Get the Data

Either use the provided .csv file or (optionally) get fresh (the freshest?) data from running an SQL query on StackExchange:

Follow this link to run the query from StackExchange to get your own .csv file

```
select dateadd(month, datediff(month, 0, q.CreationDate), 0) m, TagName, count(*) from PostTags pt join Posts q on q.Id=pt.PostId join Tags t on t.Id=pt.TagId where TagName in ('java','c','c++','python','c#','javascript','assembly','php','perl','ruby','visual basic','swift','r','object-c','scratch','go','swift','delphi') and q.CreationDate < dateadd(month, datediff(month, 0, getdate()), 0) group by dateadd(month, datediff(month, 0, q.CreationDate), 0), TagName order by dateadd(month, datediff(month, 0, q.CreationDate), 0)
```

Import Statements

```
import pandas as pd
import matplotlib.pyplot as plt
```

Data Exploration

Challenge: Read the .csv file and store it in a Pandas dataframe

```
df = pd.read_csv('/content/SE_Code_Languages.csv')
df = df.rename(columns={'m': 'Date', 'Unnamed: 2': 'TagCount'})
## Easier
##     df = pd.read_csv('QueryResults.csv', names=['DATE', 'TAG', 'POSTS'], header=0)
```

Challenge: Examine the first 5 rows and the last 5 rows of the of the dataframe

```
print(df.head())
df.tail()
```

```
                  Date   TagName  TagCount
0  2008-07-01 00:00:00        c#         3
1  2008-08-01 00:00:00  assembly         8
2  2008-08-01 00:00:00         c        83
3  2008-08-01 00:00:00        c#       505
4  2008-08-01 00:00:00       c++       164
```

	Date	TagName	TagCount
2365	2022-10-01 00:00:00	php	3825
2366	2022-10-01 00:00:00	python	23818
2367	2022-10-01 00:00:00	r	4638
2368	2022-10-01 00:00:00	ruby	443
2369	2022-10-01 00:00:00	swift	1778

Challenge: Check how many rows and how many columns there are. What are the dimensions of the dataframe?

✓ 0s completed at 2:35 PM

```
(2370, 3)
```

Challenge: Count the number of entries in each column of the dataframe

```
df.count()
df.nunique()
```

```
Date          172
TagName        14
TagCount     1904
dtype: int64
```

Challenge: Calculate the total number of post per language. Which Programming language has had the highest total number of posts of all time?

```
df.groupby('TagName').sum('TagCount').sort_values('TagCount', ascending=False)
```

	TagCount
TagName	
javascript	2438306
python	2045942
java	1870795
c#	1564572
php	1447622
c++	779653
r	468491
c	387439
swift	320154
ruby	226112
perl	67198
go	65286
delphi	50234
assembly	41406

Some languages are older (e.g., C) and other languages are newer (e.g., Swift). The dataset starts in September 2008.

Challenge: How many months of data exist per language? Which language had the fewest months with an entry?

```
df.groupby('TagName').nunique('Date').sort_values('Date')
```

	Date	TagCount
TagName		
go	156	135
swift	163	109

r	169	165
assembly	171	138
c	171	164
c++	171	170
delphi	171	144
java	171	168
javascript	171	170
perl	171	150
php	171	167
python	171	171
ruby	171	161
c#	172	171

Data Cleaning

Let's fix the date format to make it more readable. We need to use Pandas to change format from a string of "2008-07-01 00:00:00" to a datetime object with the format of "2008-07-01"

```
df['Date'][1]
```

```
'2008-08-01 00:00:00'
```

```
df['Date'] = pd.to_datetime(df['Date'])
```

```
df.head()
```

	Date	TagName	TagCount
0	2008-07-01	c#	3
1	2008-08-01	assembly	8
2	2008-08-01	c	83
3	2008-08-01	c#	505
4	2008-08-01	c++	164

Data Manipulation

```
reshaped_df = df.pivot(index='Date', columns='TagName', values = 'TagCount')
```

