Visualization

Topic 10

Dr. Douglas Stebila

Department of Computing and Software
McMaster University

Spring/Summer 2020

Table of Contents

Table of Contents

Options for plotting in Python

- There are many different libraries for plotting in Python:
  - pylab
  - pyplot (part of matplotlib)
    - Basic syntax
    - Object-oriented syntax
  - pandas (builds on matplotlib)
- We'll use the basic pyplot interface in this course, but others can be more powerful

Installing matplotlib

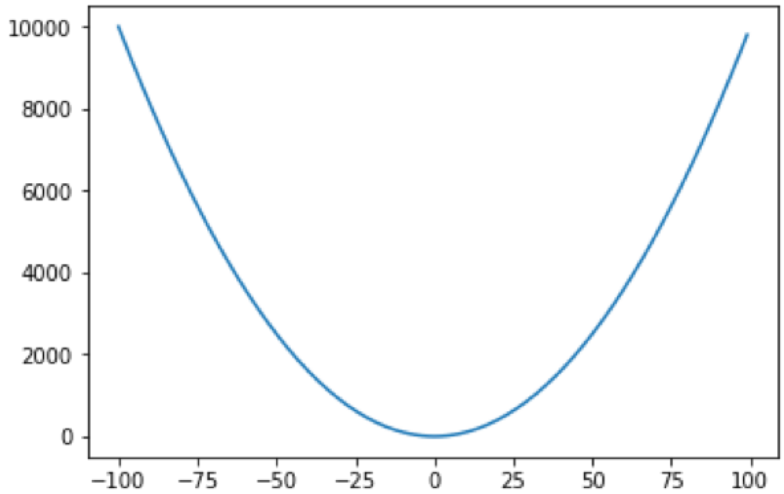
- matplotlib is an optional Python set of packages for plotting
- Pre-installed in Anaconda
- I've installed on jhub2
- To install on your local computer for use in IDLE/Jupyter, try one of the following:
  - pip3 install matplotlib
  - python3 -m pip install matplotlib
  - sudo pip3 install matplotlib

Getting plots to show up in Jupyter

- You can get plots from matplotlib to show up directly in a Jupyter notebook using a Jupyter magic command:
- %matplotlib inline
  - Displays non-interactive images in the notebook
- %matplotlib nbagg
  - Inserts interactive plots into the notebook that can be dragged etc.

A line plot from a list of points

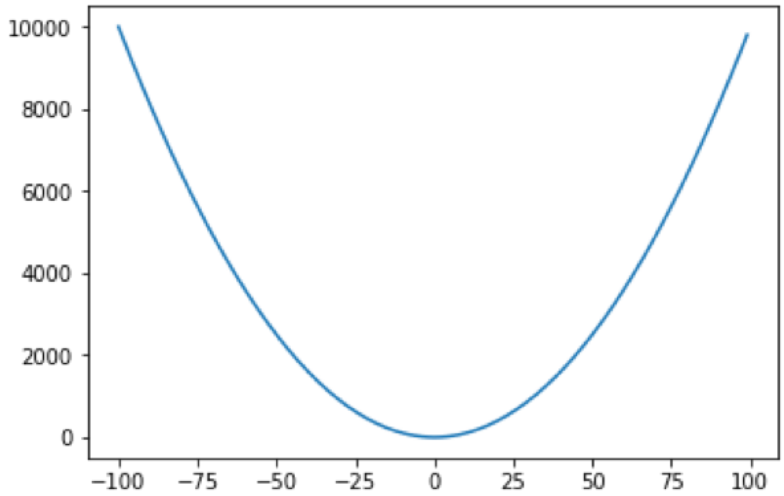
```
%matplotlib inline
import matplotlib.pyplot as pyplot
xvalues = [1,2,3]
yvalues = [4,5,6]
pyplot.plot(xvalues, yvalues)
pyplot.show()
```



A line plot from a function

- To create a line plot where the y values are computed using a function, we have to create a list of the y values by evaluating the function on the x values

