Data Visualization Crash Course

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

Nice magic command for Jupyter Notebooks

In [2]:
```
%matplotlib inline
```

When not in Jupyter, use `plt.show()` *after your plot commands.*

Also, when not in Jupyter, save before you show.

In [3]:
```
x = np.arange(0, 10)
```

In [4]:
```
x
```

Out[4]:
```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

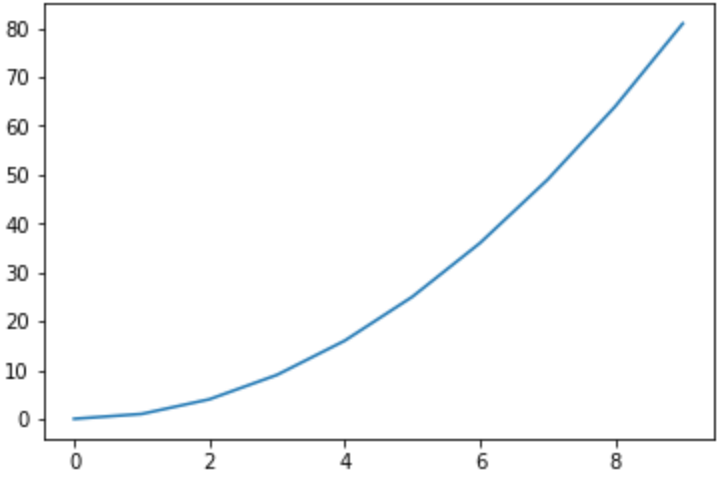
In [5]:
```
y = x**2
```

In [6]:
```
y
```

Out[6]:
```
array([ 0,  1,  4,  9, 16, 25, 36, 49, 64, 81], dtype=int32)
```

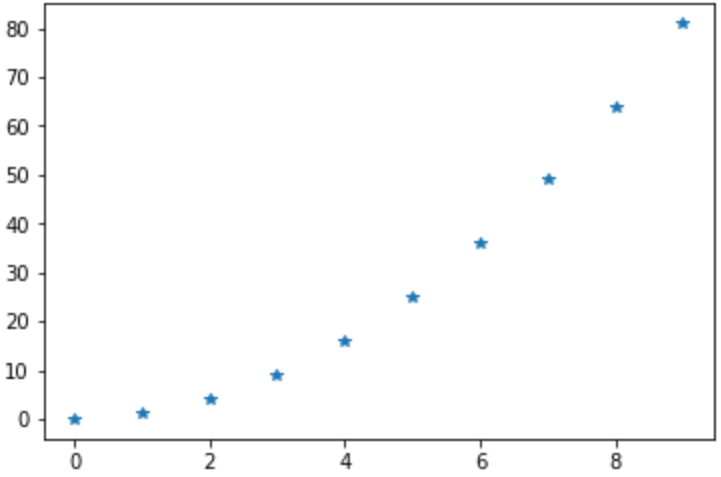
In [7]:
```
plt.plot(x, y)
```

Out[7]:
```
[<matplotlib.lines.Line2D at 0x2442440f198>]
```



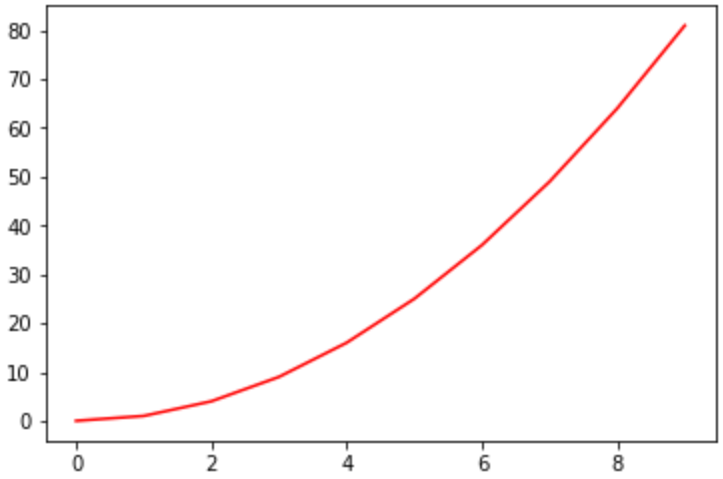
In [8]:
```
plt.plot(x, y, '*')
```

Out[8]:
```
[<matplotlib.lines.Line2D at 0x244254928d0>]
```



In [9]:
```
plt.plot(x, y, 'red')
```

Out[9]:
```
[<matplotlib.lines.Line2D at 0x24425505b38>]
```



From the tooltip (`Shift` + `Tab`)

Line style or marker

```
The following format string characters are accepted to control
the line style or marker:

================    ================================
character           description
================    ================================
``'-'``             solid line style
``'--'``            dashed line style
``'-.'``            dash-dot line style
``':'``             dotted line style
``'.'``             point marker
``','``             pixel marker
``'o'``             circle marker
``'v'``             triangle_down marker
``'^'``             triangle_up marker
``'<'``             triangle_left marker
``'>'``             triangle_right marker
``'1'``             tri_down marker
``'2'``             tri_up marker
``'3'``             tri_left marker
``'4'``             tri_right marker
``'s'``             square marker
``'p'``             pentagon marker
``'*'``             star marker
``'h'``             hexagon1 marker
``'H'``             hexagon2 marker
``'+'``             plus marker
``'x'``             x marker
``'D'``             diamond marker
``'d'``             thin_diamond marker
``'|'``             vline marker
```