Challenge: What are the dimensions of our new dataframe? How many rows and columns does it have? Print out the column names and print out the first 5 rows of the dataframe.

```
reshaped_df.shape
```

```
(172, 14)
```

```
(172, 14)
```

```
reshaped_df.columns
```

```
Index(['assembly', 'c', 'c#', 'c++', 'delphi', 'go', 'java', 'javascript',
       'perl', 'php', 'python', 'r', 'ruby', 'swift'],
      dtype='object', name='TagName')
```

```
reshaped_df.head()
```

TagName	assembly	c	c#	c++	delphi	go	java	javascript	perl	php	python	r	ruby	s
Date														
2008-07-01	NaN	NaN	3.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
2008-08-01	8.0	83.0	505.0	164.0	14.0	NaN	222.0	164.0	28.0	158.0	120.0	NaN	70.0	
2008-09-01	28.0	318.0	1646.0	753.0	104.0	NaN	1131.0	635.0	130.0	476.0	537.0	6.0	287.0	
2008-10-01	15.0	303.0	1988.0	808.0	112.0	NaN	1149.0	724.0	127.0	612.0	508.0	NaN	247.0	
2008-11-01	17.0	259.0	1731.0	734.0	141.0	NaN	957.0	581.0	97.0	501.0	451.0	1.0	158.0	

reshaped_df.head()**Challenge**: Count the number of entries per programming language. Why might the number of entries be different?

```
reshaped_df.count()
```

```
TagName
assembly      171
c             171
c#            172
c++           171
delphi        171
go            156
java          171
javascript    171
perl          171
php           171
python        171
r             169
ruby          171
swift         163
dtype: int64
```

```
reshaped_df.fillna(0, inplace=True)
```

```
reshaped_df.head()
```

TagName	assembly	c	c#	c++	delphi	go	java	javascript	perl	php	python	r	ruby	swi
Date														
2008-07-01	0.0	0.0	3.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
2008-08-01	8.0	83.0	505.0	164.0	14.0	0.0	222.0	164.0	28.0	158.0	120.0	0.0	70.0	
2008-09-01	28.0	318.0	1646.0	753.0	104.0	0.0	1131.0	635.0	130.0	476.0	537.0	6.0	287.0	
2008-10-01	15.0	303.0	1988.0	808.0	112.0	0.0	1149.0	724.0	127.0	612.0	508.0	0.0	247.0	
2008-11-01	17.0	259.0	1731.0	734.0	141.0	0.0	957.0	581.0	97.0	501.0	451.0	1.0	158.0	

