# Introduction to Data Visualization with Python

Most of the materials are taken from here

Nastaran Okati

# Outline

- Introduction
- Line Plots
- Other 2D plots
- Plot styles
- 3D plots

# 1.
# Introduction

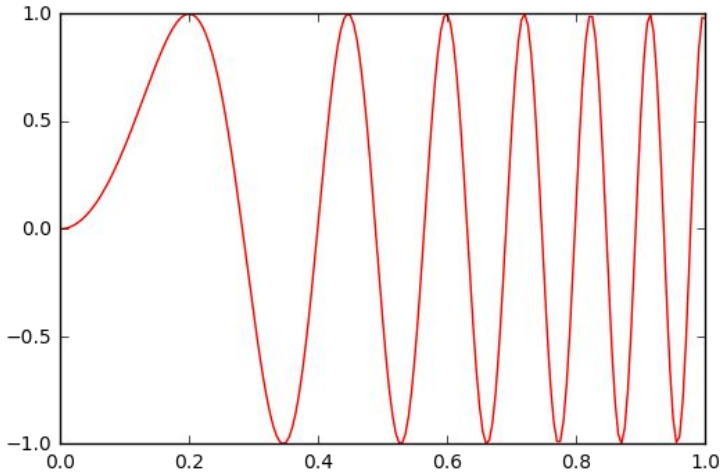
# What is Matplotlib?


- Easy to get started
- Great control of every element in a figure, including figure size and DPI.
- High-quality output in many formats, including PNG, PDF, SVG, EPS, and PGF.
- GUI for interactively exploring figures

# 2.
# Line Plots

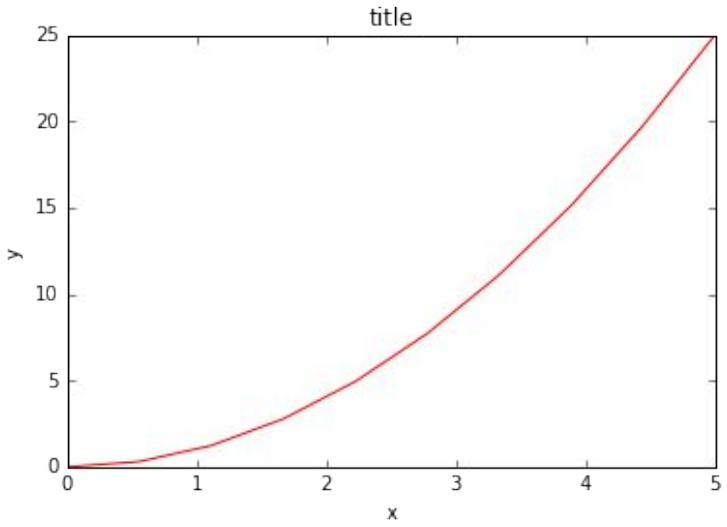
# plt.plot

```
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')


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



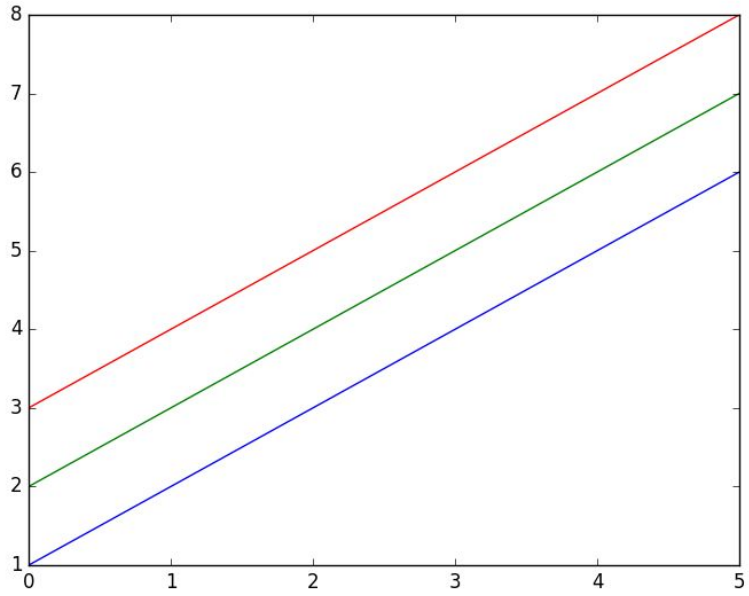
# Titles and Labels

```
In [1]: import numpy as np
In [2]: import matplotlib.pyplot as plt

…

In [3]: plt.title('title')

In [4]: plt.xlabel('x')

In [5]: plt.ylabel('y')


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



# Graphs on Common Axes

```
In [1]: import numpy as np
In [2]: import matplotlib.pyplot as plt

…

In [3]: plt.plot(x,y)

In [4]: plt.plot(x,z)

In [5]: plt.plot(x,w)


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



# Figure Size, Aspect Ration and DPI

Matplotlib allows the aspect ratio, DPI and figure size to be specified when the Figure object is created, using the figsize and dpi keyword arguments.

figsize is a tuple of the width and height of the figure in inches, and dpi is the dots-per-inch (pixel per inch).

To create an 800x400 pixel, 100 dots-per-inch figure, we can do:

