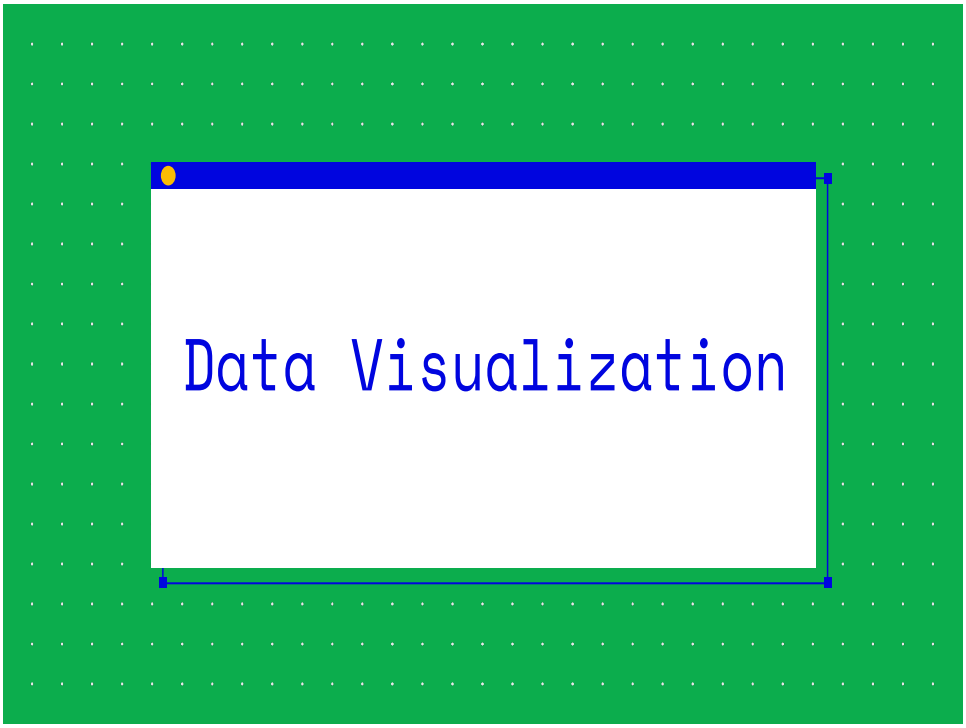

Data Visualization

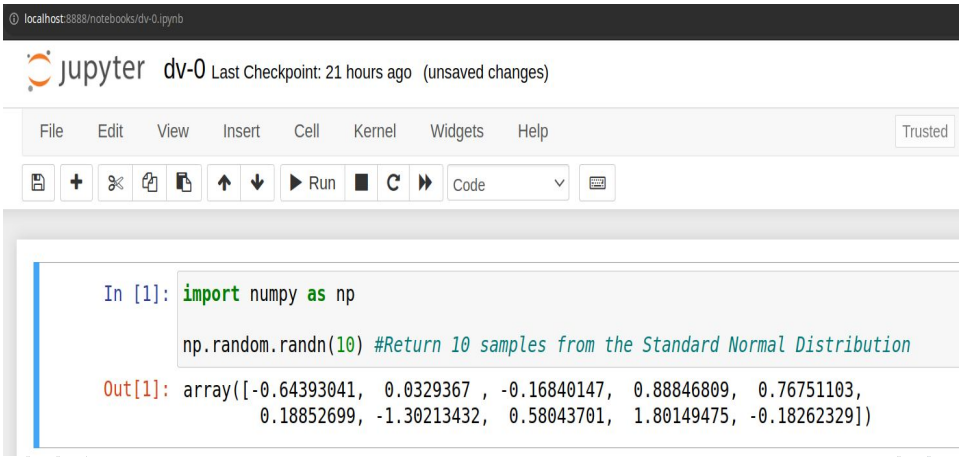
Agenda:

