Now that you are familiar with the coding environment, it's time to learn how to make your own charts!

In this tutorial, you'll learn just enough Python to create professional looking **line charts**. Then, in the following exercise, you'll put your new skills to work with a real-world dataset.

Set up the notebook

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

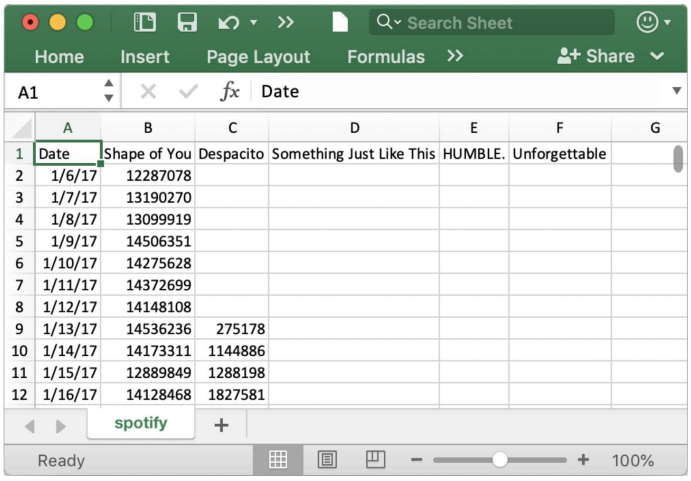
In [1]:

```
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

The dataset for this tutorial tracks global daily streams on the music streaming service Spotify. We focus on five popular songs from 2017 and 2018:

1. "Shape of You", by Ed Sheeran (link)
2. "Despacito", by Luis Fonzi (link)
3. "Something Just Like This", by The Chainsmokers and Coldplay (link)
4. "HUMBLE.", by Kendrick Lamar (link)
5. "Unforgettable", by French Montana (link)



	A	B	C	D	E	F	G
1	Date	Shape of You	Despacito	Something Just Like This	HUMBLE.	Unforgettable	
2	1/6/17	12287078					
3	1/7/17	13190270					
4	1/8/17	13099919					
5	1/9/17	14506351					
6	1/10/17	14275628					
7	1/11/17	14372699					
8	1/12/17	14148108					
9	1/13/17	14536236	275178				
10	1/14/17	14173311	1144886				
11	1/15/17	12889849	1288198				
12	1/16/17	14128468	1827581				

Notice that the first date that appears is January 6, 2017, corresponding to the release date of "The Shape of You", by Ed Sheeran. And, using the table, you can see that "The Shape of You" was streamed 12,287,078 times globally on the day of its release. Notice that the other songs have missing values in the first row, because they weren't released until later!

Load the data

As you learned in the previous tutorial, we load the dataset using the `pd.read_csv` command.

In [2]:
```
# Path of the file to read
spotify_filepath = "../input/spotify.csv"

# Read the file into a variable spotify_data
spotify_data = pd.read_csv(spotify_filepath, index_col="Date", parse_dates=True)
```

The end result of running both lines of code above is that we can now access the dataset by using `spotify_data`.

Examine the data

We can print the *first* five rows of the dataset by using the `head` command that you learned about in the previous tutorial.

In [3]:
```
# Print the first 5 rows of the data
spotify_data.head()
```

Out[3]:

	Shape of You	Despacito	Something Just Like This	HUMBLE.	Unforgettable
Date					
2017-01-06	12287078	NaN	NaN	NaN	NaN
2017-01-07	13190270	NaN	NaN	NaN	NaN
2017-01-08	13099919	NaN	NaN	NaN	NaN
2017-01-09	14506351	NaN	NaN	NaN	NaN
2017-01-10	14275628	NaN	NaN	NaN	NaN

Check now that the first five rows agree with the image of the dataset (*from when we saw what it would look like in Excel*) above.

> Empty entries will appear as `NaN`, which is short for "Not a Number".

We can also take a look at the *last* five rows of the data by making only one small change (where `.head()` becomes `.tail()`):

In [4]:
```
# Print the last five rows of the data
spotify_data.tail()
```

Out[4]:

	Shape of You	Despacito	Something Just Like This	HUMBLE.	Unforgettable
Date					
2018-01-05	4492978	3450315.0	2408365.0	2685857.0	2869783.0
2018-01-06	4416476	3394284.0	2188035.0	2559044.0	2743748.0
2018-01-07	4009104	3020789.0	1908129.0	2350985.0	2441045.0
2018-01-08	4135505	2755266.0	2023251.0	2523265.0	2622693.0
2018-01-09	4168506	2791601.0	2058016.0	2727678.0	2627334.0

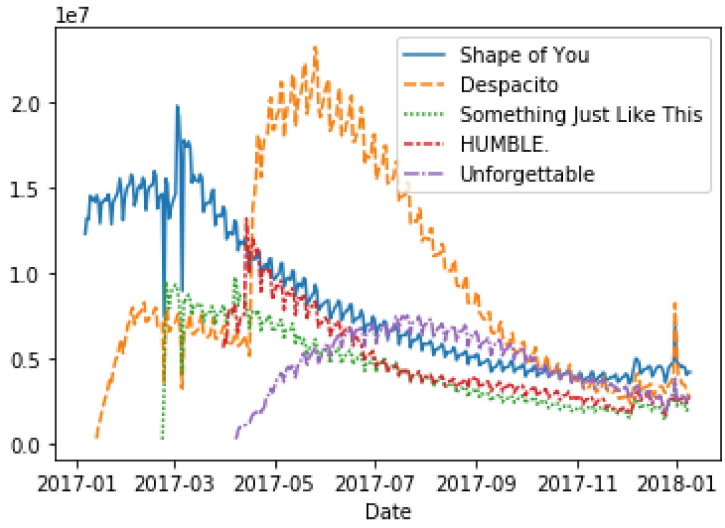
Thankfully, everything looks about right, with millions of daily global streams for each song, and we can proceed to plotting the data!

Plot the data

Now that the dataset is loaded into the notebook, we need only one line of code to make a line chart!

In [5]:
```
# Line chart showing daily global streams of each song
sns.lineplot(data=spotify_data)
```

Out[5]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0cf1118e48>
```