```
fig = plt.figure(figsize=(8,4), dpi=100)
```

# Saving figures

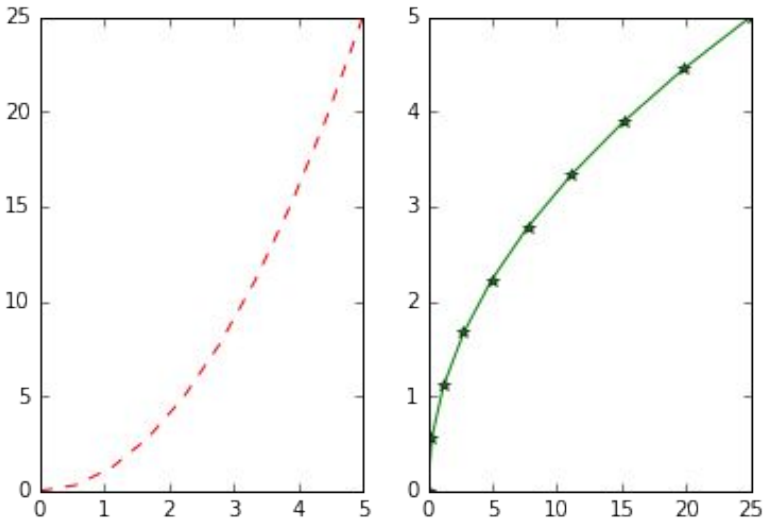
```
fig.savefig("filename.png")
```

Matplotlib can generate high-quality output in a number formats, including PNG, JPG, EPS, SVG, PGF and PDF.

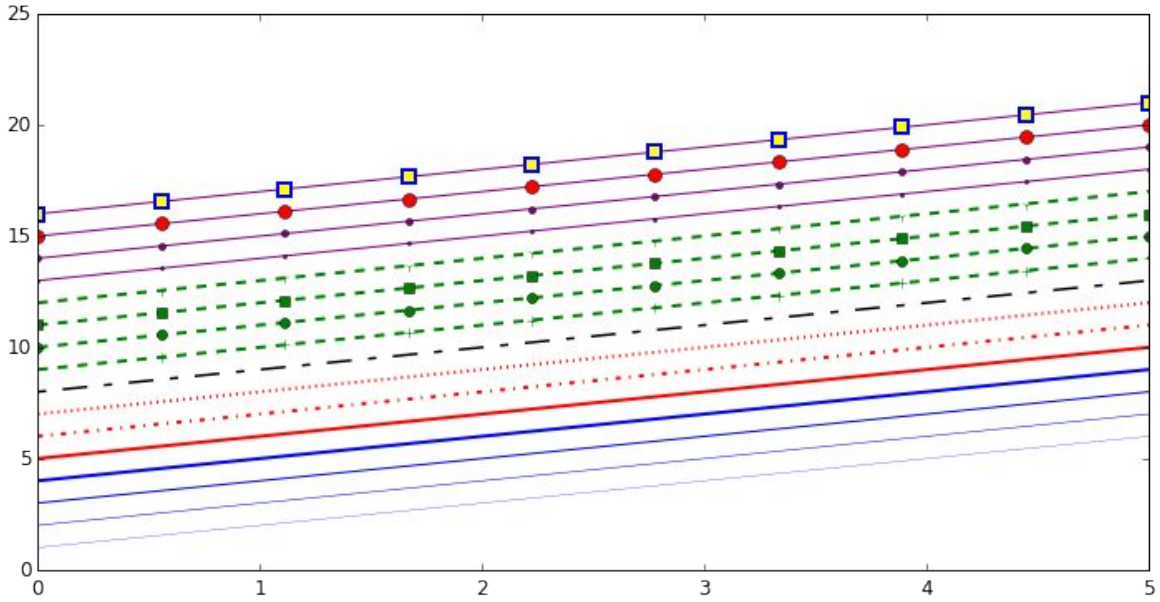
For scientific papers, use PDF whenever possible. (LaTeX documents compiled with pdflatex can include PDFs using the includegraphics command).

# Subplots and Color/Symbol Selection

```
In [1]: import numpy as np
In [2]: import matplotlib.pyplot as plt

…

In [3]: plt.subplot(1,2,1)

In [4]: plt.plot(x,y,'r--')

In [5]: plt.subplot(1,2,2)