```
``'_'``             hline marker
================    ================================
```

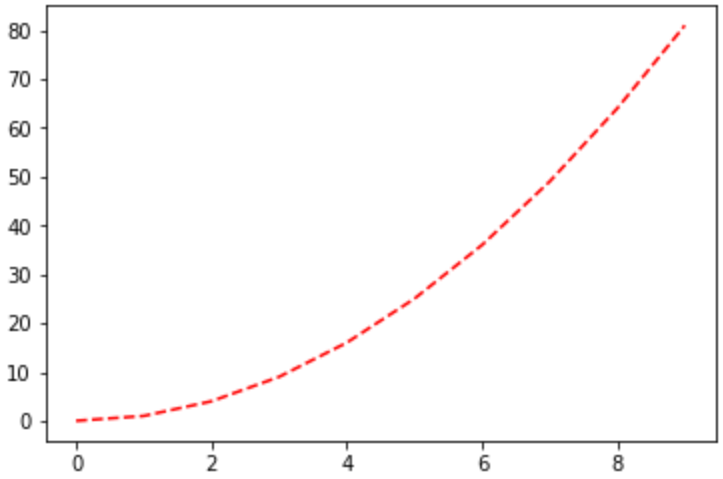
Color abbreviations

The following color abbreviations are supported:

character	color
'b'	blue
'g'	green
'r'	red
'c'	cyan
'm'	magenta
'y'	yellow
'k'	black
'w'	white

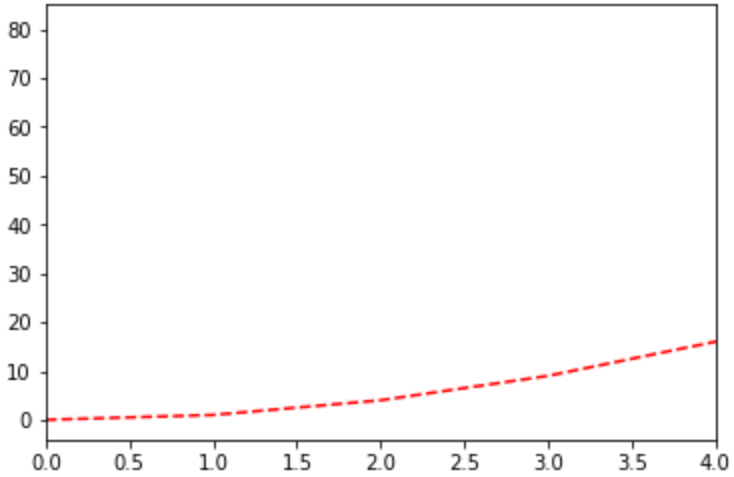
In [10]:
```
plt.plot(x, y, 'r--')
```

Out[10]:
```
[<matplotlib.lines.Line2D at 0x2442557b550>]
```



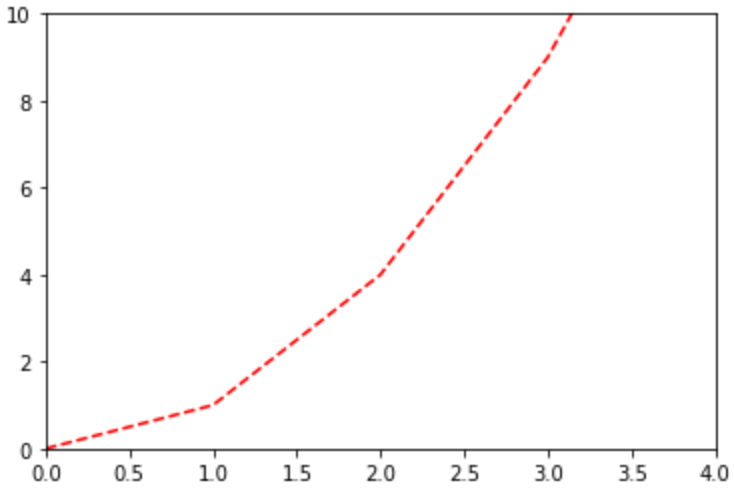
In [11]:
```
plt.plot(x, y, 'r--')
plt.xlim(0, 4)
```

Out[11]:
```
(0, 4)
```



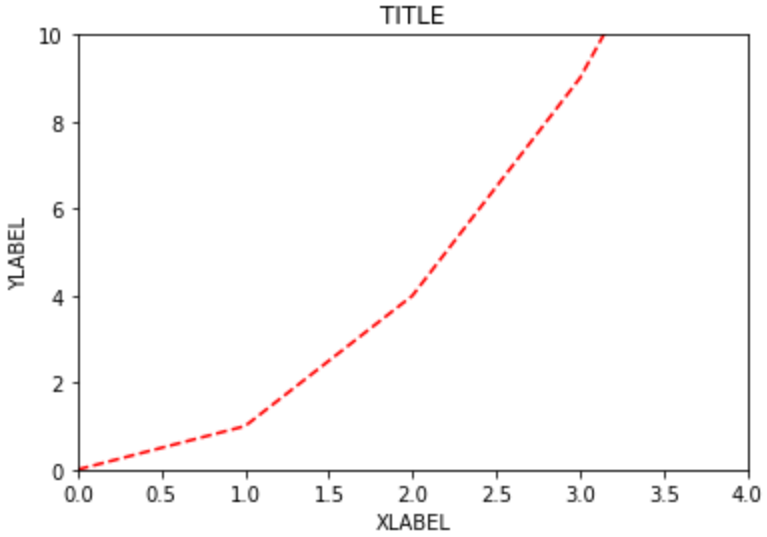
In [12]:
```
plt.plot(x, y, 'r--')
plt.xlim(0, 4)
plt.ylim(0, 10)
```

Out[12]: (0, 10)



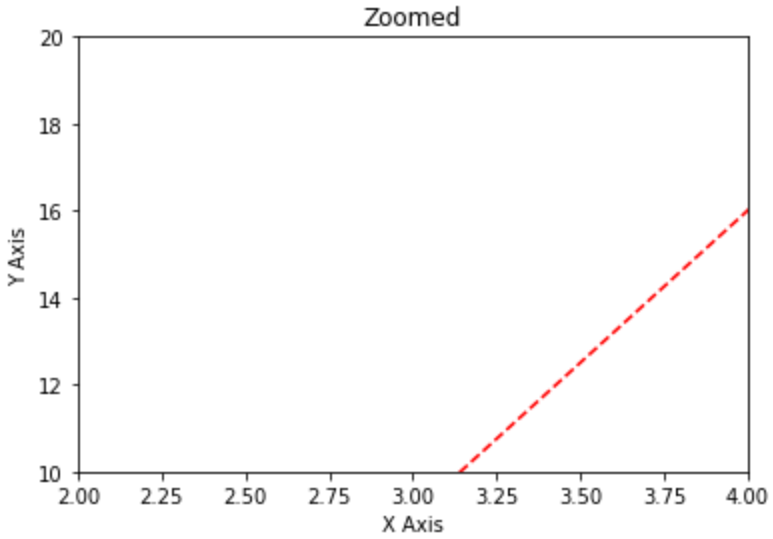
In [13]:
```
plt.plot(x, y, 'r--')
plt.xlim(0, 4)
plt.ylim(0, 10)
plt.title("TITLE")
plt.xlabel('XLABEL')
plt.ylabel('YLABEL')
```

Out[13]: <matplotlib.text.Text at 0x244255b1b70>



In [14]:
```
plt.plot(x, y, 'r--')
plt.xlim(2, 4)
plt.ylim(10, 20)
plt.title("Zoomed")
plt.xlabel("X Axis")
plt.ylabel("Y Axis")
```

Out[14]: <matplotlib.text.Text at 0x244256897b8>



Great! Now, let's plot a 2-D Matrix according to its indices

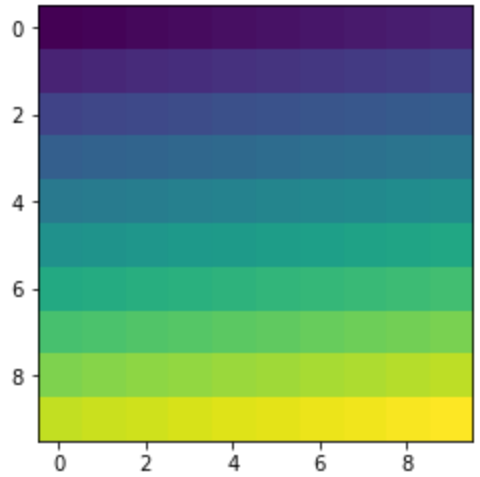
In [15]:
```
mat = np.arange(0, 100).reshape(10, 10)
```

In [16]:
```
mat
```

Out[16]:
```
array([[ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9],
       [10, 11, 12, 13, 14, 15, 16, 17, 18, 19],
       [20, 21, 22, 23, 24, 25, 26, 27, 28, 29],
       [30, 31, 32, 33, 34, 35, 36, 37, 38, 39],
       [40, 41, 42, 43, 44, 45, 46, 47, 48, 49],
       [50, 51, 52, 53, 54, 55, 56, 57, 58, 59],
       [60, 61, 62, 63, 64, 65, 66, 67, 68, 69],
       [70, 71, 72, 73, 74, 75, 76, 77, 78, 79],
       [80, 81, 82, 83, 84, 85, 86, 87, 88, 89],
       [90, 91, 92, 93, 94, 95, 96, 97, 98, 99]])
```

In [17]:
```
plt.imshow(mat)
```

Out[17]:
```
<matplotlib.image.AxesImage at 0x24425739940>
```



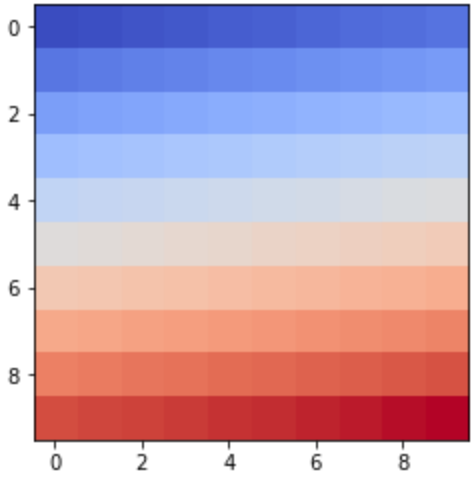
In [38]:
```
# Let's sneak out all of the color maps (rather than doing a straight docs check)
do_show_output_error_for_info = False