1. Matplotlib Scripting Layer Examples
2. Histogram
3. Bar Chart
4. Questions

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

Matplotlib Scripting Layer Examples

NumPy Example



NumPy: Numerical Python library for **Scientific Computing**

https://github.dev/numpy/numpy/blob/main/numpy/random/__init__.py

NumPy: Numerical Python **library**

NumPy.random is a **Package**

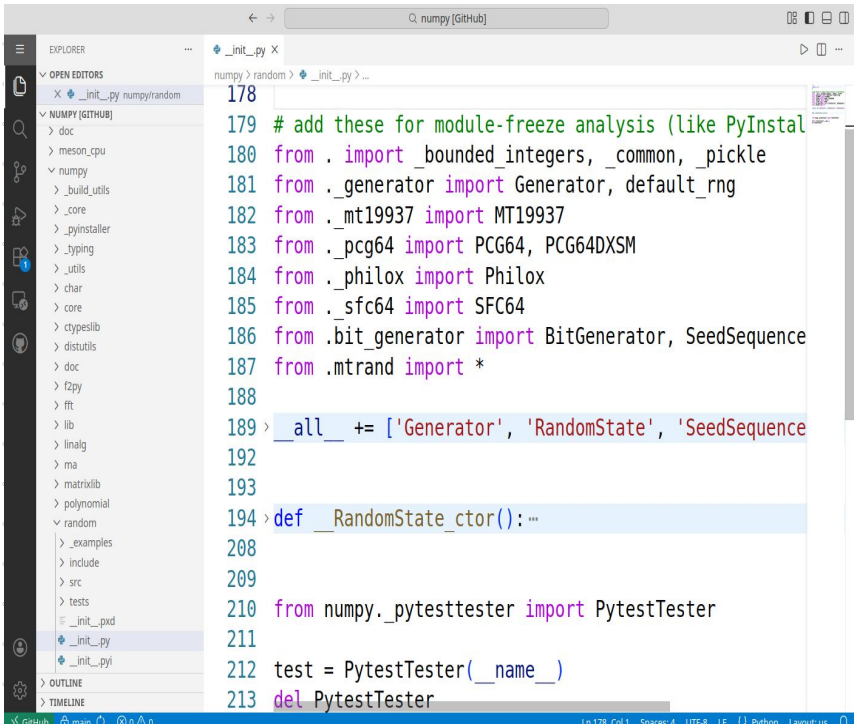
- provides functions for generating random numbers from various probability distributions.
- These functions are used to create random arrays, matrices, and other data structures.
- Here are some common functions:
    - **rand():** Generates random numbers uniformly distributed between 0 and 1.
    - **randn()**: Generates random numbers from a standard normal distribution.
    - **randint()**: Generates random integers within a specified range.

Check the code

- https://github.dev/numpy/numpy/blob/main/numpy/random/__init__.py
- https://github.dev/numpy/numpy/blob/main/numpy/matlib.py
- Ctrl+Shift+P to open command palette in vscode
- Then write the command >fold all (Ctrl+k+0)

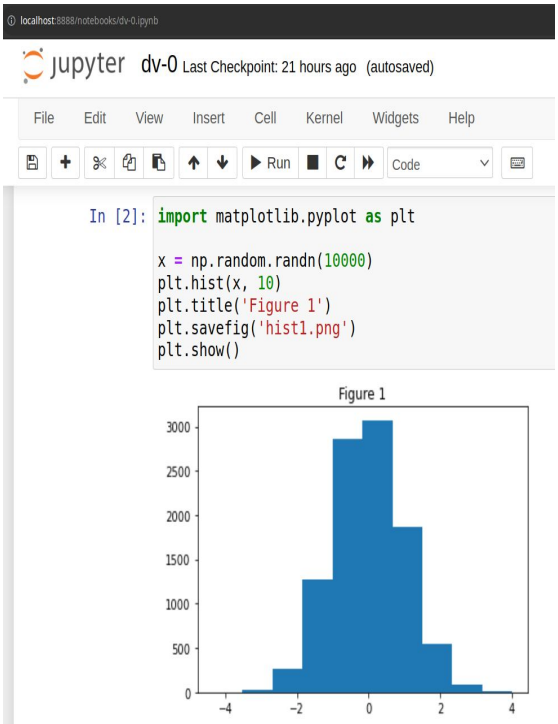
__init__.py file

- used to mark a directory as a **Python package**
- used to initialize the package when it is imported
- can contain
    - code (that will be executed when the package is imported)
    - function definitions
    - variable assignments


