Data Visualization

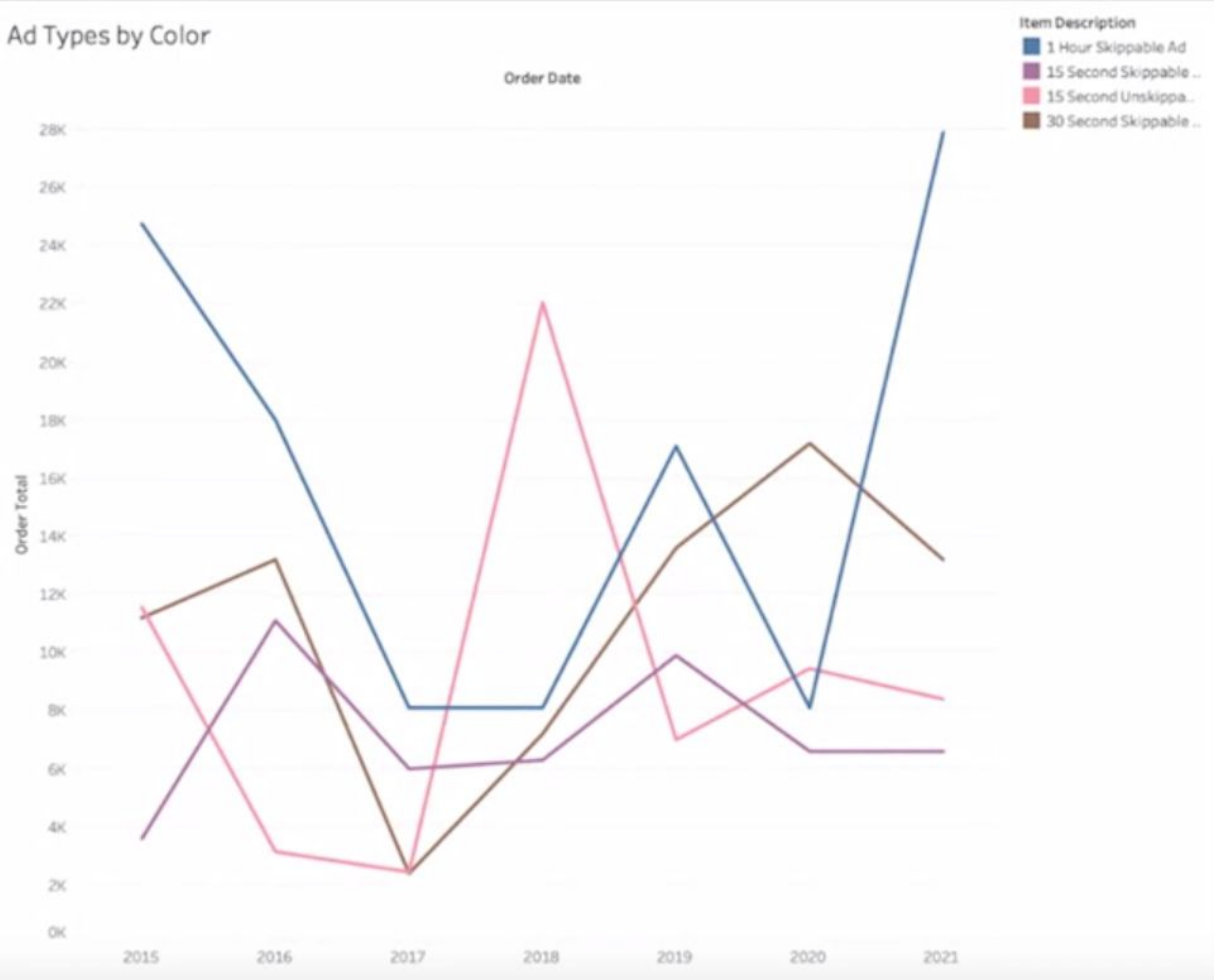
Agenda

1. Accessibility for CVD Examples

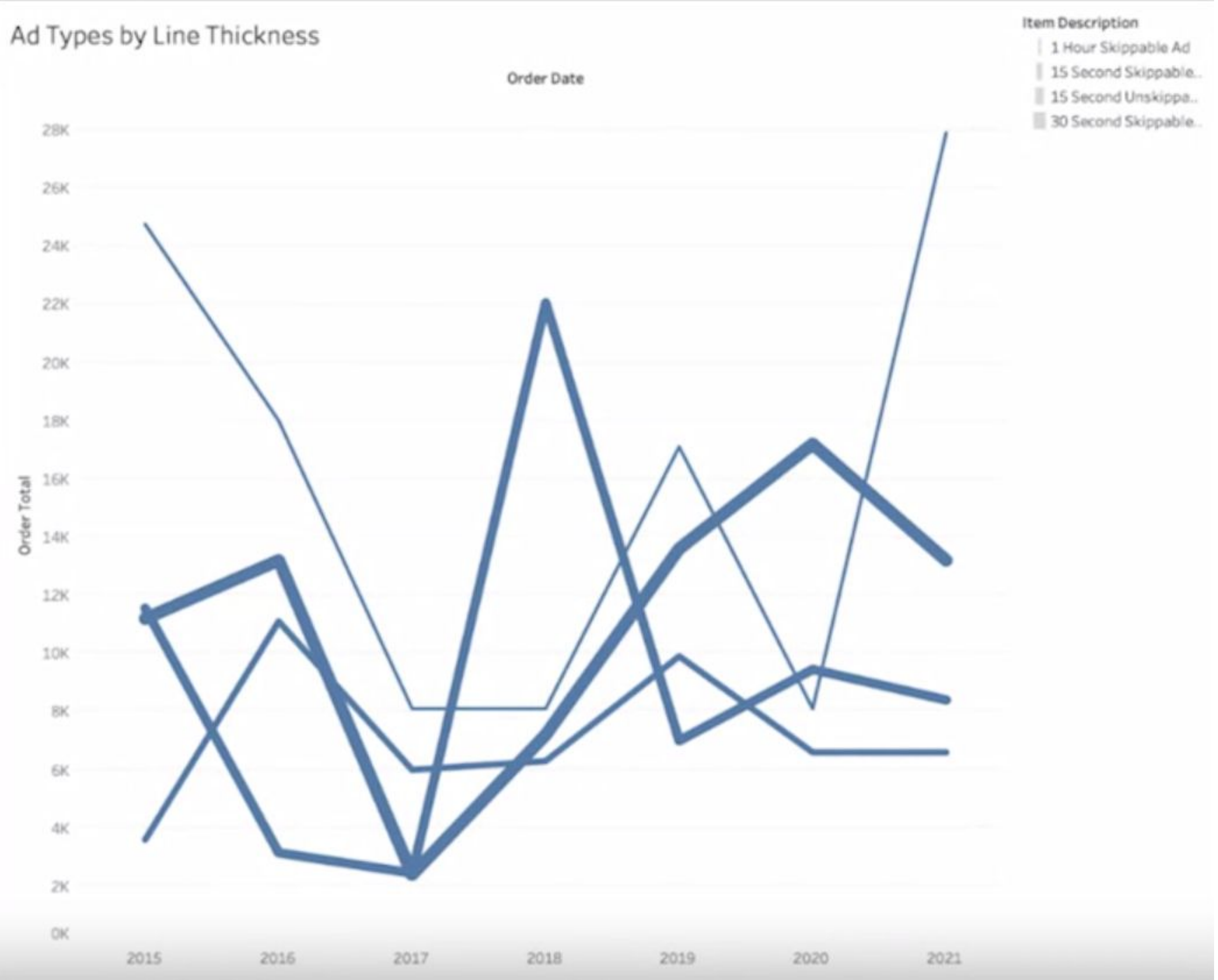
2. Matplotlib Examples

Accessibility for CVD Examples

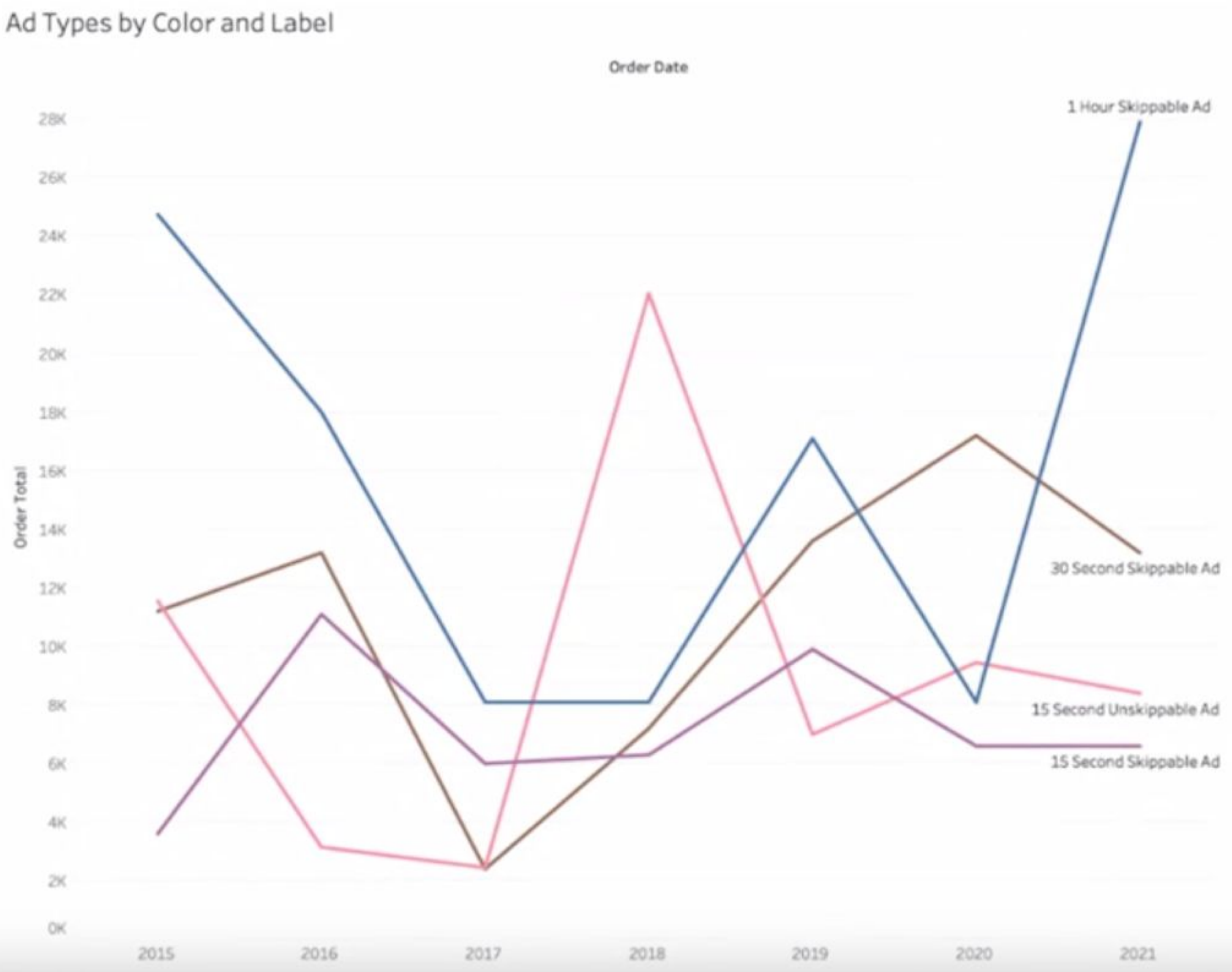
Colors


Ad Types by Color
Order Date
Item Description
1 Hour Skippable Ad