if do_show_output_error_for_info:
    plt.imshow(mat, cmap='idonotexist') #  This will give us an error with the
                                        #+ names of all the color maps
##endof:  if do_show_output_error_for_info
```

Here are the possible values from that last error (2023-11-12)

```
ValueError: Colormap idonotexist is not recognized. Possible values are: Accent, Accent_r, Blues, Blues_r, BrBG, B
rBG_r, BuGn, BuGn_r, BuPu, BuPu_r, CMRmap, CMRmap_r, Dark2, Dark2_r, GnBu, GnBu_r, Greens, Greens_r, Greys, Greys_
r, OrRd, OrRd_r, Oranges, Oranges_r, PRGn, PRGn_r, Paired, Paired_r, Pastel1, Pastel1_r, Pastel2, Pastel2_r, PiYG,
PiYG_r, PuBu, PuBuGn, PuBuGn_r, PuBu_r, PuOr, PuOr_r, PuRd, PuRd_r, Purples, Purples_r, RdBu, RdBu_r, RdGy, RdGy_
r, RdPu, RdPu_r, RdYlBu, RdYlBu_r, RdYlGn, RdYlGn_r, Reds, Reds_r, Set1, Set1_r, Set2, Set2_r, Set3, Set3_r, Spect
ral, Spectral_r, Vega10, Vega10_r, Vega20, Vega20_r, Vega20b, Vega20b_r, Vega20c, Vega20c_r, Wistia, Wistia_r, YlG
n, YlGnBu, YlGnBu_r, YlGn_r, YlOrBr, YlOrBr_r, YlOrRd, YlOrRd_r, afmhot, afmhot_r, autumn, autumn_r, binary, binar
y_r, bone, bone_r, brg, brg_r, bwr, bwr_r, cool, cool_r, coolwarm, coolwarm_r, copper, copper_r, cubehelix, cubehe
lix_r, flag, flag_r, gist_earth, gist_earth_r, gist_gray, gist_gray_r, gist_heat, gist_heat_r, gist_ncar, gist_nca
r_r, gist_rainbow, gist_rainbow_r, gist_stern, gist_stern_r, gist_yarg, gist_yarg_r, gnuplot, gnuplot2, gnuplot2_
r, gnuplot_r, gray, gray_r, hot, hot_r, hsv, hsv_r, inferno, inferno_r, jet, jet_r, magma, magma_r, nipy_spectral,
nipy_spectral_r, ocean, ocean_r, pink, pink_r, plasma, plasma_r, prism, prism_r, rainbow, rainbow_r, seismic, seis
mic_r, spectral, spectral_r, spring, spring_r, summer, summer_r, tab10, tab10_r, tab20, tab20_r, tab20b, tab20b_r,
tab20c, tab20c_r, terrain, terrain_r, viridis, viridis_r, winter, winter_r
```

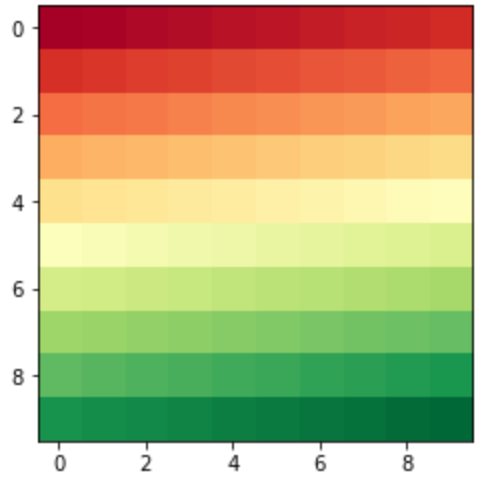
In [19]:
```
plt.imshow(mat, cmap="coolwarm")
```

Out[19]:
```
<matplotlib.image.AxesImage at 0x244259ddda0>
```



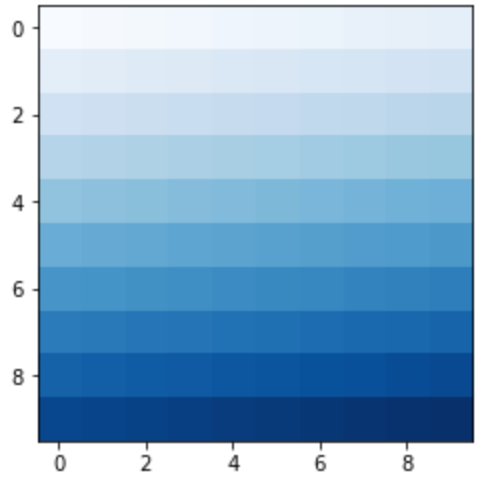
In [20]:
```
plt.imshow(mat, cmap='RdYlGn')
```

Out[20]:
```
<matplotlib.image.AxesImage at 0x24425a42898>
```



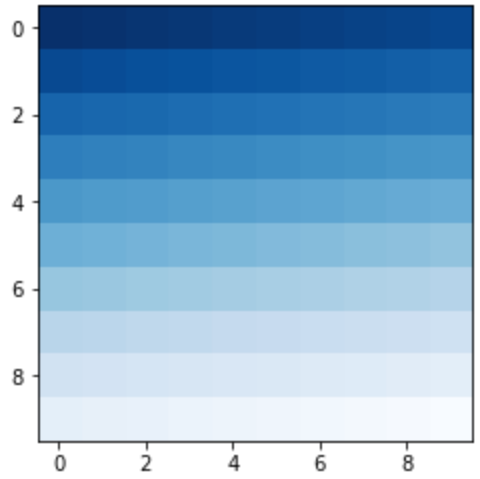
In [21]:
```
plt.imshow(mat, cmap='Blues')
```

Out[21]:
```
<matplotlib.image.AxesImage at 0x24425aaa048>
```



In [22]:
```
plt.imshow(mat, cmap='Blues_r')
```

Out[22]:
```
<matplotlib.image.AxesImage at 0x24425b0b828>
```

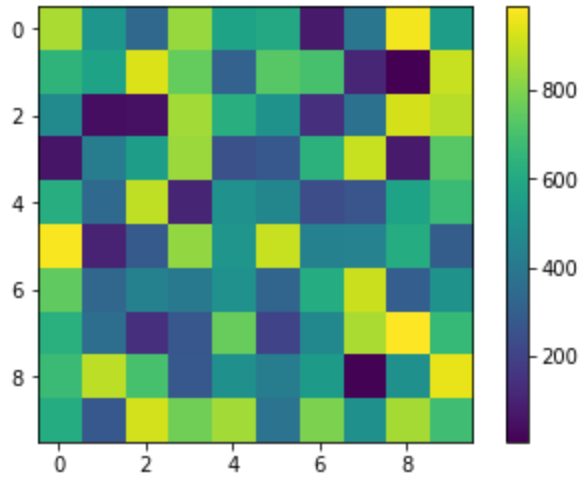


In [23]:
```
#  He doesn't use 101 this time - #
#+ or maybe he's just continued   #
#+ from where he was in the       #
#+ lecture. Anyway, no biggie.    #
#------ INVESTIGATION ------------#
#  He did use 101. See below.     #