In [6]: plt.plot(y,x,'g*-')
```

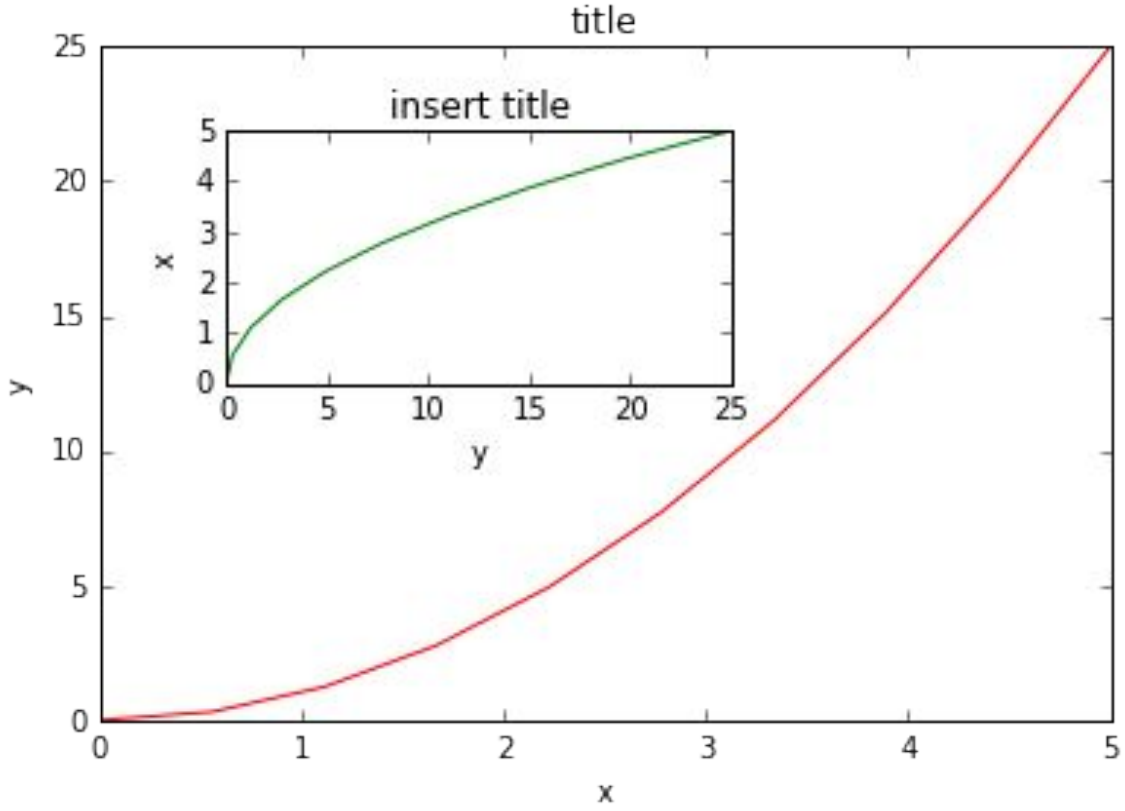


# Line and Marker Styles



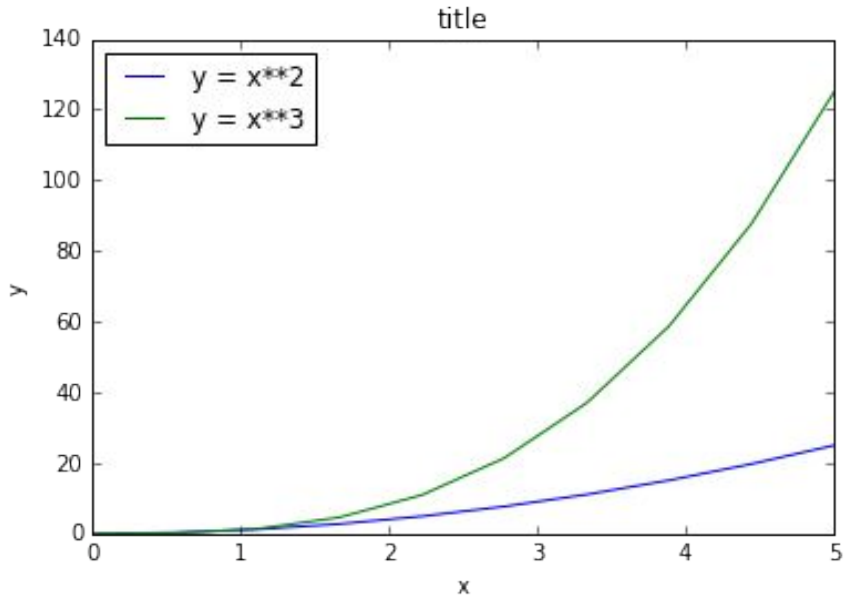
# add_axes

Exercise: Generate the figure using fig.add_axes()



# Legends

```
In [1]: import numpy as np
In [2]: import matplotlib.pyplot as plt

…

In [3]: plt.plot(x,x**2,label="y=x**2")

In [4]: plt.plot(x,x**3,label="y=x**3")

