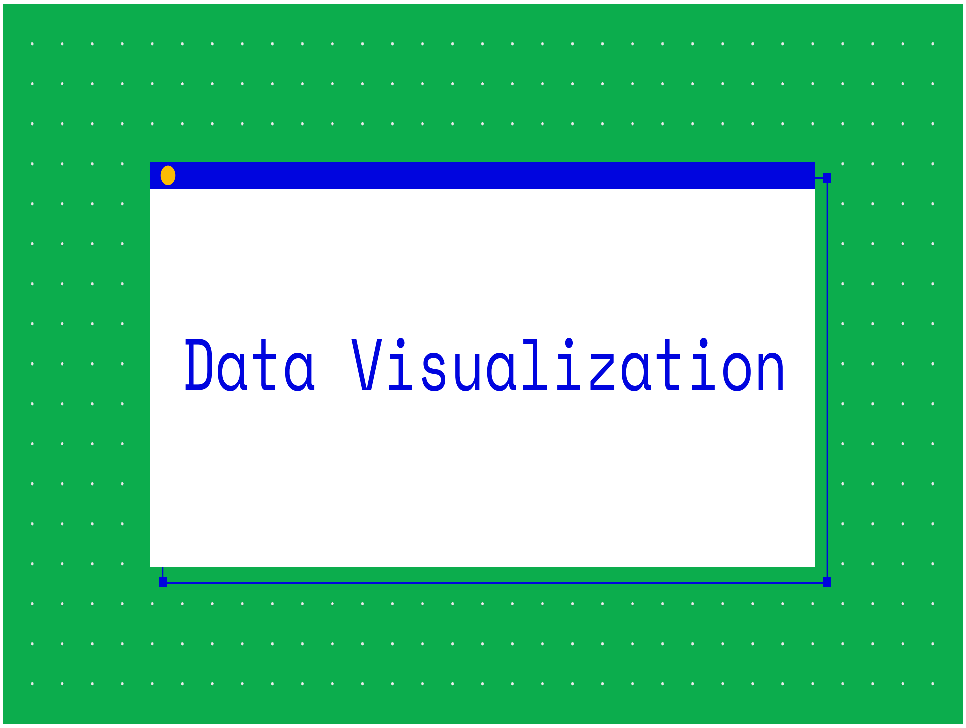
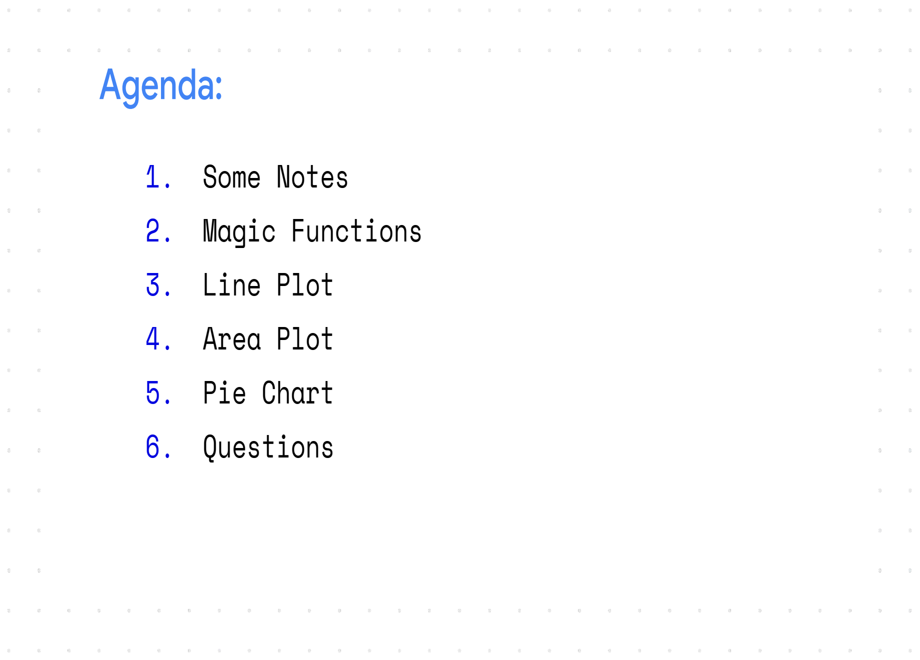


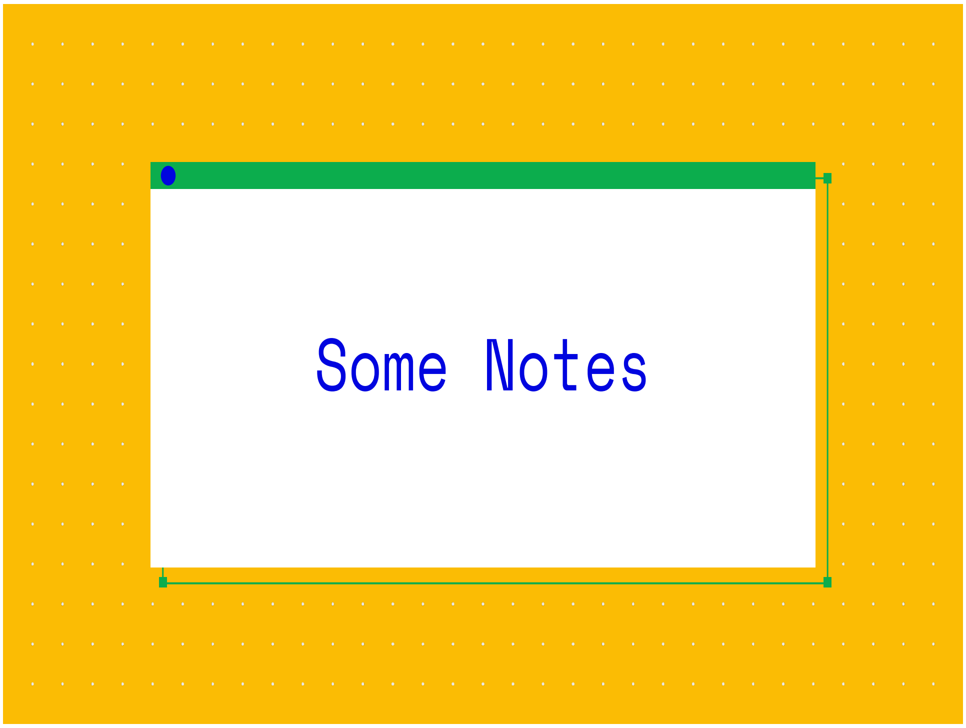
Data Visualization



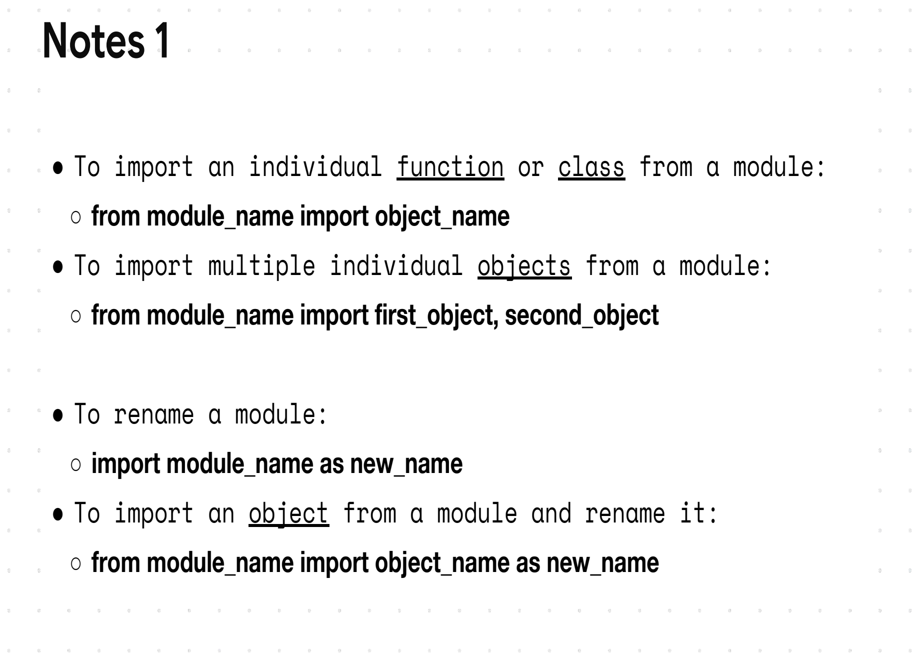
Agenda:

1. Some Notes
2. Magic Functions
3. Line Plot
4. Area Plot
5. Pie Chart
6. Questions

Here is a quick overview of what we'll cover today.



Some Notes



Notes 1

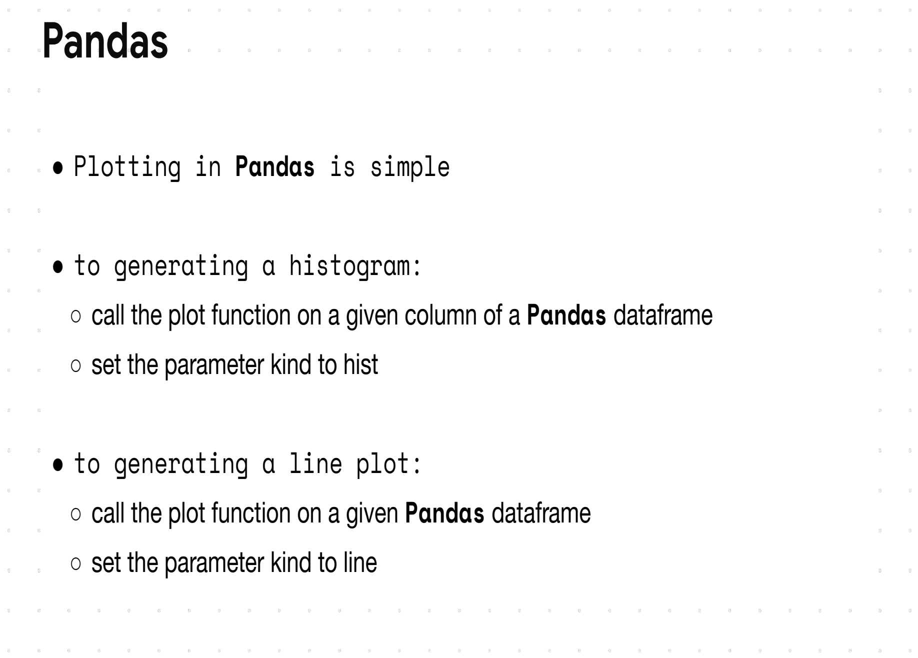
- To import an individual function or class from a module:
  - **from module_name import object_name**
- To import multiple individual objects from a module:
  - **from module_name import first_object, second_object**

- To rename a module:
  - **import module_name as new_name**
- To import an object from a module and rename it:
  - **from module_name import object_name as new_name**

Notes 2

- To import every object individually from a module:
  - **from module_name import ***
    - DO NOT DO THIS

- To use all of the objects from a module, use the following:
  - **import module_name**
    - And access each of the objects with the dot notation.