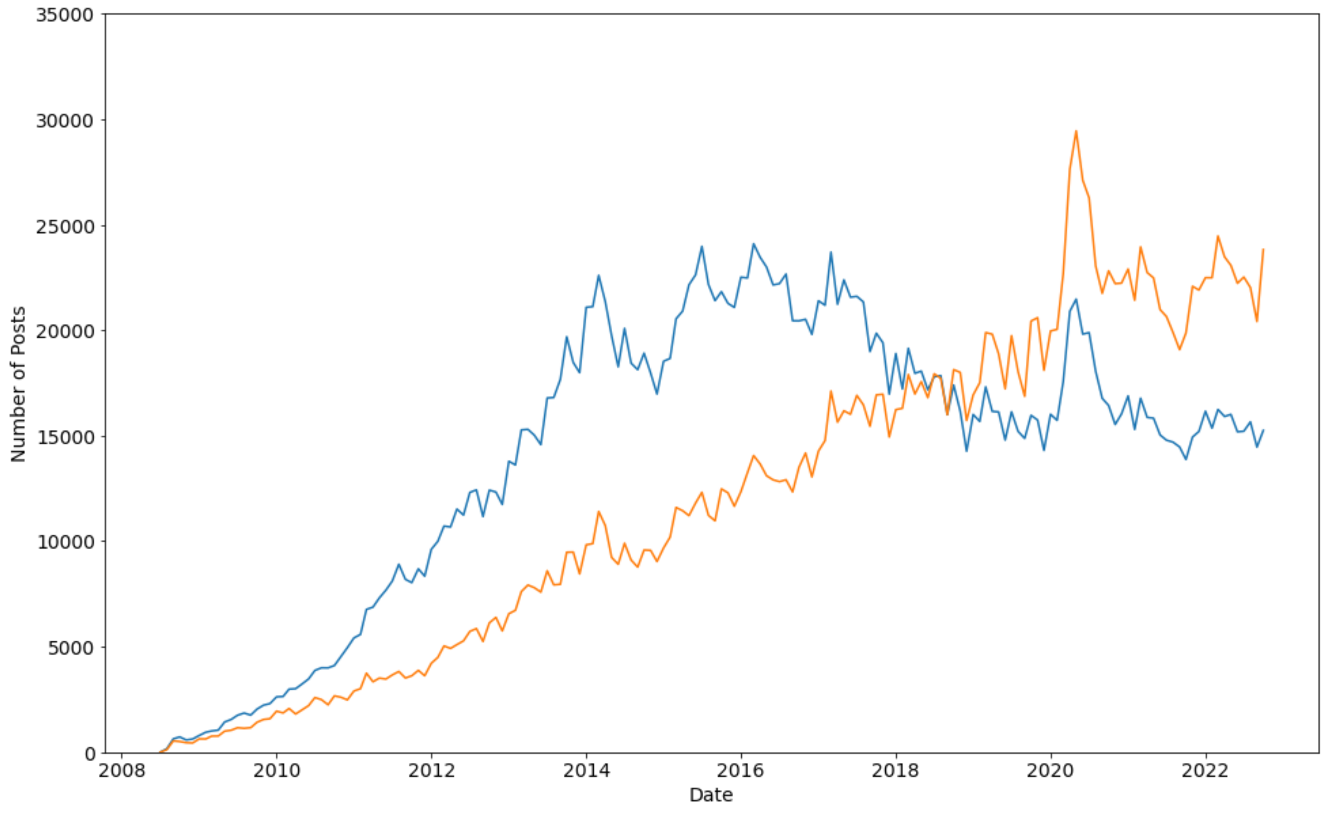
Data Visualisaton with with Matplotlib

Challenge: Use the matplotlib documentation to plot a single programming language (e.g., java) on a chart.

Challenge: Show two line (e.g. for Java and Python) on the same chart.