In [5]: plt.legend(loc=2)
```



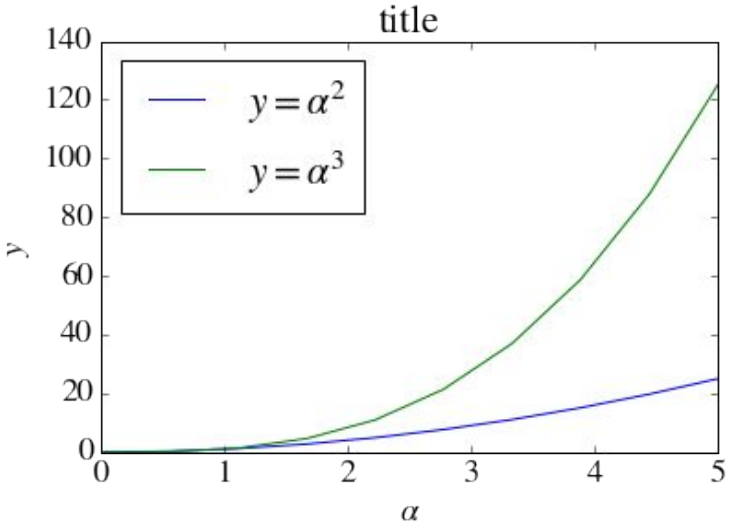
# Legend Locations

| string | code | string | code | string | code |
|---|---|---|---|---|---|
| 'upper left' | 2 | 'upper center' | 9 | 'upper right' | 1 |
| 'center left' | 6 | 'center' | 10 | 'center right' | 7 |
| 'lower left' | 3 | 'lower center' | 8 | 'lower right' | 4 |
| 'best' | 0 | | | 'right' | 5 |

# Formatting text: LaTeX, fontsize, font family

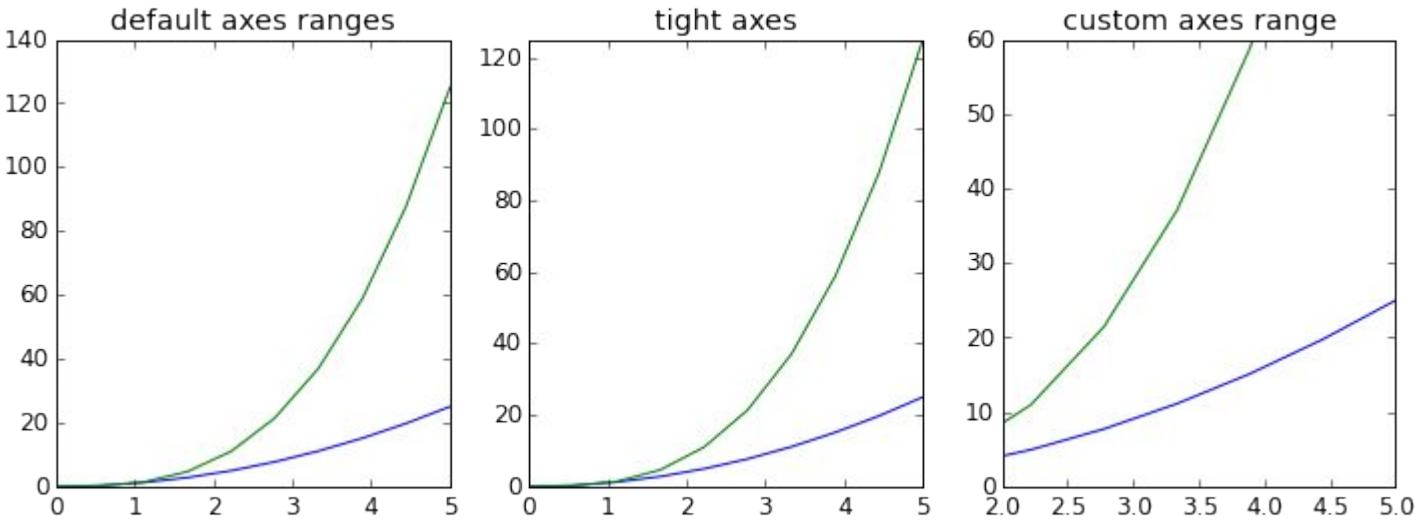
Matplotlib has great support for LaTeX. All we need to do is to use dollar signs encapsulate LaTeX in any text (legend, title, label, etc.). For example, "$y=x^3$".

Note: the backslash already has a meaning in Python strings (the escape code character). To avoid Python messing up our latex code, we need to use "raw" text strings. Raw text strings are prepended with an 'r', like r"\alpha" or r'\alpha' instead of "\alpha" or '\alpha'



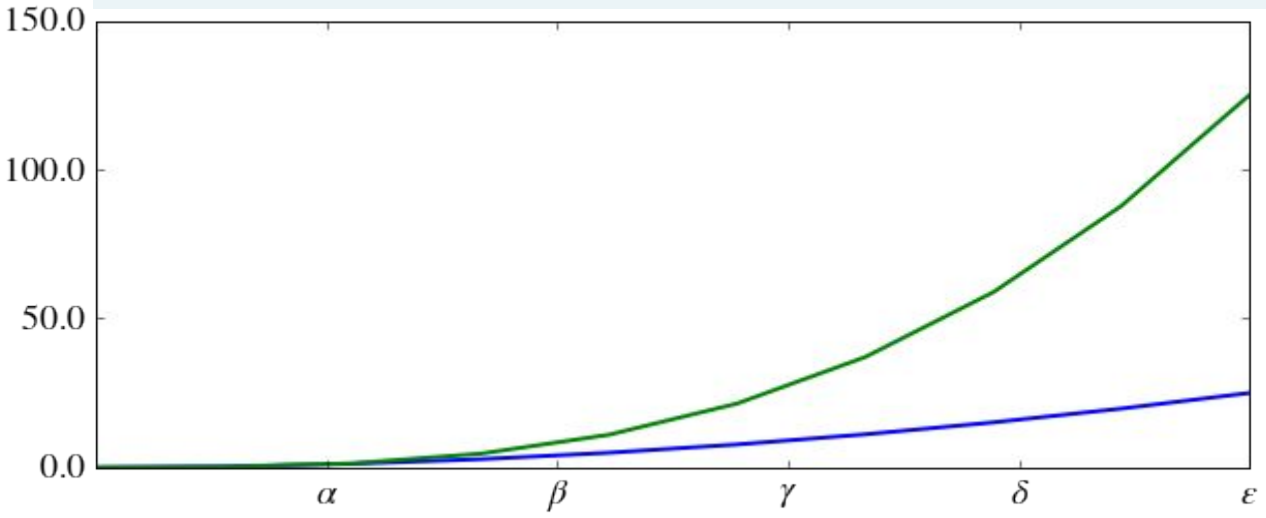
# Plot Range

```
In [1]: import numpy as np
In [2]: import matplotlib.pyplot as plt

…

In [3]: fig, axes = plt.subplots()

In [4]: axes[2].set_ylim([0,60])