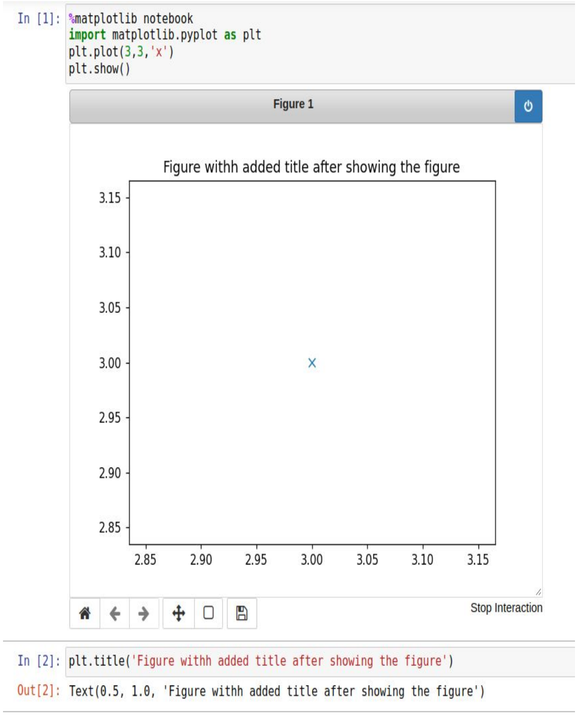
Pandas

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

- Pandas has a built-in implementation of Matplotlib

Magic Functions

%matplotlib notebook



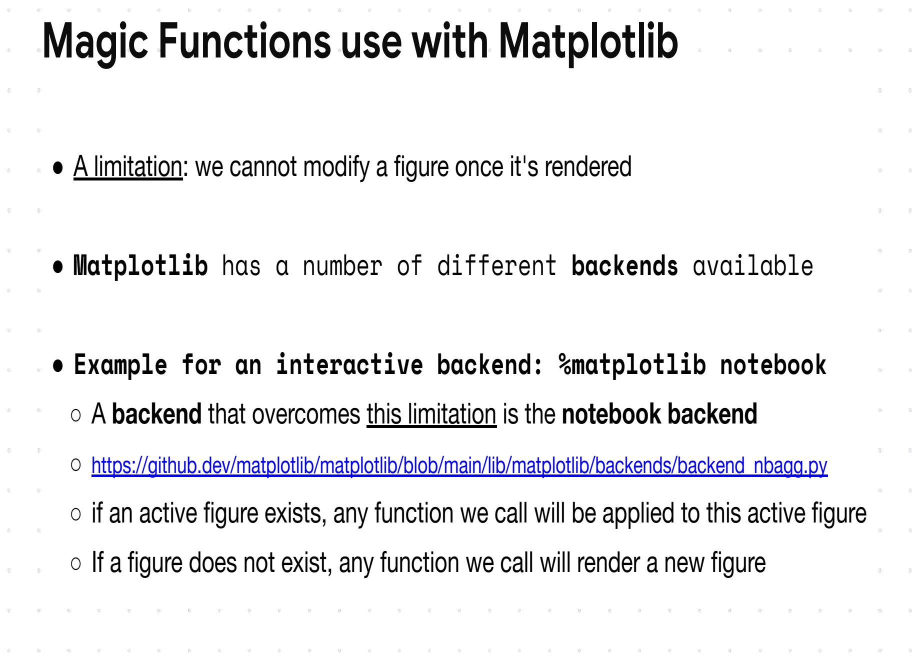
```
In [1]: %matplotlib notebook
        import matplotlib.pyplot as plt
        plt.plot(3,3,'x')
        plt.show()
```

```
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

- Python calls them on our behalf in specific circumstances.

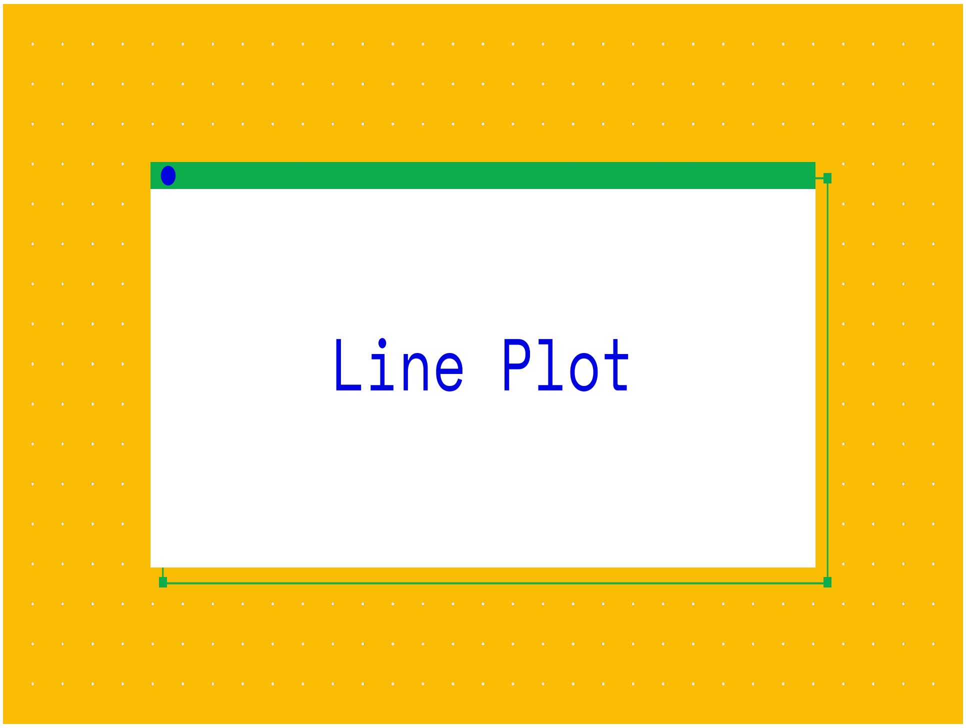
- Have this name to distinguish them from other functions



Magic Functions use with Matplotlib

- A limitation: we cannot modify a figure once it's rendered
- **Matplotlib** has a number of different **backends** available
- **Example for an interactive backend: %matplotlib notebook**
  - A **backend** that overcomes this limitation is the **notebook backend**
  - https://github.dev/matplotlib/matplotlib/blob/main/lib/matplotlib/backends/backend_nbagg.py
  - 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

%matplotlib notebook is an interactive backend for Matplotlib when used within a Jupyter Notebook or IPython environment.



Line Plot

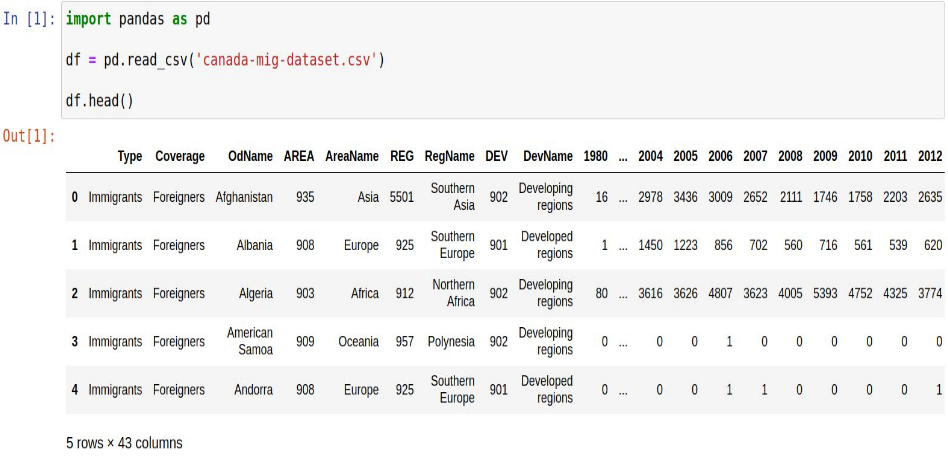
Line Plot

- 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]:

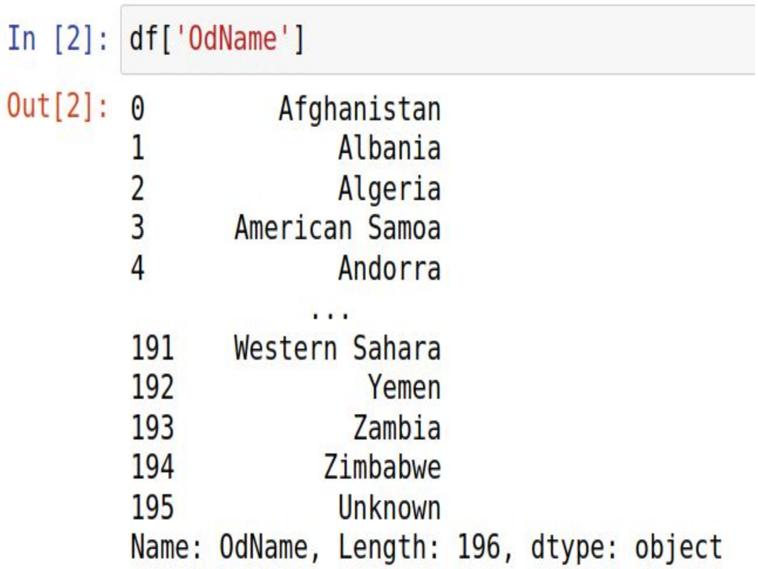
```
import pandas as pd