numpy [GitHub]
EXPLORER
init.py
OPEN EDITORS
init.py numpy/random
NUMPY [GITHUB]
doc
meson_cpu
numpy
_build_utils
_core
_pyinstaller
_typing
_utils
char
core
ctypeslib
distutils
doc
f2py
fft
lib
linalg
ma
matrixlib
polynomial
random
_examples
include
src
tests
init.pxd
init.py
init.pyi
OUTLINE
TIMELINE
numpy > random > _init_.py > ...
178
179 # add these for module-freeze analysis (like PyInstal
180 from . import _bounded_integers, _common, _pickle
181 from ._generator import Generator, default_rng
182 from ._mt19937 import MT19937
183 from ._pcg64 import PCG64, PCG64DXSM
184 from ._philox import Philox
185 from ._sfc64 import SFC64
186 from .bit_generator import BitGenerator, SeedSequence
187 from .mtrand import *
188
189 __all__ += ['Generator', 'RandomState', 'SeedSequence
192
193
194 def __RandomState_ctor():...
208
209
210 from numpy._pytesttester import PytestTester
211
212 test = PytestTester(__name__)
213 del PytestTester

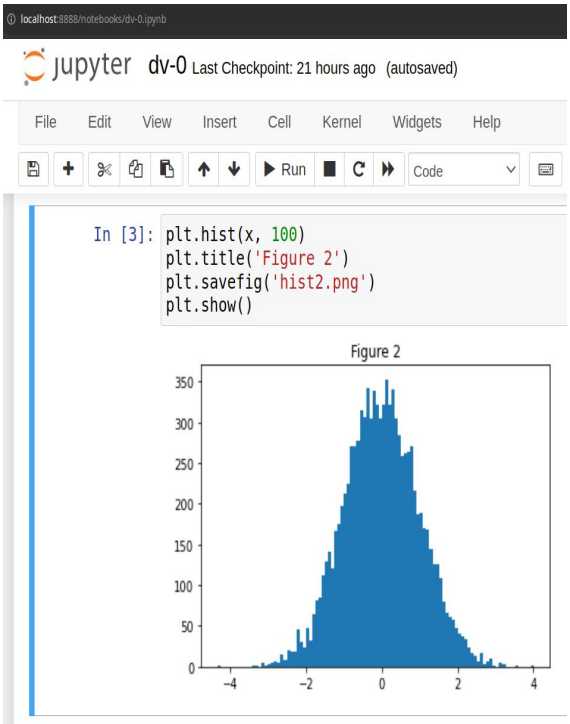
Scripting Layer Example 1

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



Scripting Layer Example 2

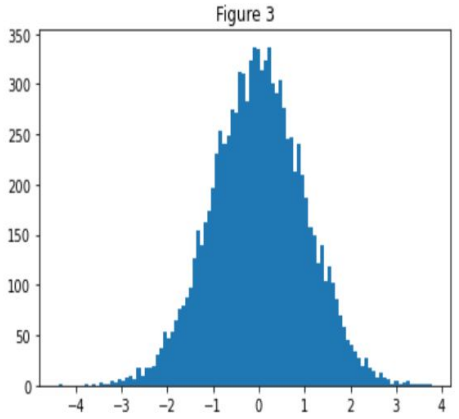
- `all methods`
  - creating/manipulating
    - histogram
    - other Artist objects
  - are part of **pyplot**