In [5]: axes[2].set_xlim([2,5])
```



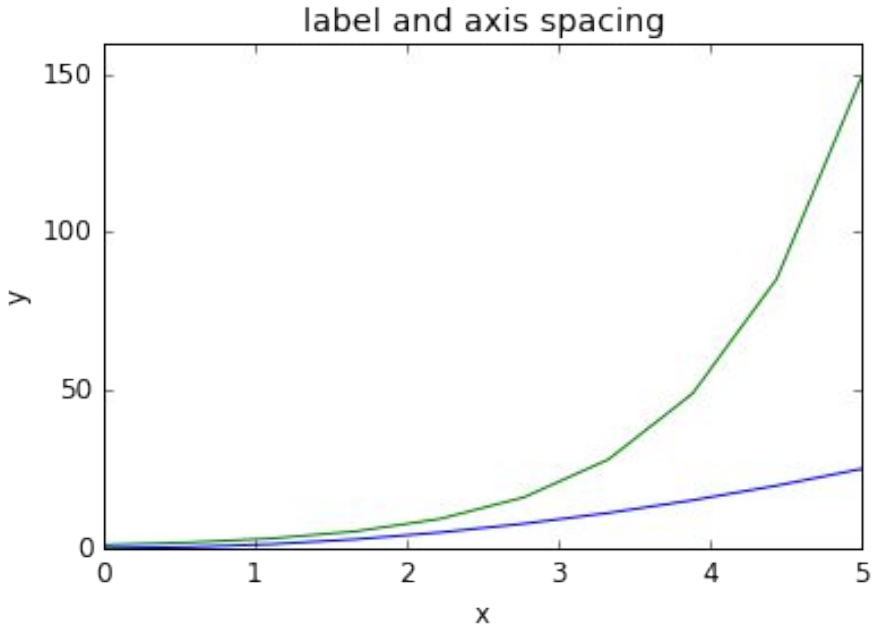
# Placement of Ticks

```
In [1]: import numpy as np
In [2]: import matplotlib.pyplot as plt

…

In [3]: fig, axes = plt.subplots()

In [4]: axes.set_xticks([1,2,3,4,5)]

In  [5]: axes.set_xticklabels([r'$\alphas$',..)
```



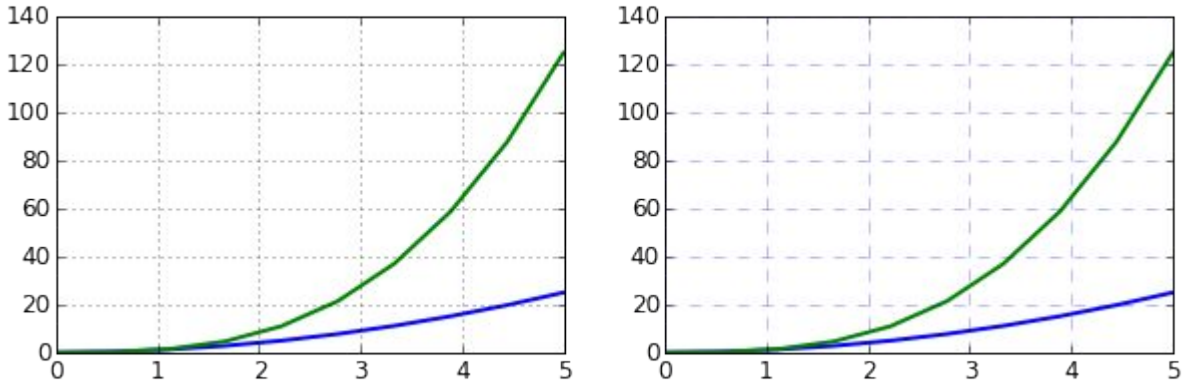
# Axis Number and Axis Label Spacing

```
In [1]: import numpy as np
In [2]: import matplotlib.pyplot as plt

…

In [3]: matplotlib.rcParams['xtick.major.pad']=5

In [4]: ax.xaxis.labelpad = 5
```



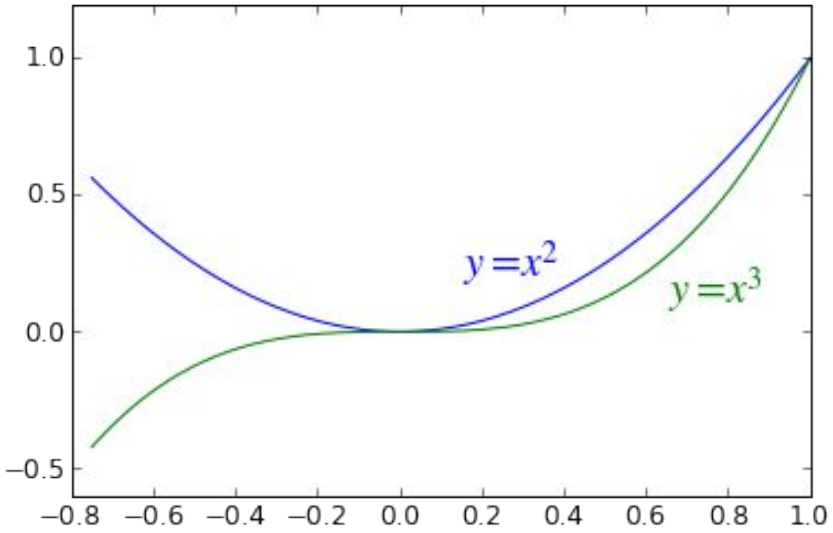
# Axis Grid

```
In [1]: import numpy as np
In [2]: import matplotlib.pyplot as plt

…

In [3]: axes[0].grid(True)

In [4]: axes[1].grid(color='b',alpha=0.5,linestyle='dashed',linewidth = 0.5)
```



# Text annotation

