Programming for Data Science (with Python)

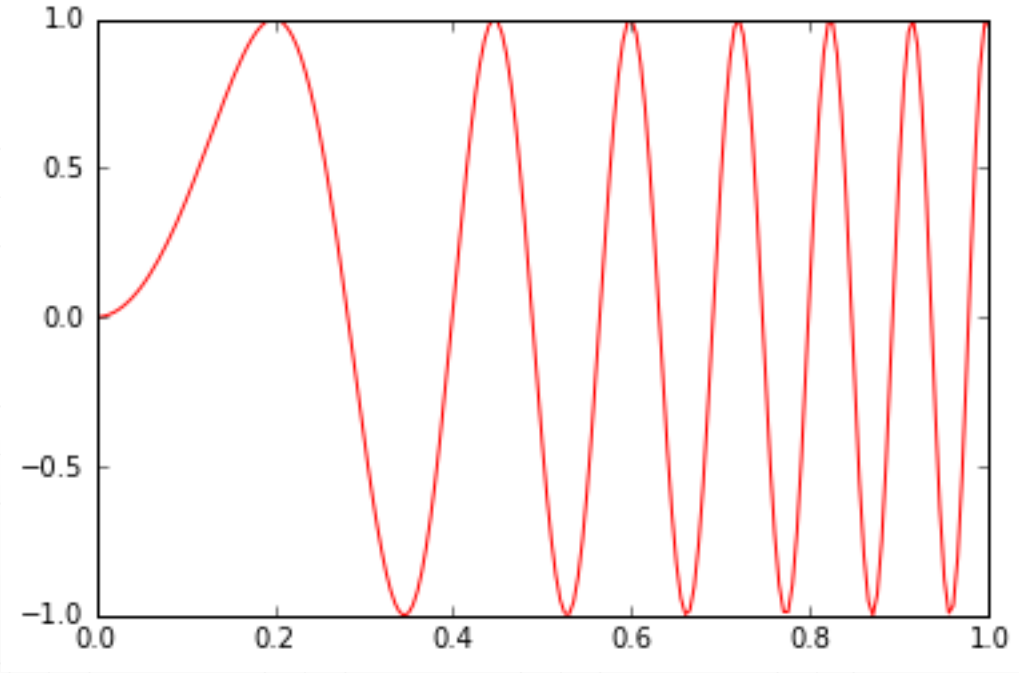
Le Trong Ngoc – http://fit.iuh.edu.vn/giangvien@letrongngoc

Contents

- Introduction to Data Science and Data Analysis
- Introduction to Python for Data Science
- **Data Visualization with Python**
- Statistical Thinking in Python
- Applied Machine Learning in Python

Remider: Line plots

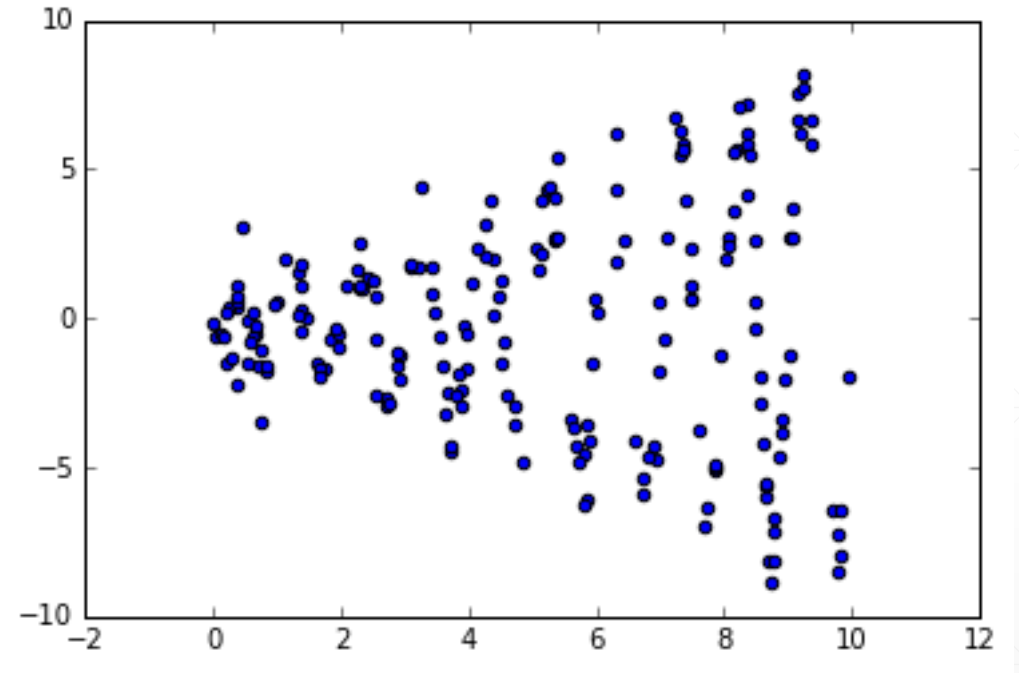
- In [**1**]: import numpy as np
- In [**2**]: import matplotlib.pyplot as plt
- In [**3**]: x = np.linspace(0, 1, 201)
- In [**4**]: y = np.sin((2*np.pi*x)**2)
- In [**5**]: plt.plot(x, y, 'red')



Remider: Scatter plots

- In [**1**]: import numpy as np
- In [**2**]: import matplotlib.pyplot as plt
- In [**3**]: x = 10*np.random.rand(200,1)
- In [**4**]: y = (0.2 + 0.8*x) * np.sin(2*np.pi*x) + \

  ...: np.random.randn(200,1)
- In [**5**]: plt.scatter(x,y)

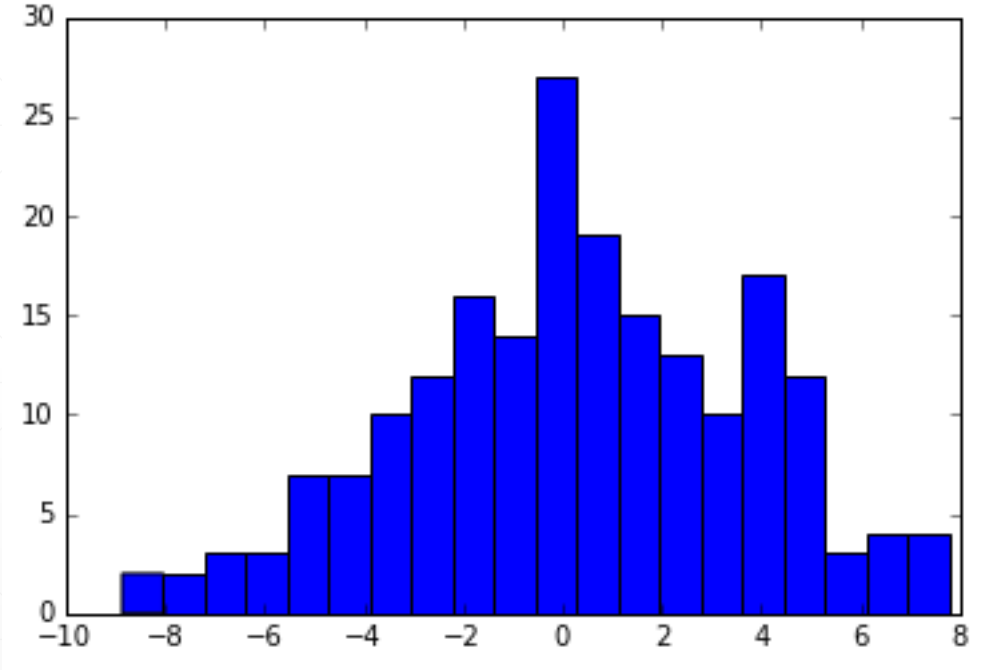


Remider: Histogram

- In [**1**]: import numpy as np
- In [**2**]: import matplotlib.pyplot as plt
- In [**3**]: x = 10*np.random.rand(200,1)
- In [**4**]: y = (0.2 + 0.8*x) * \