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()
```



Histogram

Histogram

- represents the **frequency distribution** of a **numeric dataset**
- is a way to represent the **frequency distribution** of a variable

- The way it works is:
  - partitions the spread of the numeric data into bins
  - assigns each datapoint in the dataset to a bin
  - counts the number of datapoints that have been assigned to each bin

- **Vertical axis**
  - is actually the frequency or the number of datapoints in each bin

Histogram

- If you have a **quantitative variable**,
  - then your **univariable plot** of choice will probably be the **histogram**

- When creating a **histogram**,
  - Play around with different bin sizes
  - to better understanding the plotted variable

- https://www.youtube.com/watch?v=RLez9L0htGQ&t=84s

If you have a quantitative variable, then your univariable plot of choice will probably be the histogram.
When creating a histogram, playing around with different bin sizes is an excellent way to understand the plotted variable better.

Histogram 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

- Python library for data analysis and data manipulation
- offers data structures and operations for manipulating numerical tables and time series

Histogram Example - Cell 2

In [2]:
```
df1 = df.set_index('OdName')
df1.head()
```

Out[2]:

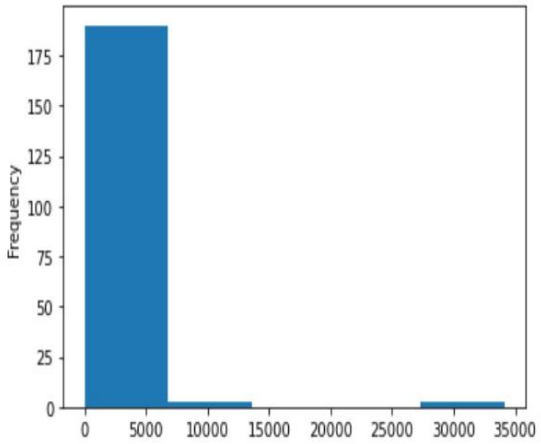
OdName	Type	Coverage	AREA	AreaName	REG	RegName	DEV	DevName	1980	1981	...	2004	2005	2006	2007	2008	2009	2010	2011	2
Afghanistan	Immigrants	Foreigners	935	Asia	5501	Southern Asia	902	Developing regions	16	39	...	2978	3436	3009	2652	2111	1746	1758	2203	2
Albania	Immigrants	Foreigners	908	Europe	925	Southern 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	3
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

Histogram Example - Cell 3

```
In [3]: df1['2013'].plot(kind='hist', bins = 5)
```

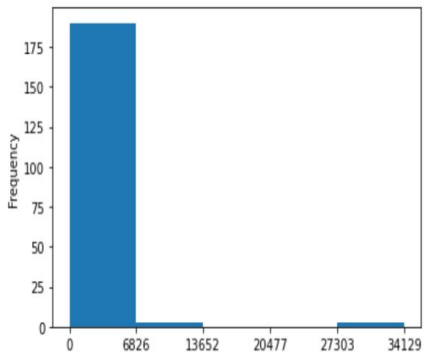
```
Out[3]: <AxesSubplot:ylabel='Frequency'>
```



Histogram Example - Cell 4

In [4]:
```
import numpy as np
count, bin_edges = np.histogram(df1['2013'], bins=5)
df1['2013'].plot(kind='hist', bins = 5, xticks = bin_edges)
```

Out[4]: <AxesSubplot:ylabel='Frequency'>




```
count = [190   3   0   0   3]
bin_edges = [    0.   6825.8 13651.6 20477.4 27303.2 34129. ]
```

Histogram

- **Numpy histogram function**
  - partitions the spread of the column data into bins of equal width
  - computes the number of data points that fall in each bin
  - returns the **frequency of each bin** (count) & **bin edges** (bin_edges)

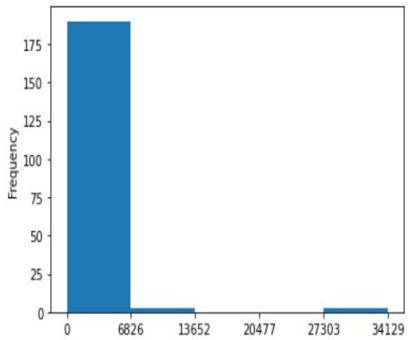
- We then pass these **bin edges** as an additional parameter in our plot function to generate the histogram

Histogram - Complete Example

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

df0 = pd.read_csv('canada-mig-dataset.csv')
df1 = df0.set_index('OdName')
count, bin_edges = np.histogram(df1['2013'], bins=5)
df1['2013'].plot(kind='hist', bins = 5, xticks = bin_edges)
```