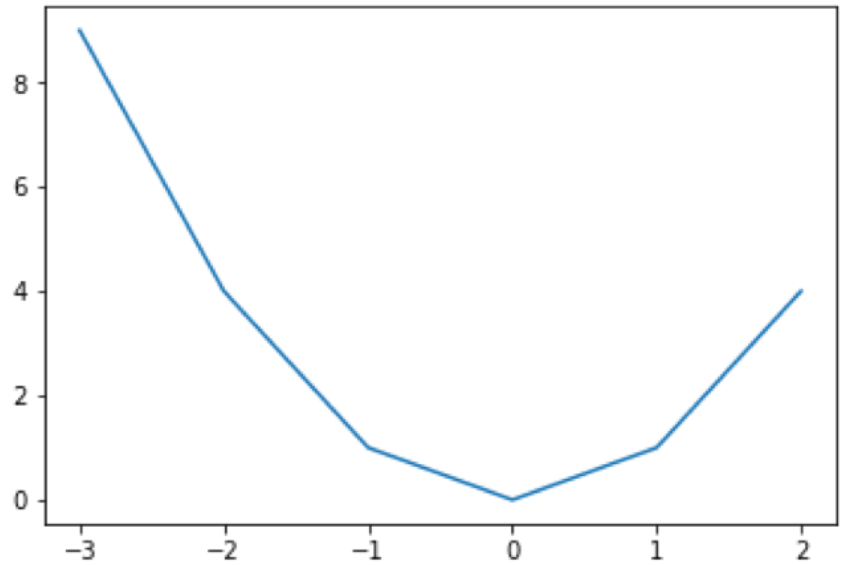
```
xvalues = range(-100, 100)
yvalues = [x**2 for x in xvalues]
pyplot.plot(xvalues, yvalues)
pyplot.show()
```



A line plot from a function

- If we are trying to create a smooth "continuous"-looking plot, we have to provide sufficiently many data points

```
xvalues = range(-3, 3)
yvalues = [x**2 for x in xvalues]
pyplot.plot(xvalues, yvalues)
pyplot.show()
```



A line plot from a function

- If we are trying to create a smooth "continuous"-looking plot, we have to provide sufficiently many data points.

```
xvalues = range(-3, 3, 0.01)
yvalues = [x**2 for x in xvalues]
pyplot.plot(xvalues, yvalues)
pyplot.show()
```

range can only step by integers

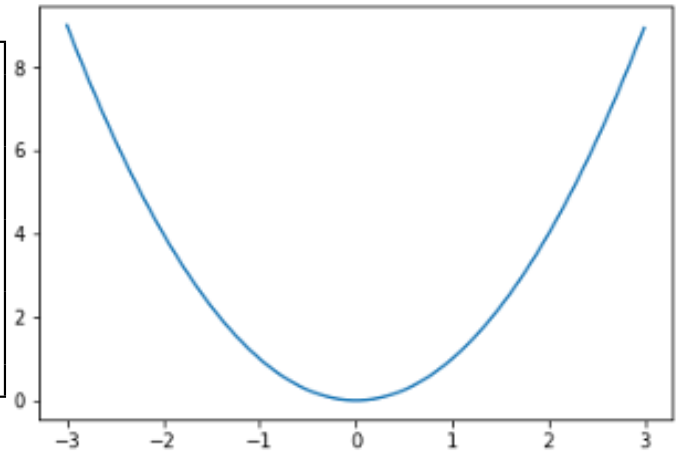
```
TypeError
Traceback (most recent call last)
<ipython-input-51-f2d2ede20032> in <module>()
----> 1 xvalues = range(-3,3,0.01)
      2 yvalues = [x**2 for x in xvalues]
      3 pyplot.plot(xvalues, yvalues)

TypeError: range() integer step argument expected, got float.
```

A line plot from a function

- If we are trying to create a smooth "continuous"-looking plot, we have to provide sufficiently many data points
- Use our custom frange function to create a range stepped by floating points

```
xvalues = frange(-3, 3, 0.01)
yvalues = [x**2 for x in xvalues]
pyplot.plot(xvalues, yvalues)
pyplot.show()
```



