Now that you can create your own line charts, it's time to learn about more chart types!

> By the way, if this is your first experience with writing code in Python, you should be *very proud* of all that you have accomplished so far, because it's never easy to learn a completely new skill! If you stick with the micro-course, you'll notice that everything will only get easier (while the charts you'll build will get more impressive!), since the code is pretty similar for all of the charts. Like any skill, coding becomes natural over time, and with repetition.

In this tutorial, you'll learn about **bar charts** and **heatmaps.**

Set up the notebook

As always, we begin by setting up the coding environment. (*This code is hidden, but you can un-hide it by clicking on the "Code" button immediately below this text, on the right.*)

Hide Output

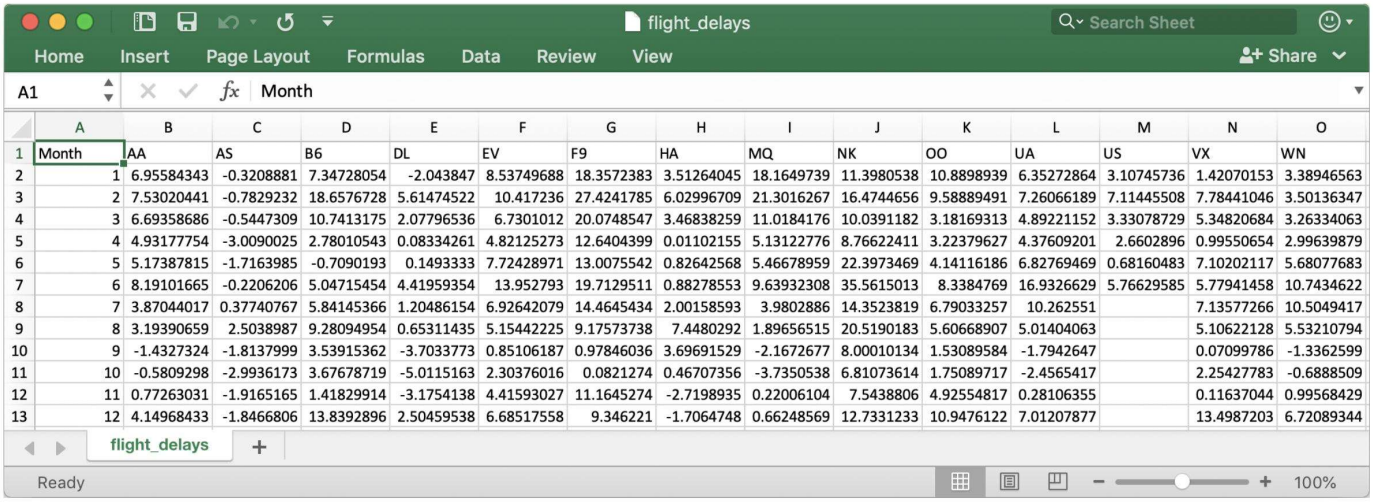
In [1]:

```
import pandas as pd
pd.plotting.register_matplotlib_converters()
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
print("Setup Complete")
```

Select a dataset

In this tutorial, we'll work with a dataset from the US Department of Transportation that tracks flight delays.