As you can see above, the line of code is relatively short and has two main components:

- `sns.lineplot` tells the notebook that we want to create a line chart.
  - *Every command that you learn about in this course will start with* `sns`*, which indicates that the command comes from the seaborn package. For instance, we use* `sns.lineplot` *to make line charts. Soon, you'll learn that we use* `sns.barplot` *and* `sns.heatmap` *to make bar charts and heatmaps, respectively.*
- `data=spotify_data` selects the data that will be used to create the chart.

Note that you will always use this same format when you create a line chart, and ***the only thing that changes with a new dataset is the name of the dataset***. So, if you were working with a different dataset named `financial_data`, for instance, the line of code would appear as follows:

```
sns.lineplot(data=financial_data)
```

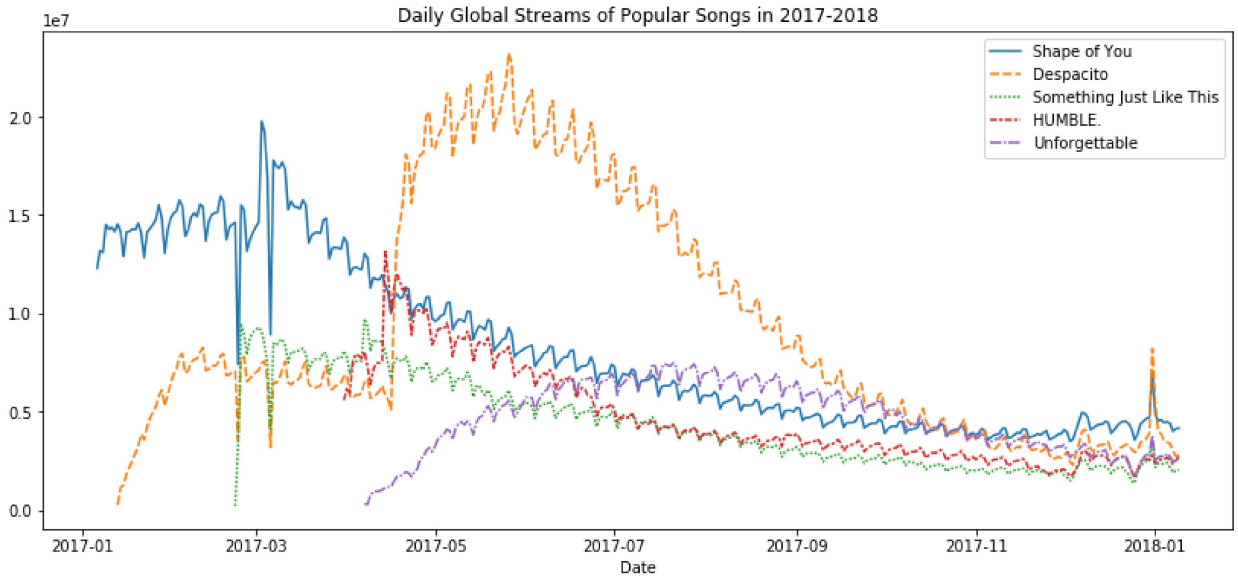
Sometimes there are additional details we'd like to modify, like the size of the figure and the title of the chart. Each of these options can easily be set with a single line of code.

