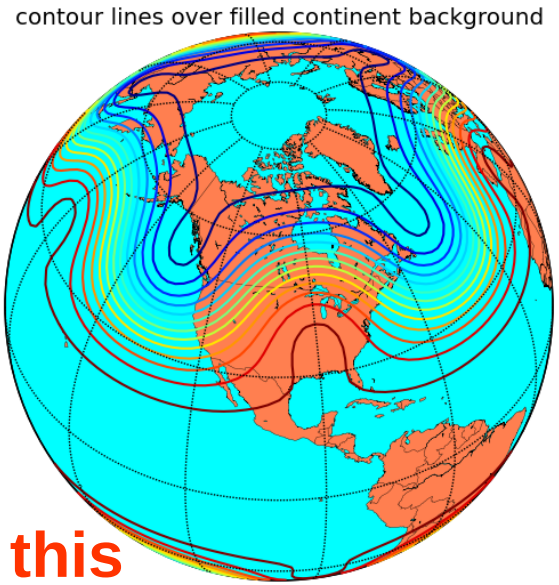
Day 8: Matplotlib

What is Matplotlib?

Matplotlib **is a third party Python library intended for data visualization**

- **Supports many types of 2D and 3D plots**
- **Built on NumPy**
- **Highly flexible and capable at the expense of a code-based interface**

We will learn how to use Matplotlib in conjunction with Python to visualize Some common types of data



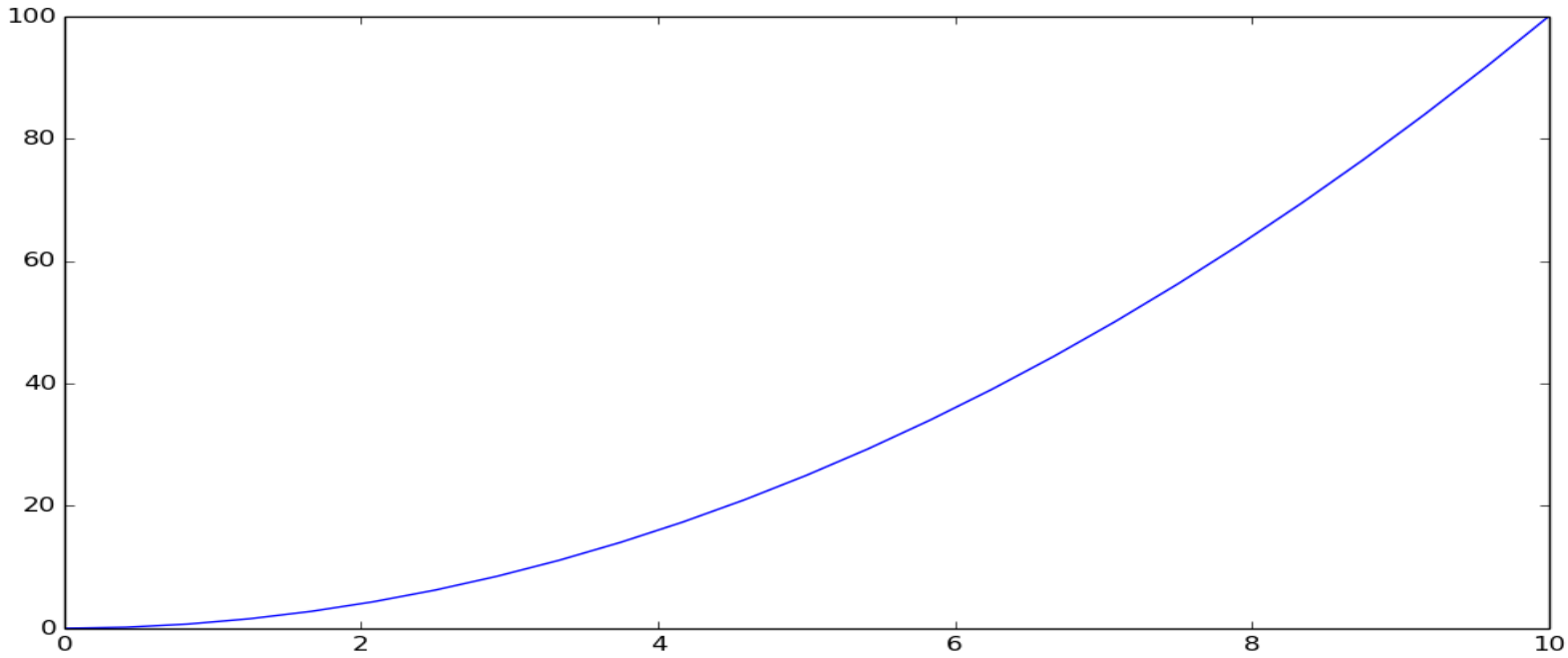
Today's lecture will be massively based on practical examples

You are guaranteed to need to know this at some point this summer!