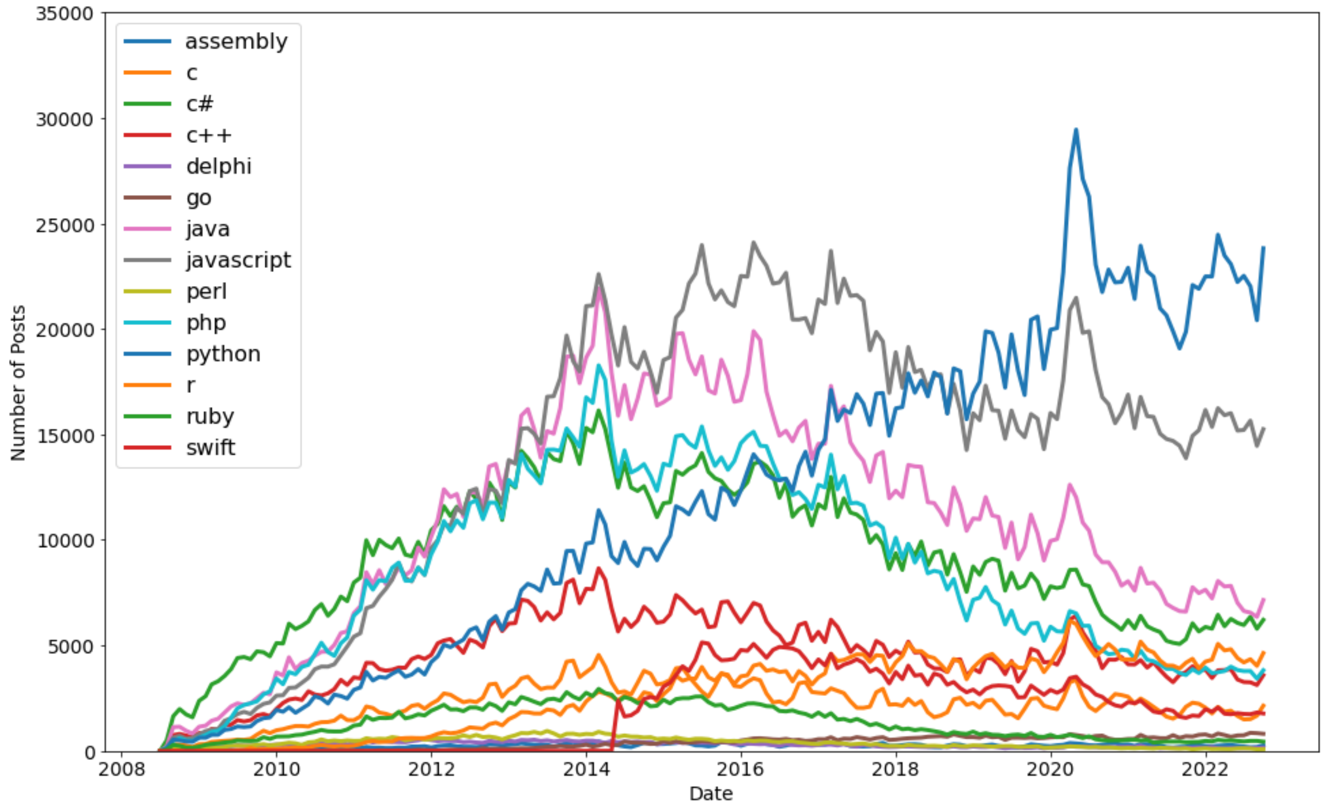
```
#Increase Graph Size
plt.figure(figsize=(16,10))
#Increase Font size
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
#Label Axis
plt.xlabel('Date', fontsize=14)
plt.ylabel('Number of Posts', fontsize=14)
#Limit the Plot
plt.ylim(0, 35000)
#Make Plot
plt.plot(reshaped_df.index, reshaped_df.javascript)
plt.plot(reshaped_df.index, reshaped_df.python)
```

```
[<matplotlib.lines.Line2D at 0x7f81ea45ef50>]
```



```
    #Increase Graph Size
plt.figure(figsize=(16,10))
#Increase Font size
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
#Label Axis
plt.xlabel('Date', fontsize=14)
plt.ylabel('Number of Posts', fontsize=14)
#Limit the Plot
plt.ylim(0, 35000)
    for column in reshaped_df.columns:
        plt.plot(reshaped_df.index, reshaped_df[column],
             linewidth=3, label=reshaped_df[column].name)

plt.legend(fontsize=16)
```

```
<matplotlib.legend.Legend at 0x7f81e89ddd10>
```