cntr_101 = 0

np.random.seed(101)
```

In [24]:
```
# # Uncomment to leave a guard against resetting the seed and the counter.
# print("breakme) # making sure we don't come back here and re-set the count.
```

In [25]:
```
mat = np.random.randint(0, 1000, (10, 10)); cntr_101 += 1
plt.imshow(mat)
plt.colorbar()
print("\ncntr_101: " + str(cntr_101))
```

```
cntr_101: 1
```



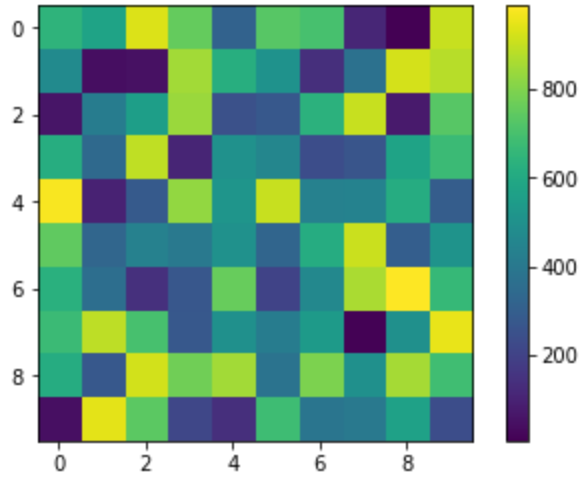
I forgot about the offset no necessarily being a multiple of 100.

I figured it out, though. We can get what he has on the lecture with the following.

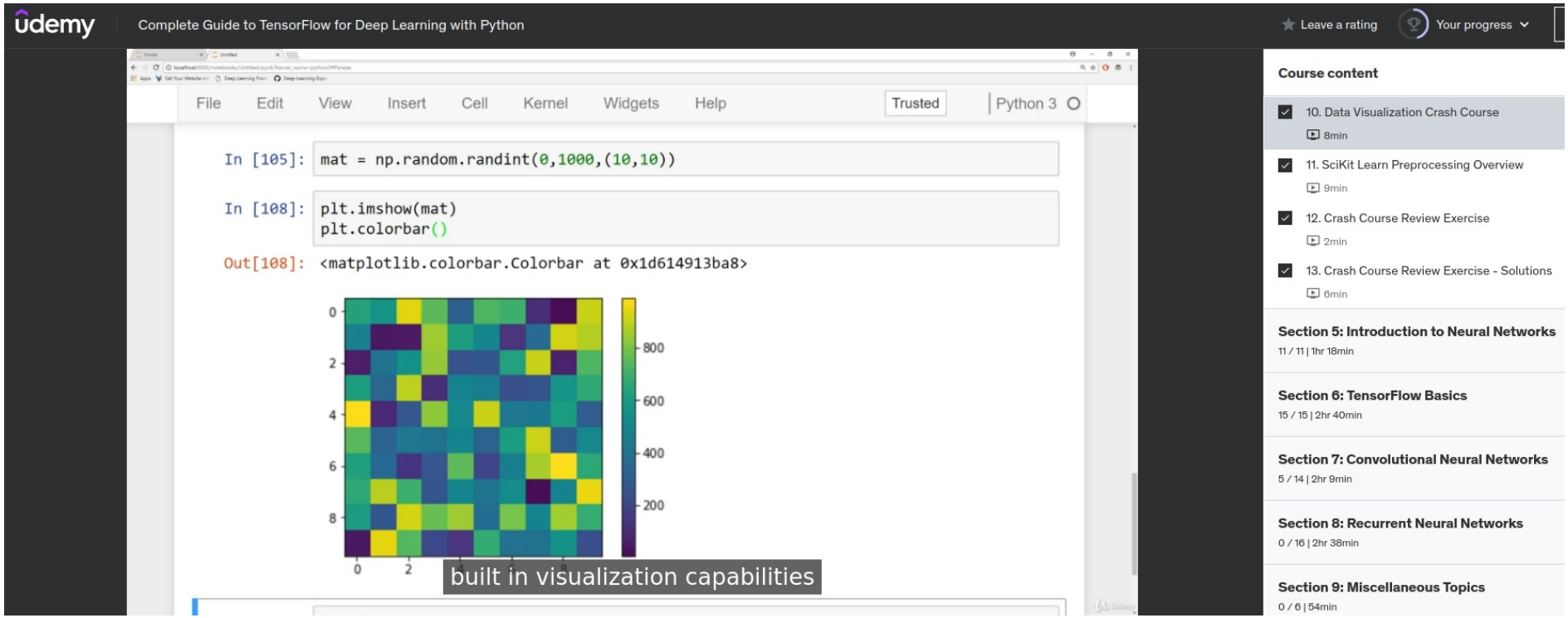
In [26]:
```
np.random.seed(101)
print(np.random.randint(1, 1000, (1, 10)))