```
In [1]: import numpy as np
In [2]: import matplotlib.pyplot as plt

…

In [3]: ax.text(0.15,0.2,r"$y=x^2$",fontsize=20,color='blue')
```



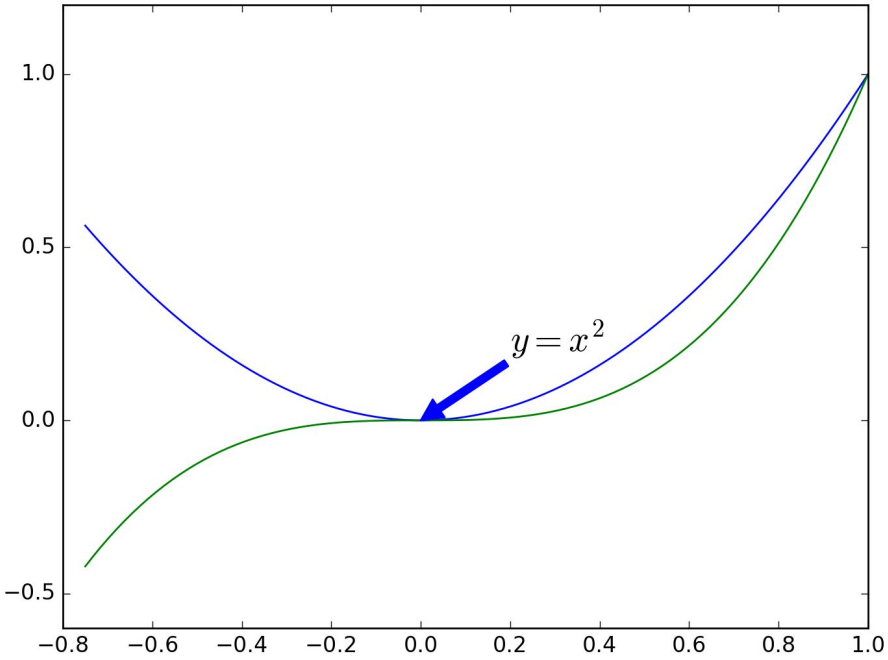
# Plot Annotations

- Text labels and arrows using annotate() method
- Flexible specification of coordinates
- Keyword arrowprops: dict of arrow properties (width,color,etc)

# Options for annotate()

| option | description |
| --- | --- |
| s | text of label |
| xy | coordinates to annotate |
| xytext | coordinates of label |
| arrowprops | controls drawing of arrow |

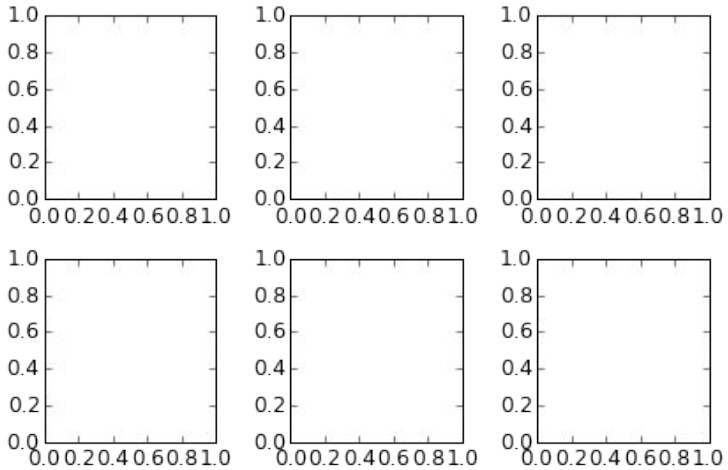
# Using Annotate() for Arrows



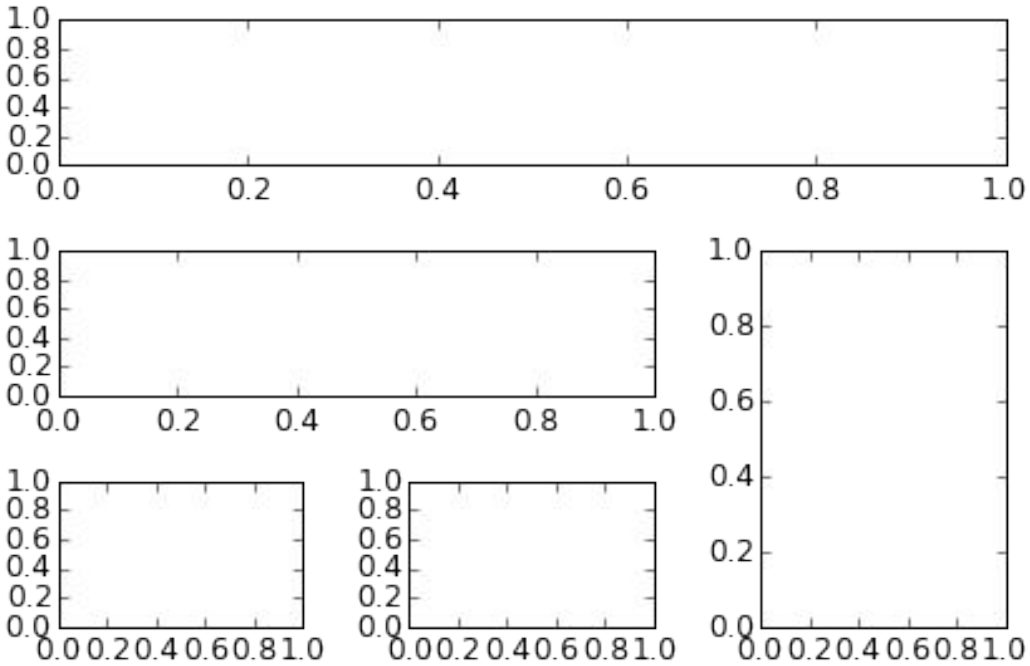
# The Subplot() Command

- Syntax: subplot(nrows, ncols, nsubplot)
- Subplot ordering: Row-wise from top left and indexed from 1