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

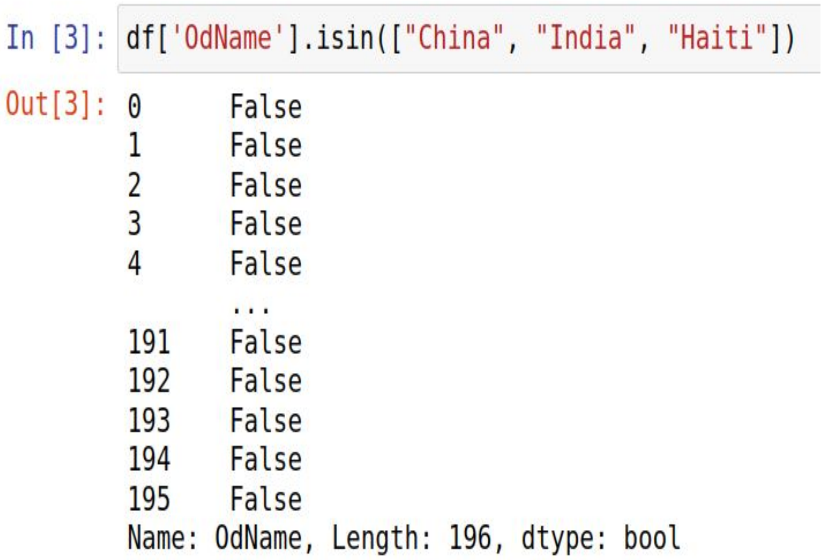
Pandas Plot Example - Cell 2



```
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
```

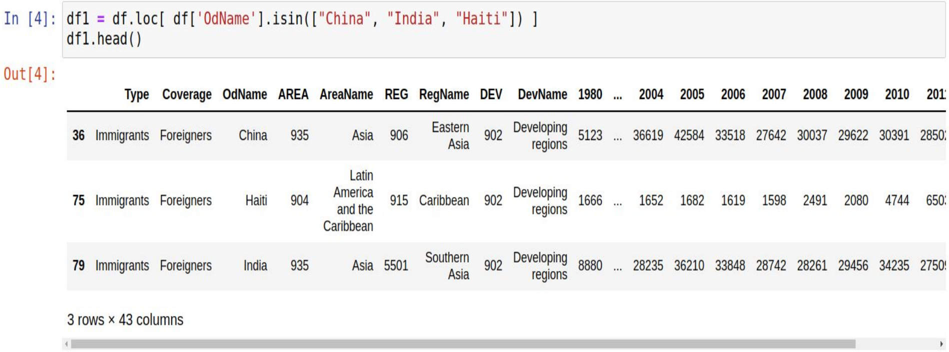
Pandas Plot Example - Cell 3



```
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



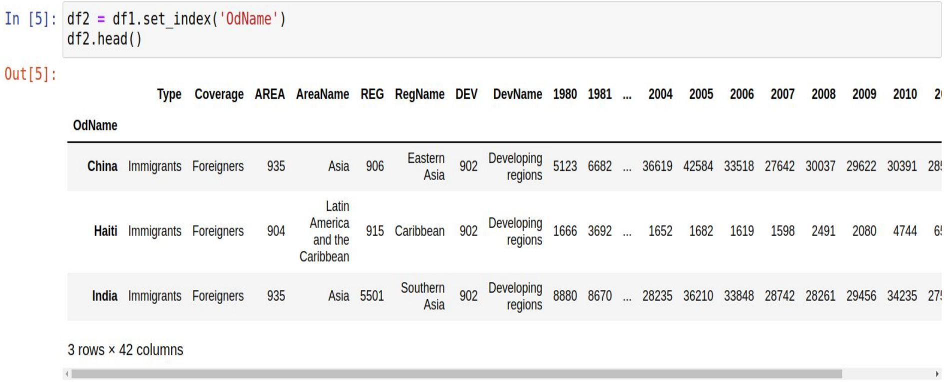
```
In [4]: df1 = df.loc[ df['OdName'].isin(["China", "India", "Haiti"]) ]
        df1.head()
```

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



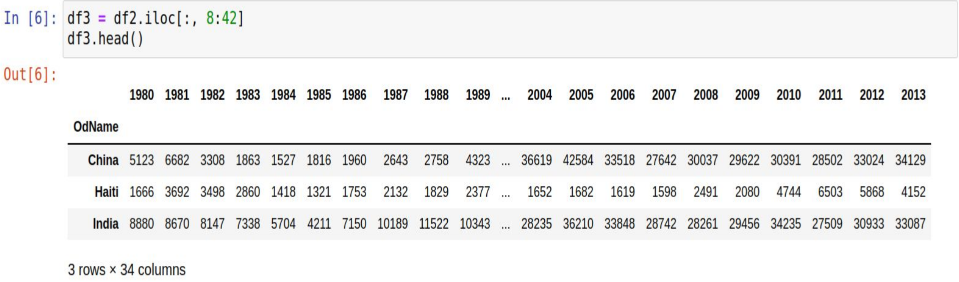
In [5]:
```
df2 = df1.set_index('OdName')
df2.head()
```

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



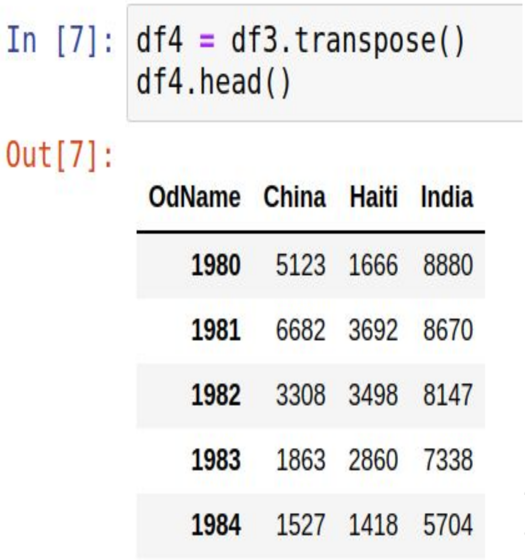
```
In [6]: df3 = df2.iloc[:, 8:42]
        df3.head()
```

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

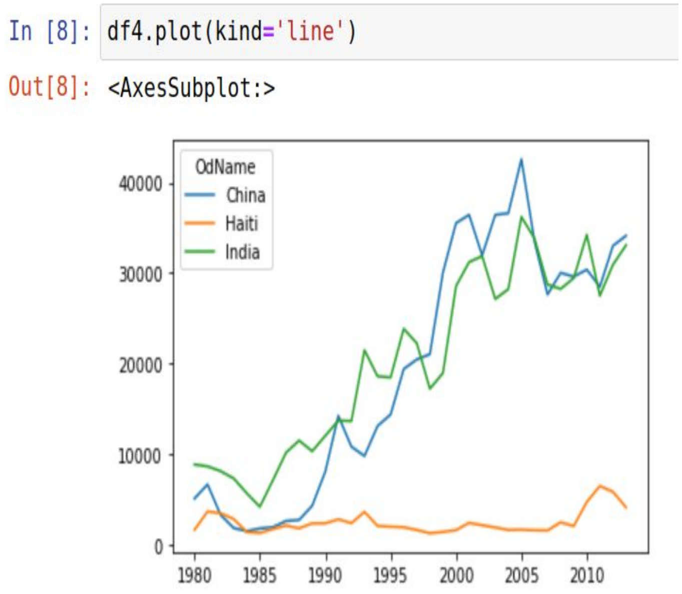
Pandas Plot Example - Cell 7



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

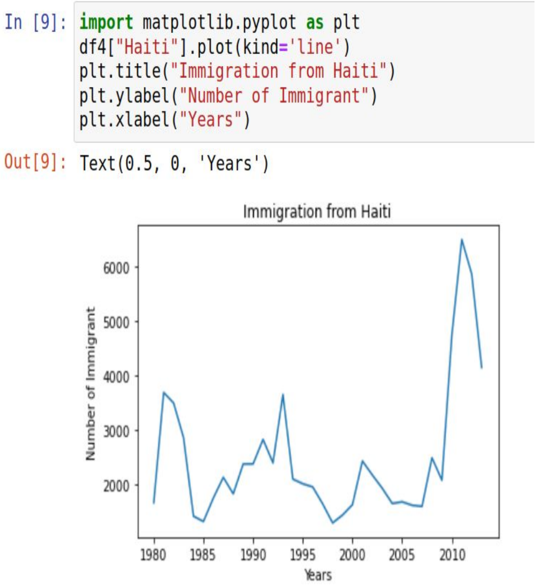
OdName	China	Haiti	India
1980	5123	1666	8880
1981	6682	3692	8670
1982	3308	3498	8147
1983	1863	2860	7338
1984	1527	1418	5704

Pandas Plot Example - Cell 8



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

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

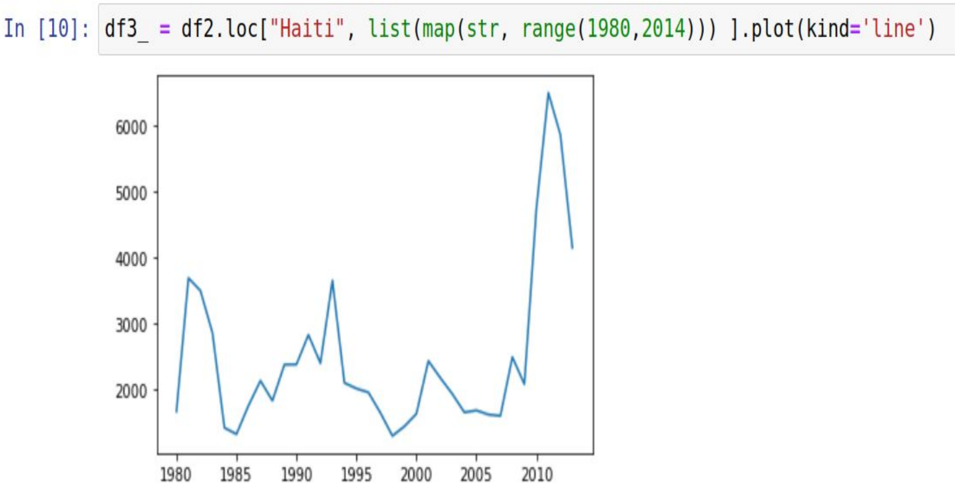
Pandas Plot Example - Cell 9