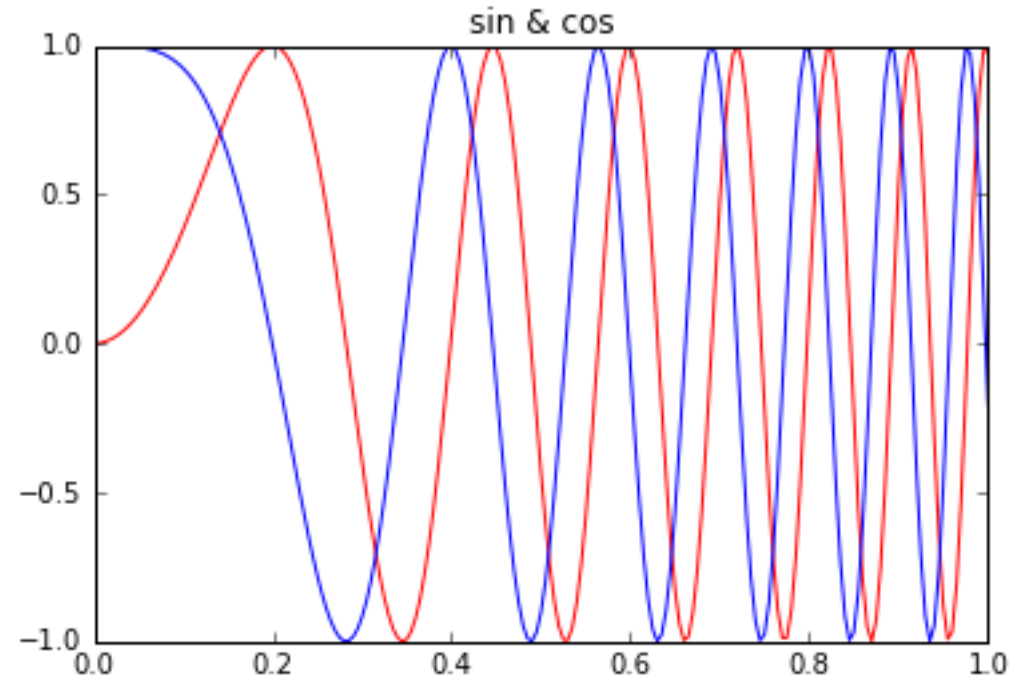
  ...: np.sin(2*np.pi*x) + \

  ...: np.random.randn(200,1)
- In [**5**]: plt.hist(y, bins=20)



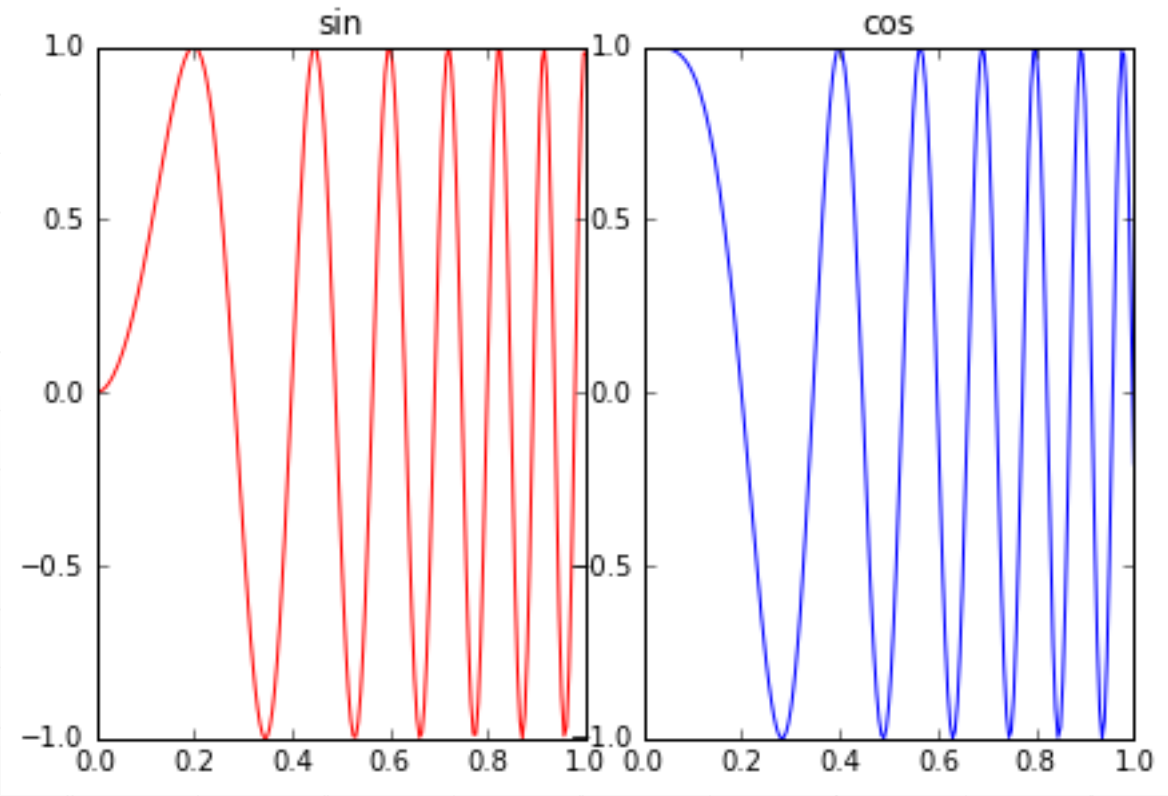
Plotting multiple graphs

- import numpy as np
- import matplotlib.pyplot as plt
- x = np.linspace(0, 1, 201)
- c,s=np.cos((2*np.pi*x)**2),np.sin((2*np.pi*x)**2)
- plt.plot(x,s,'red')
- plt.plot(x,c,'blue')
- plt.title('sin & cos')
- plt.show()