Basics – A simple 2D line plot

```
import numpy as np
import matplotlib.pyplot as plt

x = np.linspace(0, 10, 25)
y = x**2

plt.plot(x,y)
plt.show()
```

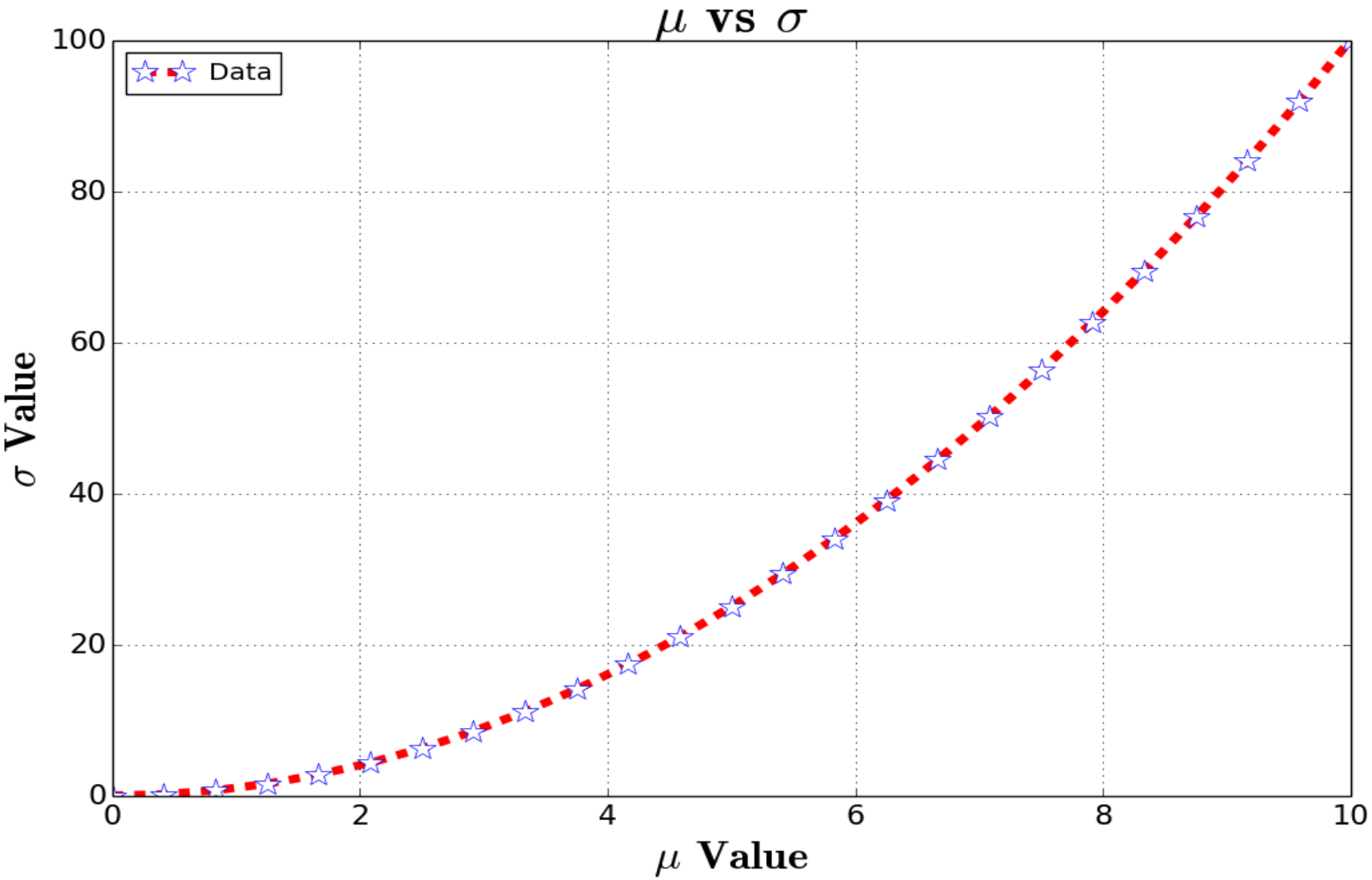
This looks utterly terrible. Let's go over some basic *presentation standards* for production visualizations