mat = np.random.randint(0, 1000, (10, 10))
plt.imshow(mat)
plt.colorbar()
```

```
[[864 524 338 839 576 600  76 394 974 553]]
```

Out[26]:
```
<matplotlib.colorbar.Colorbar at 0x24425b44a20>
```



Compare

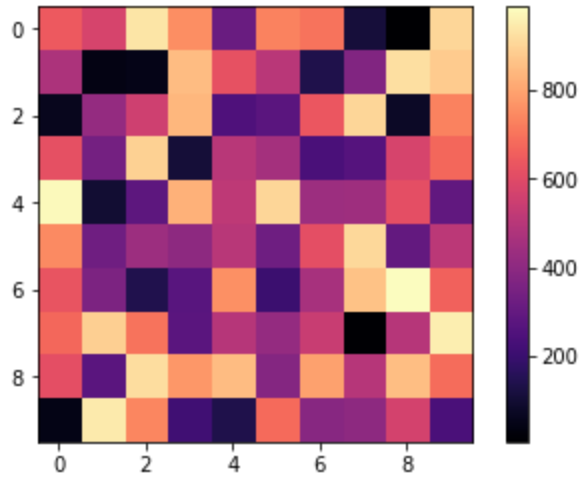

udemy
Complete Guide to TensorFlow for Deep Learning with Python
Leave a rating
Your progress
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
In [105]: mat = np.random.randint(0,1000,(10,10))
In [108]: plt.imshow(mat)
plt.colorbar()
Out[108]: <matplotlib.colorbar.Colorbar at 0x1d614913ba8>
built in visualization capabilities
Course content
10. Data Visualization Crash Course
8min
11. SciKit Learn Preprocessing Overview
9min
12. Crash Course Review Exercise
2min
13. Crash Course Review Exercise - Solutions
6min
Section 5: Introduction to Neural Networks
11 / 11 | 1hr 18min
Section 6: TensorFlow Basics
15 / 15 | 2hr 40min
Section 7: Convolutional Neural Networks
5 / 14 | 2hr 9min
Section 8: Recurrent Neural Networks
0 / 16 | 2hr 38min
Section 9: Miscellaneous Topics
0 / 6 | 54min

I want to try a few other color maps. The list, again.