Plotting multiple graphs

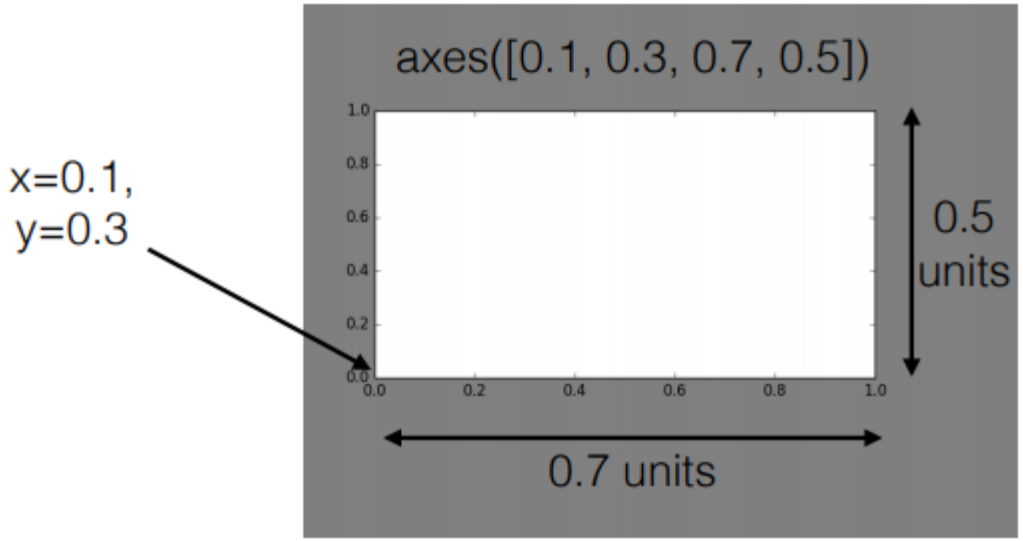
- import numpy as np
- import matplotlib.pyplot as plt
- x = np.linspace(0, 1, 201)
- c,s=np.cos((2*np.pi*x)**2),np.sin((2*np.pi*x)**2)
- **plt.axes([0.05,0.05,0.425,0.9])**
- plt.title('sin')
- plt.plot(x,s,'red')
- **plt.axes([0.525,0.05,0.425,0.9])**
- plt.title('cos')
- plt.plot(x,c,'blue')
- plt.show()



Plotting multiple graphs

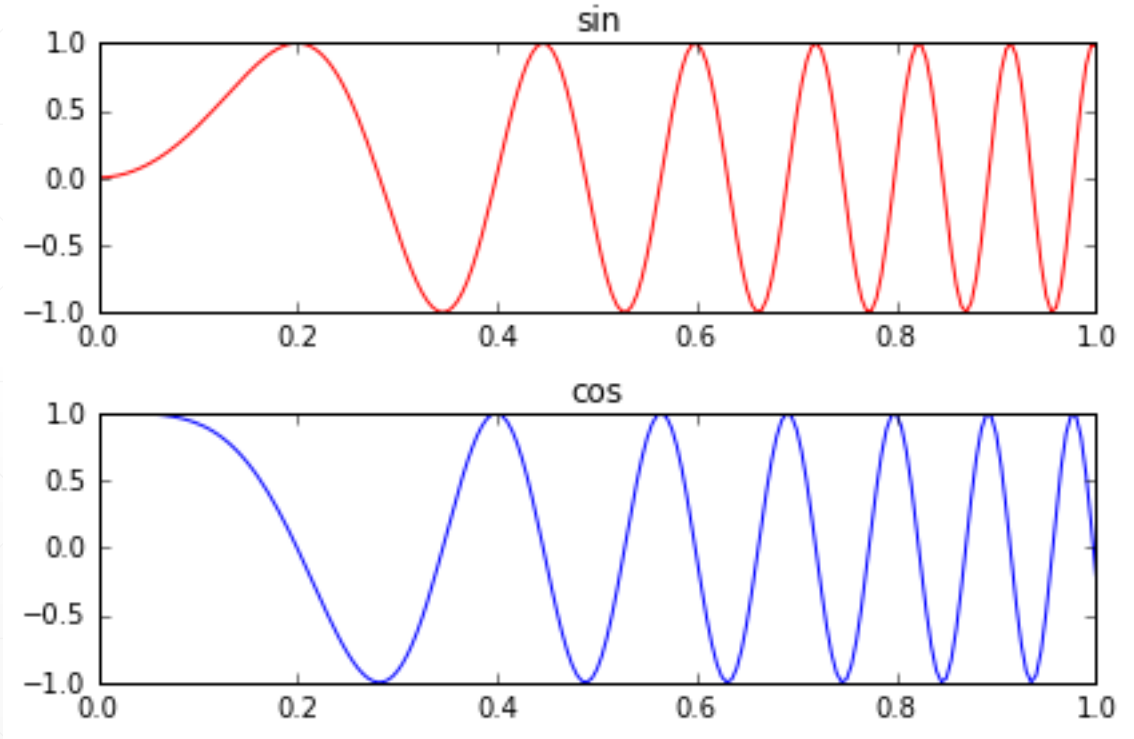
The axes() command

- Syntax: axes([x_lo, y_lo, width, height])
- Units between 0 and 1 (figure dimensions)



Plotting multiple graphs

- import numpy as np
- import matplotlib.pyplot as plt
- x = np.linspace(0, 1, 201)
- c,s=np.cos((2*np.pi*x)**2),np.sin((2*np.pi*x)**2)
- **plt.subplot(2,1,1)**
- plt.title('sin')
- plt.plot(x,s,'red')
- **plt.subplot(2,1,2)**
- plt.title('cos')
- plt.plot(x,c,'blue')
- plt.tight_layout()
- plt.show()



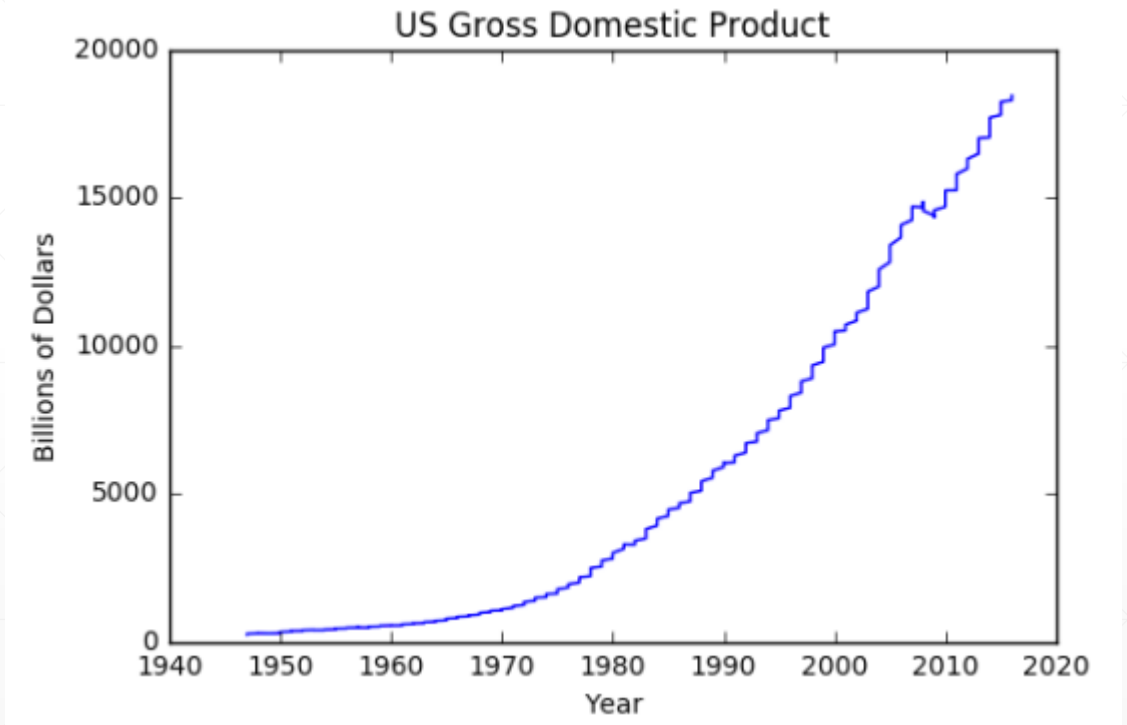
Customizing axes

- Controlling axis extents
  - axis([xmin, xmax, ymin, ymax]) sets axis extents
  - Control over individual axis extents
    - xlim([xmin, xmax])
    - ylim([ymin, ymax])
  - Can use tuples, lists for extents
    - e.g., xlim((-2, 3)) works
    - e.g., xlim([-2, 3]) works also

