Use the **to_period() function** along with the addition of a frequency code to convert the DatetimeIndexes from Question 6 to PeriodIndexes following the guidelines found below:

- date_1 --> frequency = day
- date_2 --> frequency = week
- date_3 --> frequency = month
- date_4 --> frequency = year

In [11]:
```
print(date_1.to_period('D'))
print(date_2.to_period('W'))
print(date_3.to_period('M'))
print(date_4.to_period('Y'))
```

```
2016-03-19
2019-04-01/2019-04-07
2020-11
2021
```

**Question 8:**

Use the **pd.to_datetime function** to create a **DatetimeIndex** named "dates"using the following dates:

- datetime(2021, 12, 25)
- 4th of July, 2022
- 2022-Aug-16
- 10-10-2022
- 20221125

**Note:** To do this question, first run the following line in your code cell

- **from** datetime **import** datetime

In [12]:
```
dates = pd.to_datetime([datetime(2021, 12, 25),"4th of July, 2022","2022-Aug
print(dates)
```

```
DatetimeIndex(['2021-12-25', '2022-07-04', '2022-08-16', '2022-10-10',
               '2022-11-25'],
              dtype='datetime64[ns]', freq=None)
```

**Question 9:**

Use the **to_period() function** along with the addition of a frequency code to convert the "dates" **DatetimeIndex** from Question 8 to **PeriodIndexes** following the guidelines found below:

a. PeriodIndex frequency = day
b. PeriodIndex frequency = week
c. PeriodIndex frequency = month
d. PeriodIndex frequency = year

In [13]:
```
print(dates.to_period('D'))
print(dates.to_period('W'))
print(dates.to_period('M'))
print(dates.to_period('Y'))
```

```
PeriodIndex(['2021-12-25', '2022-07-04', '2022-08-16', '2022-10-10',
             '2022-11-25'],
            dtype='period[D]')
PeriodIndex(['2021-12-20/2021-12-26', '2022-07-04/2022-07-10',
             '2022-08-15/2022-08-21', '2022-10-10/2022-10-16',
             '2022-11-21/2022-11-27'],
            dtype='period[W-SUN]')
PeriodIndex(['2021-12', '2022-07', '2022-08', '2022-10', '2022-11'], dtype
='period[M]')
PeriodIndex(['2021', '2022', '2022', '2022', '2022'], dtype='period[A-DE
C]')
```

**Question 10:**

Create a **TimedeltaIndex** by subtracting the first date in the "dates" **DatetimeIndex** (see Question 8) from all of the dates in the "dates" **DatetimeIndex**

In [14]:
```
for date in dates:
    print(dates - date)
```

```
TimedeltaIndex(['0 days', '191 days', '234 days', '289 days', '335 day
s'], dtype='timedelta64[ns]', freq=None)
TimedeltaIndex(['-191 days', '0 days', '43 days', '98 days', '144 days'],
dtype='timedelta64[ns]', freq=None)
TimedeltaIndex(['-234 days', '-43 days', '0 days', '55 days', '101 day
s'], dtype='timedelta64[ns]', freq=None)
TimedeltaIndex(['-289 days', '-98 days', '-55 days', '0 days', '46 day
s'], dtype='timedelta64[ns]', freq=None)
TimedeltaIndex(['-335 days', '-144 days', '-101 days', '-46 days', '0 day
s'], dtype='timedelta64[ns]', freq=None)
```

# PART FOUR

<div class="alert alert-block alert-info"

# QUESTIONS 11-15:

**REGULAR SEQUENCES USING pd.date_range()**

**Question 11:**

Use the **pd.date_range() function** to create a sequence of timestamps that consists of the days from Jan. 3, 2022 to Jan. 27, 2022.

In [15]:
```
pd.date_range('01-03-2022','01-27-2022')
```

Out[15]:
```
DatetimeIndex(['2022-01-03', '2022-01-04', '2022-01-05', '2022-01-06',
               '2022-01-07', '2022-01-08', '2022-01-09', '2022-01-10',
               '2022-01-11', '2022-01-12', '2022-01-13', '2022-01-14',
               '2022-01-15', '2022-01-16', '2022-01-17', '2022-01-18',
               '2022-01-19', '2022-01-20', '2022-01-21', '2022-01-22',
               '2022-01-23', '2022-01-24', '2022-01-25', '2022-01-26',
               '2022-01-27'],
              dtype='datetime64[ns]', freq='D')
```

**Question 12:**

Use the **pd.date_range() function** to create a sequence of timestampd that consists of the 10 consecutive days starting with March 13, 2020.