Opening this CSV file in Excel shows a row for each month (where `1` = January, `2` = February, etc) and a column for each airline code.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	Month	AA	AS	B6	DL	EV	F9	HA	MQ	NK	OO	UA	US	VX	WN
2	1	6.95584343	-0.3208881	7.34728054	-2.043847	8.53749688	18.3572383	3.51264045	18.1649739	11.3980538	10.8898939	6.35272864	3.10745736	1.42070153	3.38946563
3	2	7.53020441	-0.7829232	18.6576728	5.61474522	10.417236	27.4241785	6.02996709	21.3016267	16.4744656	9.58889491	7.26066189	7.11445508	7.78441046	3.50136347
4	3	6.69358686	-0.5447309	10.7413175	2.07796536	6.7301012	20.0748547	3.46838259	11.0184176	10.0391182	3.18169313	4.89221152	3.33078729	5.34820684	3.26334063
5	4	4.93177754	-3.0090025	2.78010543	0.08334261	4.82125273	12.6404399	0.01102155	5.13122776	8.76622411	3.22379627	4.37609201	2.6602896	0.99550654	2.99639879
6	5	5.17387815	-1.7163985	-0.7090193	0.1493333	7.72428971	13.0075542	0.82642568	5.46678959	22.3973469	4.14116186	6.82769469	0.68160483	7.10202117	5.68077683
7	6	8.19101665	-0.2206206	5.04715454	4.41959354	13.952793	19.7129511	0.88278553	9.63932308	35.5615013	8.3384769	16.9326629	5.76629585	5.77941458	10.7434622
8	7	3.87044017	0.37740767	5.84145366	1.20486154	6.92642079	14.4645434	2.00158593	3.9802886	14.3523819	6.79033257	10.262551		7.13577266	10.5049417
9	8	3.19390659	2.5038987	9.28094954	0.65311435	5.15442225	9.17573738	7.4480292	1.89656515	20.5190183	5.60668907	5.01404063		5.10622128	5.53210794
10	9	-1.4327324	-1.8137999	3.53915362	-3.7033773	0.85106187	0.97846036	3.69691529	-2.1672677	8.00010134	1.53089584	-1.7942647		0.07099786	-1.3362599
11	10	-0.5809298	-2.9936173	3.67678719	-5.0115163	2.30376016	0.0821274	0.46707356	-3.7350538	6.81073614	1.75089717	-2.4565417		2.25427783	-0.6888509
12	11	0.77263031	-1.9165165	1.41829914	-3.1754138	4.41593027	11.1645274	-2.7198935	0.22006104	7.5438806	4.92554817	0.28106355		0.11637044	0.99568429
13	12	4.14968433	-1.8466806	13.8392896	2.50459538	6.68517558	9.346221	-1.7064748	0.66248569	12.7331233	10.9476122	7.01207877		13.4987203	6.72089344

Each entry shows the average arrival delay (in minutes) for a different airline and month (all in year 2015). Negative entries denote flights that (*on average*) tended to arrive early. For instance, the average American Airlines flight (*airline code:* ***AA***) in January arrived roughly 7 minutes late, and the average Alaska Airlines flight (*airline code:* ***AS***) in April arrived roughly 3 minutes early.

Load the data

As before, we load the dataset using the `pd.read_csv` command.

In [2]:

```
# Path of the file to read
flight_filepath = "../input/flight_delays.csv"

# Read the file into a variable flight_data
flight_data = pd.read_csv(flight_filepath, index_col="Month")
```

You may notice that the code is slightly shorter than what we used in the previous tutorial. In this case, since the row labels (from the `'Month'` column) don't correspond to dates, we don't add `parse_dates=True` in the parentheses. But, we keep the first two pieces of text as before, to provide both:

- the filepath for the dataset (in this case, `flight_filepath`), and
- the name of the column that will be used to index the rows (in this case, `index_col="Month"`).

Examine the data

Since the dataset is small, we can easily print all of its contents. This is done by writing a single line of code with just the name of the dataset.

In [3]:
```
# Print the data
flight_data
```

Out[3]:

	AA	AS	B6	DL	EV	F9	HA	MQ
Month								
1	6.955843	-0.320888	7.347281	-2.043847	8.537497	18.357238	3.512640	18.164974
2	7.530204	-0.782923	18.657673	5.614745	10.417236	27.424179	6.029967	21.301627
3	6.693587	-0.544731	10.741317	2.077965	6.730101	20.074855	3.468383	11.018418
4	4.931778	-3.009003	2.780105	0.083343	4.821253	12.640440	0.011022	5.131228
5	5.173878	-1.716398	-0.709019	0.149333	7.724290	13.007554	0.826426	5.466790
6	8.191017	-0.220621	5.047155	4.419594	13.952793	19.712951	0.882786	9.639323
7	3.870440	0.377408	5.841454	1.204862	6.926421	14.464543	2.001586	3.980289
8	3.193907	2.503899	9.280950	0.653114	5.154422	9.175737	7.448029	1.896565
9	-1.432732	-1.813800	3.539154	-3.703377	0.851062	0.978460	3.696915	-2.167268
10	-0.580930	-2.993617	3.676787	-5.011516	2.303760	0.082127	0.467074	-3.735054
11	0.772630	-1.916516	1.418299	-3.175414	4.415930	11.164527	-2.719894	0.220061
12	4.149684	-1.846681	13.839290	2.504595	6.685176	9.346221	-1.706475	0.662486