```
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')
```

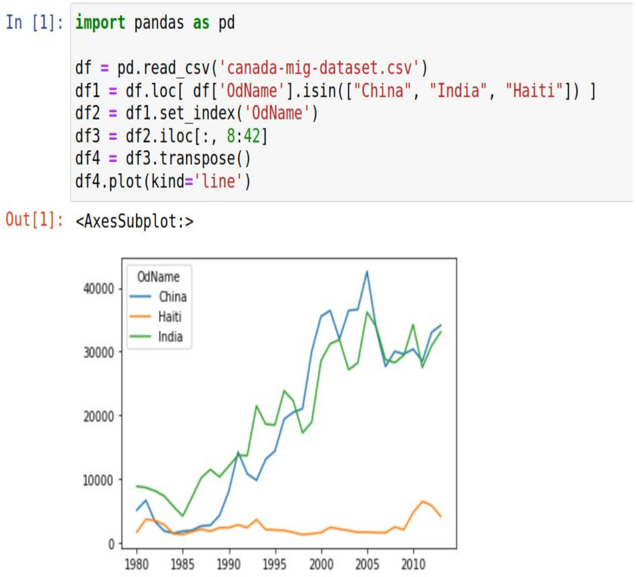
Pandas Plot Example - Cell 10



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

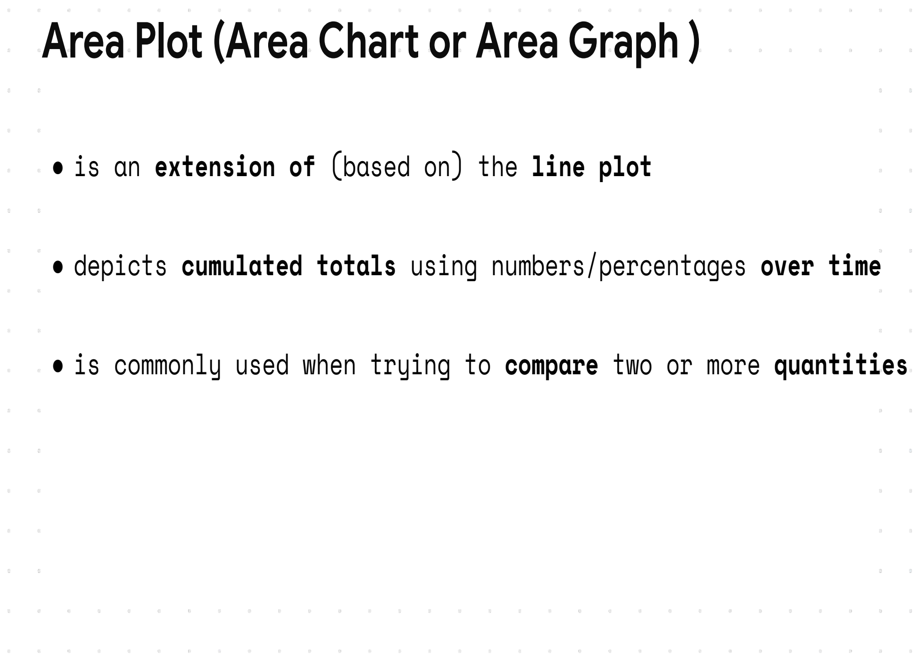
- **range(0,3)**
  - returns a class of immutable iterable objects that lets you iterate over them
  - does not produce lists
  - does not store all the elements in the range in memory
  - instead produces the elements on the fly (as you are iterating over them)
- **list(range(0,3))**
  - produces a list (by iterating over all the elements and appending to the list)
- if you only want to iterate over that range of values
  - **range(0,3)** would be faster because **list(range(0,3))** has the overhead of producing a list before you start iterating over it

Line Plot - Complete Example



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

        df = pd.read_csv('canada-mig-dataset.csv')
        df1 = df.loc[ df['OdName'].isin(["China", "India", "Haiti"]) ]
        df2 = df1.set_index('OdName')
        df3 = df2.iloc[:, 8:42]
        df4 = df3.transpose()
        df4.plot(kind='line')

Out[1]: <AxesSubplot:>
```

Area Plot



Area Plot (Area Chart or Area Graph)

- is an **extension of** (based on) the **line plot**
- depicts **cumulated totals** using numbers/percentages **over time**
- is commonly used when trying to **compare** two or more **quantities**

Chart

- A Chart is the broadest and most general term.
- It refers to a graphic representation of data, where the primary purpose is usually to categorize or compare information for a general audience.
- **Main Goal**: Simplify data to show comparisons, proportions, and simple relationships.
- **Data Focus:** Often deals with a mix of categorical (qualitative) and quantitative data.
- **Examples**:
  - Pie Charts (showing proportions)
  - Donut Charts
  - Bar Charts (comparing discrete categories)

Graph

- The term Graph is typically used to highlight relationships, trends, and patterns between **two or more quantitative variables**, often over a continuous measure like time.
- **Main Goal**: Explore relationships and changes in data over a continuous range.
- **Data Focus**: Usually involves two or more quantitative (numeric) variables.
- **Examples**:
    - Line Graphs (time series)

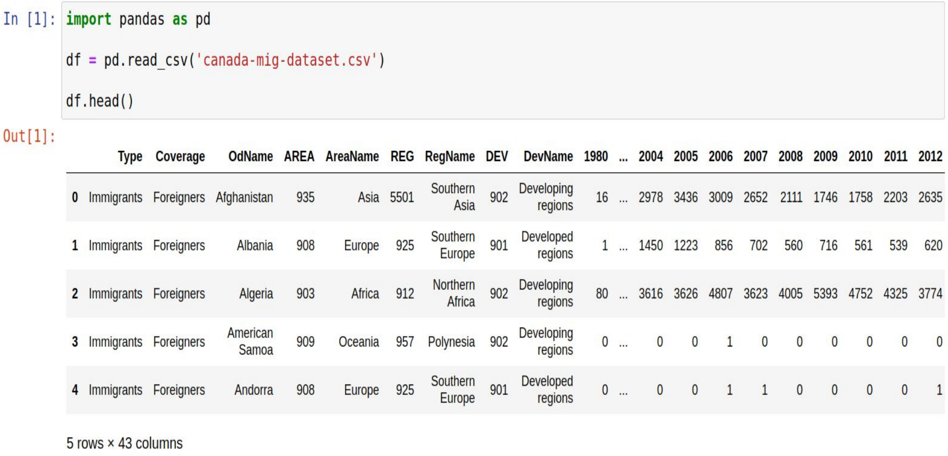
Plot

- A Plot is an action and also the resulting diagram.
- It is most often used to describe the result of plotting (marking) points or data on a coordinate system (like an X-Y plane) to visualize the relationship between variables.
- **Main Goal**: Visualize the relationship and distribution of data points.
- **Data Focus**: Highly focused on quantitative data points.
- **Examples**:
    - Scatter Plot
    - Line Plot (often synonymous with Line Graph)

Diagram

- A Diagram is the most general term in this context
- It refers to any simplified, structural, or abstract visual representation of information, concepts, systems, or relationships.
- It is not necessarily tied to numerical data or coordinate axes.

Area Plot Example - Cell 1



In [1]:

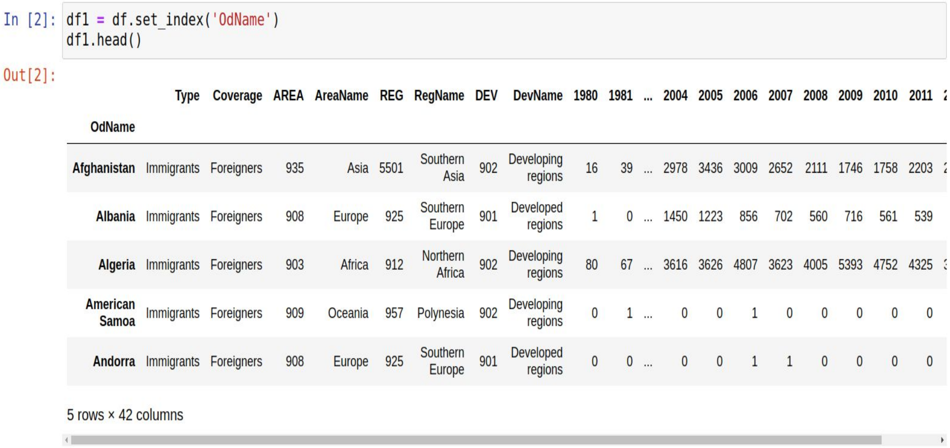
```
import pandas as pd

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

Area Plot Example - Cell 2



In [2]:

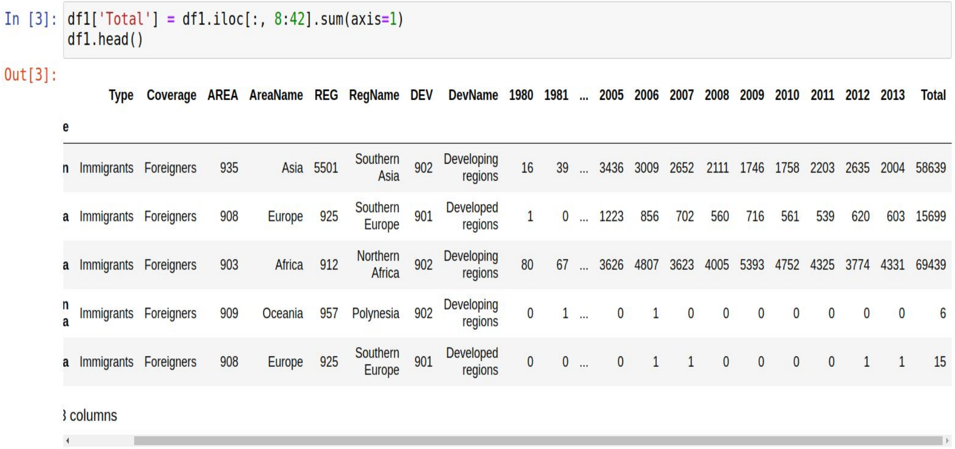
```
df1 = df.set_index('OdName')
df1.head()
```

Out[2]:

	Type	Coverage	AREA	AreaName	REG	RegName	DEV	DevName	1980	1981	...	2004	2005	2006	2007	2008	2009	2010	2011
OdName																			
Afghanistan	Immigrants	Foreigners	935	Asia	5501	Southern Asia	902	Developing regions	16	39	...	2978	3436	3009	2652	2111	1746	1758	2203
Albania	Immigrants	Foreigners	908	Europe	925	Southern Europe	901	Developed regions	1	0	...	1450	1223	856	702	560	716	561	539
Algeria	Immigrants	Foreigners	903	Africa	912	Northern Africa	902	Developing regions	80	67	...	3616	3626	4807	3623	4005	5393	4752	4325
American Samoa	Immigrants	Foreigners	909	Oceania	957	Polynesia	902	Developing regions	0	1	...	0	0	1	0	0	0	0	0
Andorra	Immigrants	Foreigners	908	Europe	925	Southern Europe	901	Developed regions	0	0	...	0	0	1	1	0	0	0	0

5 rows × 42 columns

Area Plot Example - Cell 3



In [3]:

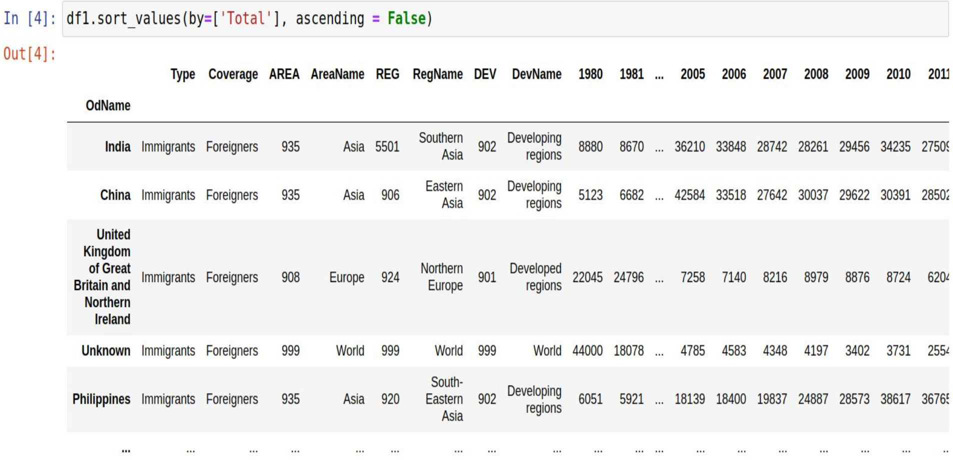
```
df1['Total'] = df1.iloc[:, 8:42].sum(axis=1)
df1.head()
```

Out[3]:

	Type	Coverage	AREA	AreaName	REG	RegName	DEV	DevName	1980	1981	...	2005	2006	2007	2008	2009	2010	2011	2012	2013	Total
e																					
n	Immigrants	Foreigners	935	Asia	5501	Southern Asia	902	Developing regions	16	39	...	3436	3009	2652	2111	1746	1758	2203	2635	2004	58639
a	Immigrants	Foreigners	908	Europe	925	Southern Europe	901	Developed regions	1	0	...	1223	856	702	560	716	561	539	620	603	15699
a	Immigrants	Foreigners	903	Africa	912	Northern Africa	902	Developing regions	80	67	...	3626	4807	3623	4005	5393	4752	4325	3774	4331	69439
n a	Immigrants	Foreigners	909	Oceania	957	Polynesia	902	Developing regions	0	1	...	0	1	0	0	0	0	0	0	0	6
a	Immigrants	Foreigners	908	Europe	925	Southern Europe	901	Developed regions	0	0	...	0	1	1	0	0	0	0	1	1	15

3 columns

Area Plot Example - Cell 4

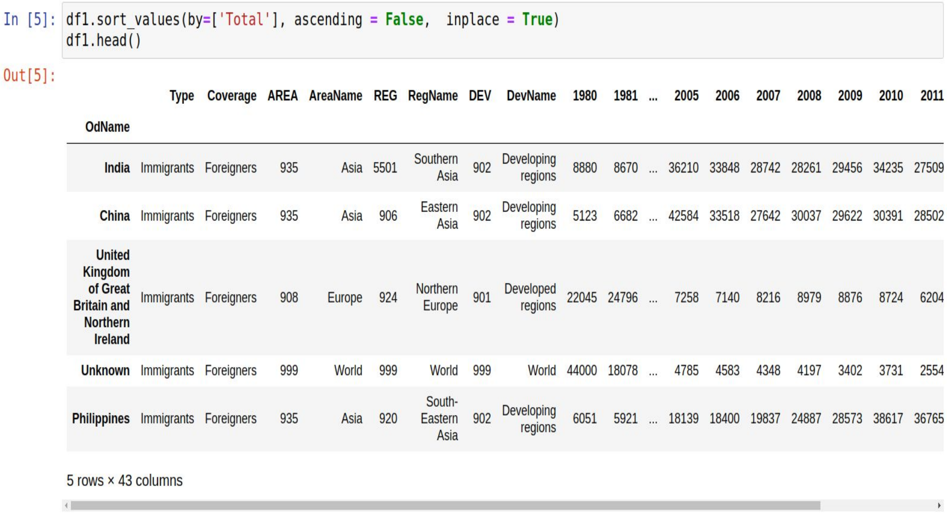