In [16]:
```
pd.date_range('03-13-2020',periods=10, freq='D')
```

Out[16]:
```
DatetimeIndex(['2020-03-13', '2020-03-14', '2020-03-15', '2020-03-16',
               '2020-03-17', '2020-03-18', '2020-03-19', '2020-03-20',
               '2020-03-21', '2020-03-22'],
              dtype='datetime64[ns]', freq='D')
```

**Question 13:**

Use the **pd.date_range() function** to create a sequence of 28 consecutive hourly timestamps starting on May 11, 2016.

In [17]:
```
pd.date_range('05-11-2016',periods=28, freq='H')
```

Out[17]:
```
DatetimeIndex(['2016-05-11 00:00:00', '2016-05-11 01:00:00',
               '2016-05-11 02:00:00', '2016-05-11 03:00:00',
               '2016-05-11 04:00:00', '2016-05-11 05:00:00',
               '2016-05-11 06:00:00', '2016-05-11 07:00:00',
               '2016-05-11 08:00:00', '2016-05-11 09:00:00',
               '2016-05-11 10:00:00', '2016-05-11 11:00:00',
               '2016-05-11 12:00:00', '2016-05-11 13:00:00',
               '2016-05-11 14:00:00', '2016-05-11 15:00:00',
               '2016-05-11 16:00:00', '2016-05-11 17:00:00',
               '2016-05-11 18:00:00', '2016-05-11 19:00:00',
               '2016-05-11 20:00:00', '2016-05-11 21:00:00',
               '2016-05-11 22:00:00', '2016-05-11 23:00:00',
               '2016-05-12 00:00:00', '2016-05-12 01:00:00',
               '2016-05-12 02:00:00', '2016-05-12 03:00:00'],
              dtype='datetime64[ns]', freq='H')
```

**Question 14:**

Use the **pd.timedelta_range() function** to create a **TimedeltaIndex** that has 12 periods, and has a frequency of 8 hours and 15 minutes.

In [18]:
```
pd.timedelta_range(0,periods=12, freq='8H15T')
```

Out[18]:
```
TimedeltaIndex(['0 days 00:00:00', '0 days 08:15:00', '0 days 16:30:00',
                '1 days 00:45:00', '1 days 09:00:00', '1 days 17:15:00',
                '2 days 01:30:00', '2 days 09:45:00', '2 days 18:00:00',
                '3 days 02:15:00', '3 days 10:30:00', '3 days 18:45:00'],
               dtype='timedelta64[ns]', freq='495T')
```

**Question 15:**

Use the **pd.date_range() function** to create a **DatetimeIndex** that starts on Dec. 13, 2021, has 7 periods, and has a business day offset.

**Note:** To do this question, first run the following line in your code cell

- **from** pandas.tseries.offsets **import** BDay

In [19]:
```
pd.date_range('12-13-2021',periods=7,freq=BDay())
```

Out[19]:
```
DatetimeIndex(['2021-12-13', '2021-12-14', '2021-12-15', '2021-12-16',
               '2021-12-17', '2021-12-20', '2021-12-21'],
              dtype='datetime64[ns]', freq='B')
```

# PART FIVE

<div class="alert alert-block alert-info"

# QUESTIONS 16-:

### DATES AND TIMES IN PANDAS -- ADDITIONAL QUESTIONS

For Questions 16-29: We will be using the **'ufo.csv' dataset** and the **ufo DataFrame**. 

**Question 16:**

Read in the dataset found at **'http://bit.ly/uforeports'** (or use the link **ufo.csv**) and store the results in a DataFrame named **ufo**.

In [20]:
```
ufo = pd.read_csv('ufo.csv')
ufo
```

Out[20]:

| | City | Colors Reported | Shape Reported | State | Time |
|---|---|---|---|---|---|
| **0** | Ithaca | NaN | TRIANGLE | NY | 6/1/1930 22:00 |
| **1** | Willingboro | NaN | OTHER | NJ | 6/30/1930 20:00 |
| **2** | Holyoke | NaN | OVAL | CO | 2/15/1931 14:00 |
| **3** | Abilene | NaN | DISK | KS | 6/1/1931 13:00 |
| **4** | New York Worlds Fair | NaN | LIGHT | NY | 4/18/1933 19:00 |
| **...** | ... | ... | ... | ... | ... |
| **18236** | Grant Park | NaN | TRIANGLE | IL | 12/31/2000 23:00 |
| **18237** | Spirit Lake | NaN | DISK | IA | 12/31/2000 23:00 |
| **18238** | Eagle River | NaN | NaN | WI | 12/31/2000 23:45 |
| **18239** | Eagle River | RED | LIGHT | WI | 12/31/2000 23:45 |
| **18240** | Ybor | NaN | OVAL | FL | 12/31/2000 23:59 |

18241 rows × 5 columns

**Question 17:**

Use the **dtypes attribute** to check the data type of the 'Time' column.