A More Complete Line Plot

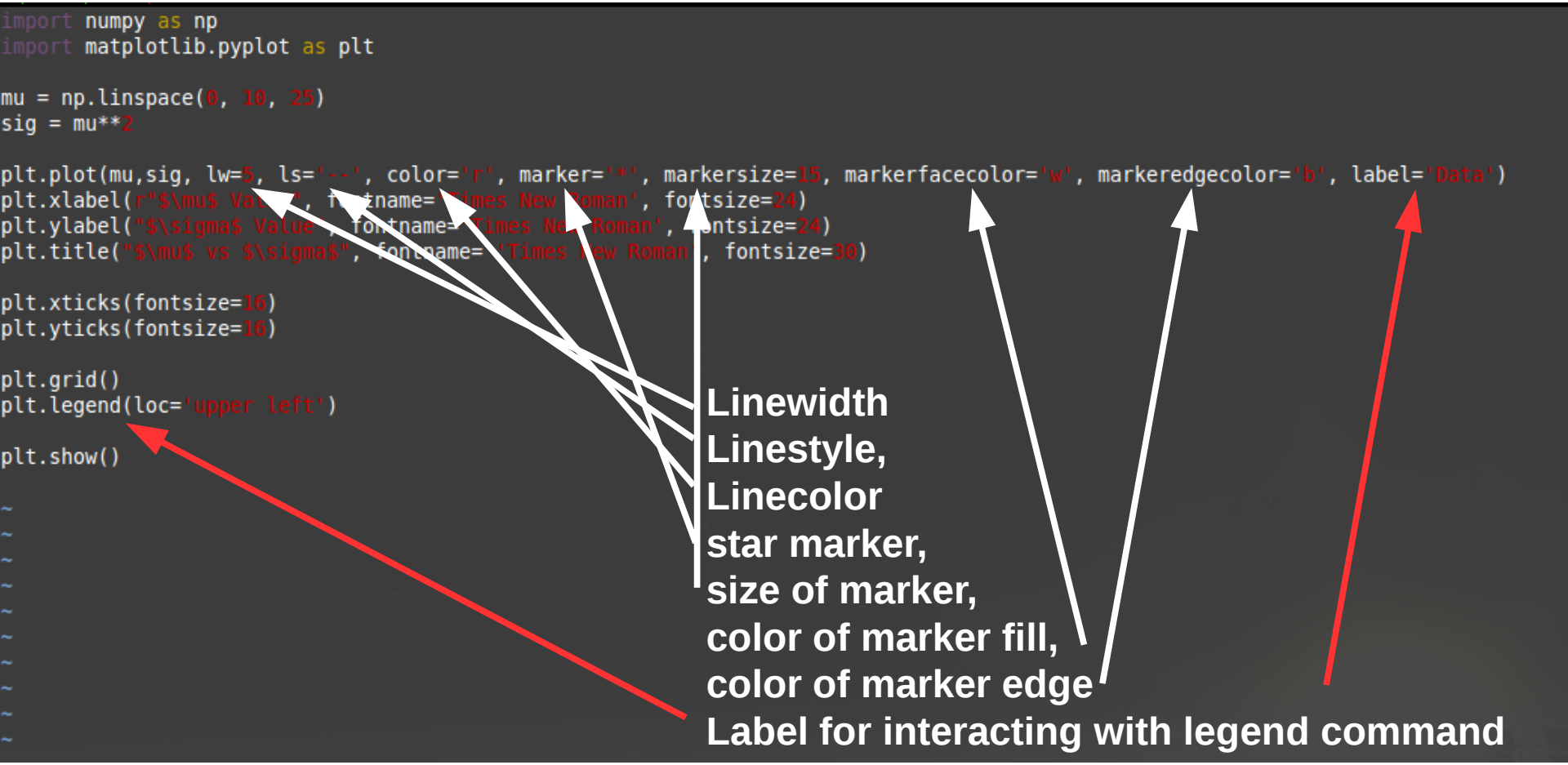
```
import numpy as np
import matplotlib.pyplot as plt

mu = np.linspace(0, 10, 25)
sig = mu**2

plt.plot(mu,sig, lw=5, ls='--', color='r', marker='*', markersize=15, markerfacecolor='w', markeredgecolor='b', label='Data')
plt.xlabel(r"$\mu$ Value", fontname='Times New Roman', fontsize=24)
plt.ylabel("$\sigma$ Value", fontname='Times New Roman', fontsize=24)
plt.title("$\mu$ vs $\sigma$", fontname= 'Times New Roman', fontsize=30)

plt.xticks(fontsize=16)
plt.yticks(fontsize=16)

plt.grid()
plt.legend(loc='upper left')

plt.show()

~
~
~
~
~
~
~
~
~
~
```