Bar chart

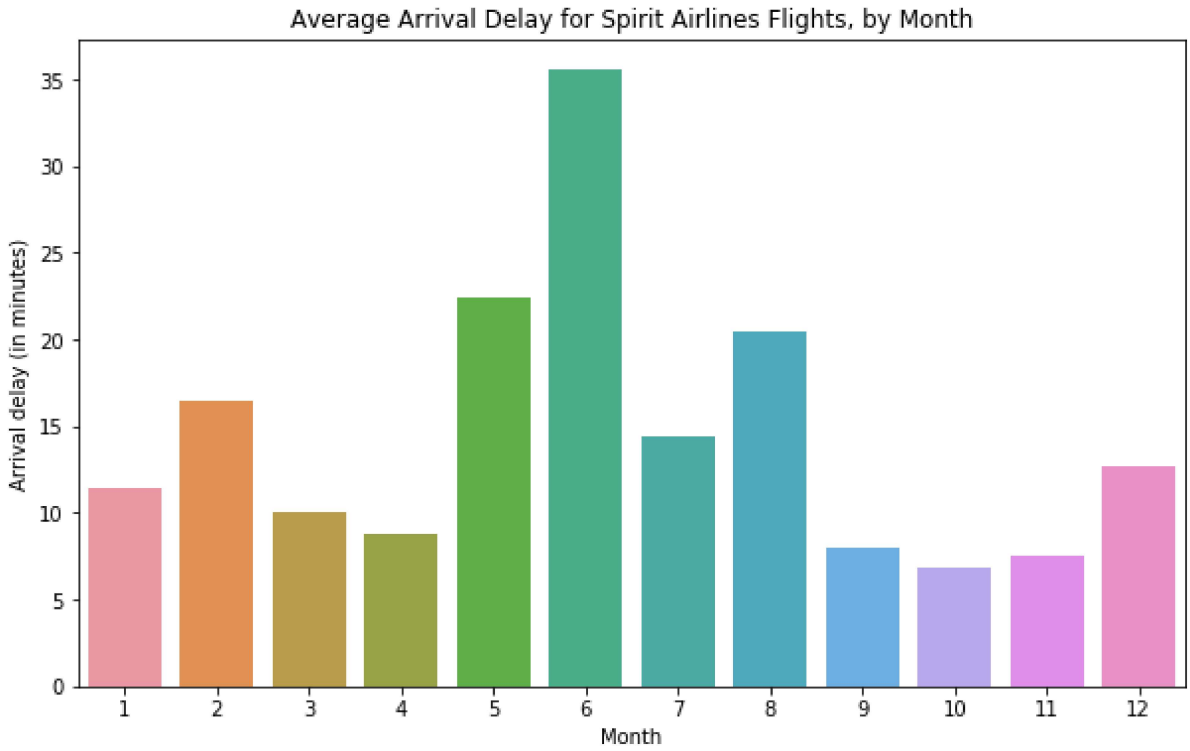
Say we'd like to create a bar chart showing the average arrival delay for Spirit Airlines (*airline code:* ***NK***) flights, by month.

In [4]:

```
# Set the width and height of the figure
plt.figure(figsize=(10,6))

# Add title
plt.title("Average Arrival Delay for Spirit Airlines Flights, by Month")

# Bar chart showing average arrival delay for Spirit Airlines flights by month
sns.barplot(x=flight_data.index, y=flight_data['NK'])

# Add label for vertical axis
plt.ylabel("Arrival delay (in minutes)")
```

Out[4]:

```
Text(0, 0.5, 'Arrival delay (in minutes)')
```



The commands for customizing the text (title and vertical axis label) and size of the figure are familiar from the previous tutorial. The code that creates the bar chart is new:

```
# Bar chart showing average arrival delay for Spirit Airlines flights by month
sns.barplot(x=flight_data.index, y=flight_data['NK'])
```

It has three main components:

- `sns.barplot` - This tells the notebook that we want to create a bar chart.
  - *Remember that `sns` refers to the seaborn package, and all of the commands that you use to create charts in this course will start with this prefix.*
- `x=flight_data.index` - This determines what to use on the horizontal axis. In this case, we have selected the column that ***index***es the rows (in this case, the column containing the months).
- `y=flight_data['NK']` - This sets the column in the data that will be used to determine the height of each bar. In this case, we select the `'NK'` column.

> **Important Note**: You must select the indexing column with `flight_data.index`, and it is not possible to use `flight_data['Month']` (*which will return an error*). This is because when we loaded the dataset, the `"Month"` column was used to index the rows. **We always have to use this special notation to select the indexing column.**

Heatmap

We have one more plot type to learn about: **heatmaps**!