```
In [4]: df1.sort_values(by=['Total'], ascending = False)
```

Out[4]:

	Type	Coverage	AREA	AreaName	REG	RegName	DEV	DevName	1980	1981	...	2005	2006	2007	2008	2009	2010	2011
OdName																		
India	Immigrants	Foreigners	935	Asia	5501	Southern Asia	902	Developing regions	8880	8670	...	36210	33848	28742	28261	29456	34235	27509
China	Immigrants	Foreigners	935	Asia	906	Eastern Asia	902	Developing regions	5123	6682	...	42584	33518	27642	30037	29622	30391	28502
United Kingdom of Great Britain and Northern Ireland	Immigrants	Foreigners	908	Europe	924	Northern Europe	901	Developed regions	22045	24796	...	7258	7140	8216	8979	8876	8724	6204
Unknown	Immigrants	Foreigners	999	World	999	World	999	World	44000	18078	...	4785	4583	4348	4197	3402	3731	2554
Philippines	Immigrants	Foreigners	935	Asia	920	South-Eastern Asia	902	Developing regions	6051	5921	...	18139	18400	19837	24887	28573	38617	36765
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	..

Area Plot Example - Cell 5



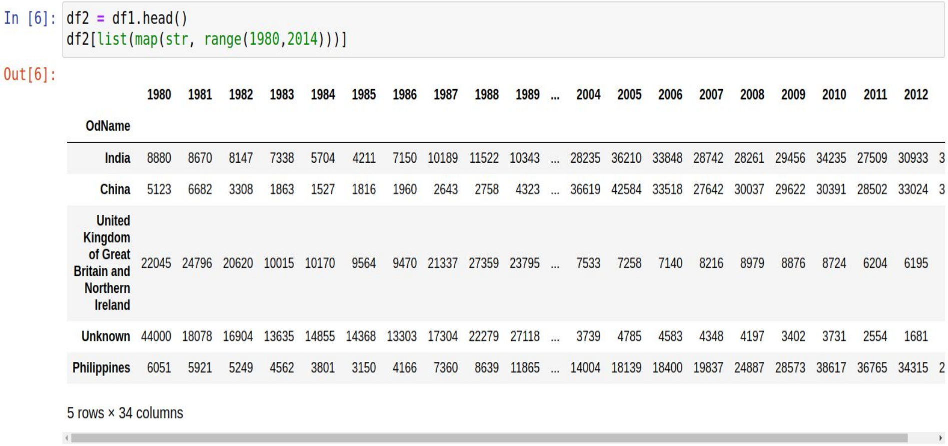
```
In [5]: df1.sort_values(by=['Total'], ascending = False,  inplace = True)
        df1.head()
```

Out[5]:

	Type	Coverage	AREA	AreaName	REG	RegName	DEV	DevName	1980	1981	...	2005	2006	2007	2008	2009	2010	2011
OdName																		
India	Immigrants	Foreigners	935	Asia	5501	Southern Asia	902	Developing regions	8880	8670	...	36210	33848	28742	28261	29456	34235	27509
China	Immigrants	Foreigners	935	Asia	906	Eastern Asia	902	Developing regions	5123	6682	...	42584	33518	27642	30037	29622	30391	28502
United Kingdom of Great Britain and Northern Ireland	Immigrants	Foreigners	908	Europe	924	Northern Europe	901	Developed regions	22045	24796	...	7258	7140	8216	8979	8876	8724	6204
Unknown	Immigrants	Foreigners	999	World	999	World	999	World	44000	18078	...	4785	4583	4348	4197	3402	3731	2554
Philippines	Immigrants	Foreigners	935	Asia	920	South-Eastern Asia	902	Developing regions	6051	5921	...	18139	18400	19837	24887	28573	38617	36765

5 rows × 43 columns

Area Plot Example - Cell 6



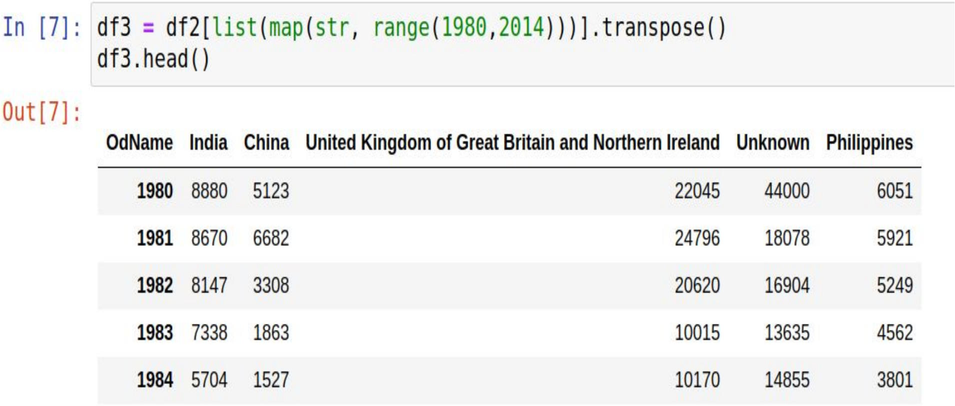
```
In [6]: df2 = df1.head()
        df2[list(map(str, range(1980,2014)))]
```

Out[6]:

	1980	1981	1982	1983	1984	1985	1986	1987	1988	1989	...	2004	2005	2006	2007	2008	2009	2010	2011	2012	
OdName																					
India	8880	8670	8147	7338	5704	4211	7150	10189	11522	10343	...	28235	36210	33848	28742	28261	29456	34235	27509	30933	3
China	5123	6682	3308	1863	1527	1816	1960	2643	2758	4323	...	36619	42584	33518	27642	30037	29622	30391	28502	33024	3
United Kingdom of Great Britain and Northern Ireland	22045	24796	20620	10015	10170	9564	9470	21337	27359	23795	...	7533	7258	7140	8216	8979	8876	8724	6204	6195	
Unknown	44000	18078	16904	13635	14855	14368	13303	17304	22279	27118	...	3739	4785	4583	4348	4197	3402	3731	2554	1681	
Philippines	6051	5921	5249	4562	3801	3150	4166	7360	8639	11865	...	14004	18139	18400	19837	24887	28573	38617	36765	34315	2

5 rows × 34 columns

Area Plot Example - Cell 7



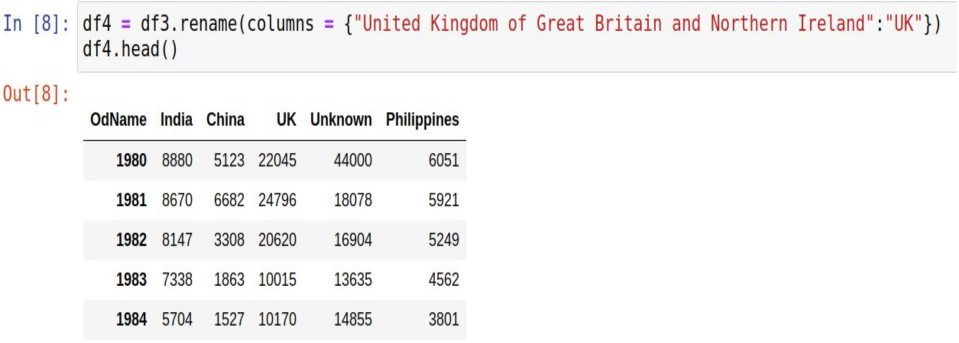
In [7]:

```
df3 = df2[list(map(str, range(1980,2014)))].transpose()
df3.head()
```

Out[7]:

OdName	India	China	United Kingdom of Great Britain and Northern Ireland	Unknown	Philippines
1980	8880	5123	22045	44000	6051
1981	8670	6682	24796	18078	5921
1982	8147	3308	20620	16904	5249
1983	7338	1863	10015	13635	4562
1984	5704	1527	10170	14855	3801

Area Plot Example - Cell 8

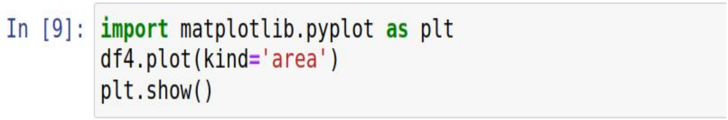


```
In [8]: df4 = df3.rename(columns = {"United Kingdom of Great Britain and Northern Ireland":"UK"})
        df4.head()
```

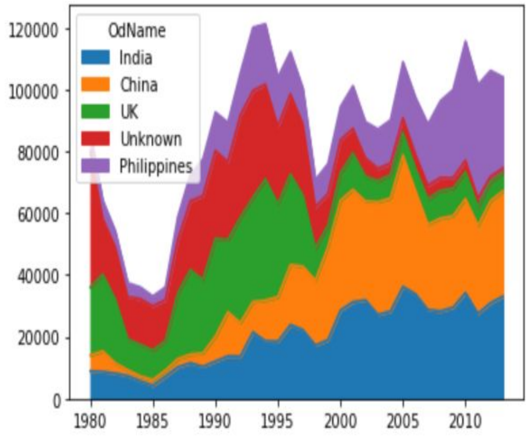
Out[8]:

OdName	India	China	UK	Unknown	Philippines
1980	8880	5123	22045	44000	6051
1981	8670	6682	24796	18078	5921
1982	8147	3308	20620	16904	5249
1983	7338	1863	10015	13635	4562
1984	5704	1527	10170	14855	3801

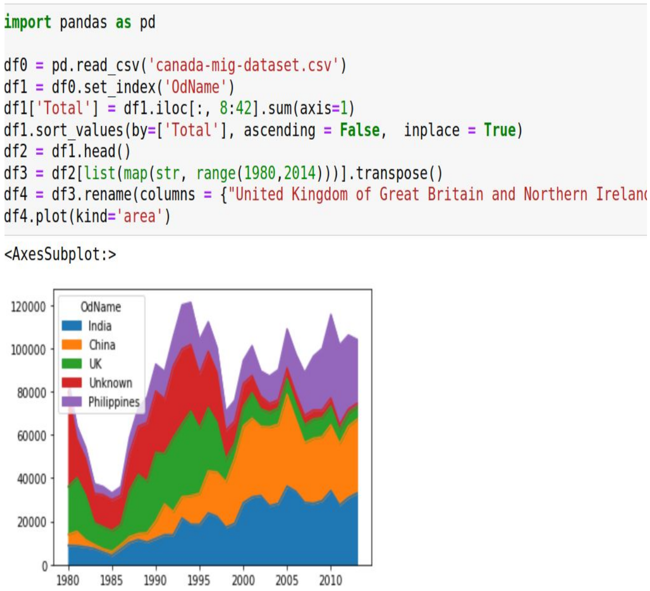
Area Plot Example - Cell 9



```
In [9]: import matplotlib.pyplot as plt
        df4.plot(kind='area')
        plt.show()
```



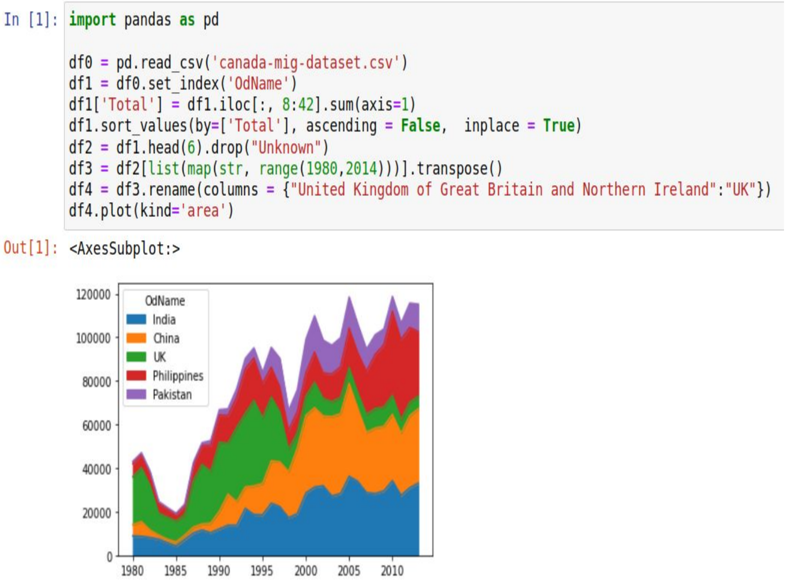
Area Plot - Complete Example



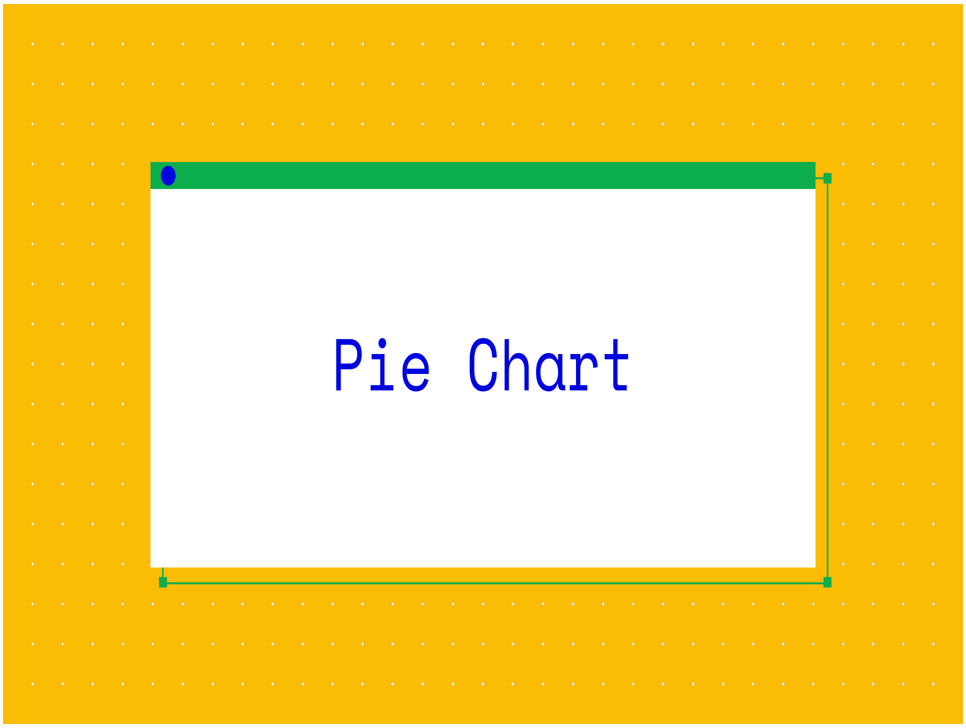
```
import pandas as pd