A More Complete Line Plot



Detail: Main Plot Command

```
import numpy as np
import matplotlib.pyplot as plt

mu = np.linspace(0, 10, 25)
sig = mu**2

plt.plot(mu,sig, lw=5, ls='--', color='r', marker='*', markersize=15, markerfacecolor='w', markeredgecolor='b', label='Data')
plt.xlabel(r"$\mu$ Value", fontname='Times New Roman', fontsize=24)
plt.ylabel("$\sigma$ Value", fontname='Times New Roman', fontsize=24)
plt.title("$\mu$ vs $\sigma$", fontname='Times New Roman', fontsize=30)

plt.xticks(fontsize=16)
plt.yticks(fontsize=16)

plt.grid()
plt.legend(loc='upper left')

plt.show()
```



What's Available

Linestyles

linestyle	description
`'-'` or `'solid'`	solid line
`'--'` or `'dashed'`	dashed line
`'-.'` or `'dash_dot'`	dash-dotted line
`':'` or `'dotted'`	dotted line
`'None'`	draw nothing
`' '`	draw nothing
`''`	draw nothing

Markerstyles (+many more)

marker	description
"."	point
","	pixel
"o"	circle
"v"	triangle_down
"^"	triangle_up
"<"	triangle_left
">"	triangle_right
"1"	tri_down
"2"	tri_up
"3"	tri_left
"4"	tri_right
"8"	octagon
"s"	square
"p"	pentagon
"*"	star

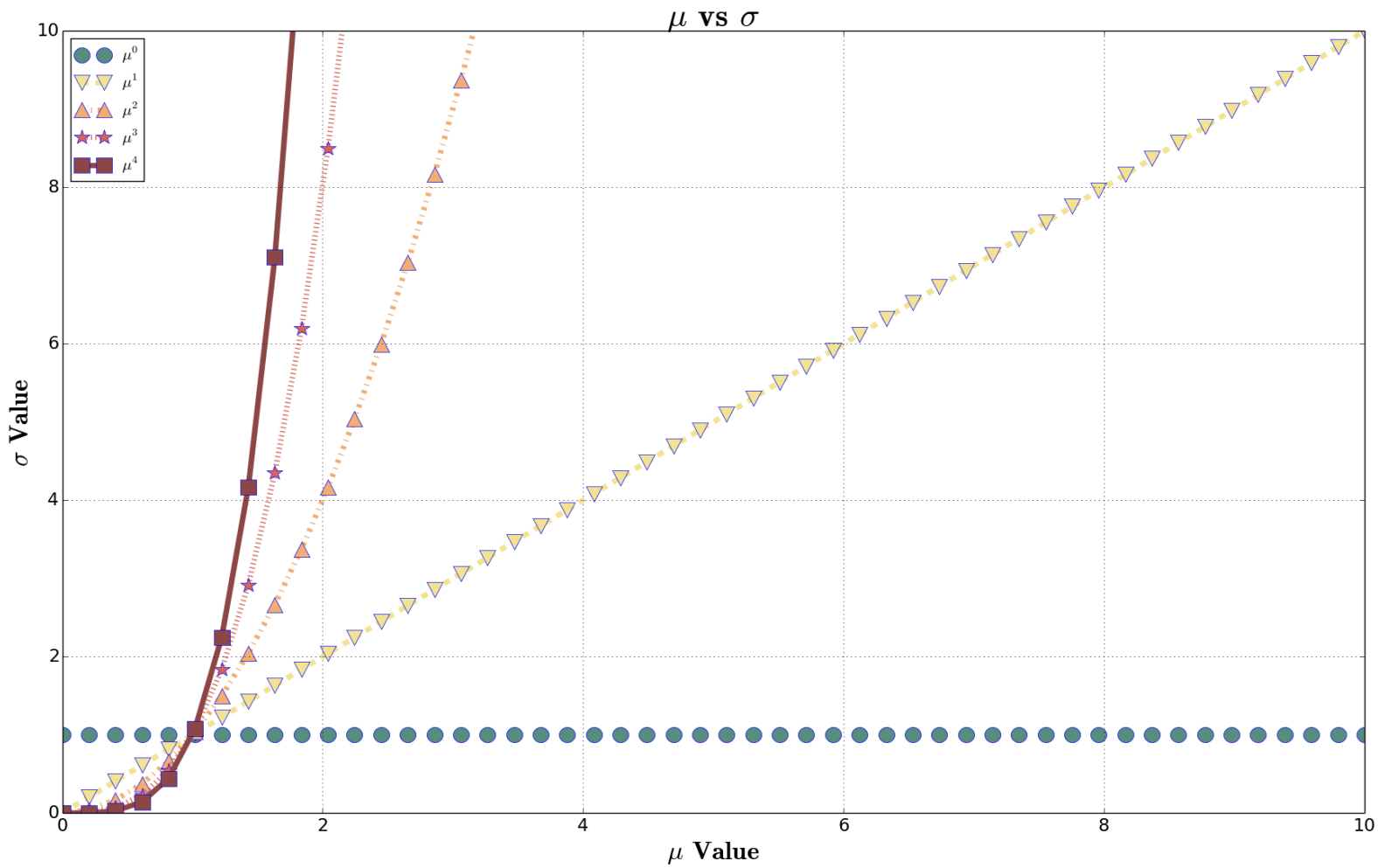
All of this information is easily accessible via the Matplotlib documentation which can be found online using GOOGLE!