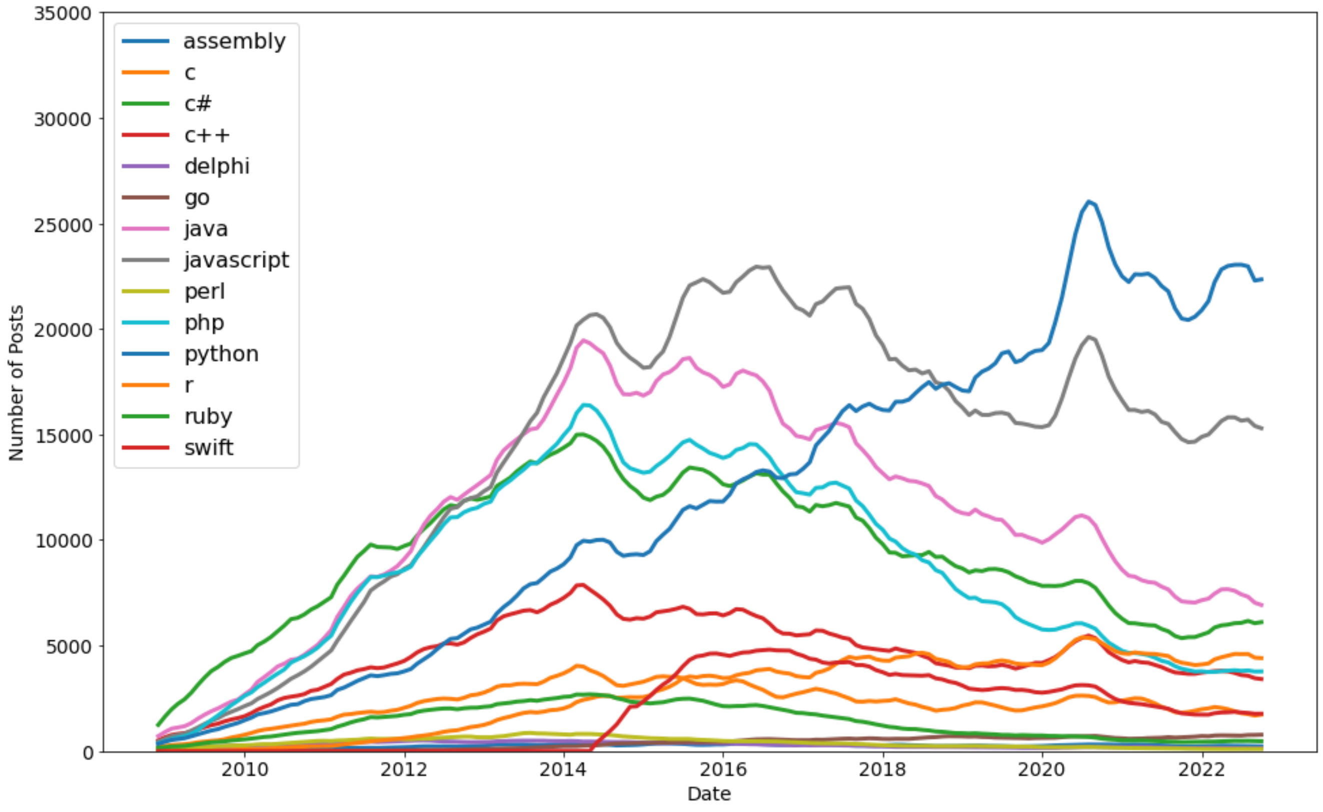


Smoothing out Time Series Data

Time series data can be quite noisy, with a lot of up and down spikes. To better see a trend we can plot an average of, say 6 or 12 observations. This is called the rolling mean. We calculate the average in a window of time and move it forward by one overservation. Pandas has two handy methods already built in to work this out: rolling() and mean().