```
Accent, Accent_r, Blues, Blues_r, BrBG, BrBG_r, BuGn, BuGn_r, BuPu, BuPu_r, CMRmap, CMRmap_r, Dark2, Dark2_r, GnB
u, GnBu_r, Greens, Greens_r, Greys, Greys_r, OrRd, OrRd_r, Oranges, Oranges_r, PRGn, PRGn_r, Paired, Paired_r, Pas
tel1, Pastel1_r, Pastel2, Pastel2_r, PiYG, PiYG_r, PuBu, PuBuGn, PuBuGn_r, PuBu_r, PuOr, PuOr_r, PuRd, PuRd_r, Pur
ples, Purples_r, RdBu, RdBu_r, RdGy, RdGy_r, RdPu, RdPu_r, RdYlBu, RdYlBu_r, RdYlGn, RdYlGn_r, Reds, Reds_r, Set1,
Set1_r, Set2, Set2_r, Set3, Set3_r, Spectral, Spectral_r, Vega10, Vega10_r, Vega20, Vega20_r, Vega20b, Vega20b_r,
Vega20c, Vega20c_r, Wistia, Wistia_r, YlGn, YlGnBu, YlGnBu_r, YlGn_r, YlOrBr, YlOrBr_r, YlOrRd, YlOrRd_r, afmhot,
afmhot_r, autumn, autumn_r, binary, binary_r, bone, bone_r, brg, brg_r, bwr, bwr_r, cool, cool_r, coolwarm, coolwa
rm_r, copper, copper_r, cubehelix, cubehelix_r, flag, flag_r, gist_earth, gist_earth_r, gist_gray, gist_gray_r, gi
st_heat, gist_heat_r, gist_ncar, gist_ncar_r, gist_rainbow, gist_rainbow_r, gist_stern, gist_stern_r, gist_yarg, g
ist_yarg_r, gnuplot, gnuplot2, gnuplot2_r, gnuplot_r, gray, gray_r, hot, hot_r, hsv, hsv_r, inferno, inferno_r, je
t, jet_r, magma, magma_r, nipy_spectral, nipy_spectral_r, ocean, ocean_r, pink, pink_r, plasma, plasma_r, prism, p
rism_r, rainbow, rainbow_r, seismic, seismic_r, spectral, spectral_r, spring, spring_r, summer, summer_r, tab10, t