A More Advanced Example

```
import numpy as np
import matplotlib.pyplot as plt

mu = np.linspace(0, 10, 50)


markers = ['o', 'v', '^', '*', 's']
styles  = ['.', '--', '-.', ':', '-']
colors  = ['#588C7E', '#F2E394','#F2AE72','#D96459','#8C4646']
for i in range(0,5):
        plt.plot(mu, mu**i, lw=5, ls=styles[i], color=colors[i], marker=markers[i], markersize=15, markerfacecolor=colors[i], markeredgecolor='b', label=r'$\mu^{%s}$' % i)



plt.xlabel(r"$\mu$ Value", fontname='Times New Roman', fontsize=24)
plt.ylabel("$\sigma$ Value", fontname='Times New Roman', fontsize=24)
plt.title("$\mu$ vs $\sigma$", fontname= 'Times New Roman', fontsize=30)

plt.ylim(0,10)
plt.xlim(0,10)

plt.xticks(fontsize=16)
plt.yticks(fontsize=16)

plt.grid()
plt.legend(loc='upper left')

plt.show()

~
~
~
```

You may call plt.plot multiple times to put additional “plot objects” onto the same “figure object” as in the loop above.

A More Advanced Example



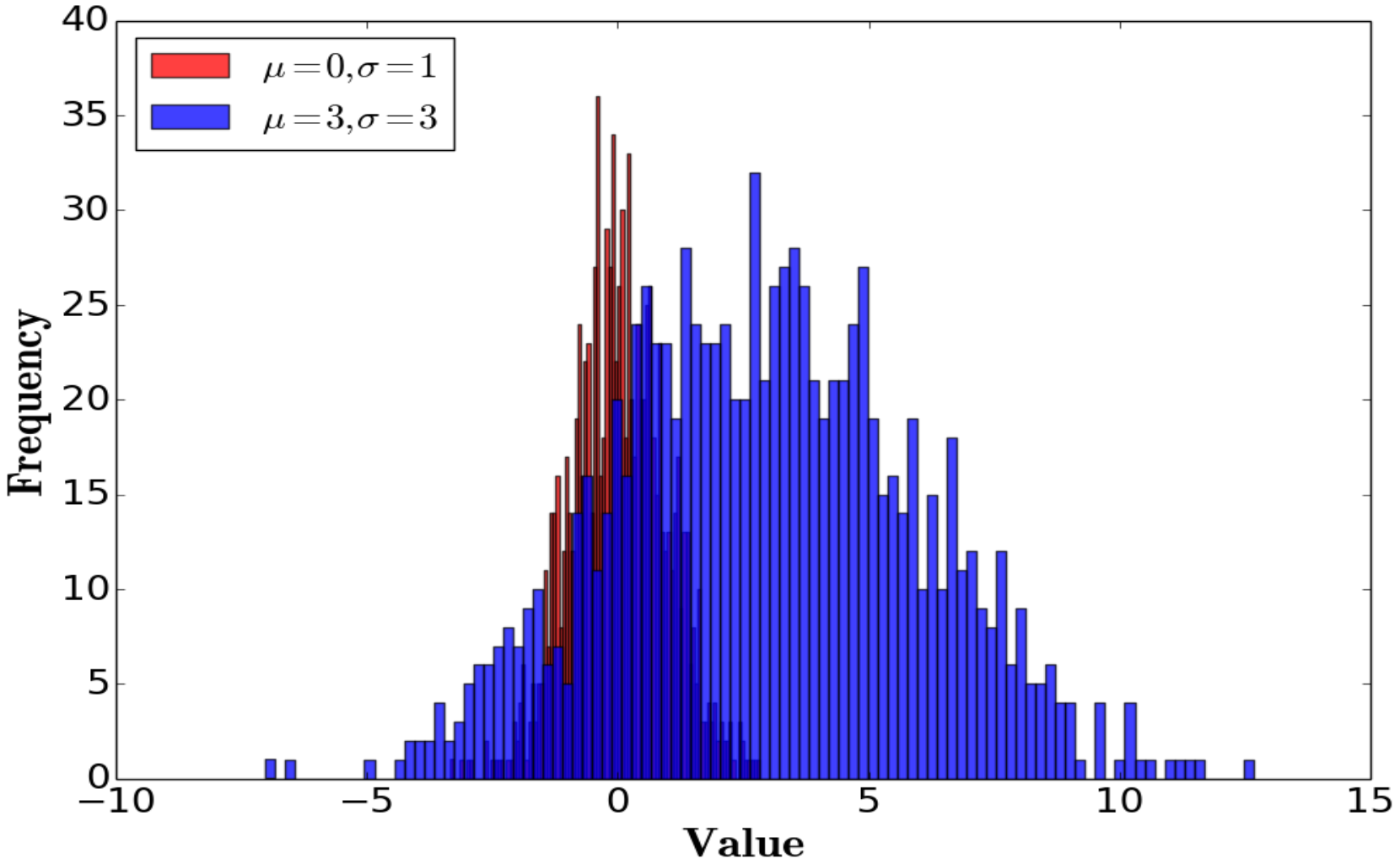
Histograms

```
import numpy as np
import matplotlib.pyplot as plt

data1 = np.random.normal(loc=0.0, scale=1.0, size=1000)
data2 = np.random.normal(loc=3.0, scale=3.0, size=1000)
plt.hist(data1, bins=100, label=r'$\mu = 0, \sigma = 1$', color='r', alpha=0.75)
plt.hist(data2, bins=100, label=r'$\mu = 3, \sigma = 3$', color='b', alpha=0.75)
plt.xticks(fontsize=20)