A floating-point friendly range function

```
def frange(start, stop, step, places=5):
    x = start
    L = []
    while round(x, places) < round(stop, places):
        L.append(round(x, places))
        x += step
    return L
```

```
print(frange(0, 1, 0.1))
```

```
[0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9]
```

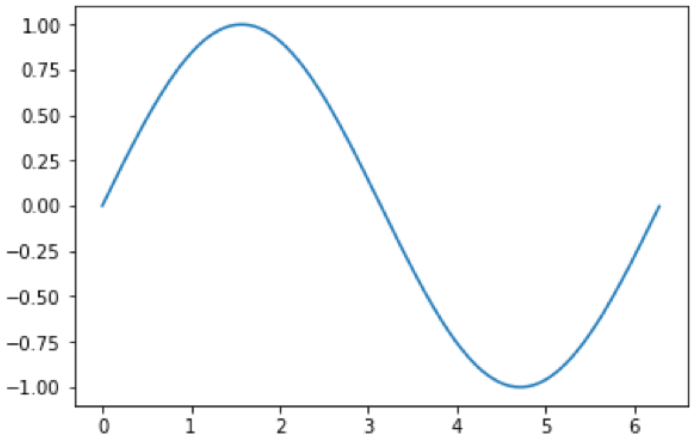
```
print(frange(0, 1, 0.1, places=20))
```

```
[0.0, 0.1, 0.2, 0.30000000000000004, 0.4, 0.5, 0.6, 0.7,
0.7999999999999999, 0.8999999999999999, 0.9999999999999999]
```

A line plot from a function

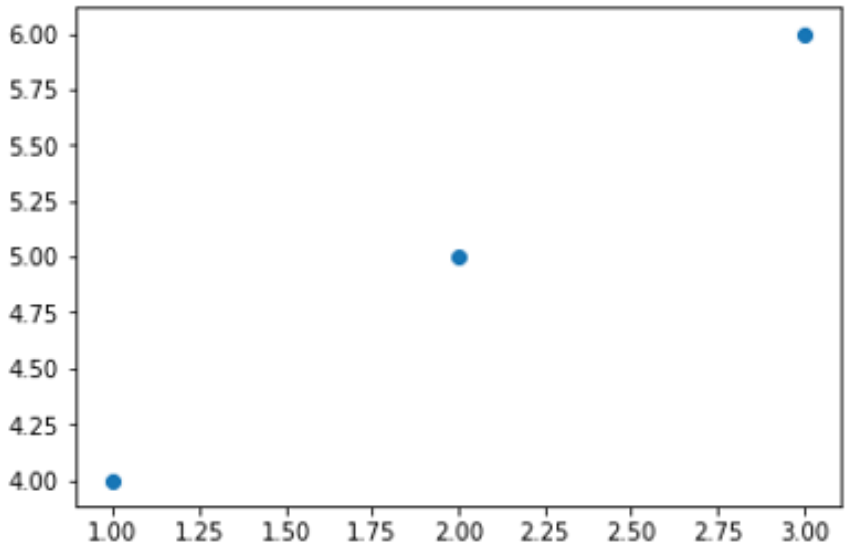
- If we are trying to create a smooth "continuous"-looking plot, we have to provide sufficiently many data points
- Use our custom frange function to create a range stepped by floating points.

```
import math
xvalues = frange(0,2 * math.pi,0.01)
yvalues = [math.sin(x) for x in xvalues]
pyplot.plot(xvalues, yvalues)
pyplot.show()
```



Other types of plots: dot (scatter)

```
xvalues = [1, 2, 3]
yvalues = [4, 5, 6]
pyplot.scatter(xvalues, yvalues)
pyplot.show()
```