In [6]:

```
# Set the width and height of the figure
plt.figure(figsize=(14,6))

# Add title
plt.title("Daily Global Streams of Popular Songs in 2017-2018")

# Line chart showing daily global streams of each song
sns.lineplot(data=spotify_data)
```

Out[6]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0cf0fd9da0>
```



The first line of code sets the size of the figure to `14` inches (in width) by `6` inches (in height). To set the size of *any figure*, you need only copy the same line of code as it appears. Then, if you'd like to use a custom size, change the provided values of `14` and `6` to the desired width and height.

The second line of code sets the title of the figure. Note that the title must *always* be enclosed in quotation marks (`"..."`)!

Plot a subset of the data

So far, you've learned how to plot a line for *every* column in the dataset. In this section, you'll learn how to plot a *subset* of the columns.

We'll begin by printing the names of all columns. This is done with one line of code and can be adapted for any dataset by just swapping out the name of the dataset (in this case, `spotify_data`).

In [7]:
```
list(spotify_data.columns)
```

Out[7]:
```
['Shape of You',
 'Despacito',
 'Something Just Like This',
 'HUMBLE.',
 'Unforgettable']
```

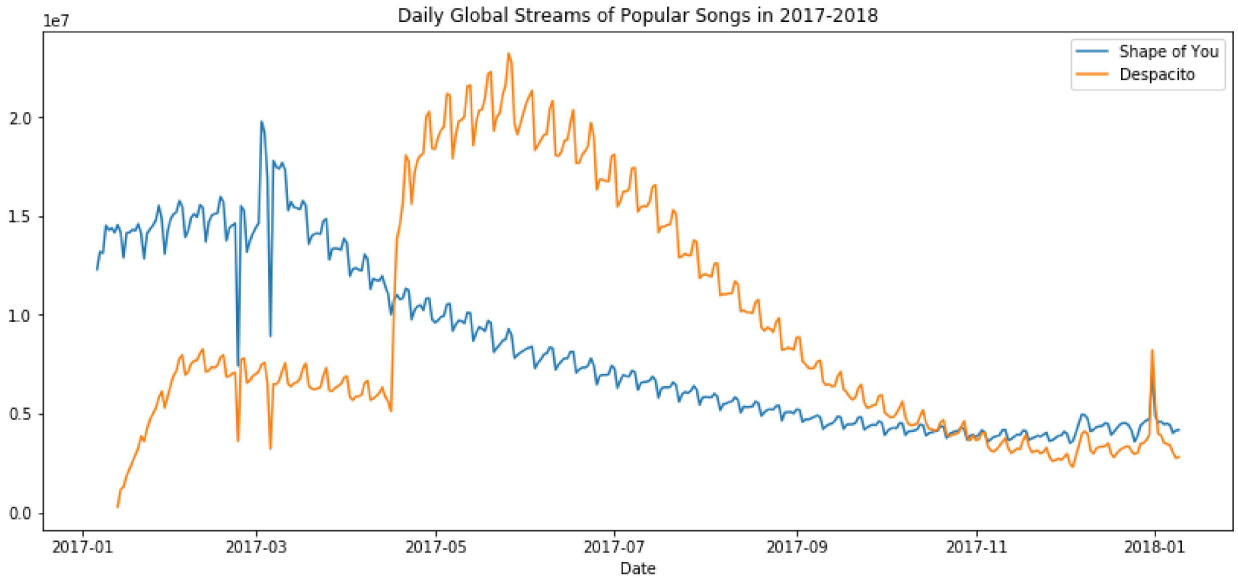
In the next code cell, we plot the lines corresponding to the first two columns in the dataset.

In [8]:

```
# Set the width and height of the figure
plt.figure(figsize=(14,6))

# Add title
plt.title("Daily Global Streams of Popular Songs in 2017-2018")

# Line chart showing daily global streams of 'Shape of You'
sns.lineplot(data=spotify_data['Shape of You'], label="Shape of You")

# Line chart showing daily global streams of 'Despacito'
sns.lineplot(data=spotify_data['Despacito'], label="Despacito")

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

Out[8]:

```
Text(0.5, 0, 'Date')
```



The first two lines of code set the title and size of the figure (*and should look very familiar!*).

The next two lines each add a line to the line chart. For instance, consider the first one, which adds the line for "Shape of You":

```
# Line chart showing daily global streams of 'Shape of You'
sns.lineplot(data=spotify_data['Shape of You'], label="Shape of You")
```

This line looks really similar to the code we used when we plotted every line in the dataset, but it has a few key differences:

- Instead of setting `data=spotify_data`, we set `data=spotify_data['Shape of You']`. In general, to plot only a single column, we use this format with putting the name of the column in single quotes and enclosing it in square brackets. (*To make sure that you correctly specify the name of the column, you can print the list of all column names using the command you learned above.*)
- We also add `label="Shape of You"` to make the line appear in the legend and set its corresponding label.

The final line of code modifies the label for the horizontal axis (or x-axis), where the desired label is placed in quotation marks (`"..."`).

What's next?

Put your new skills to work in a **coding exercise**!