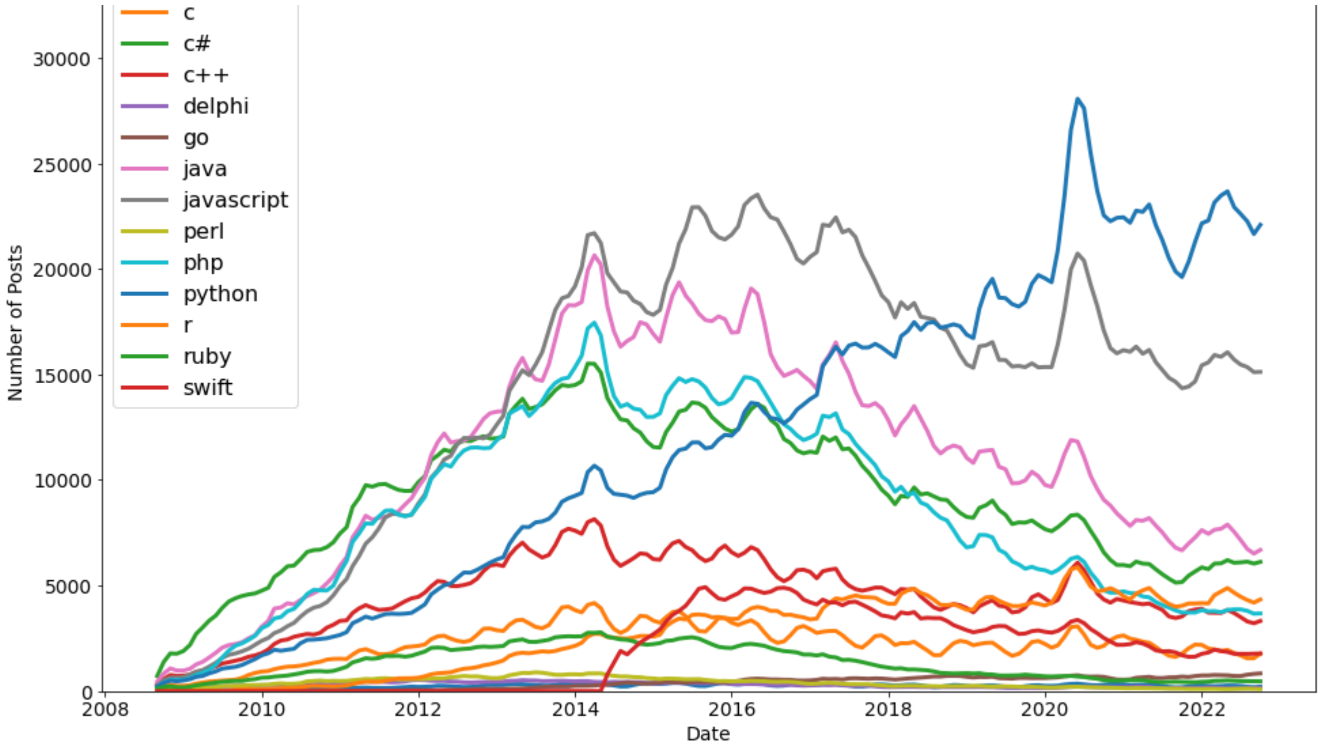
```
# The window is number of observations that are averaged
roll_df = reshaped_df.rolling(window=6).mean()

plt.figure(figsize=(16,10))
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
plt.xlabel('Date', fontsize=14)
plt.ylabel('Number of Posts', fontsize=14)
plt.ylim(0, 35000)

# plot the roll_df instead
for column in roll_df.columns:
    plt.plot(roll_df.index, roll_df[column],
             linewidth=3, label=roll_df[column].name)

plt.legend(fontsize=16)
```

```
<matplotlib.legend.Legend at 0x7f81e88c9450>
```