In [21]:
```
ufo.dtypes
```

Out[21]:
```
City              object
Colors Reported   object
Shape Reported    object
State             object
Time              object
dtype: object
```

**Question 18:**

Convert 'Time' from a string to a datetime format and then use the **dtypes attribute** to check that the 'Time' column is now a datetime object.

In [22]:
```
ufo['Time'] = pd.to_datetime(ufo.Time)
ufo.dtypes
```

Out[22]:
```
City                        object
Colors Reported             object
Shape Reported              object
State                       object
Time                datetime64[ns]
dtype: object
```

Documentation for `to_datetime`

**Question 19:**

Use the **dt.hour attribute** to find the hour for each item in the 'Time" column. Use the head() method to view the first 5 rows of this Series.

In [23]:
```
pd.Series(ufo['Time'].head()).dt.hour
```

Out[23]:
```
0    22
1    20
2    14
3    13
4    19
Name: Time, dtype: int64
```

**Question 20:**

Use the **dt.minute attribute** to find the minutes for each item in the 'Time" column. Use the tail() method to view the last 5 rows of this Series.

In [24]:
```
pd.Series(ufo['Time'].tail()).dt.minute
```

Out[24]:
```
18236     0
18237     0
18238    45
18239    45
18240    59
Name: Time, dtype: int64
```

**Question 21:**

Use the **value_counts and sort_values methods** to find the most common day of the week for sighting a UFO.

In [25]:
```
pd.Series(ufo['Time']).dt.dayofweek.value_counts()
```

Out[25]:
```
1    2822
6    2689
5    2687
4    2669
3    2598
2    2476
0    2300
Name: Time, dtype: int64
```

**Question 22:**

Use the **dt.dayofyear attribute** to find the day of the year for each item in the 'Time" column. Use the head() method to view the first 5 rows of this Series.

In [26]:
```
ufo.Time.dt.dayofyear.head()
```

Out[26]:
```
0    152
1    181
2     46
3    152
4    108
Name: Time, dtype: int64
```

**Question 23:**

Use the **value_counts and sort_values methods** to find the most common day of the year for sighting a UFO.

In [32]:
```
pd.Series(ufo['Time']).dt.dayofyear.value_counts().sort
```

Out[32]:
```
152    651
166    472
196    374
182    352
227    311
      ...
68      13
77      12
19      11
123     11
21      10
Name: Time, Length: 366, dtype: int64
```

API reference for **datetime properties and methods**

**Question 24:**

Convert the string '4th of July, 2000' to the datetime format to output a timestamp object that is named 'timestamp'.

In [44]:
```
timestamp = pd.Timestamp(2000, 7, 4)
timestamp
```

Out[44]:
```
Timestamp('2000-07-04 00:00:00')
```

**Question 25:**

a. Find all of the UFO's that were sighted after the timestamp from Question 25 (i.e. after the '4th of July, 2000').
b. How many of UFO's were sighted after the timestamp from Question 25 (i.e. after the '4th of July, 2000').

In [53]:
```
ufo.loc[(ufo['Time'] > timestamp)]
ufo.loc[(ufo['Time'] < timestamp)]
```

Out[53]:

| | City | Colors Reported | Shape Reported | State | Time |
|---|---|---|---|---|---|
| **16783** | Colorado Springs | NaN | LIGHT | CO | 2000-07-04 02:30:00 |
| **16784** | Columbus | NaN | LIGHT | GA | 2000-07-04 03:00:00 |
| **16785** | Loves Park | NaN | CIGAR | IL | 2000-07-04 17:30:00 |
| **16786** | Potsdam | NaN | TRIANGLE | NY | 2000-07-04 18:00:00 |
| **16787** | St. Louis | NaN | FIREBALL | MO | 2000-07-04 20:40:00 |
| **...** | ... | ... | ... | ... | ... |
| **18236** | Grant Park | NaN | TRIANGLE | IL | 2000-12-31 23:00:00 |
| **18237** | Spirit Lake | NaN | DISK | IA | 2000-12-31 23:00:00 |
| **18238** | Eagle River | NaN | NaN | WI | 2000-12-31 23:45:00 |
| **18239** | Eagle River | RED | LIGHT | WI | 2000-12-31 23:45:00 |
| **18240** | Ybor | NaN | OVAL | FL | 2000-12-31 23:59:00 |