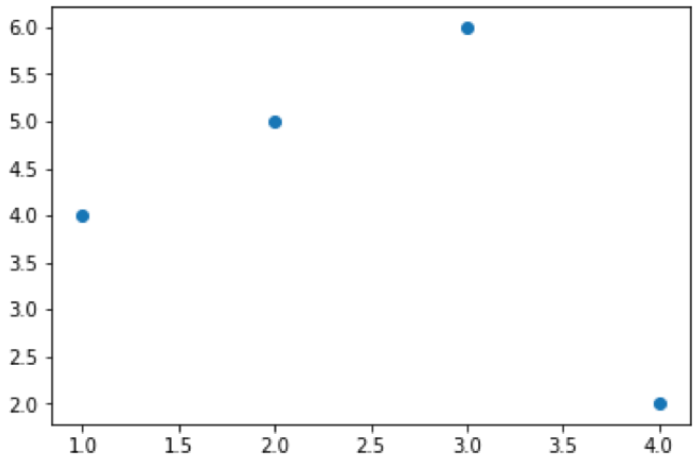
Plotting data from a CSV file

- Basic approach:
  - Read data from CSV file using approach from previous lectures
  - Construct xvalues and yvalues lists by iterating through the rows of the CSV file
- Various libraries can be used to do this more automatically:
  - numpy.loadtxt
  - pandas
- Can also load data from a network file using approach from previous lectures

Plotting data from a CSV file

```
import csv
with open('sample.csv', 'r') as fh:
    rows = csv.reader(fh)
    xvalues = []
    yvalues = []
    for row in rows:
        xvalues.append(row[0])
        yvalues.append(row[1])
    pyplot.scatter(xvalues, yvalues)
    pyplot.show()
```

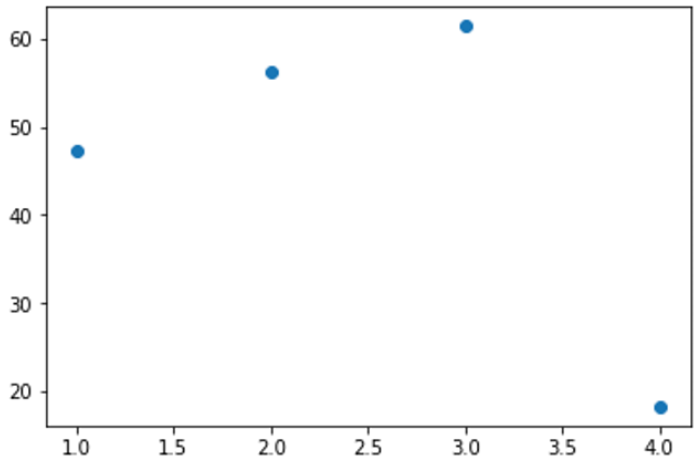
1,4
2,5
3,6
4,2



Plotting data from a CSV file with a header row

```
import csv
with open('sample2.csv', 'r') as fh:
    rows = csv.DictReader(fh)
    xvalues = []
    yvalues = []
    for row in rows:
        xvalues.append(row['Day'])
        yvalues.append(row['Close'])
    pyplot.scatter(xvalues, yvalues)
    pyplot.show()
```