Customizing axes

- GDP over time

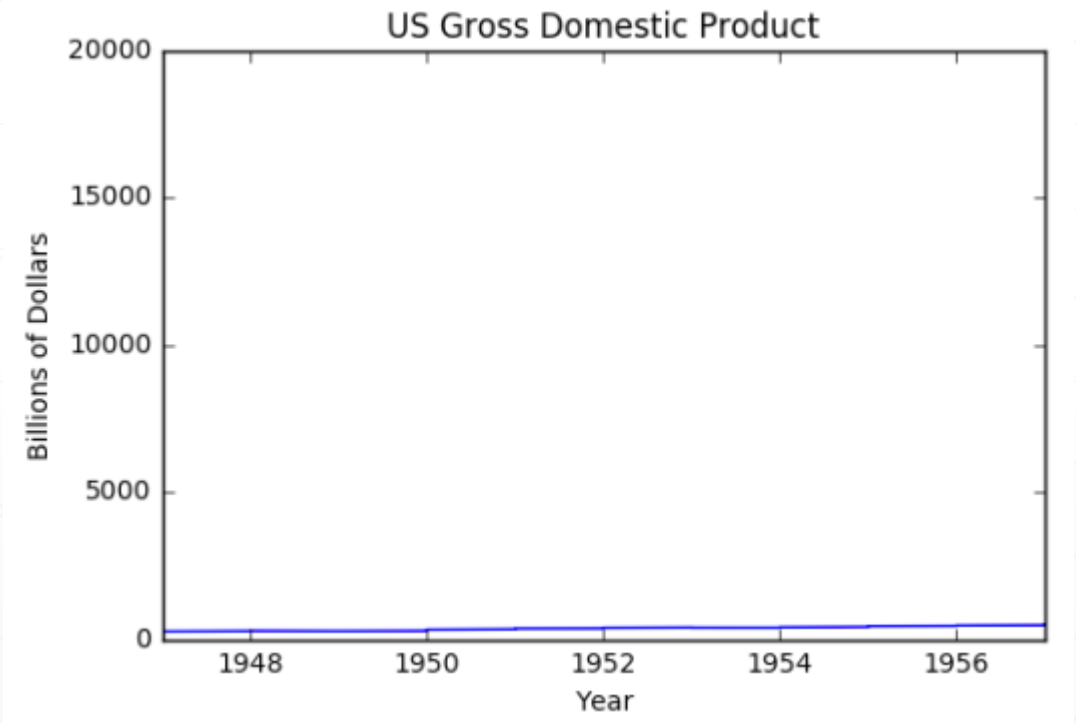
```
In [1]: import matplotlib.pyplot as plt

In [2]: plt.plot(yr, gdp)

In [3]: plt.xlabel('Year')

In [4]: plt.ylabel('Billions of Dollars')

In [5]: plt.title('US Gross Domestic Product')

In [6]: plt.show()
```



Customizing axes

- Using xlim()

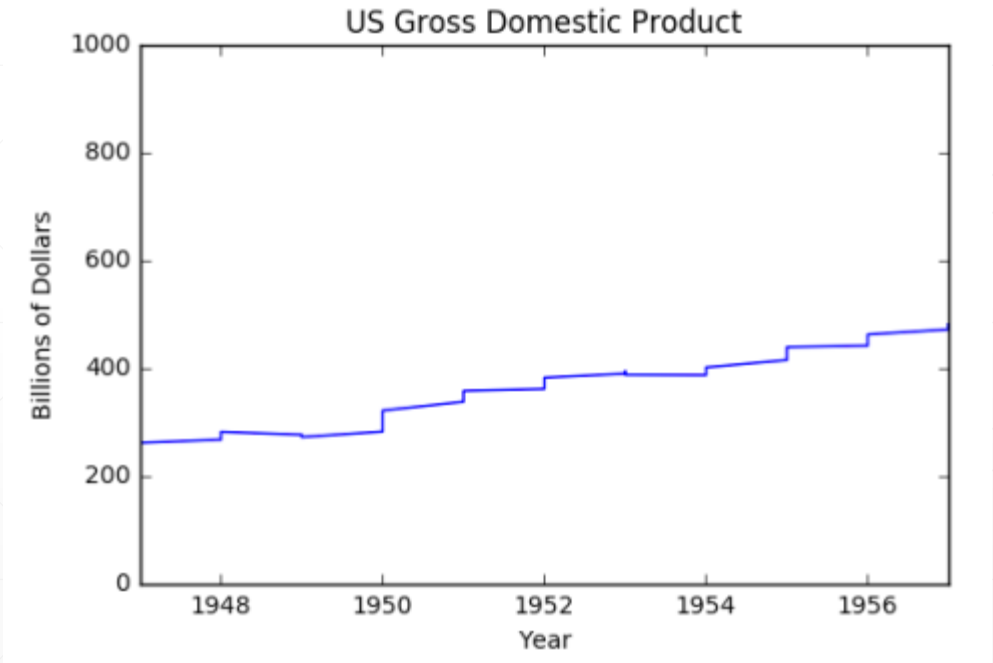
```
In [1]: plt.plot(yr, gdp)

In [2]: plt.xlabel('Year')

In [3]: plt.ylabel('Billions of Dollars')

In [4]: plt.title('US Gross Domestic Product')

In [5]: plt.xlim((1947, 1957))

In [6]: plt.show()
```



Customizing axes

- Using xlim() & ylim()