Out[1]: <AxesSubplot:ylabel='Frequency'>



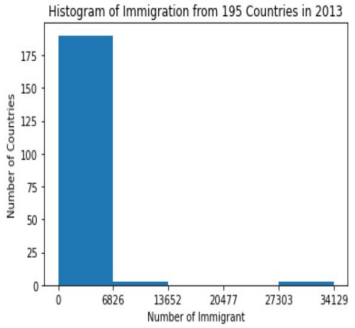
Histogram - Complete Example

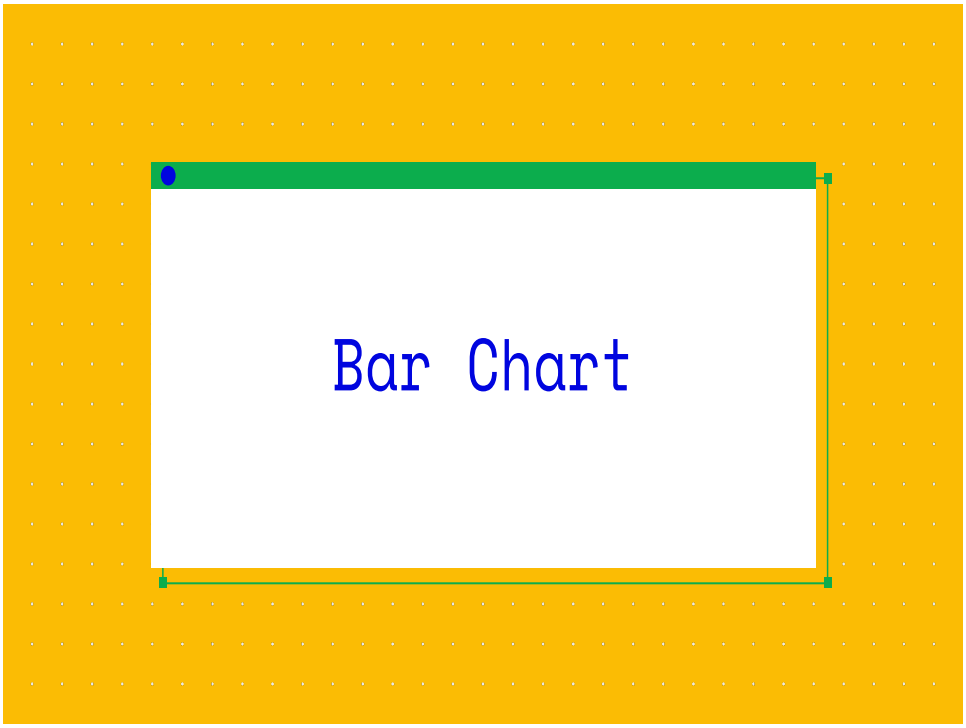
- Add Title
- Add X Label
- Add Y Label

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

df0 = pd.read_csv('canada-mig-dataset.csv')
df1 = df0.set_index('OdName')
count, bin_edges = np.histogram(df1['2013'], bins=5)
df1['2013'].plot(kind='hist', bins = 5, xticks = bin_edges)

plt.title('Histogram of Immigration from 195 Countries in 2013')
plt.xlabel('Number of Immigrant')
plt.ylabel('Number of Countries')
plt.show()
```




Bar Chart

Bar Chart (Bar Graph)

- is a very popular visualization tool
  - the length of each bar is proportional to the value of the item that it represents

- commonly used:
  - to compare the values of a variable at given points in time

- Example:
  - visualizing in discrete fashion how immigration from Iceland to Canada evolved
  - bar height represents total immigration from Iceland to Canada in a specific year

Bar 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

Bar Chart Example - Cell 2

In [2]:
```
df1 = df.set_index('OdName')
df1.head()
```

Out[2]:

OdName	Type	Coverage	AREA	AreaName	REG	RegName	DEV	DevName	1980	1981	...	2004	2005	2006	2007	2008	2009	2010	2011	2
Afghanistan	Immigrants	Foreigners	935	Asia	5501	Southern Asia	902	Developing regions	16	39	...	2978	3436	3009	2652	2111	1746	1758	2203	2
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	3
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

Explain: list(map(str, range(1,5)))