Day,Open,Close
1,13.3,47.3
2,47.3,56.2
3,56.2,61.4
4,61.4,18.3

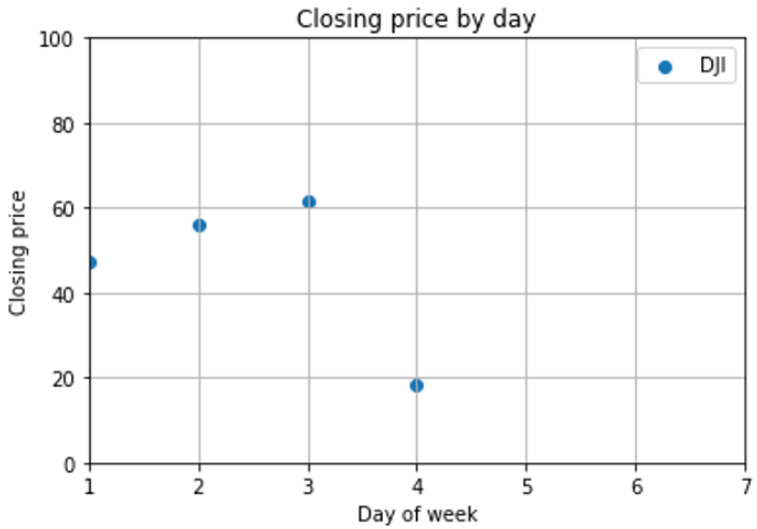


Customizing plots

- Title
  - pyplot.title('Closing price by day')
- x-axis and y-axis titles
  - pyplot.xlabel('Day of week')
  - pyplot.ylabel('Closing price')
- x-axis and y-axis lower/upper bounds
  - pyplot.axis([1, 7, 0, 100])
    # xlower, xupper, ylower, yupper
- Legend
  - pyplot.legend(['DJI'])
    # list of labels, one label for each set of y data
- Gridlines
  - pyplot.grid(True)
- Colors, line styles, marker styles, . . .

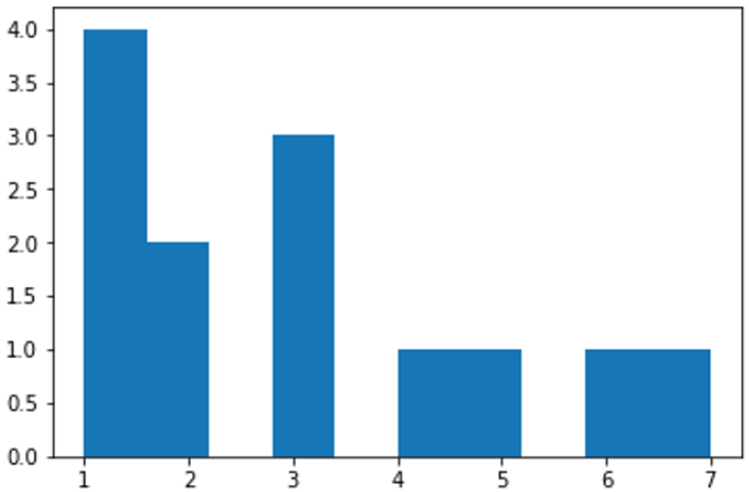
Customizing plots

```
import csv
with open('sample2.csv', 'r') as fh:
    rows = csv.DictReader(fh)
    xvalues = []
    yvalues = []
    for row in rows:
        xvalues.append(row['Day'])
        yvalues.append(row['Close'])
    pyplot.title('Closing price by day')
    pyplot.xlabel('Day of week')
    pyplot.ylabel('Closing price')
    pyplot.axis([1, 7, 0, 100])
    pyplot.grid(True)
    pyplot.scatter(xvalues, yvalues)
    pyplot.legend(['DJI'])
    pyplot.show()
```



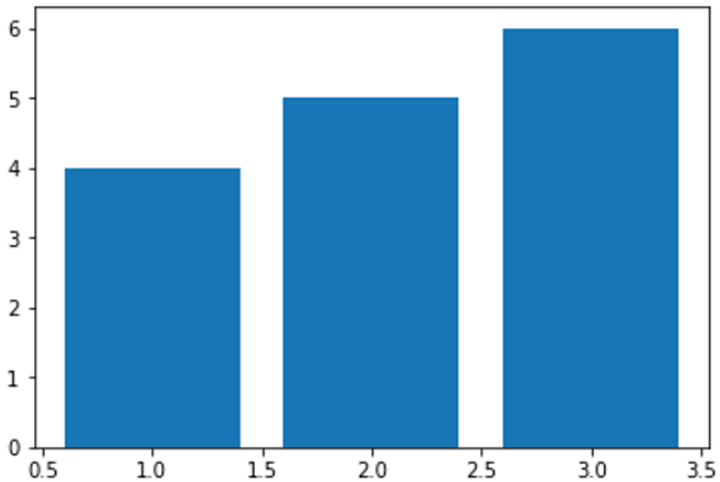
Different types of plots – Histograms

```
data = [1,1,3,5,6,7,4,3,2,1,2,3,1]
pyplot.hist(data)
pyplot.show()
```