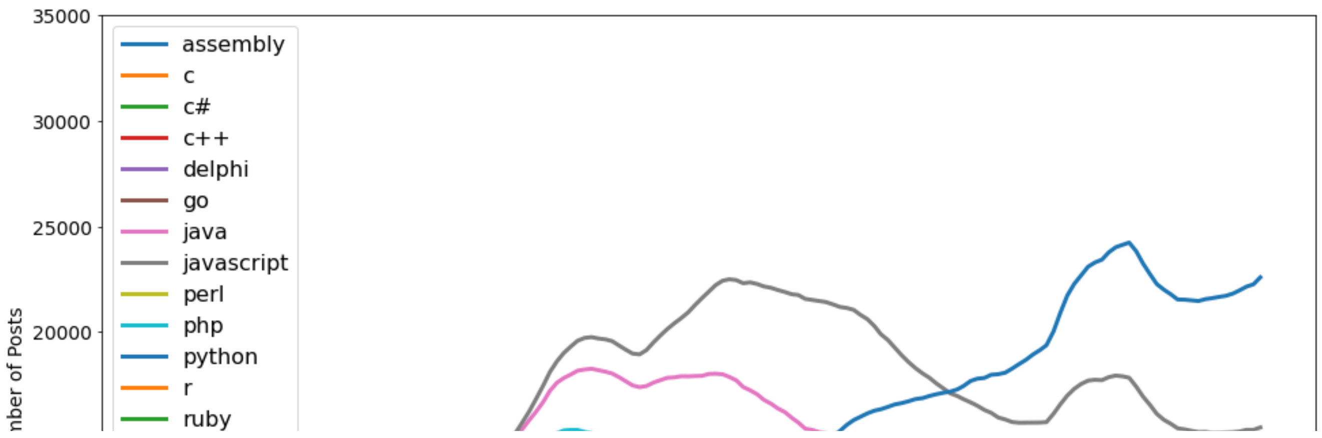
```
# The window is number of observations that are averaged
roll_df = reshaped_df.rolling(window=3).mean()

plt.figure(figsize=(16,10))
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
plt.xlabel('Date', fontsize=14)
plt.ylabel('Number of Posts', fontsize=14)
plt.ylim(0, 35000)

# plot the roll_df instead
for column in roll_df.columns:
    plt.plot(roll_df.index, roll_df[column],
             linewidth=3, label=roll_df[column].name)

plt.legend(fontsize=16)
```

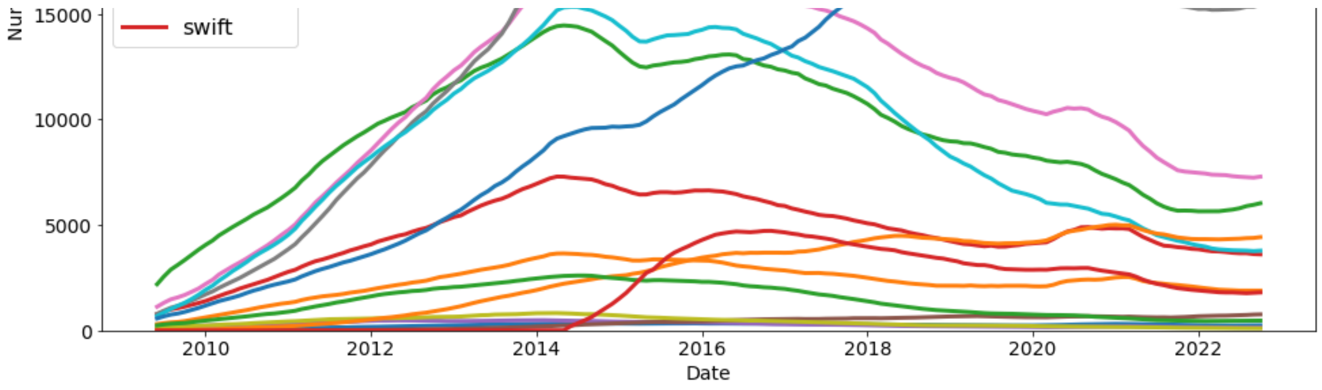
```
<matplotlib.legend.Legend at 0x7f81e88372d0>
```



```
# The window is number of observations that are averaged
roll_df = reshaped_df.rolling(window=12).mean()

plt.figure(figsize=(16,10))
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
plt.xlabel('Date', fontsize=14)
plt.ylabel('Number of Posts', fontsize=14)
plt.ylim(0, 35000)

# plot the roll_df instead
for column in roll_df.columns:
    plt.plot(roll_df.index, roll_df[column],
             linewidth=3, label=roll_df[column].name)

plt.legend(fontsize=16)
```