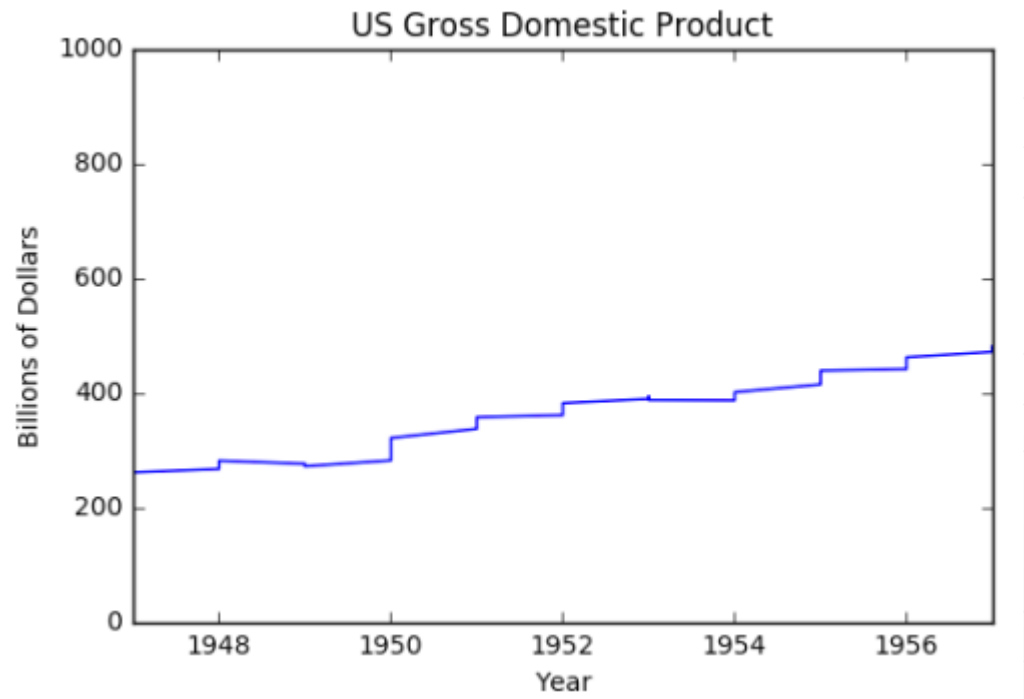
```
In [1]: plt.plot(yr, gdp)

In [2]: plt.xlabel('Year')

In [3]: plt.ylabel('Billions of Dollars')

In [4]: plt.title('US Gross Domestic Product')

In [5]: plt.xlim((1947, 1957))

In [6]: plt.ylim((0, 1000))

In [7]: plt.show()
```



Customizing axes

- Using axis()

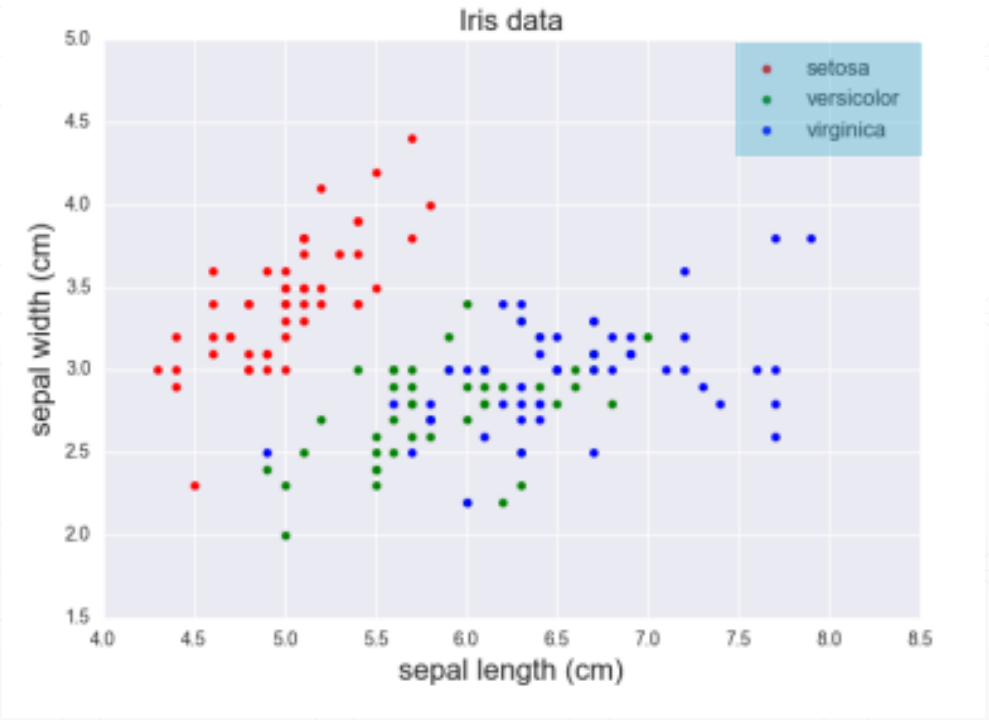
```
In [1]: plt.plot(yr, gdp)

In [2]: plt.xlabel('Year')

In [3]: plt.ylabel('Billions of Dollars')

In [4]: plt.title('US Gross Domestic Product')

In [5]: plt.axis((1947, 1957, 0, 600))

In [6]: plt.show()
```



Legends, annotation, and styles

- Using legend()

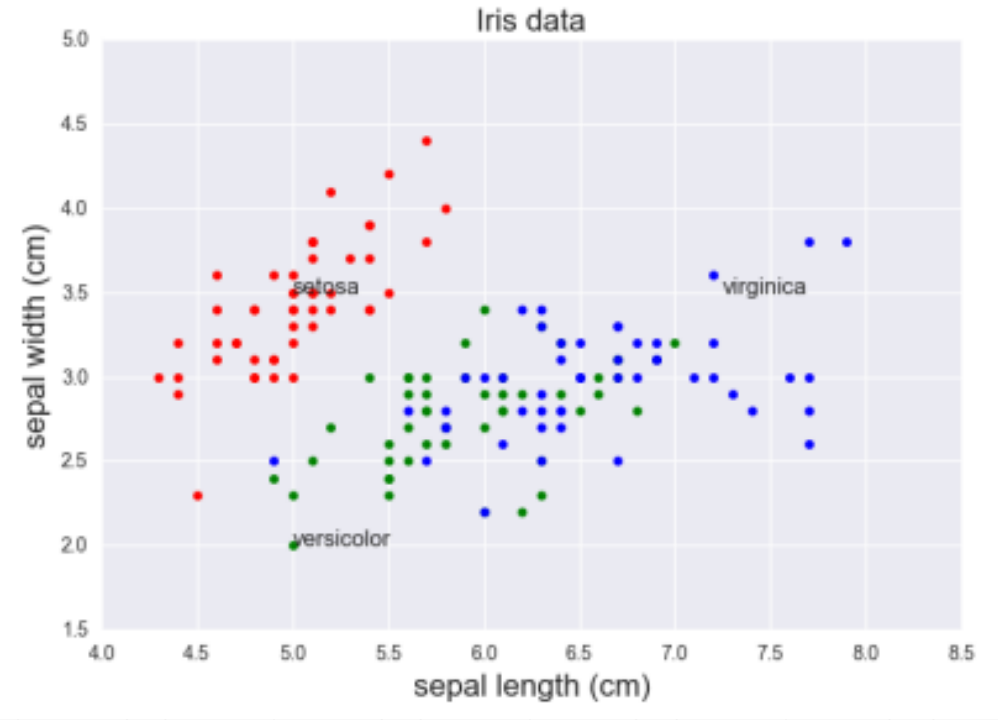
```
In [1]: import matplotlib.pyplot as plt

In [2]: plt.scatter(setosa_len, setosa_wid,
   ...:             marker='o', color='red', label='setosa')

In [3]: plt.scatter(versicolor_len, versicolor_wid,
   ...:             marker='o', color='green', label='versicolor')

In [4]: plt.scatter(virginica_len, virginica_wid,
   ...:             marker='o', color='blue', label='virginica')

In [5]: plt.legend(loc='upper right')

In [6]: plt.title('Iris data')

In [7]: plt.xlabel('sepal length (cm)')

In [8]: plt.ylabel('sepal width (cm)')

In [9]: plt.show()
```