df0 = pd.read_csv('canada-mig-dataset.csv')
df1 = df0.set_index('OdName')
df1['Total'] = df1.iloc[:, 8:42].sum(axis=1)
df1.sort_values(by=['Total'], ascending = False,  inplace = True)
df2 = df1.head()
df3 = df2[list(map(str, range(1980,2014)))].transpose()
df4 = df3.rename(columns = {"United Kingdom of Great Britain and Northern Irelanc
df4.plot(kind='area')
```

```
<AxesSubplot:>
```

Area Plot - Complete Example - No Unknown

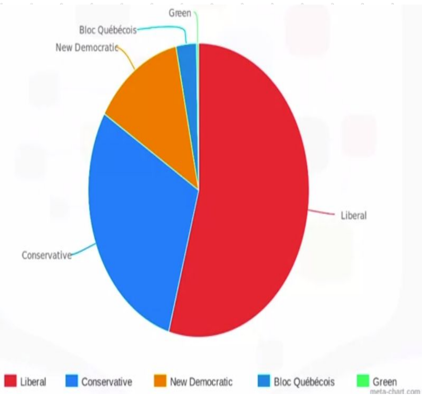


```
In [1]: import pandas as pd

        df0 = pd.read_csv('canada-mig-dataset.csv')
        df1 = df0.set_index('OdName')
        df1['Total'] = df1.iloc[:, 8:42].sum(axis=1)
        df1.sort_values(by=['Total'], ascending = False,  inplace = True)
        df2 = df1.head(6).drop("Unknown")
        df3 = df2[list(map(str, range(1980,2014)))].transpose()
        df4 = df3.rename(columns = {"United Kingdom of Great Britain and Northern Ireland":"UK"})
        df4.plot(kind='area')

Out[1]: <AxesSubplot:>
```


Pie Chart

Pie Chart

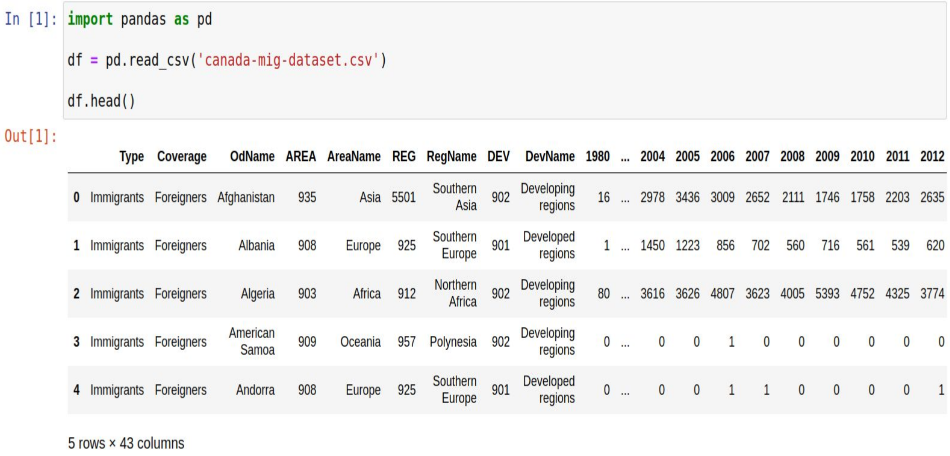
- is a circular statistical graphic
  - divided into slices
  - to illustrate numerical proportion



- **Example**
  - the Canadian federal election in 2015
  - were Liberals in red won more than 50% of the seats in the House of Commons

- There are some very vocal opponents to the use of pie charts
  - Most argue that pie charts fail to accurately display data with any consistency

Pie Chart Example - Cell 1



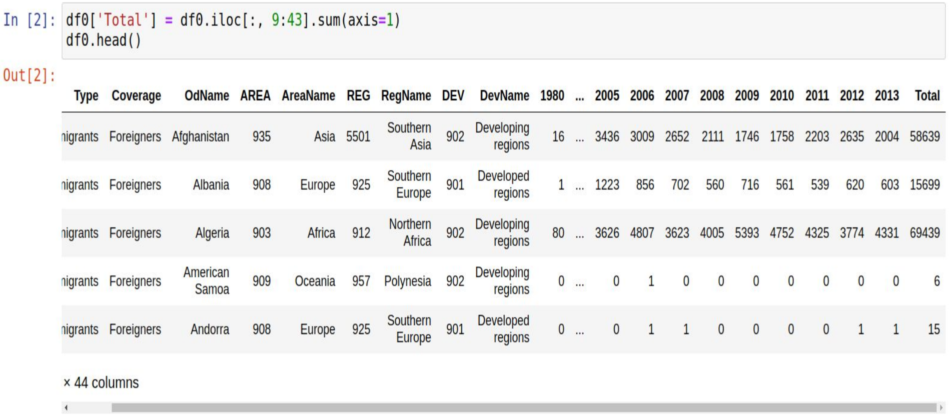
In [1]:
```
import pandas as pd

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

Pie Chart Example - Cell 2



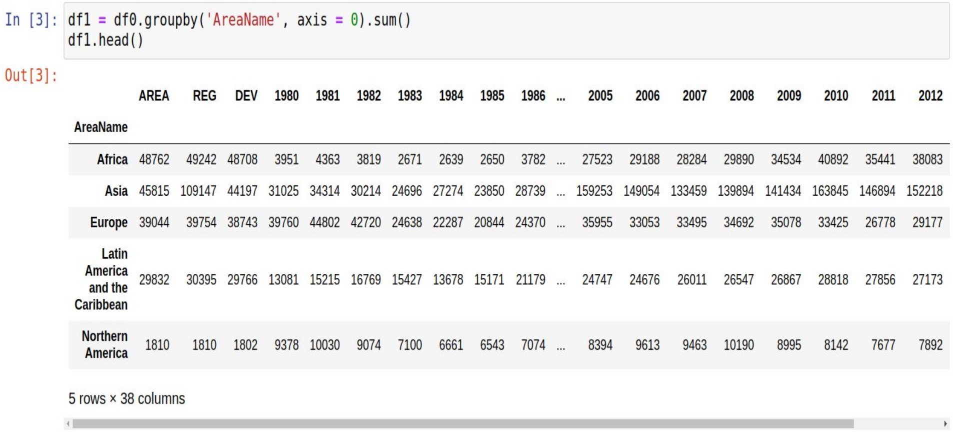
In [2]:
```
df0['Total'] = df0.iloc[:, 9:43].sum(axis=1)
df0.head()
```

Out[2]:

Type	Coverage	OdName	AREA	AreaName	REG	RegName	DEV	DevName	1980	...	2005	2006	2007	2008	2009	2010	2011	2012	2013	Total
nigrants	Foreigners	Afghanistan	935	Asia	5501	Southern Asia	902	Developing regions	16	...	3436	3009	2652	2111	1746	1758	2203	2635	2004	58639
nigrants	Foreigners	Albania	908	Europe	925	Southern Europe	901	Developed regions	1	...	1223	856	702	560	716	561	539	620	603	15699
nigrants	Foreigners	Algeria	903	Africa	912	Northern Africa	902	Developing regions	80	...	3626	4807	3623	4005	5393	4752	4325	3774	4331	69439
nigrants	Foreigners	American Samoa	909	Oceania	957	Polynesia	902	Developing regions	0	...	0	1	0	0	0	0	0	0	0	6
nigrants	Foreigners	Andorra	908	Europe	925	Southern Europe	901	Developed regions	0	...	0	1	1	0	0	0	0	1	1	15

× 44 columns

Pie Chart Example - Cell 3



In [3]:

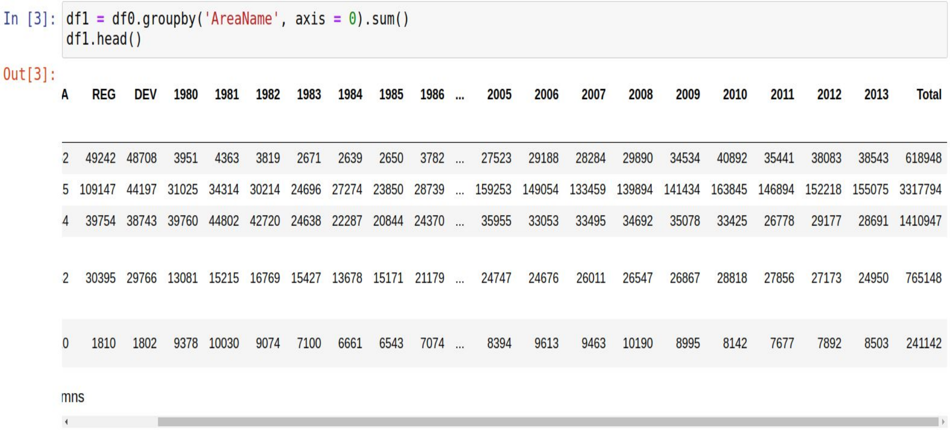
```
df1 = df0.groupby('AreaName', axis = 0).sum()
df1.head()
```