ab10_r, tab20, tab20_r, tab20b, tab20b_r, tab20c, tab20c_r, terrain, terrain_r, viridis, viridis_r, winter, winter
_r
```

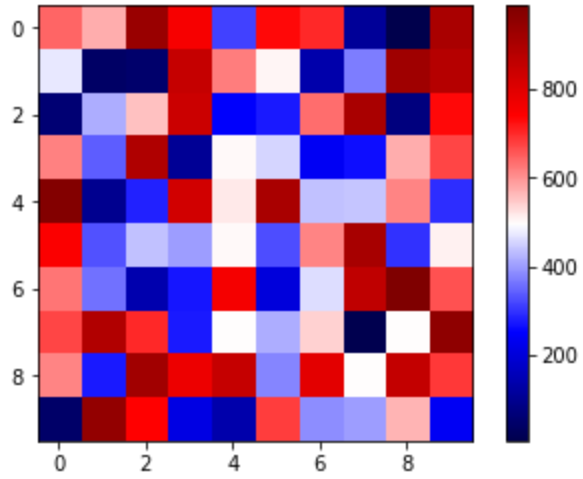
In [27]:
```
plt.imshow(mat, cmap='magma')
plt.colorbar()
```

Out[27]:
```
<matplotlib.colorbar.Colorbar at 0x244255b1f98>
```



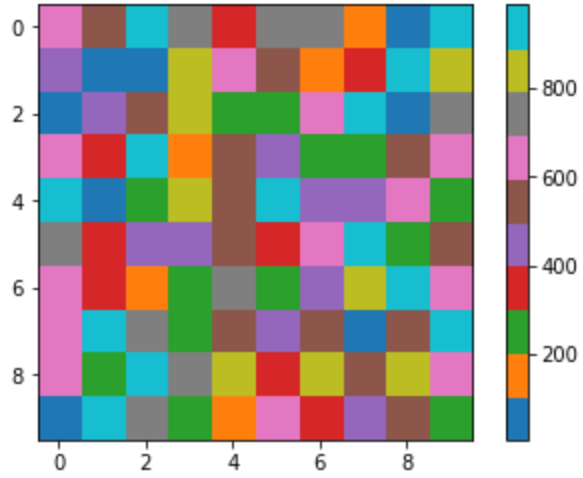
In [28]:
```
plt.imshow(mat, cmap='seismic')
plt.colorbar()
```

Out[28]:
```
<matplotlib.colorbar.Colorbar at 0x24425c58898>
```



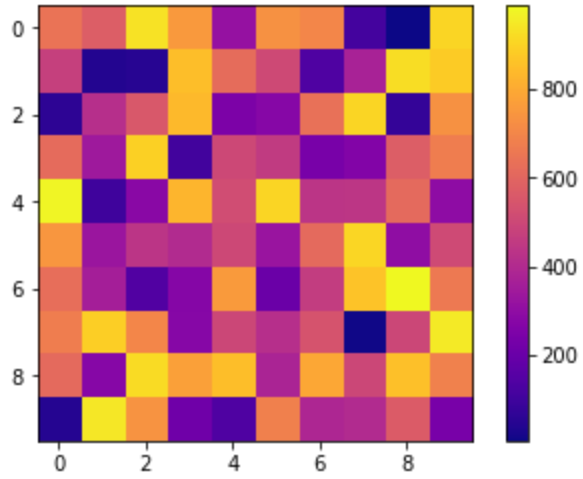
In [29]:
```
plt.imshow(mat, cmap='tab10')
plt.colorbar()
```

Out[29]:
```
<matplotlib.colorbar.Colorbar at 0x24425d07208>
```



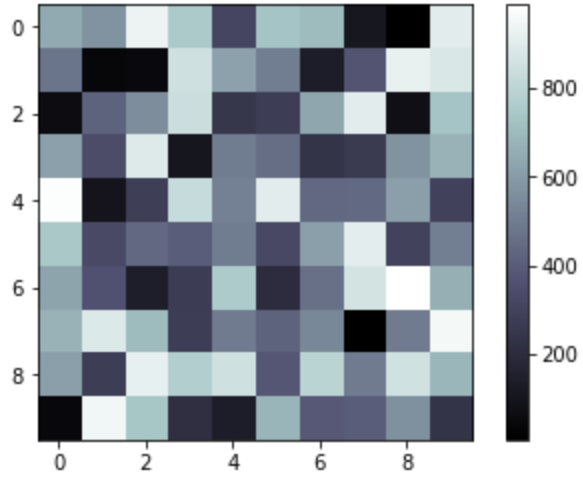
In [30]:
```
plt.imshow(mat, cmap='plasma')
plt.colorbar()
```

Out[30]: <matplotlib.colorbar.Colorbar at 0x24425dec400>



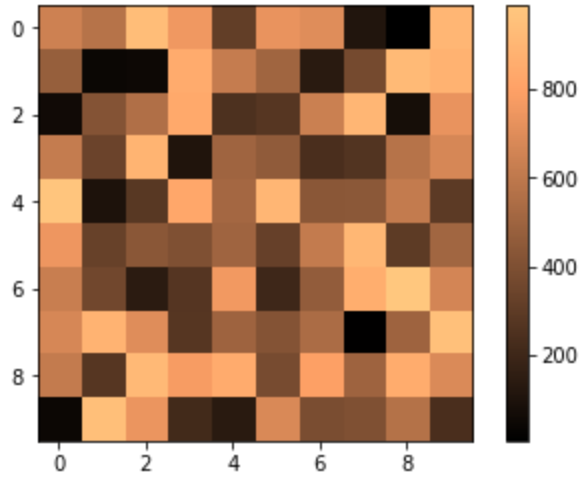
In [31]:
```
plt.imshow(mat, cmap='bone')
plt.colorbar()
```

Out[31]: <matplotlib.colorbar.Colorbar at 0x24425c6dcf8>



In [32]:
```
plt.imshow(mat, cmap='copper')
plt.colorbar()
```

Out[32]:
```
<matplotlib.colorbar.Colorbar at 0x24425f63358>
```



Pandas Plotting

In [33]:
```
df = pd.read_csv('salaries.csv')
```

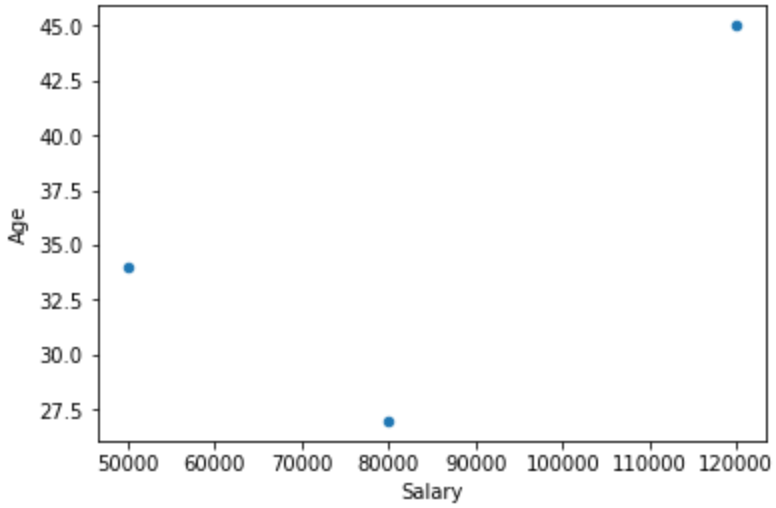
In [34]:
```
df
```

Out[34]:

	Name	Salary	Age
0	John	50000	34
1	Sally	120000	45
2	Alyssa	80000	27

In [35]:
```
df.plot(x='Salary', y='Age', kind='scatter')
```

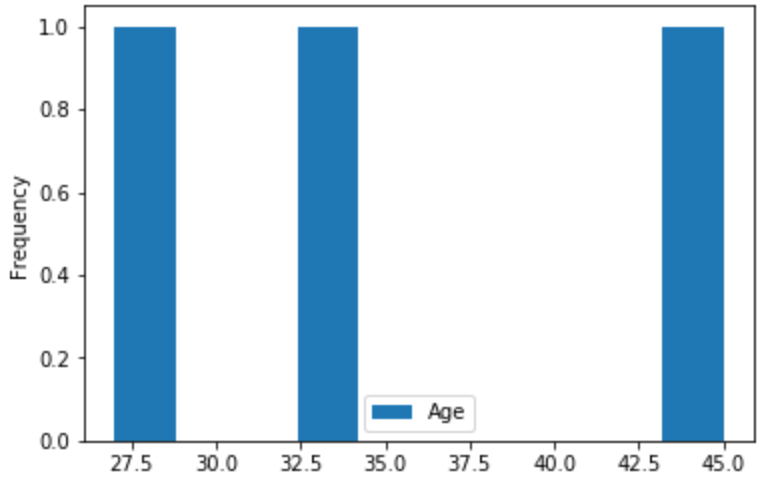
Out[35]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x24426ff1828>
```