```
range(1,5)
list(range(1,5))
```

```
[1, 2, 3, 4]
```

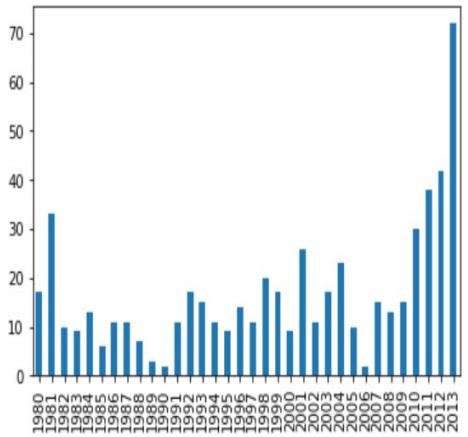
```
range(1,5)
map(str, range(1,5))
list(map(str, range(1,5)))
```

```
['1', '2', '3', '4']
```

Bar Chart Example - Cell 3

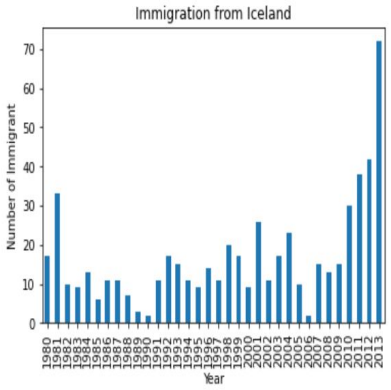
In [3]:
```
df2 = df1.loc[ "Iceland", list(map(str, range(1980,2014))) ]
df2.plot(kind='bar')
```

Out[3]: <AxesSubplot:>



Bar Chart Example - Cell 4

In [4]:
```
import numpy as np
import matplotlib.pyplot as plt
df2.plot(kind='bar')
plt.title("Immigration from Iceland")
plt.ylabel("Number of Immigrant")
plt.xlabel("Year")
plt.show()
```

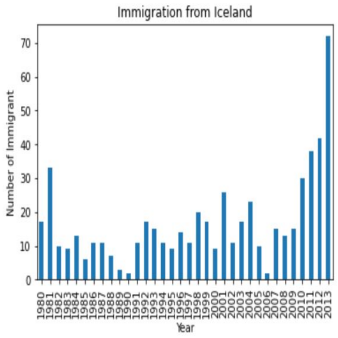


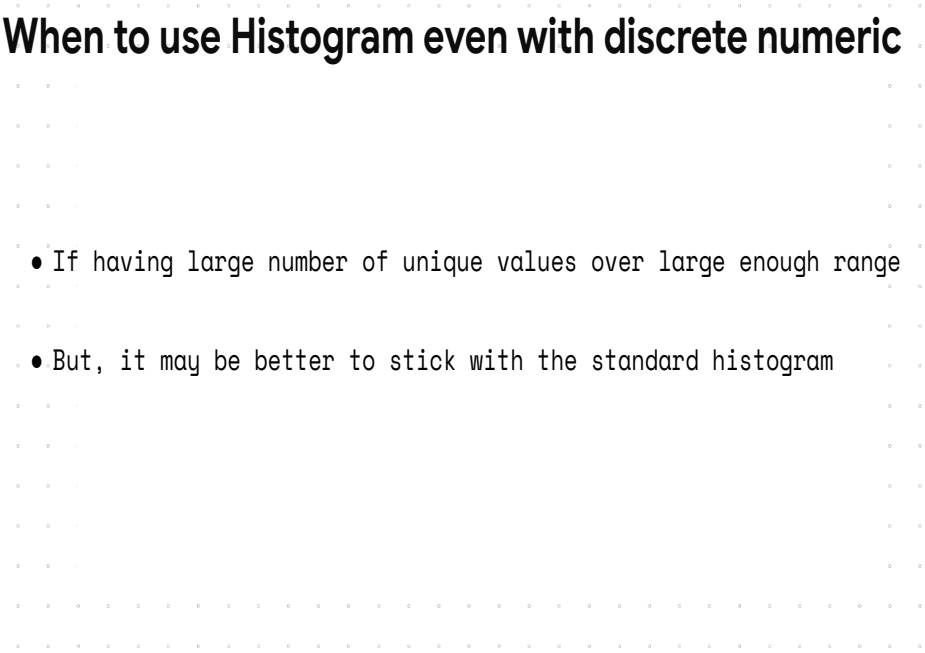
Bar Chart - Complete Example

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt

df0 = pd.read_csv('canada-mig-dataset.csv')
df1 = df0.set_index('OdName')
df2 = df1.loc[ "Iceland", list(map(str, range(1980,2014))) ]
df2.plot(kind='bar')

plt.title("Immigration from Iceland")
plt.ylabel("Number of Immigrant")
plt.xlabel("Year")
plt.show()
```