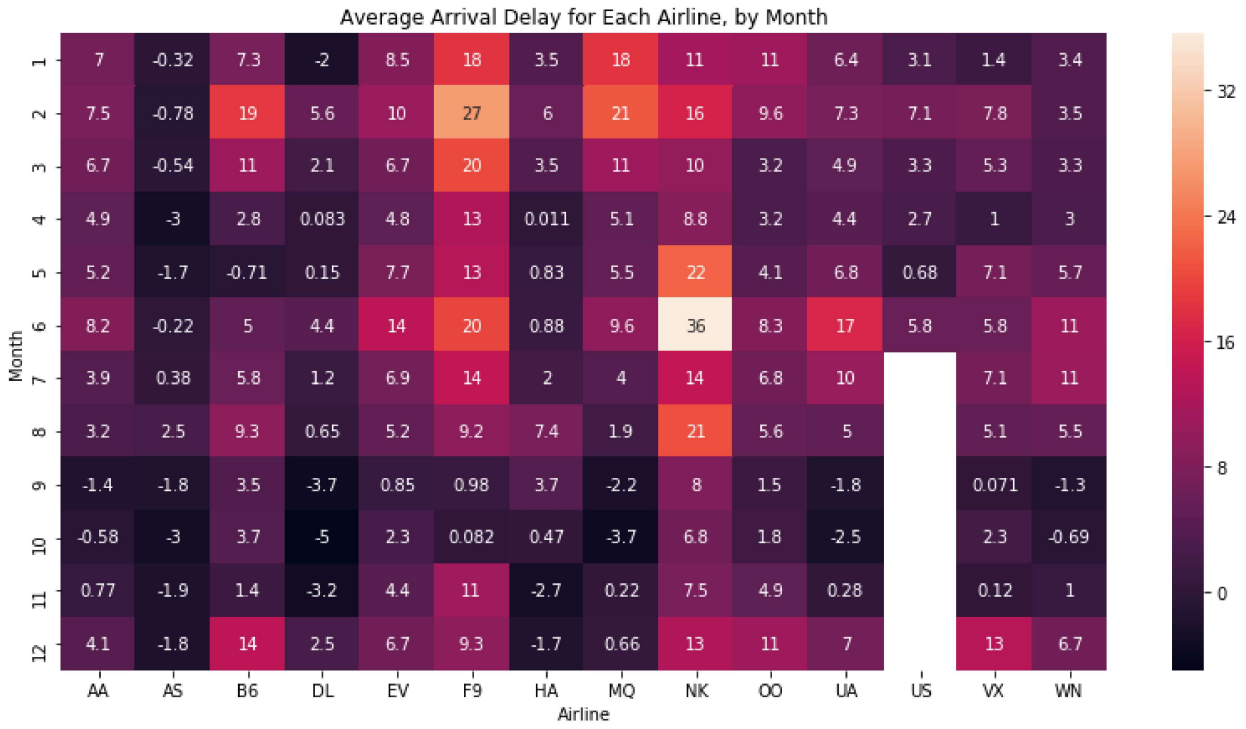
In the code cell below, we create a heatmap to quickly visualize patterns in `flight_data`. Each cell is color-coded according to its corresponding value.

In [5]:

```
# Set the width and height of the figure
plt.figure(figsize=(14,7))

# Add title
plt.title("Average Arrival Delay for Each Airline, by Month")

# Heatmap showing average arrival delay for each airline by month
sns.heatmap(data=flight_data, annot=True)

# Add label for horizontal axis
plt.xlabel("Airline")
```

Out[5]:

```
Text(0.5, 42.0, 'Airline')
```



The relevant code to create the heatmap is as follows:

```
# Heatmap showing average arrival delay for each airline by month
sns.heatmap(data=flight_data, annot=True)
```

This code has three main components:

- `sns.heatmap` - This tells the notebook that we want to create a heatmap.
- `data=flight_data` - This tells the notebook to use all of the entries in `flight_data` to create the heatmap.
- `annot=True` - This ensures that the values for each cell appear on the chart. (*Leaving this out removes the numbers from each of the cells!*)

What patterns can you detect in the table? For instance, if you look closely, the months toward the end of the year (especially months 9-11) appear relatively dark for all airlines. This suggests that airlines are better (on average) at keeping schedule during these months!

What's next?

Create your own visualizations with a **coding exercise**!