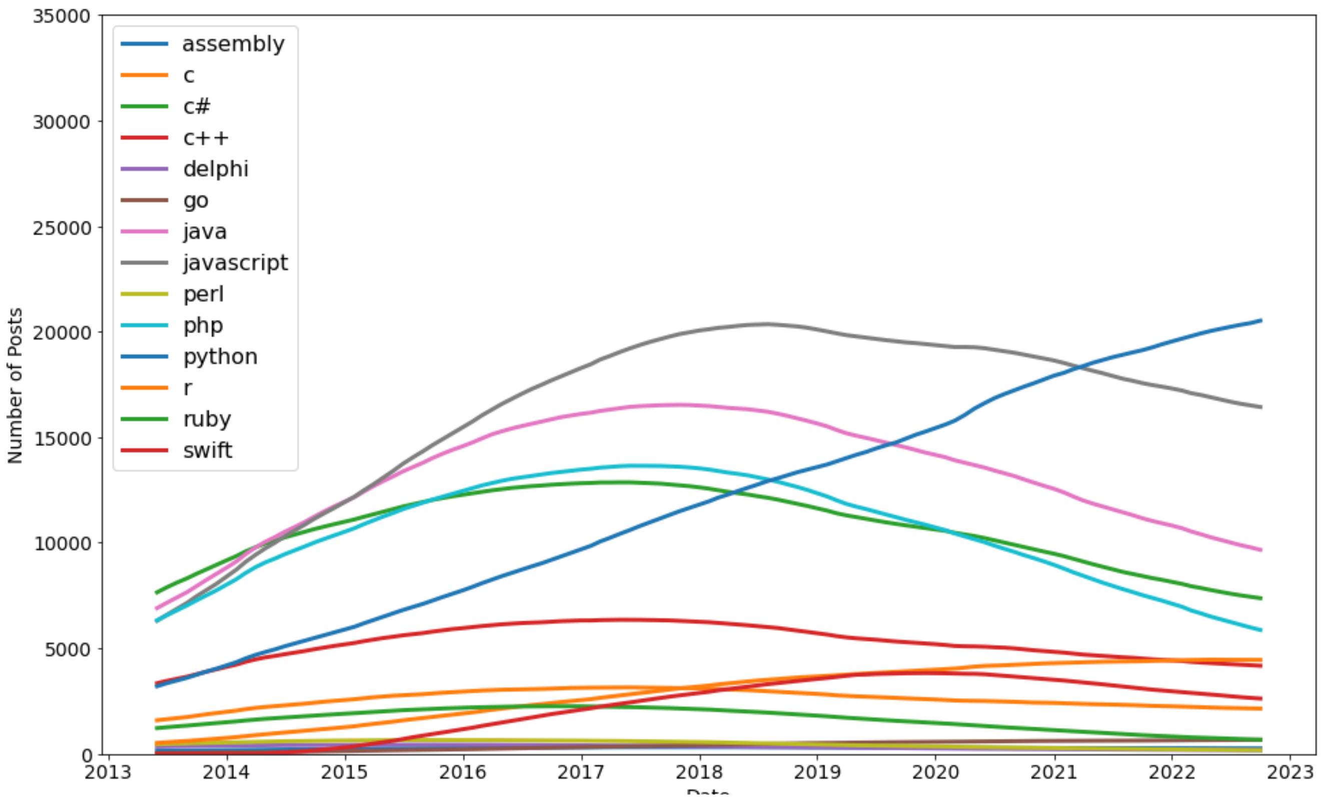
```
<matplotlib.legend.Legend at 0x7f81e8734410>
```





```
# The window is number of observations that are averaged
roll_df = reshaped_df.rolling(window=60).mean()

plt.figure(figsize=(16,10))
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
plt.xlabel('Date', fontsize=14)
plt.ylabel('Number of Posts', fontsize=14)
plt.ylim(0, 35000)

# plot the roll_df instead
for column in roll_df.columns:
    plt.plot(roll_df.index, roll_df[column],
             linewidth=3, label=roll_df[column].name)

plt.legend(fontsize=16)
```

```
<matplotlib.legend.Legend at 0x7f81e8613910>
```



Date