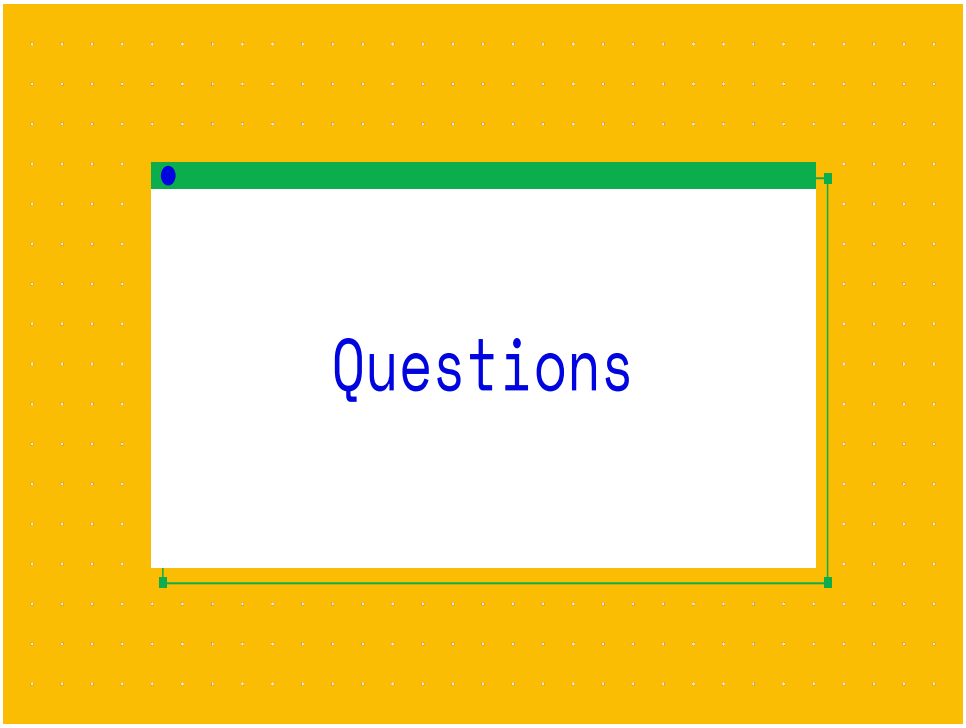
Use Case for using Histogram even with discrete numeric

- If the discrete variable (like the number of orders) has a wide range of values (e.g., from 0 to 100), a standard **bar chart** that shows the count for each specific number would be cluttered and hard to interpret.
- By using a histogram and binning the data, you simplify the visual and highlight the overall shape of the distribution.

There are viewpoints that refuse using histograms for discrete numeric data, arguing it is an inappropriate application that can be misleading, especially when the number of unique discrete values is small.

Pyplot

https://github.com/matplotlib/matplotlib/blob/main/lib/matplotlib/pyplot.py


Questions

Links

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

References

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