```
In [1]: import numpy as np
In [2]: import matplotlib.pyplot as plt

…

In [3]: fig,ax = plt.subplots(2,3)
```



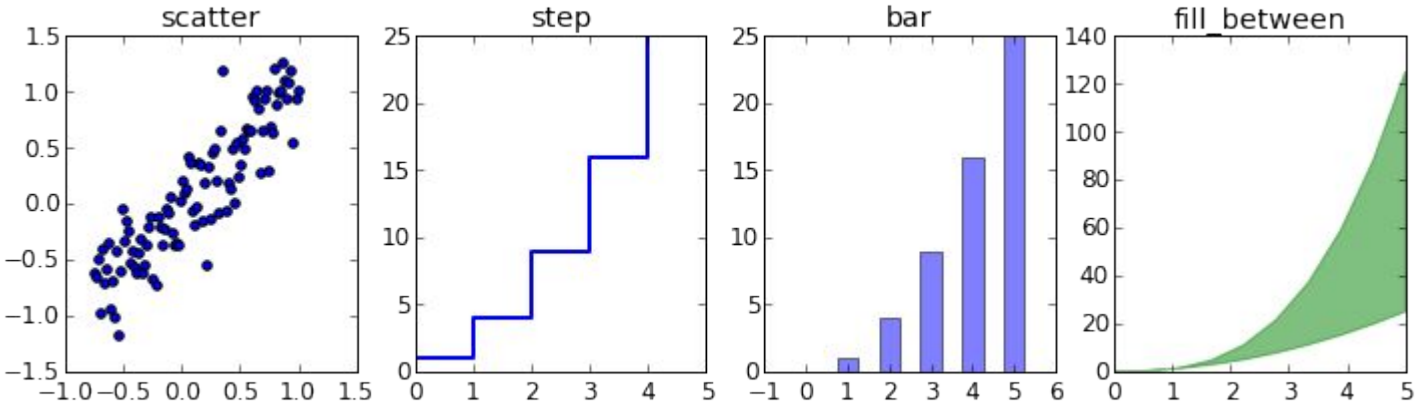
```
In [1]: import numpy as np
In [2]: import matplotlib.pyplot as plt