Legends, annotation, and styles

- Using annotate() for text

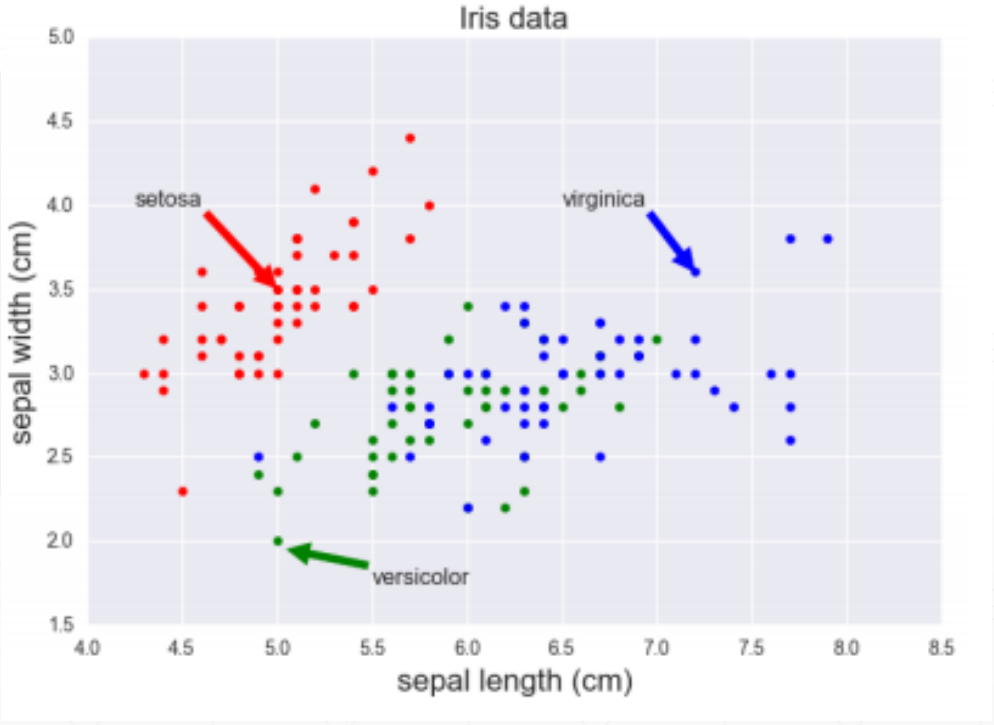
```
In [1]: plt.annotate('setosa', xy=(5.0, 3.5))

In [2]: plt.annotate('virginica', xy=(7.25, 3.5))

In [3]: plt.annotate('versicolor', xy=(5.0, 2.0))

In [4]: plt.show()
```



Legends, annotation, and styles

- Using annotate() for arrows

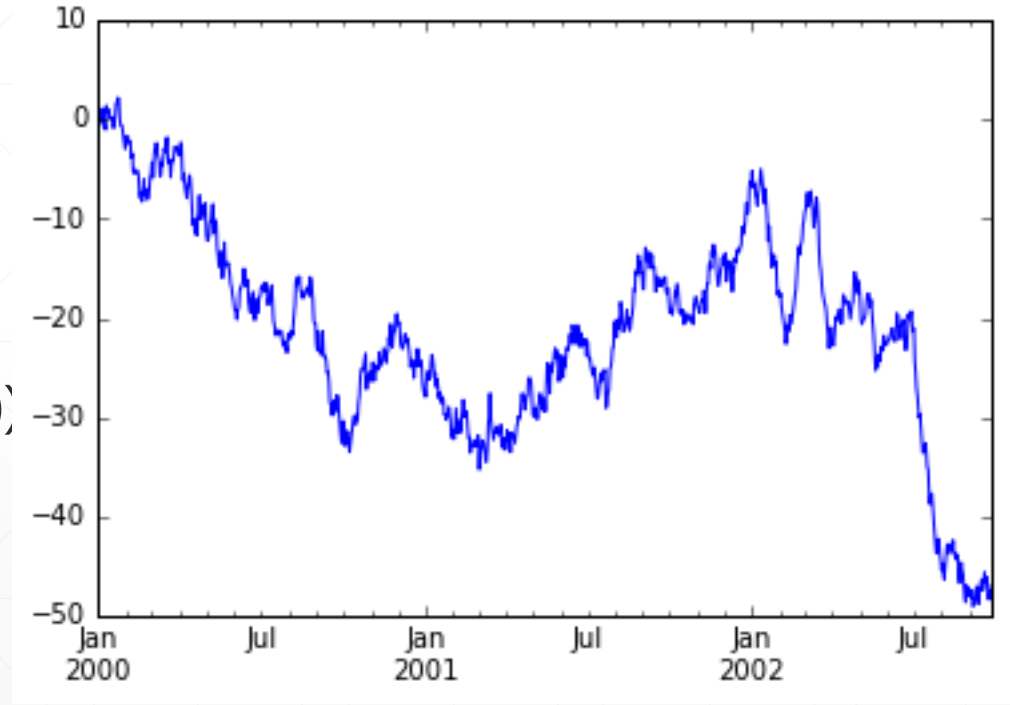
```
In [1]: plt.annotate('setosa', xy=(5.0, 3.5),
   ...:              xytext=(4.25, 4.0), arrowprops={'color':'red'})

In [2]: plt.annotate('virginica', xy=(7.2, 3.6),
   ...:              xytext=(6.5, 4.0), arrowprops={'color':'blue'})

In [3]: plt.annotate('versicolor', xy=(5.05, 1.95),
   ...:              xytext=(5.5, 1.75),
   ...:              arrowprops={'color':'green'})

In [4]: plt.show()
```



Pandas

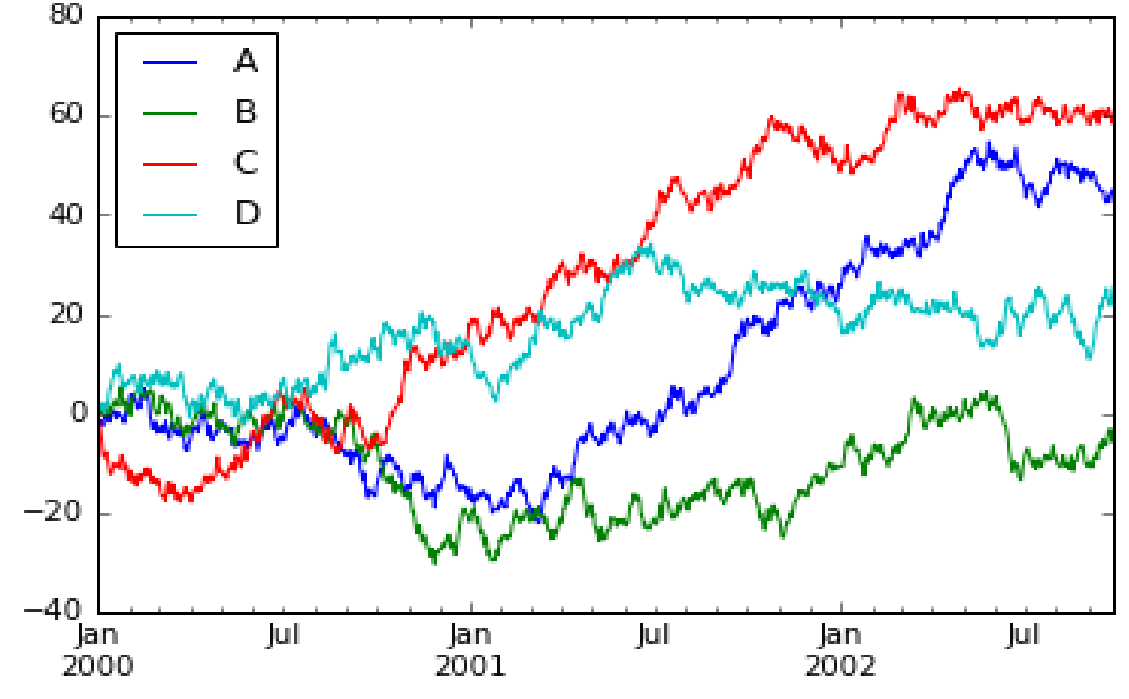
- import matplotlib.pyplot as plt
- import pandas as pd
- import numpy as np
- ts = pd.Series(np.random.randn(1000),

  index=pd.date_range('1/1/2000', periods=1000))
- ts = ts.cumsum()
- ts.plot()