15 Second Skippable ..
15 Second Unskippa..
30 Second Skippable ..
Order Total
28K
26K
24K
22K
20K
18K
16K
14K
12K
10K
8K
6K
4K
2K
0K
2015
2016
2017
2018
2019
2020
2021

Lines Thickness



Colors and Labels



Matplotlib Examples

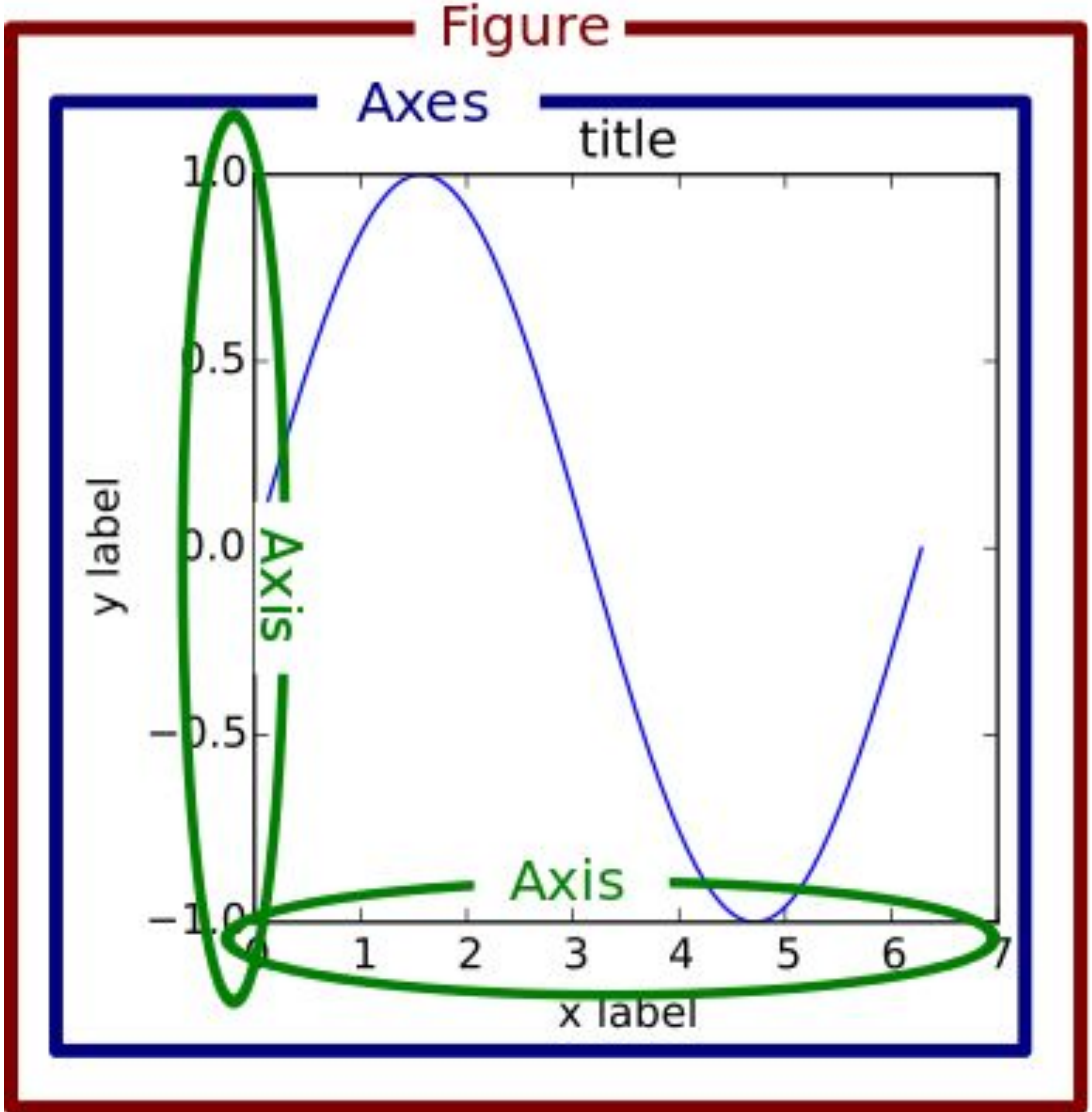
Matplotlib Architecture: 2) Artist Layer Types