…

In [3]: ax1 =plt.subplot2grid((3,3),(0,0),colspan=3)
```



# 3.
# Other 2D Plots

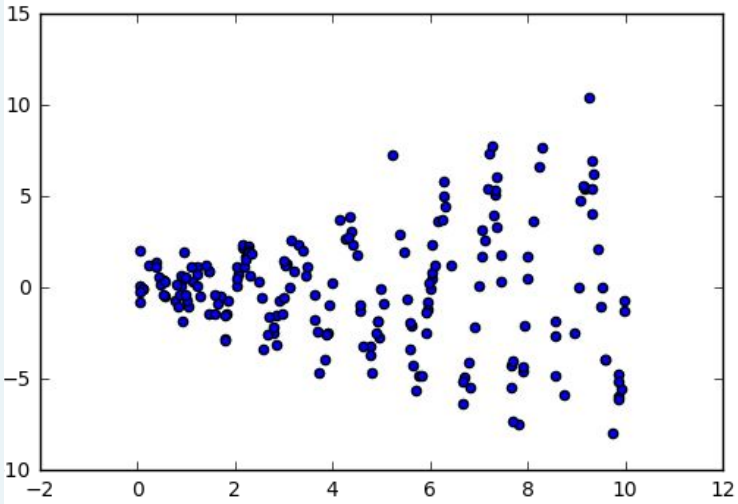
# Other 2D Plot Styles

- axes[0].scatter(...)
- axes[1].step(...)
- axes[2].bar(...)
- axes[3].fill_between(...)



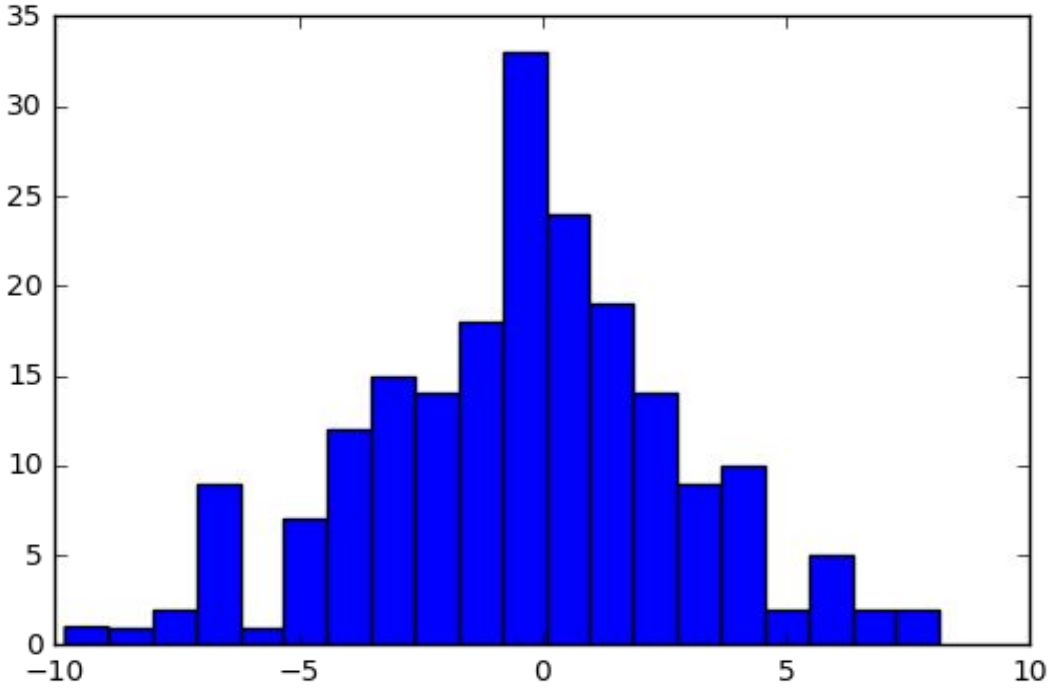
# Scatter Plots

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



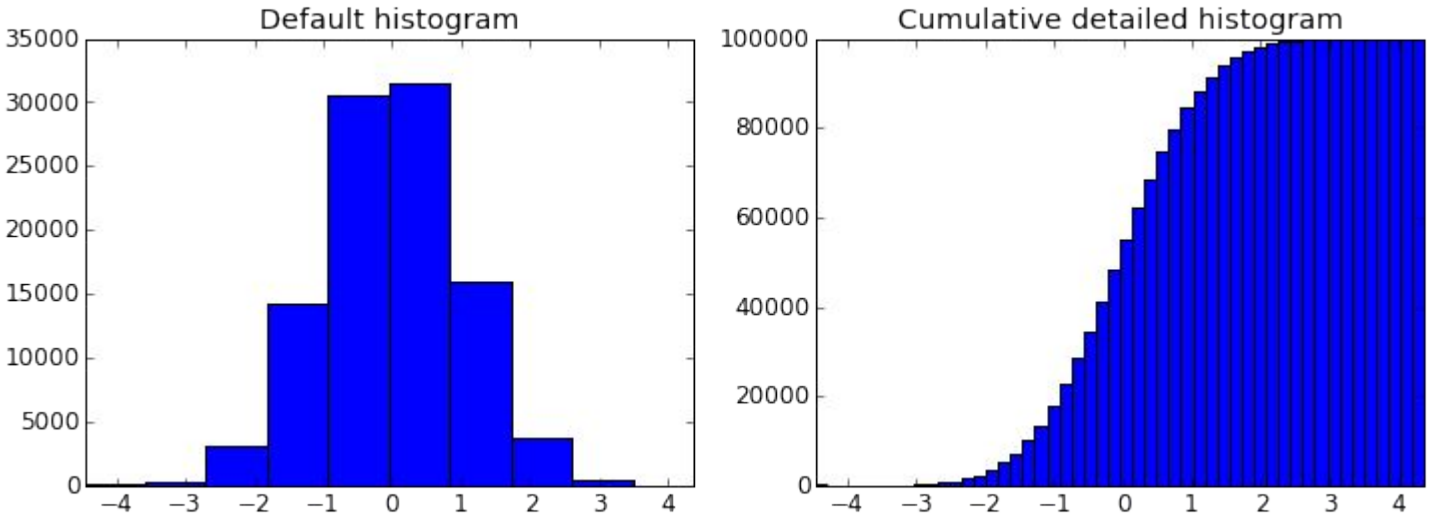
# Histograms

- axes[0].hist(y, bins=...,cumulative=...)



# Exercise

Plot this diagram!



# 4.
# Plot Styles

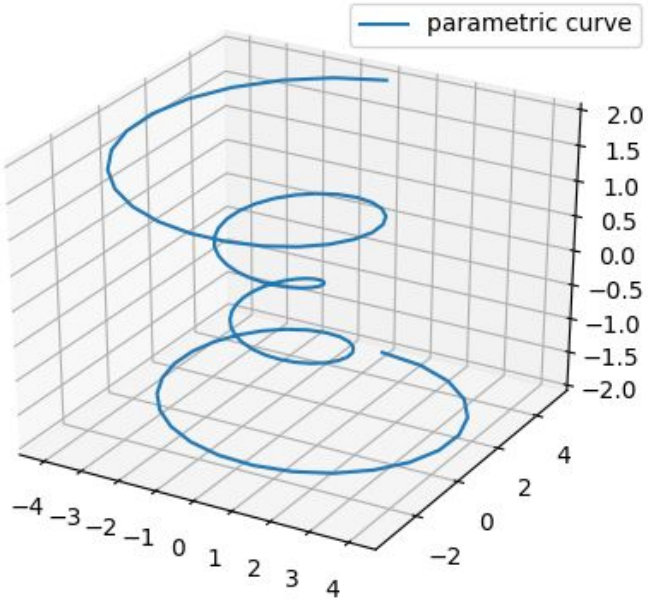
# Plot Styles

- Style sheets in matplotlib
- Default for lines, points, backgrounds, etc
- Switch styles globally with plt.style.use()
- Plt.style.available: list of styles

# 5.
# 3D Plots

# 3D Plots

- from mpl_toolkits import mplot3d
- ax = plt.axes(projection="3d")



# References

- https://github.com/jrjohansson/scientific-python-lectures/blob/master/Lecture-4-Matplotlib.ipynb
- https://www.datacamp.com/courses/introduction-to-data-visualization-with-python

# Thanks!

**Any questions?**