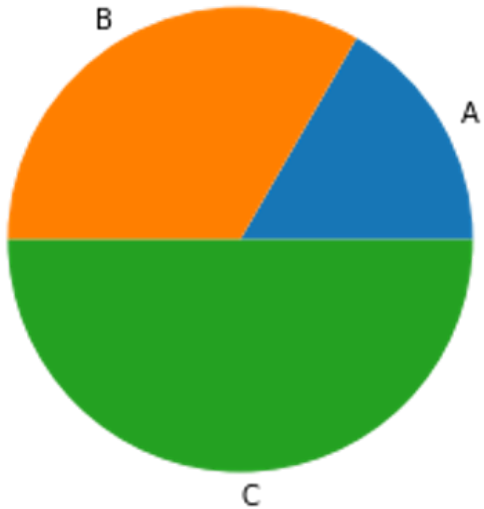
Different types of plots – Bar charts

```
xvalues = [1,2,3]
yvalues = [4,5,6]
pyplot.bar(xvalues, yvalues)
# use barh for horizontal bars
pyplot.show()
```



Different types of plots – Pie charts

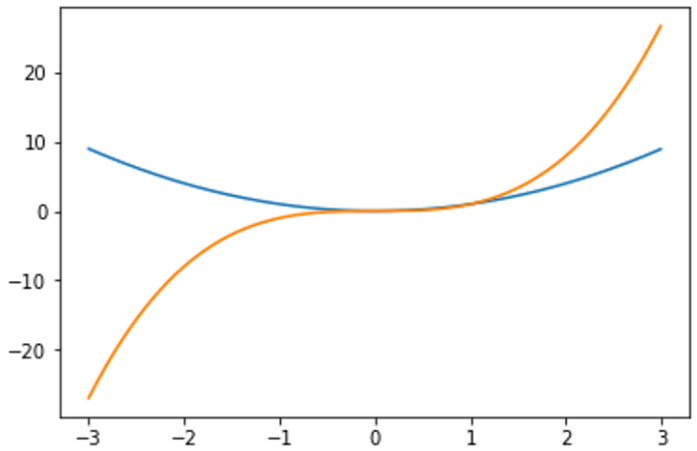
```
data = [1, 2, 3]
categories = ['A', 'B', 'C']
pyplot.subplot(aspect=1) # looks weird without this
pyplot.pie(data, labels=categories)
pyplot.show()
```



Plotting multiple data sets

- Some plot types can plot multiple data sets against each other
  - Line graphs, scatter plots, bar graphs, ...

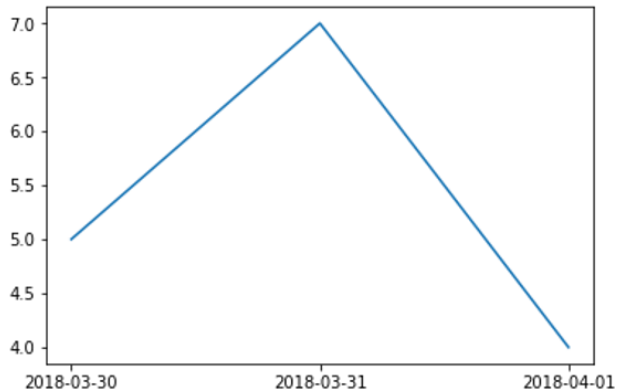
```
xvalues = frange(-3,3,0.01)
yvalues = [x**2 for x in xvalues]
moreyvalues = [x**3 for x in xvalues]
pyplot.plot(xvalues, yvalues, '', xvalues, moreyvalues)
pyplot.show()
```