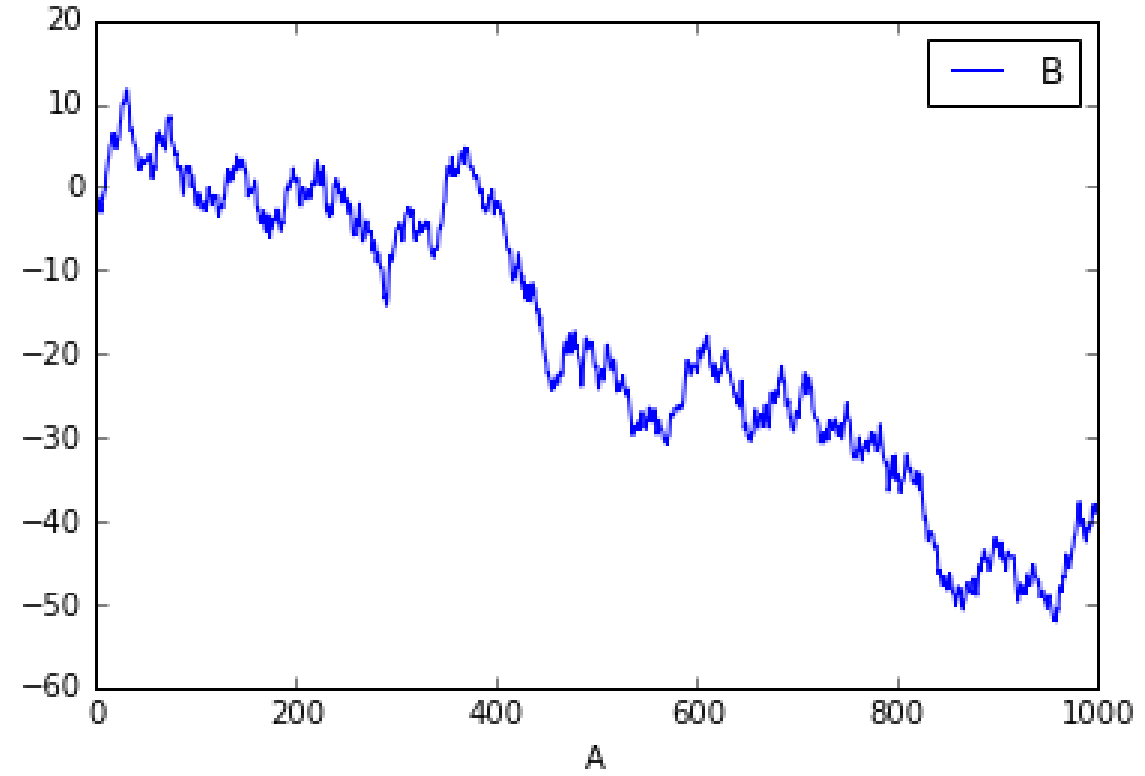
Pandas

- df = pd.DataFrame(np.random.randn(1000, 4), index=ts.index, columns=list('ABCD'))
- df = df.cumsum()
- plt.figure(); df.plot();



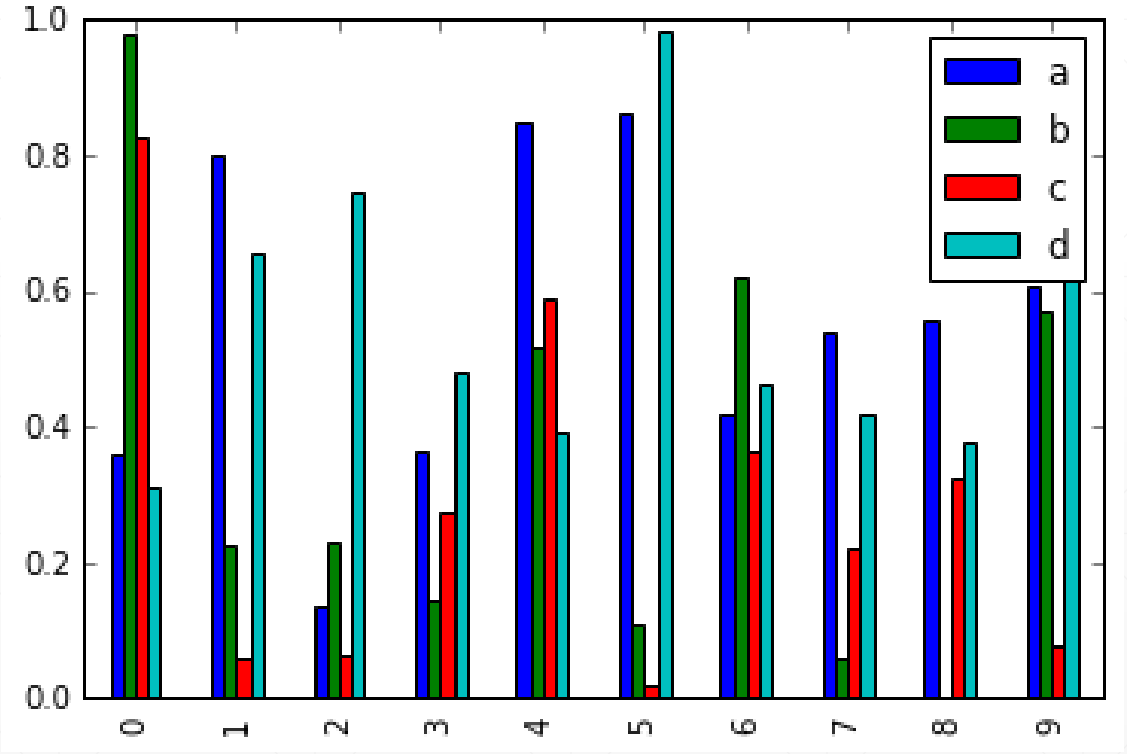
Pandas

- df3 = pd.DataFrame(np.random.randn(1000, 2), columns=['B', 'C']).cumsum()
- df3['A'] = pd.Series(list(range(len(df))))
- df3.plot(x='A', y='B')