plt.yticks(fontsize=20)
plt.xlabel('Value', fontname='Times New Roman', fontsize=25)
plt.ylabel('Frequency', fontname='Times New Roman', fontsize=25)
plt.legend(loc='upper left', fontsize=20)
plt.show()
~
~
~
```

This is probably the most likely plot that you will have to make. Simply called through plt.hist

Notice the adjusted levels of *alpha* (opacity) meant to show histogram overlap. this is a very common design choice.

Histograms


$\mu = 0, \sigma = 1$
$\mu = 3, \sigma = 3$
Frequency
Value
0
5
10
15
20
25
30
35
40
−10
−5
0
5
10
15

2D Histograms

```
import numpy as np
import matplotlib.pyplot as plt

data1 = np.random.normal(loc=0.0, scale=1.0, size=100000)
data2 = np.random.normal(loc=3.0, scale=3.0, size=100000)
plt.hist2d(data1, data2, bins=25, label=r'$\mu = 0, \sigma = 1$', cmap='bone')
plt.xticks(fontsize=20)
plt.yticks(fontsize=20)
plt.xlabel('Data 1', fontname='Times New Roman', fontsize=25)
plt.ylabel('Data 2', fontname='Times New Roman', fontsize=25)
plt.legend(loc='upper left', fontsize=20)
plt.show()
~
```

Two dimensional histograms are used to represent cross-correlation between two sets of sampled data

A two dimensional histogram must use color to represent the frequency or height along the z-axis. However the second case is better handled with a surface plot for continuous functions

2D Histograms