Plotting datetime data

```
import datetime
day1 = datetime.date(2018, 3, 30)
day2 = datetime.date(2018, 3, 31)
day3 = datetime.date(2018, 4, 1)
xvalues = [day1, day2, day3]
yvalues = [5, 7, 4]
pyplot.xticks(xvalues, xvalues)
pyplot.plot(xvalues, yvalues)
```

Can just use datetime objects as x or y values
Can be tricky to get x-axis labels spaced correctly if you have lots of data



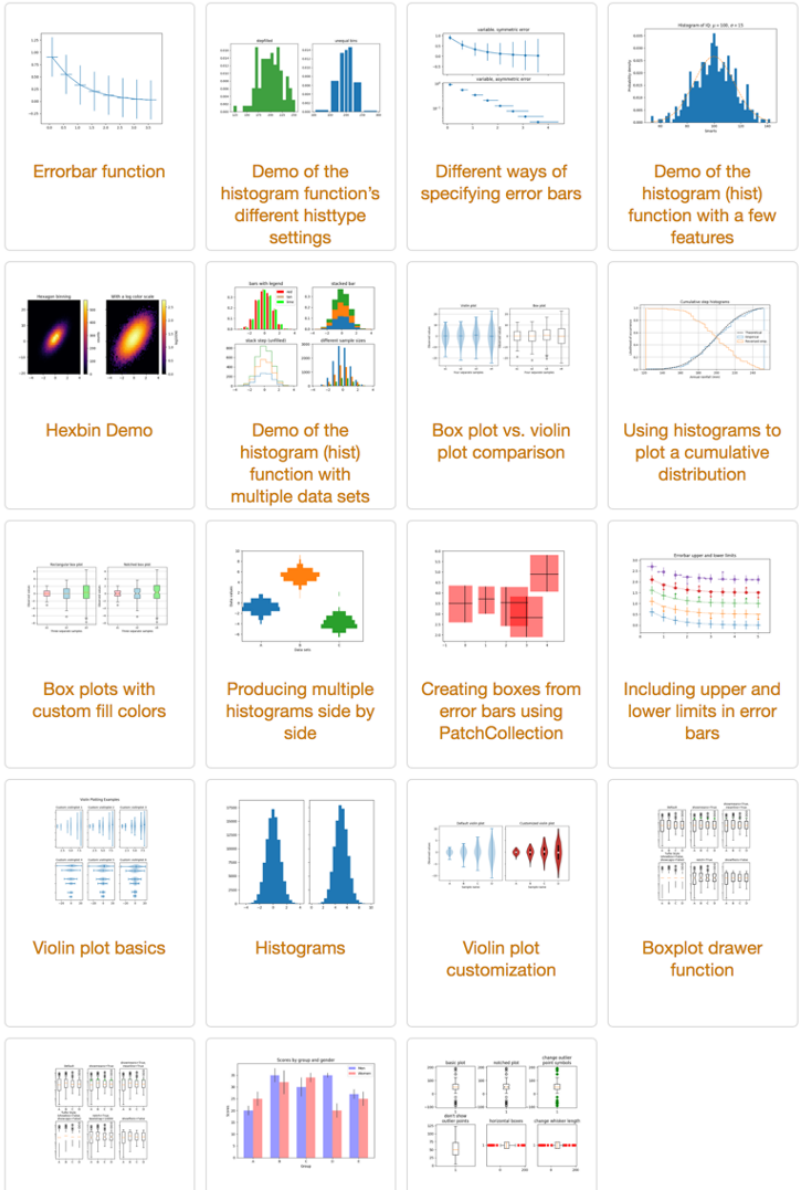
Saving the plot to a file

```
data = [1, 2, 3]
categories = ['A', 'B', 'C']
pyplot.subplot(aspect=1) # looks weird without this
pyplot.pie(data, labels=categories)

pyplot.savefig('mypiechart.png') # or .pdf or .svg or others
pyplot.savefig('mypiechart.png', dpi=200) # higher resolution
```

Many more plots

- Polar
- 3D
- Subplots
- Adding text and arrows
- Error bars
- Customization



https://matplotlib.org/gallery/index.html