And now, a histogram as done in the course files.

In [36]:
```
df.plot(x='Salary', kind='hist')
```

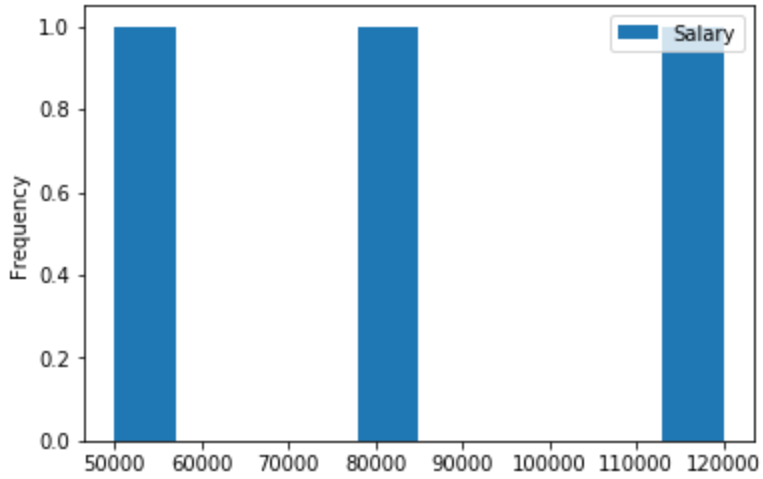
Out[36]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x24426fca128>
```



Not my favorite legend, nor my favorite way of things being shown, considering what we entered. The entries of the histogram correspond to ages, not salaries.

In [37]:
```
df.plot(x='Age', kind='hist')
```

Out[37]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x24427129208>
```



Well, that's what I would have expected from the other command. Meh!

That's all for now!

https://www.udemy.com/course/complete-guide-to-tensorflow-for-deep-learning-with-python/learn/lecture/7982588 (https://www.udemy.com/course/complete-guide-to-tensorflow-for-deep-learning-with-python/learn/lecture/7982588)