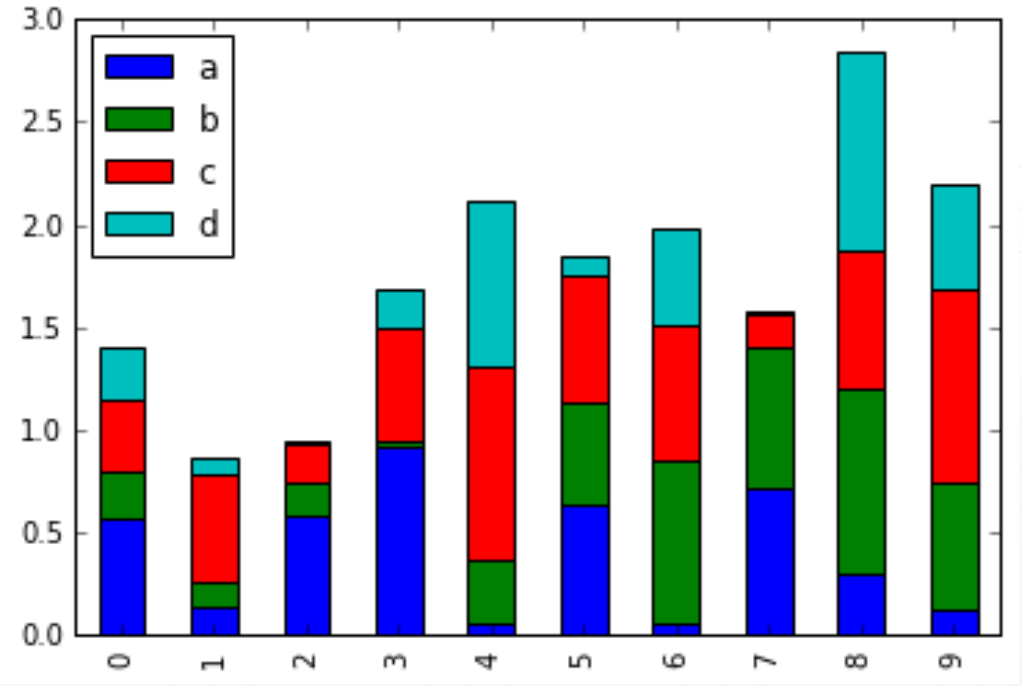
Pandas

- df2 = pd.DataFrame(np.random.rand(10, 4), columns=['a', 'b', 'c', 'd'])
- df2.plot.bar();



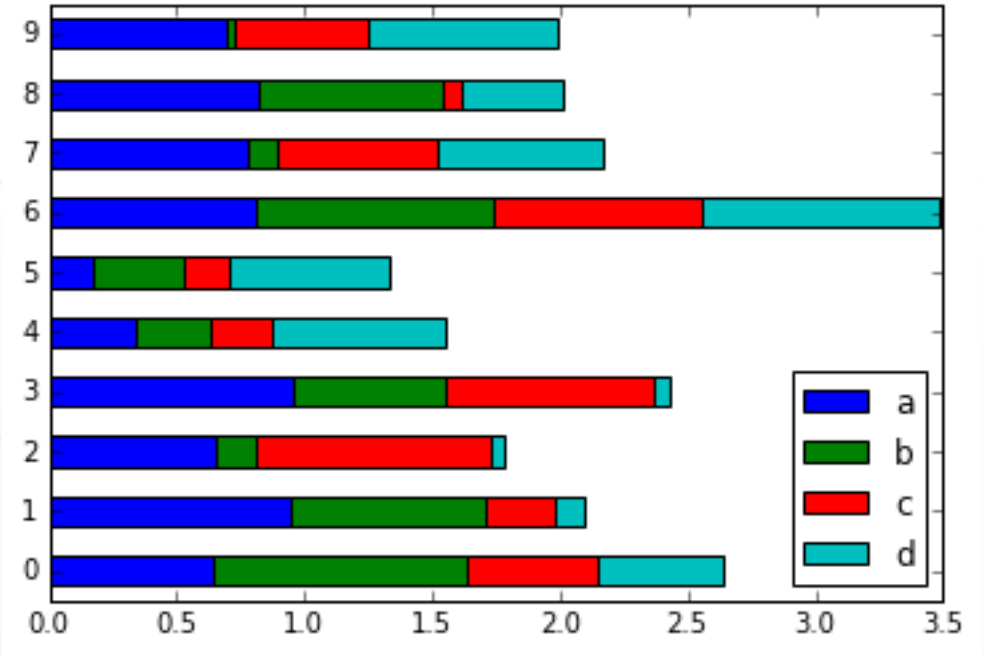
Pandas

- df2 = pd.DataFrame(np.random.rand(10, 4), columns=['a', 'b', 'c', 'd'])
- df2.plot.bar(stacked=True);



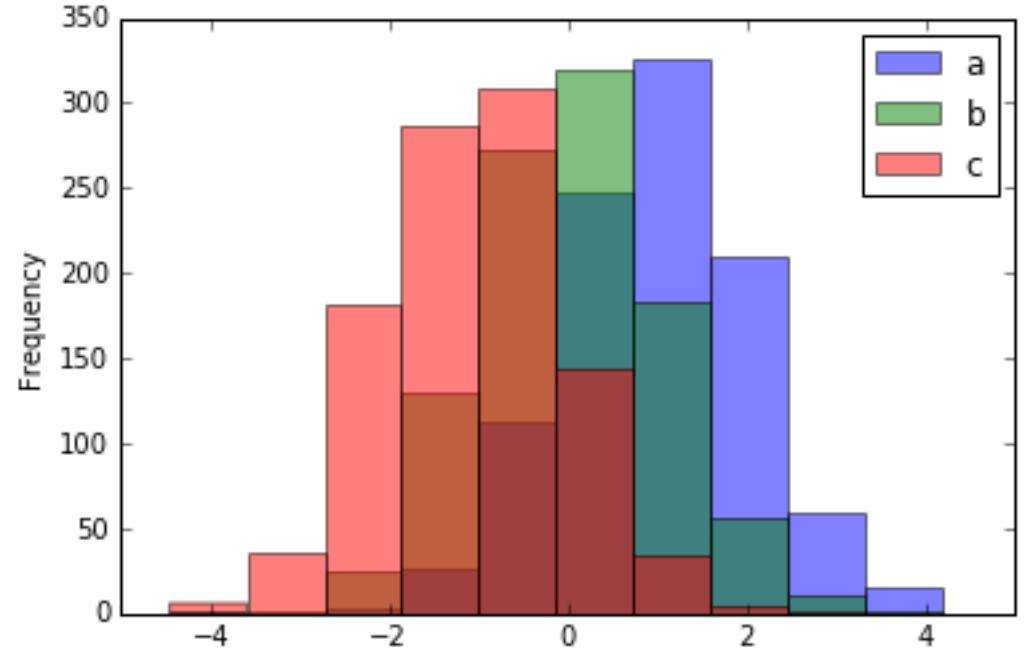
Pandas

- df2 = pd.DataFrame(np.random.rand(10, 4), columns=['a', 'b', 'c', 'd'])
- df2.plot.barh(stacked=True);



Pandas

- df4 = pd.DataFrame({'a': np.random.randn(1000) + 1, 'b': np.random.randn(1000),'c': np.random.randn(1000) - 1}, columns=['a', 'b', 'c'])
- plt.figure();
- df4.plot.hist(alpha=0.5)



Pandas

- df4 = pd.DataFrame({'a': np.random.randn(1000) + 1, 'b': np.random.randn(1000),'c': np.random.randn(1000) - 1}, columns=['a', 'b', 'c'])
- plt.figure();
- df4.plot.hist(stacked=True, bins=20)