Out[3]:

AreaName	AREA	REG	DEV	1980	1981	1982	1983	1984	1985	1986	...	2005	2006	2007	2008	2009	2010	2011	2012
Africa	48762	49242	48708	3951	4363	3819	2671	2639	2650	3782	...	27523	29188	28284	29890	34534	40892	35441	38083
Asia	45815	109147	44197	31025	34314	30214	24696	27274	23850	28739	...	159253	149054	133459	139894	141434	163845	146894	152218
Europe	39044	39754	38743	39760	44802	42720	24638	22287	20844	24370	...	35955	33053	33495	34692	35078	33425	26778	29177
Latin America and the Caribbean	29832	30395	29766	13081	15215	16769	15427	13678	15171	21179	...	24747	24676	26011	26547	26867	28818	27856	27173
Northern America	1810	1810	1802	9378	10030	9074	7100	6661	6543	7074	...	8394	9613	9463	10190	8995	8142	7677	7892

5 rows × 38 columns

Pie Chart Example - Cell 3 (showing Total)



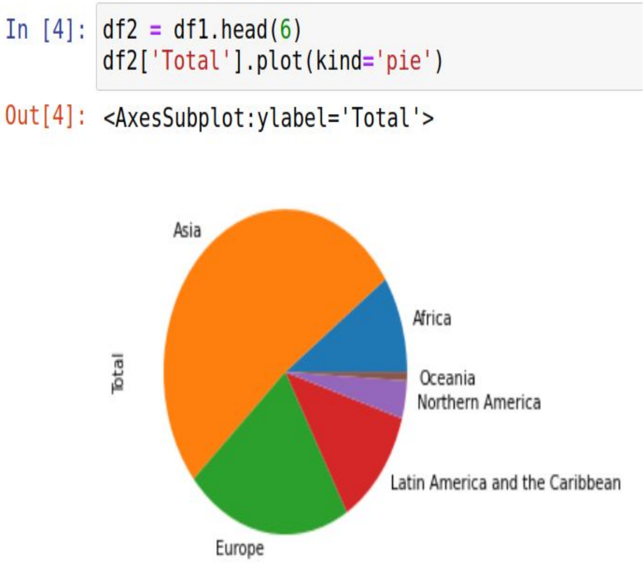
In [3]:
```
df1 = df0.groupby('AreaName', axis = 0).sum()
df1.head()
```

Out[3]:

A	REG	DEV	1980	1981	1982	1983	1984	1985	1986	...	2005	2006	2007	2008	2009	2010	2011	2012	2013	Total
2	49242	48708	3951	4363	3819	2671	2639	2650	3782	...	27523	29188	28284	29890	34534	40892	35441	38083	38543	618948
5	109147	44197	31025	34314	30214	24696	27274	23850	28739	...	159253	149054	133459	139894	141434	163845	146894	152218	155075	3317794
4	39754	38743	39760	44802	42720	24638	22287	20844	24370	...	35955	33053	33495	34692	35078	33425	26778	29177	28691	1410947
2	30395	29766	13081	15215	16769	15427	13678	15171	21179	...	24747	24676	26011	26547	26867	28818	27856	27173	24950	765148
0	1810	1802	9378	10030	9074	7100	6661	6543	7074	...	8394	9613	9463	10190	8995	8142	7677	7892	8503	241142

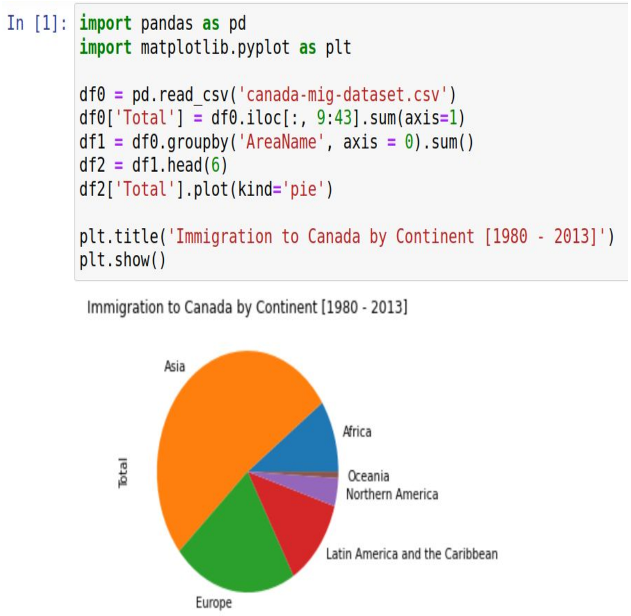
mns

Pie Chart Example - Cell 4



```
In [4]: df2 = df1.head(6)
        df2['Total'].plot(kind='pie')

Out[4]: <AxesSubplot:ylabel='Total'>
```

Pie Chart - Complete Example

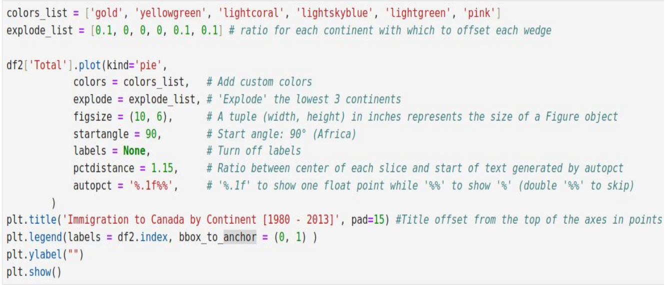


In [1]:

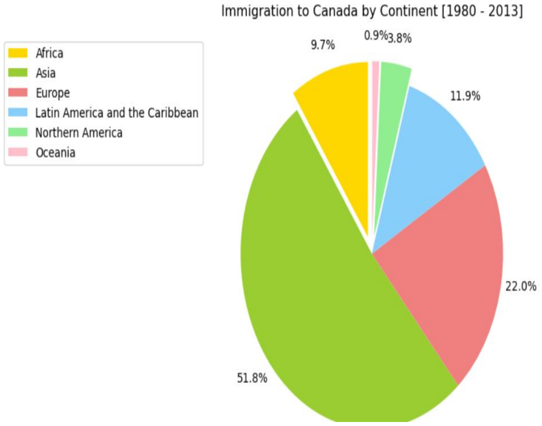
```
import pandas as pd
import matplotlib.pyplot as plt

df0 = pd.read_csv('canada-mig-dataset.csv')
df0['Total'] = df0.iloc[:, 9:43].sum(axis=1)
df1 = df0.groupby('AreaName', axis = 0).sum()
df2 = df1.head(6)
df2['Total'].plot(kind='pie')

plt.title('Immigration to Canada by Continent [1980 - 2013]')
plt.show()
```

Immigration to Canada by Continent [1980 - 2013]

Pie Chart Example - Enhancement



```
colors_list = ['gold', 'yellowgreen', 'lightcoral', 'lightskyblue', 'lightgreen', 'pink']
explode_list = [0.1, 0, 0, 0, 0.1, 0.1] # ratio for each continent with which to offset each wedge

df2['Total'].plot(kind='pie',
            colors = colors_list,   # Add custom colors
            explode = explode_list, # 'Explode' the lowest 3 continents
            figsize = (10, 6),      # A tuple (width, height) in inches represents the size of a Figure object
            startangle = 90,        # Start angle: 90° (Africa)
            labels = None,          # Turn off labels
            pctdistance = 1.15,     # Ratio between center of each slice and start of text generated by autopct
            autopct = '%.1f%%',     # '%.1f' to show one float point while '%%' to show '%' (double '%%' to skip)
        )
plt.title('Immigration to Canada by Continent [1980 - 2013]', pad=15) #Title offset from the top of the axes in points
plt.legend(labels = df2.index, bbox_to_anchor = (0, 1) )
plt.ylabel("")
plt.show()
```



Points in Matplotlib title function refer to **typographic points**

- A **typographic point** is a unit of measurement used in typography, approximately equal to 1/72 of an inch.
- In the context of Matplotlib, it's used to specify distances and sizes in a way that's independent of the specific display resolution.
- So, when you set the **pad parameter** to a certain number of points,
  - you're essentially telling Matplotlib to move the title that many points away from the top of the **axes.**
  - This ensures that the title is positioned consistently, regardless of the size of the figure or the font size used.

Pie Chart Notes 1

- Limit the number of slices plotted
- A **pie chart** works best with two or three slices
    - It's possible to plot four/five slices as long as the wedge sizes can be distinguished

- If we have
        - many categories or
        - categories with small representation
    - we can
        - re-group them together so that fewer wedges are plotted, or
        - use an 'Other' category to handle them

Pie Chart Notes 2

- One typical method of plotting a pie chart
    - Start from the top of the circle,
    - then plot each categorical level clockwise from most frequent to least frequent

- If you have three categories and are
    - interested in the comparison of two of them
        - You may place the two categories on either side of the 12 o'clock direction
        - with the third category filling in the remaining space at the bottom

- **Donut Chart**
    - Use its hole to make use of available space better (add statistics)

Pie Chart vs Bar chart

- There are some very vocal opponents to the use of pie charts
  - Most argue that pie charts fail to accurately display data with any consistency

- Bar charts are much better when it comes to representing the data in a consistent way and getting the message

Questions

Links

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

References

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