1. **Primitive Artist**: as Line, Rectangle, Circle, Text

2. **Composite Artist:** may contain other **Artists**
   - **Example 1: Figure Artist**
     - top-level Matplotlib object
     - contains and manages all of the elements in a given graphic
     - https://github.dev/matplotlib/matplotlib/blob/main/lib/matplotlib/figure.py
   - **Example 2: Axes Artist**
     - most important Composite Artist
     - where most of the Matplotlib API plotting methods are defined
       - including methods to create and manipulate ticks, axis lines, grid, background
   - **Other Examples: Tick Artist**

Axes Artist

- The plotting area
  - including all axis
- Don’t mean plural of **Axis**



Matplotlib Architecture: 3) Scripting Layer

- Developed for scientists who are not professional programmers

- Essentially the **Matplotlib.pyplot** that automates:
  - defining **FigureCanvas**
  - defining **Figure Artist**
  - connecting **FigureCanvas** and **Figure Artist**
  - https://github.dev/matplotlib/matplotlib/blob/main/lib/matplotlib/pyplot.py

- Comparing with **Layer 2 (Artist Layer)** which is:
  - heavy and for developers
  - not for individuals who want to perform quick **Exploratory Analysis** of some data

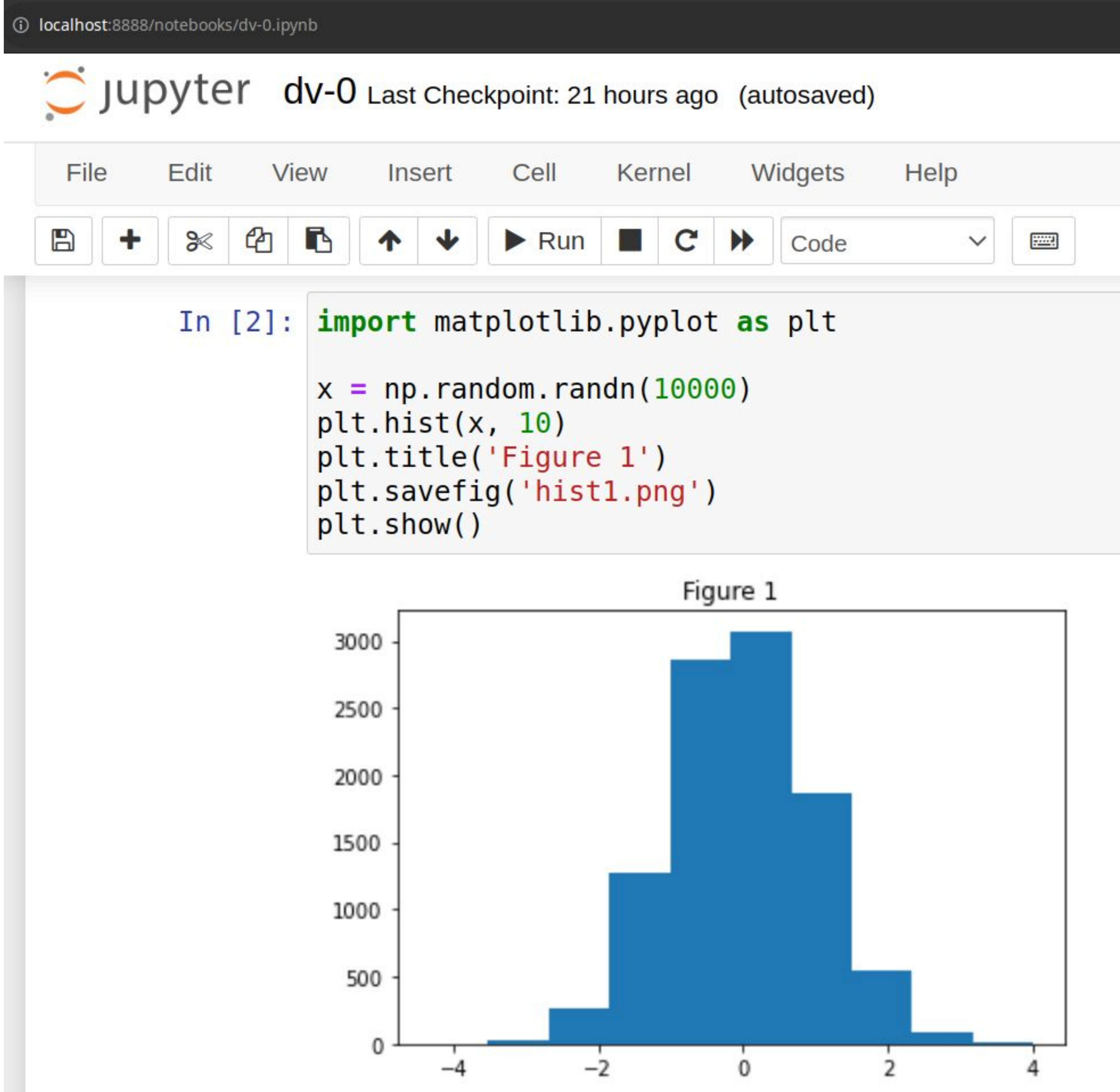
Scripting Layer Example (1/3)

localhost:8888/notebooks/dv-0.ipynb
jupyter
dv-0 Last Checkpoint: 21 hours ago (unsaved changes)
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Trusted
Run
Code

```
In [1]: import numpy as np

        np.random.randn(10) #Return 10 samples from the Standard Normal Distribution

Out[1]: array([-0.64393041,  0.0329367 , -0.16840147,  0.88846809,  0.76751103,
                0.18852699, -1.30213432,  0.58043701,  1.80149475, -0.18262329])
```

Scripting Layer Example (2/3)

- `import` **`pyplot`**
  - ○ from the **matplotlib library**

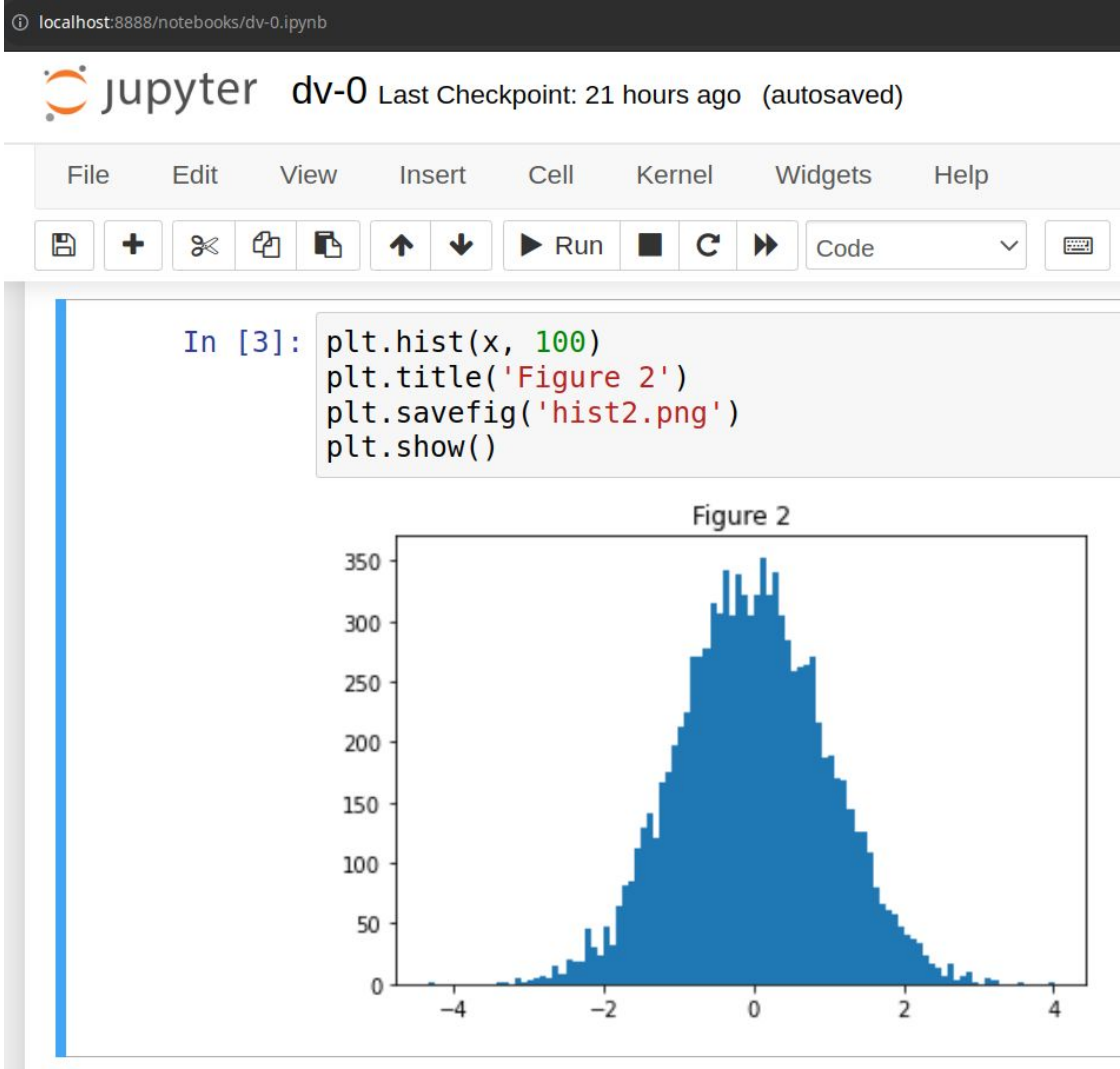


localhost:8888/notebooks/dv-0.ipynb
jupyter
dv-0 Last Checkpoint: 21 hours ago (autosaved)
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Run
Code
In [2]:

```
import matplotlib.pyplot as plt

x = np.random.randn(10000)
plt.hist(x, 10)
plt.title('Figure 1')
plt.savefig('hist1.png')
plt.show()
```

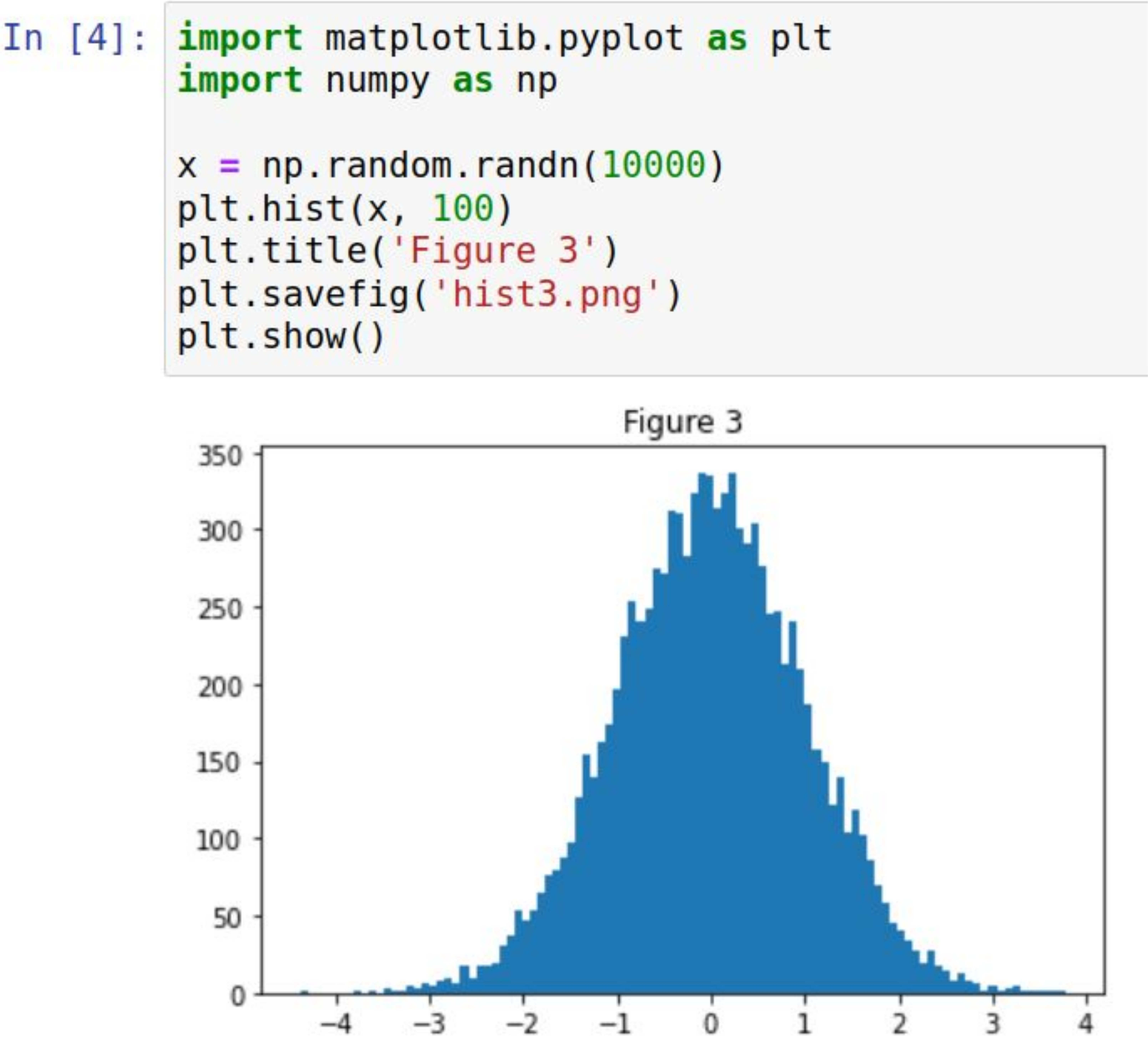
Scripting Layer Example (3/3)

- `all methods`
  - creating
    - histogram
    - other Artist objects
  - manipulating them
- `are part of` **`pyplot`**



```
plt.hist(x, 100)
plt.title('Figure 2')
plt.savefig('hist2.png')
plt.show()
```

Scripting Layer Complete Example



In [4]:

```
import matplotlib.pyplot as plt
import numpy as np

x = np.random.randn(10000)
plt.hist(x, 100)
plt.title('Figure 3')
plt.savefig('hist3.png')
plt.show()
```

Artist Layer Example 1

In [1]:

```
from matplotlib.backends.backend_agg import FigureCanvasAgg as FigureCanvas
from matplotlib.figure import Figure
import numpy as np

x = np.random.randn(10000)

fig = Figure()                          #Create Figure Artist
canvas = FigureCanvas(fig)              #Create FigureCanvas and attach Figure Artist to it
ax = fig.add_subplot(111)               #Create Axes Artist
ax.hist(x, 100)                         #Call hist method to generate the histogram
ax.set_title('Figure 4')
fig.savefig('hist4.png')
```

Artist Layer Example 1 - Notes

- Use **Artist Layer** to generate histogram of 10000 random numbers

- **Anti Grain Geometry (AGG)**
  - a high-performance library that produces attractive images

- use **111** (from MATLAB convention)
  - creates a grid with 1 row and 1 column
  - uses the first cell in that grid for the location of the new **Axes Artist**

- **hist method**
  - creates a sequence of **Rectangle Artists**

Artist Layer Example 2

In [1]:

```
import numpy as np
x = np.random.randn(10000)

from matplotlib.backends.backend_agg import FigureCanvasAgg as FigureCanvas
from matplotlib.figure import Figure
fig = Figure()
canvas = FigureCanvas(fig)

ax1 = fig.add_subplot(321)
ax1.hist(x, 10)
ax1.set_title('10 bins')

ax2 = fig.add_subplot(324)
ax2.hist(x, 40)
ax2.set_title('40 bins')

ax3 = fig.add_subplot(3,4,10)
ax3.hist(x, 70)
ax3.set_title('70 bins')

fig.savefig('3-axes.png')
```

Artist Layer Example 3

In [2]:

```
import numpy as np
x = np.random.randn(10000)

from matplotlib.backends.backend_agg import FigureCanvasAgg as FigureCanvas
from matplotlib.figure import Figure
fig = Figure()
canvas = FigureCanvas(fig)
gs = fig.add_gridspec(3, 3)

ax1 = fig.add_subplot(gs[0, 0])
ax1.hist(x, 10)
ax1.set_title('10 bins')

ax2 = fig.add_subplot(gs[0, 2])
ax2.hist(x, 40)
ax2.set_title('40 bins')

ax3 = fig.add_subplot(gs[2, 2])
ax3.hist(x, 70)
ax3.set_title('70 bins')

ax4 = fig.add_subplot(gs[2, 0])
ax4.hist(x, 100)
ax4.set_title('100 bins')

fig.savefig('4-axes.png')
```

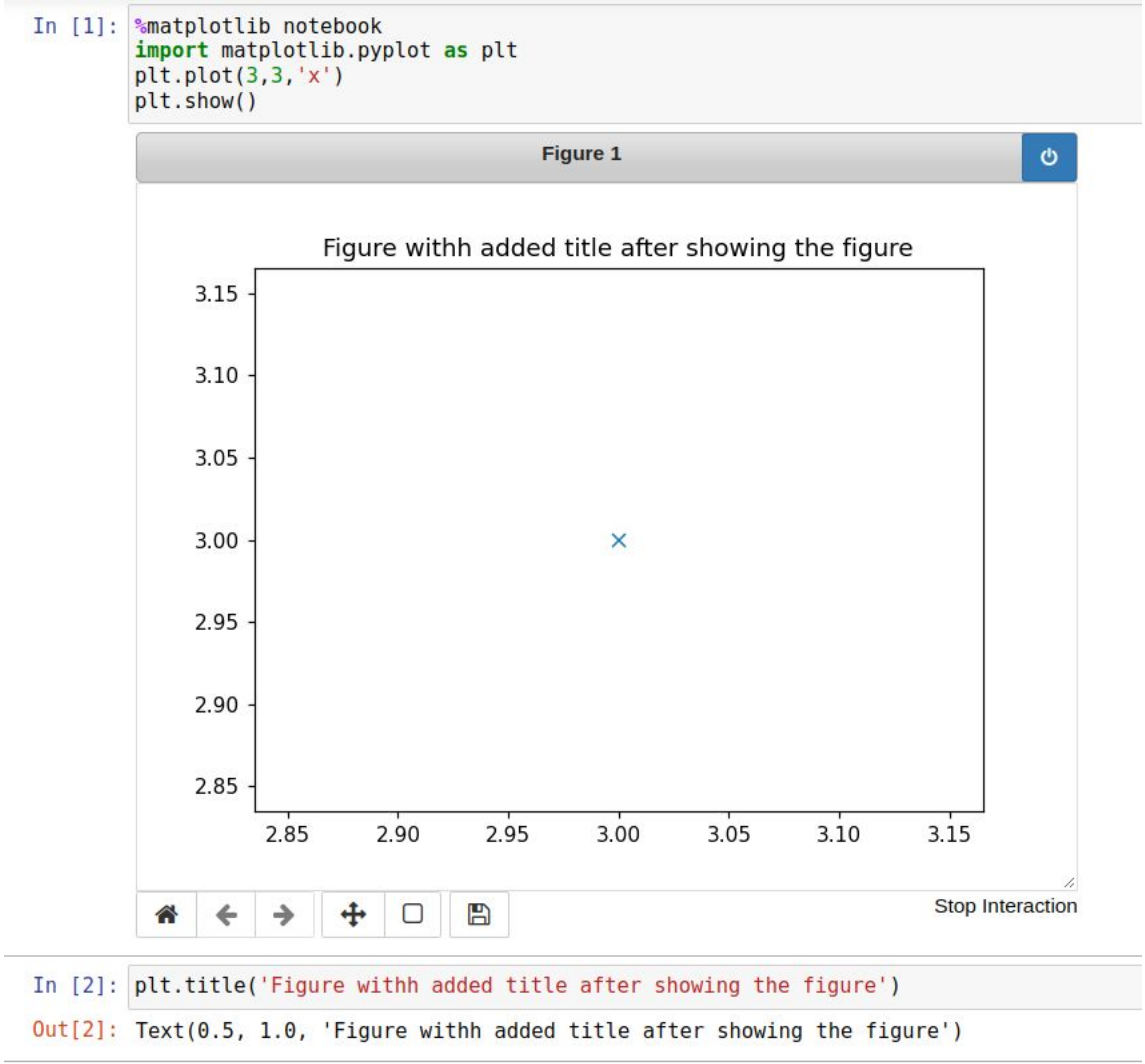
Artist Layer Example 4

In [3]:

```
import numpy as np
x = np.random.randn(10000)

from matplotlib.backends.backend_agg import FigureCanvasAgg as FigureCanvas
from matplotlib.figure import Figure
fig = Figure()
canvas = FigureCanvas(fig)
gs = fig.add_gridspec(3, 2)

ax1 = fig.add_subplot(gs[0, 0])
ax1.hist(x, 100)
ax1.set_title('100 bins')

ax2 = fig.add_subplot(gs[2, 0])
ax2.hist(x, 10)
ax2.set_title('10 bins')

fig.savefig('2-axes.png')
```

100 bins
200
0
−4
−2
0
2
4
10 bins
2000
1000
0
−4
−2
0
2
4

Matplotlib
Plot Function



```
In [1]: %matplotlib notebook
        import matplotlib.pyplot as plt
        plt.plot(3,3,'x')
        plt.show()
```

```
In [2]: plt.title('Figure withh added title after showing the figure')
Out[2]: Text(0.5, 1.0, 'Figure withh added title after showing the figure')
```

Magic Functions

- `Start with` **`%matplotlib`**
  - Matplotlib has a number of different **backends** available

- **`Example for backend: %matplotlib notebook`**
  - if an active figure exists, any function we call will be applied to this active figure
  - If a figure does not exist, any function we call will render a new figure

Pandas

- **Pandas** has a built-in implementation of **Matplotlib**

- Plotting in **Pandas** is simple
    - to generating a line plot:
        - call the plot function on a given **Pandas** dataframe
        - set the parameter kind to line
    - to generating a histogram:
        - call the plot function on a given column of a **Pandas** dataframe
        - set the parameter kind to hist

Line Plot

- Common in many fields, not just data science
- One of the most basic types of plot

- Displays info
  - as a series of data points (**markers**) connected by **straight-line segments**

- **When to use?**
  - best use case: **continuous dataset** to be visualized **over a period of time**
  - **Example:** Plotting the trend of immigrants from Haiti to Canada over time

Pandas Plot Example - Cell 1

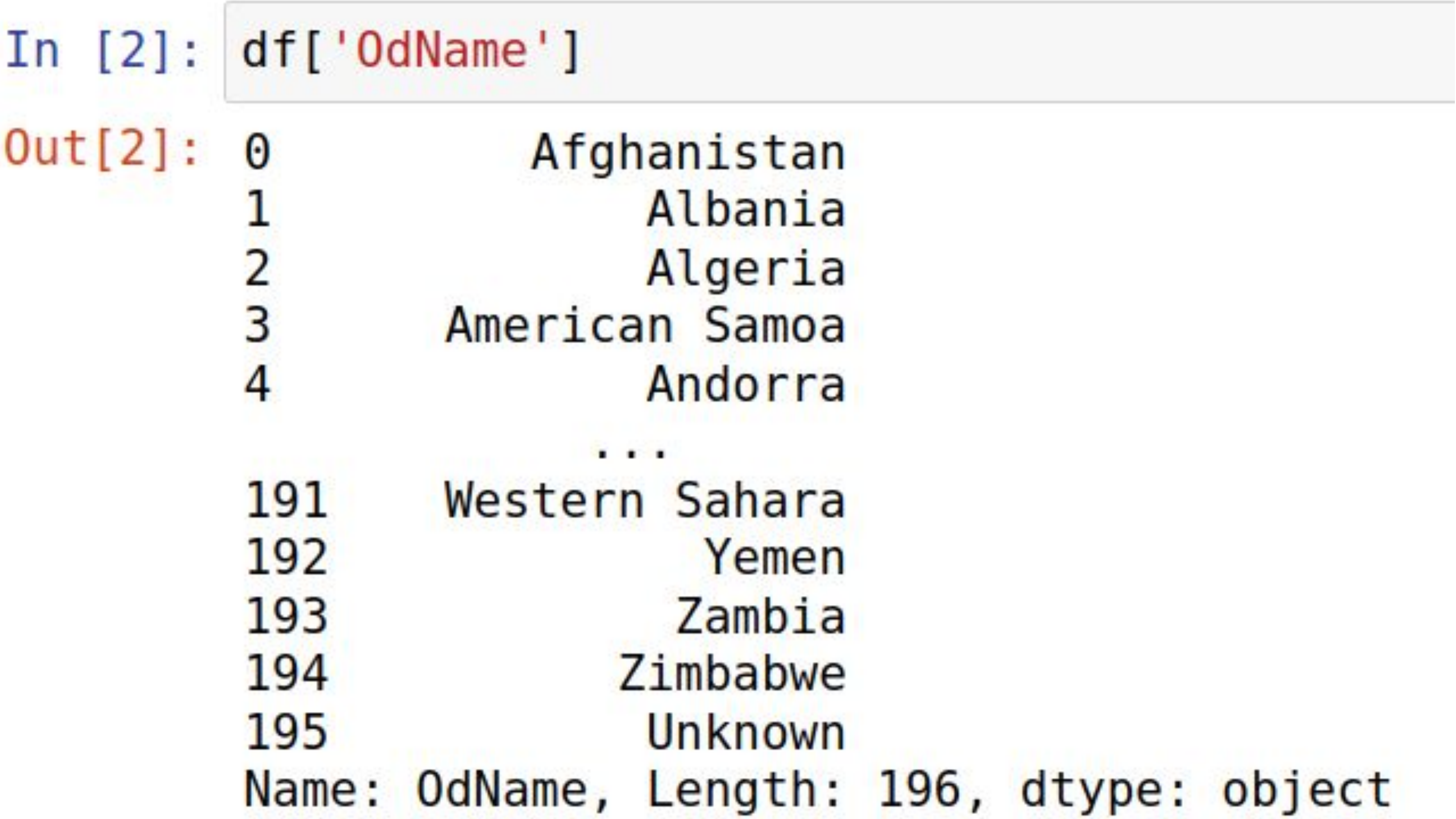
```
In [1]: import pandas as pd

        df = pd.read_csv('canada-mig-dataset.csv')

        df.head()
```

Out[1]:

	Type	Coverage	OdName	AREA	AreaName	REG	RegName	DEV	DevName	1980	...	2004	2005	2006	2007	2008	2009	2010	2011	2012
0	Immigrants	Foreigners	Afghanistan	935	Asia	5501	Southern Asia	902	Developing regions	16	...	2978	3436	3009	2652	2111	1746	1758	2203	2635
1	Immigrants	Foreigners	Albania	908	Europe	925	Southern Europe	901	Developed regions	1	...	1450	1223	856	702	560	716	561	539	620
2	Immigrants	Foreigners	Algeria	903	Africa	912	Northern Africa	902	Developing regions	80	...	3616	3626	4807	3623	4005	5393	4752	4325	3774
3	Immigrants	Foreigners	American Samoa	909	Oceania	957	Polynesia	902	Developing regions	0	...	0	0	1	0	0	0	0	0	0
4	Immigrants	Foreigners	Andorra	908	Europe	925	Southern Europe	901	Developed regions	0	...	0	0	1	1	0	0	0	0	1

5 rows × 43 columns

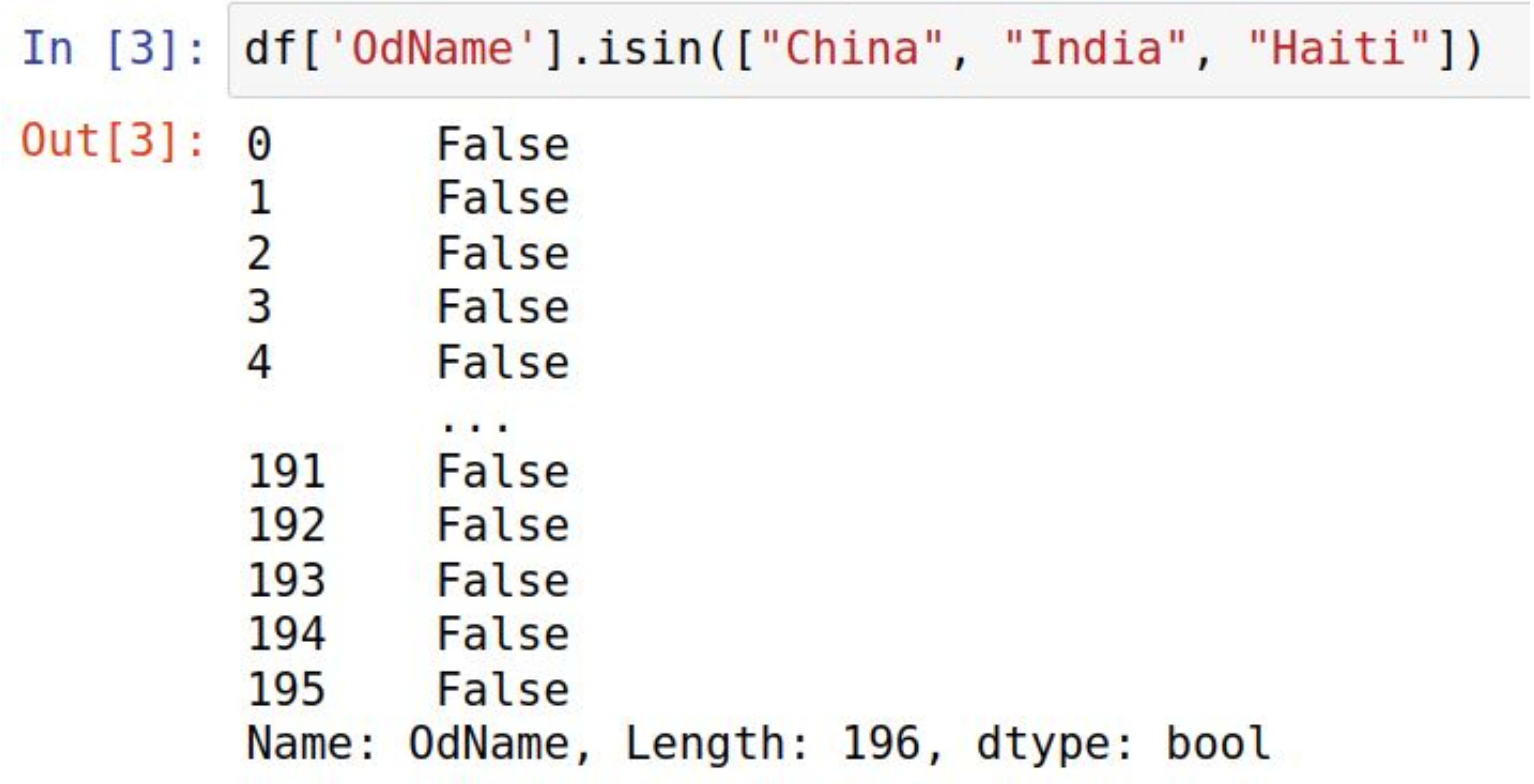
Pandas Plot Example - Cell 2



```
In [2]: df['OdName']
```

```
Out[2]: 0          Afghanistan
        1              Albania
        2              Algeria
        3       American Samoa
        4              Andorra
                    ...
        191     Western Sahara
        192              Yemen
        193             Zambia
        194           Zimbabwe
        195            Unknown
        Name: OdName, Length: 196, dtype: object
```

Pandas Plot Example - Cell 3



```
In [3]: df['OdName'].isin(["China", "India", "Haiti"])
```

```
Out[3]: 0      False
        1      False
        2      False
        3      False
        4      False
               ...
        191    False
        192    False
        193    False
        194    False
        195    False
        Name: OdName, Length: 196, dtype: bool
```

Pandas Plot Example - Cell 4

```
In [4]: df1 = df.loc[ df['OdName'].isin(["China", "India", "Haiti"]) ]
        df1.head()
```

Out[4]:

	Type	Coverage	OdName	AREA	AreaName	REG	RegName	DEV	DevName	1980	...	2004	2005	2006	2007	2008	2009	2010	201
36	Immigrants	Foreigners	China	935	Asia	906	Eastern Asia	902	Developing regions	5123	...	36619	42584	33518	27642	30037	29622	30391	2850
75	Immigrants	Foreigners	Haiti	904	Latin America and the Caribbean	915	Caribbean	902	Developing regions	1666	...	1652	1682	1619	1598	2491	2080	4744	650
79	Immigrants	Foreigners	India	935	Asia	5501	Southern Asia	902	Developing regions	8880	...	28235	36210	33848	28742	28261	29456	34235	2750

3 rows × 43 columns

Pandas Plot Example - Cell 5

```
In [5]: df2 = df1.set_index('OdName')
        df2.head()
```

Out[5]:

	Type	Coverage	AREA	AreaName	REG	RegName	DEV	DevName	1980	1981	...	2004	2005	2006	2007	2008	2009	2010	2
OdName																			
China	Immigrants	Foreigners	935	Asia	906	Eastern Asia	902	Developing regions	5123	6682	...	36619	42584	33518	27642	30037	29622	30391	28
Haiti	Immigrants	Foreigners	904	Latin America and the Caribbean	915	Caribbean	902	Developing regions	1666	3692	...	1652	1682	1619	1598	2491	2080	4744	6
India	Immigrants	Foreigners	935	Asia	5501	Southern Asia	902	Developing regions	8880	8670	...	28235	36210	33848	28742	28261	29456	34235	27

3 rows × 42 columns

Pandas Plot Example - Cell 6

In [6]:

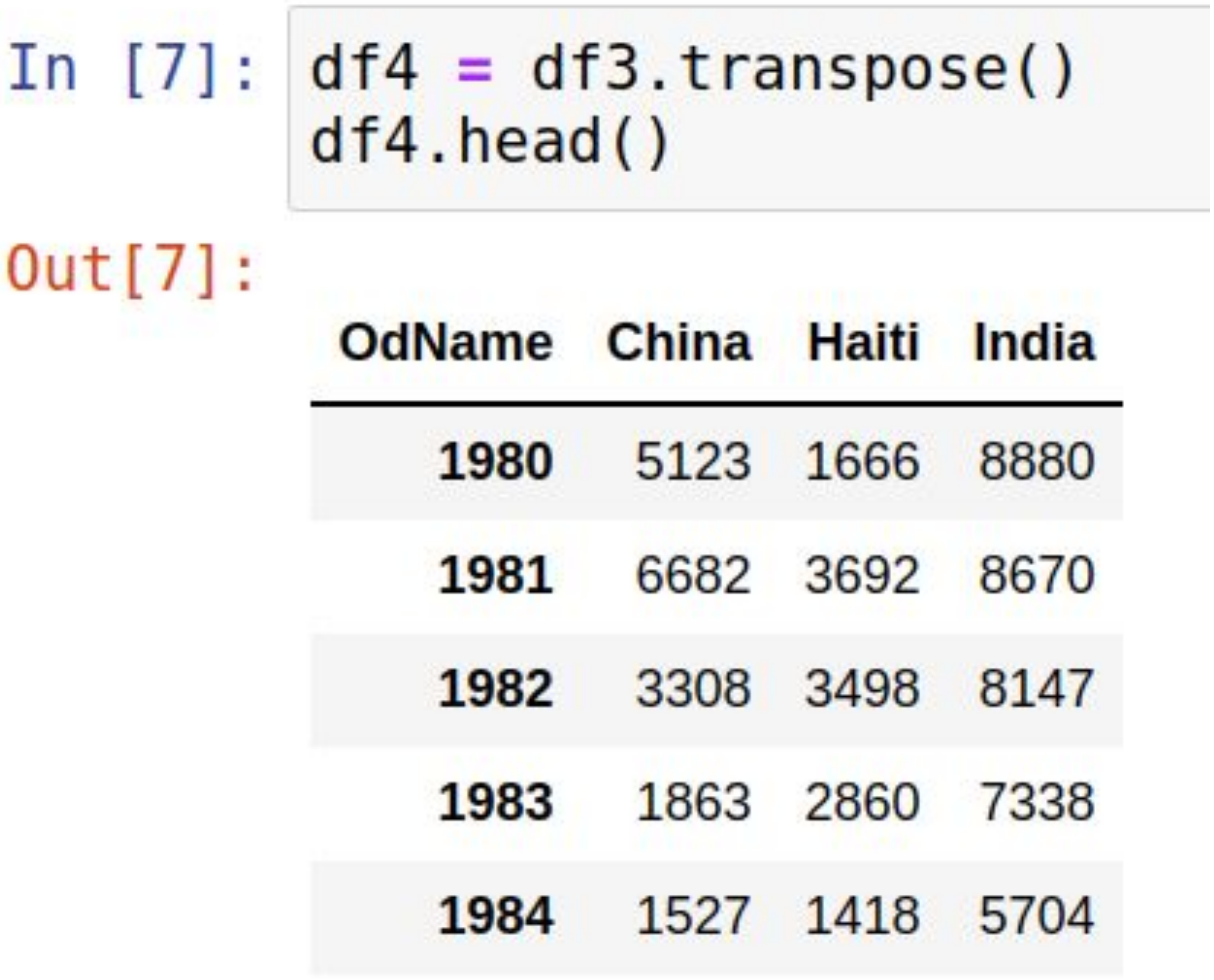
```
df3 = df2.iloc[:, 8:42]
df3.head()
```

Out[6]:

	1980	1981	1982	1983	1984	1985	1986	1987	1988	1989	...	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013
OdName																					
China	5123	6682	3308	1863	1527	1816	1960	2643	2758	4323	...	36619	42584	33518	27642	30037	29622	30391	28502	33024	34129
Haiti	1666	3692	3498	2860	1418	1321	1753	2132	1829	2377	...	1652	1682	1619	1598	2491	2080	4744	6503	5868	4152
India	8880	8670	8147	7338	5704	4211	7150	10189	11522	10343	...	28235	36210	33848	28742	28261	29456	34235	27509	30933	33087

3 rows × 34 columns

Pandas Plot Example - Cell 7



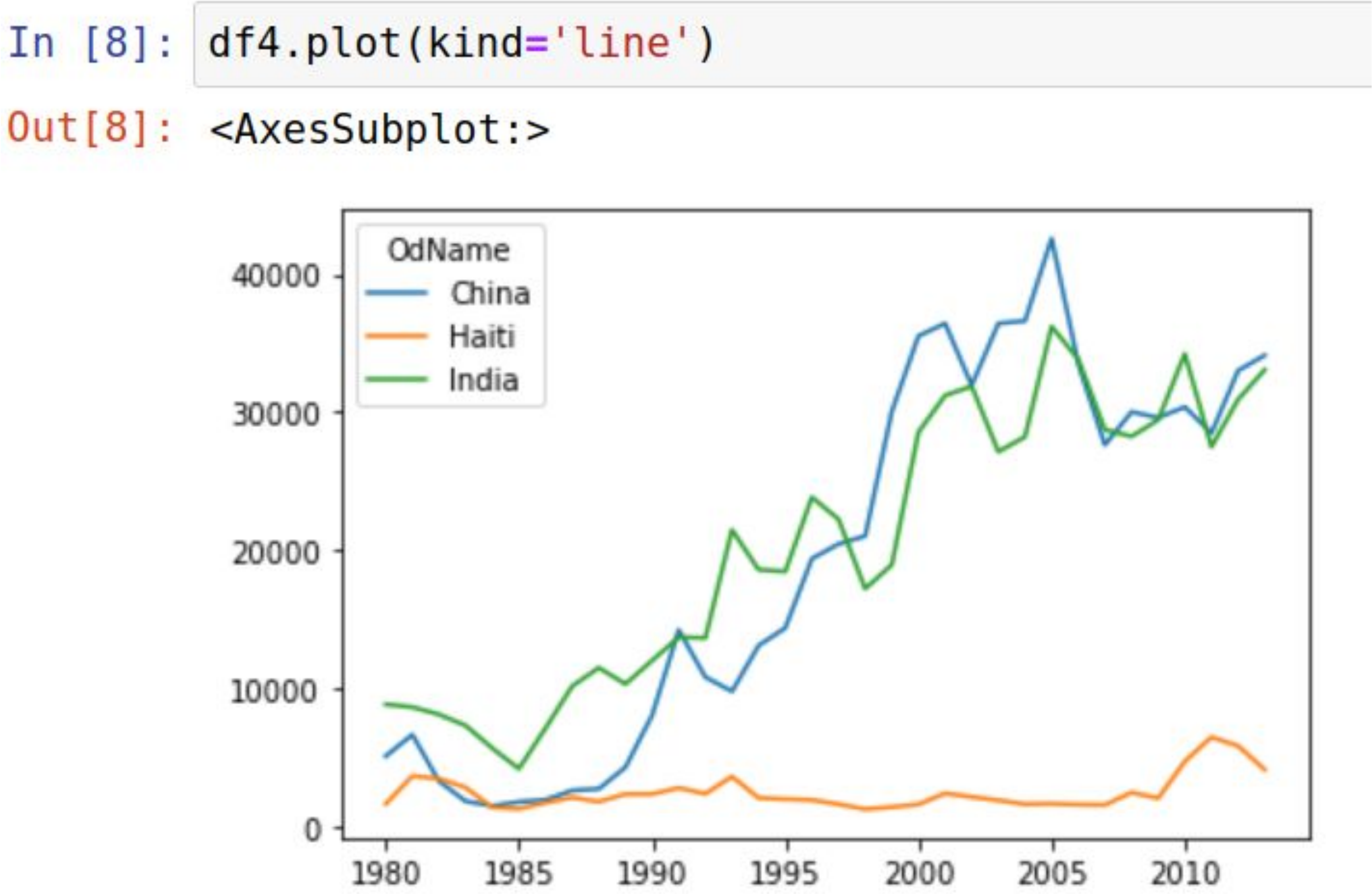
In [7]:

```
df4 = df3.transpose()
df4.head()
```

Out[7]:

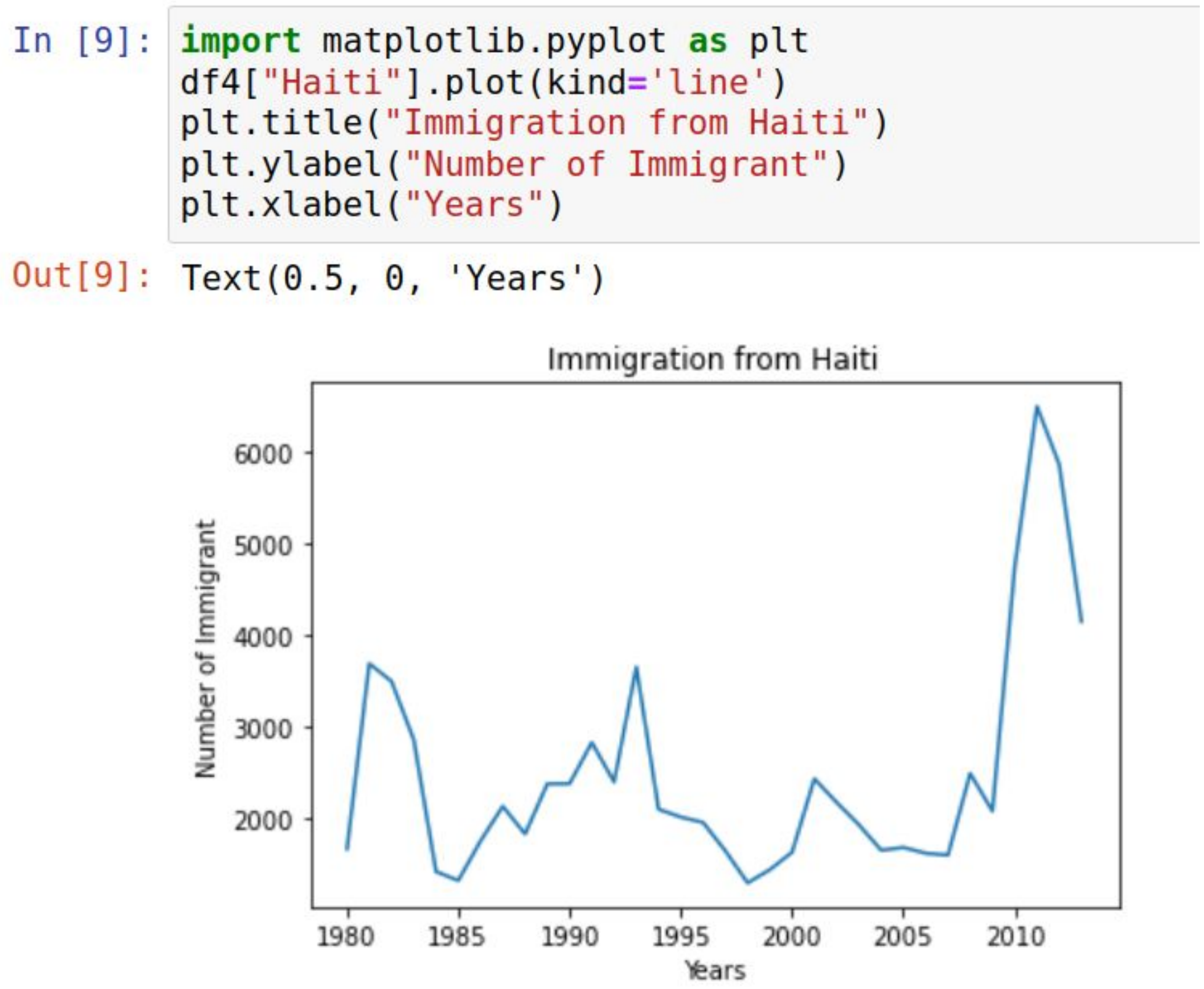
OdName	China	Haiti	India
1980	5123	1666	8880
1981	6682	3692	8670
1982	3308	3498	8147
1983	1863	2860	7338
1984	1527	1418	5704

Pandas Plot Example - Cell 8



```
In [8]: df4.plot(kind='line')

Out[8]: <AxesSubplot:>
```

Pandas Plot Example - Cell 9



```
In [9]: import matplotlib.pyplot as plt
        df4["Haiti"].plot(kind='line')
        plt.title("Immigration from Haiti")
        plt.ylabel("Number of Immigrant")
        plt.xlabel("Years")
```

```
Out[9]: Text(0.5, 0, 'Years')
```

Pandas Plot Example - Cell 10

```
In [10]: df3_ = df2.loc["Haiti", list(map(str, range(1980,2014))) ].plot(kind='line')
```

6000
5000
4000
3000
2000
1980
1985
1990
1995
2000
2005
2010

Questions

Links

https://github.com/fcai-b/dv

References

1. https://www.coursera.org/learn/python-for-data-visualization