1458 rows × 5 columns

Out[53]:

| | City | Colors Reported | Shape Reported | State | Time |
|---|---|---|---|---|---|
| **0** | Ithaca | NaN | TRIANGLE | NY | 1930-06-01 22:00:00 |
| **1** | Willingboro | NaN | OTHER | NJ | 1930-06-30 20:00:00 |
| **2** | Holyoke | NaN | OVAL | CO | 1931-02-15 14:00:00 |
| **3** | Abilene | NaN | DISK | KS | 1931-06-01 13:00:00 |
| **4** | New York Worlds Fair | NaN | LIGHT | NY | 1933-04-18 19:00:00 |
| **...** | ... | ... | ... | ... | ... |
| **16778** | Cottage Grove | NaN | RECTANGLE | WA | 2000-07-03 17:00:00 |
| **16779** | Meridian | NaN | OTHER | ID | 2000-07-03 17:00:00 |
| **16780** | Boston | NaN | LIGHT | NY | 2000-07-03 19:00:00 |
| **16781** | Anderson | NaN | LIGHT | CA | 2000-07-03 21:45:00 |
| **16782** | Colebrook | NaN | VARIOUS | NH | 2000-07-03 22:30:00 |

16783 rows × 5 columns

**Question 26:**

How many of UFO's were sighted between 'New Year\'s Day, 1998' and the '4th of July, 2000').

In [54]:
```
ufo.loc[(ufo['Time'] < timestamp) & (ufo['Time'] >
```

Out[54]:

| | City | Colors Reported | Shape Reported | State | Time |
|---|---|---|---|---|---|
| **11089** | St. Louis | BLUE | FLASH | MO | 1998 01-0 00:58:0 |
| **11090** | Allentown | NaN | LIGHT | PA | 1998 01-0 01:42:0 |
| **11091** | Elizabethtown | NaN | TRIANGLE | IL | 1998 01-0 02:00:0 |
| **11092** | Huelo | NaN | LIGHT | HI | 1998 01-0 04:00:0 |
| **11093** | Hoboken | NaN | DISK | NJ | 1998 01-0 06:00:0 |
| **...** | ... | ... | ... | ... | .. |
| **16778** | Cottage Grove | NaN | RECTANGLE | WA | 2000 07-0 17:00:0 |
| **16779** | Meridian | NaN | OTHER | ID | 2000 07-0 17:00:0 |
| **16780** | Boston | NaN | LIGHT | NY | 2000 07-0 19:00:0 |
| **16781** | Anderson | NaN | LIGHT | CA | 2000 07-0 21:45:0 |
| **16782** | Colebrook | NaN | VARIOUS | NH | 2000 07-0 22:30:0 |

5694 rows × 5 columns

**Question 27:**

How much time was there between the first UFO sighting and the last UFO sighting?

**Hint**: Use the **Time attribute**, along with the **max() and min() methods** and then subtract timestamps to output a **timedelta object**.

In [93]:
```
ufo.Time.max() - ufo.Time.min()
```

Out[93]:
```
Timedelta('25781 days 01:59:00')
```

**Question 28:**

a. Create a new column called 'Year' ... Hint: To do this use the Time, dt, and year Series attibutes
b. Use the value_counts() and sort_index() methods to count the number of UFO reports per year ... use the head() method to view the first 5 rows of the Series that will be created
c. During which year were there the most UFO sightings and how many UFO sightings were there during this year?

In [112]:
```
ufo['Year'] = ufo['Time'].dt.year
pd.Series(ufo['Year'].value_counts()).head()
```

Out[112]:
```
1999    2774
2000    2635
1998    1743
1995    1344
1997    1237
Name: Year, dtype: int64
```

## Run the command below to allow plots to appear in the notebook

%matplotlib inline

In [27]:
```
%matplotlib inline
```

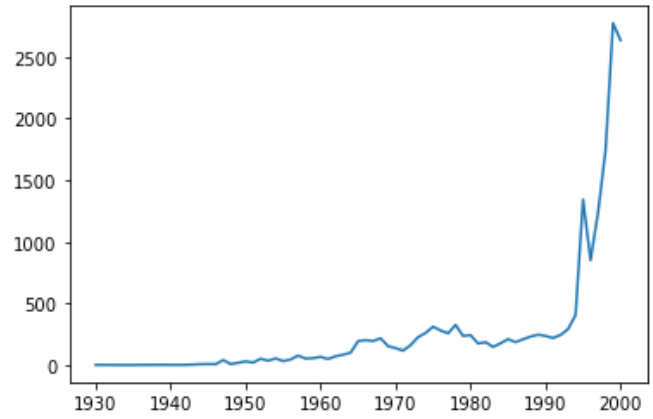
**Question 29:**

Plot the number of UFO reports per year (line plot is the default).

Use the code that has been provided.

In [106]:
```
# plot the number of UFO reports per year (lin
ufo.Year.value_counts().sort_index().plot()
```

Out[106]:
```
<AxesSubplot:>
```


2500
2000
1500
1000
500
0
1930
1940
1950
1960
1970
1980
1990
2000

In [ ]: