INTRO TO PYTHON 3

Be sure to get an account at

https://research.google.com/colaboratory/

DataNexus
ANALYZE • INTERPRET

DOWNLOAD THE DATA FOR THIS SESSION

Before we begin download

gunlaws.csv from our website

DataNexus
ANALYZE • INTERPRET

Intro_to_Python Part 3

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

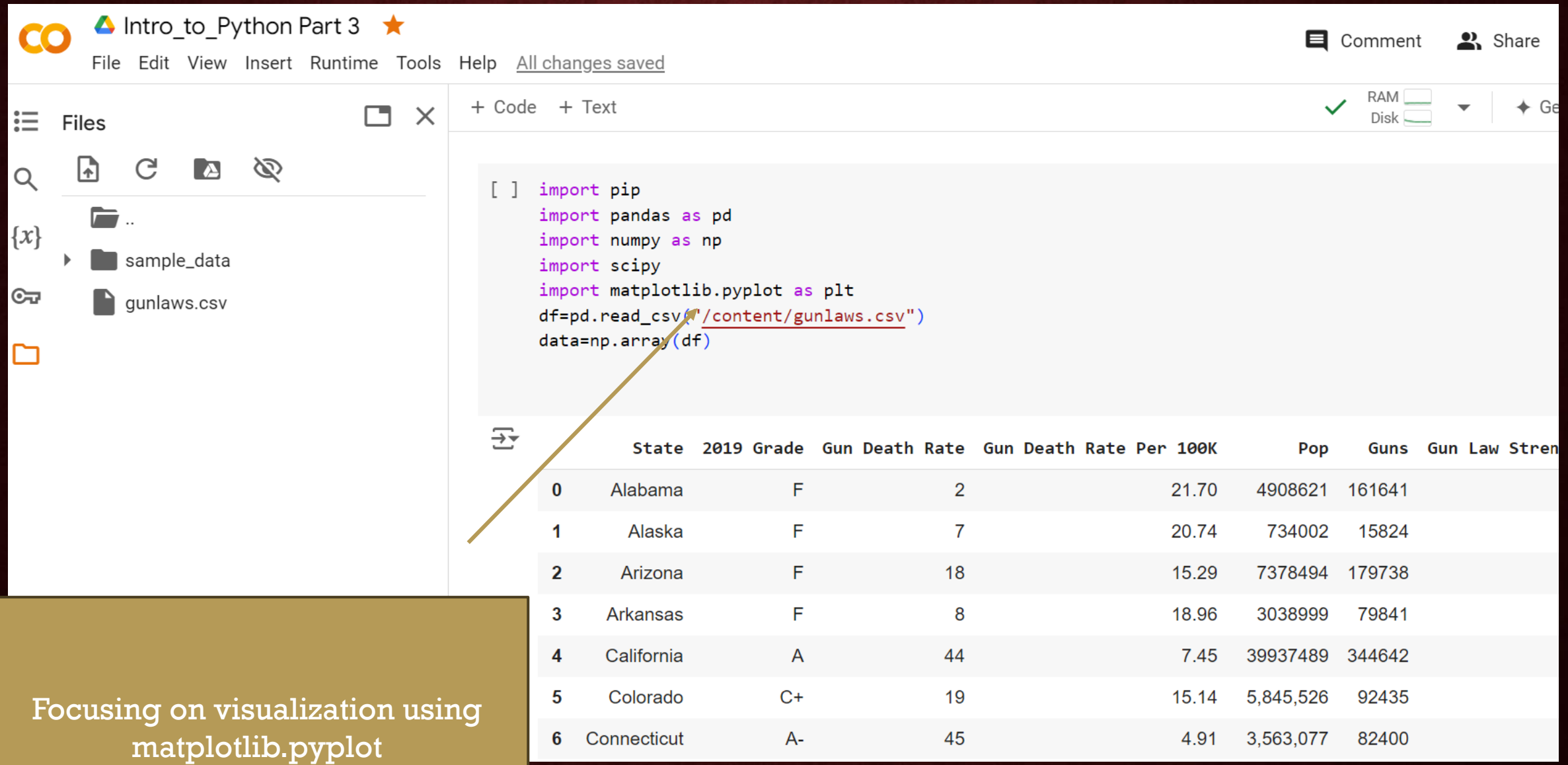
Files

+ Code + Text

RAM Disk

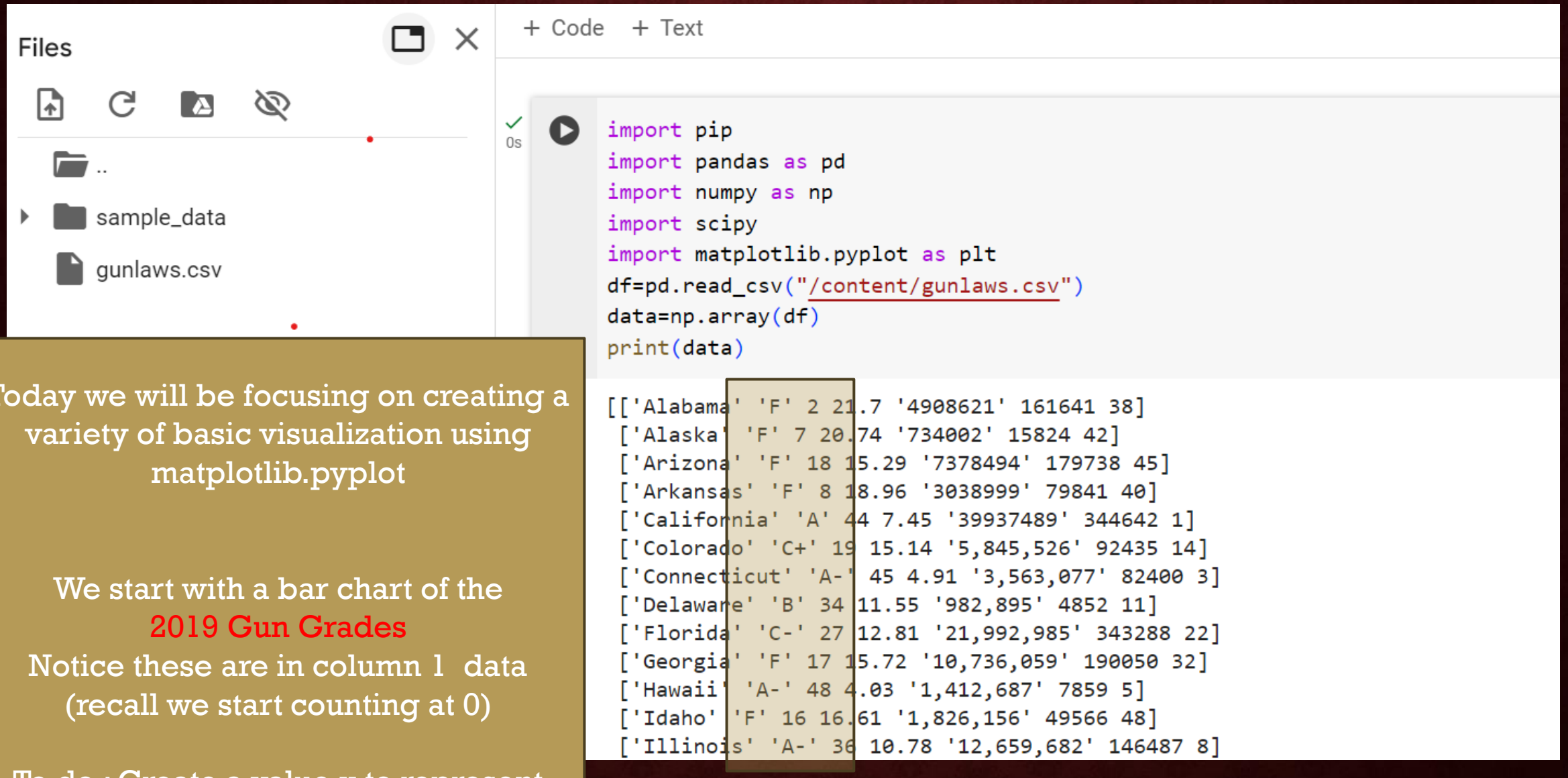
```
[ ]  import pip
     import pandas as pd
     import numpy as np
     import scipy
     import matplotlib.pyplot as plt
     df=pd.read_csv("/content/gunlaws.csv")
     data=np.array(df)
```

	State	2019 Grade	Gun Death Rate	Gun Death Rate Per 100K	Pop	Guns	Gun Law Stren
0	Alabama	F	2	21.70	4908621	161641	
1	Alaska	F	7	20.74	734002	15824	
2	Arizona	F	18	15.29	7378494	179738	
3	Arkansas	F	8	18.96	3038999	79841	
4	California	A	44	7.45	39937489	344642	
5	Colorado	C+	19	15.14	5,845,526	92435	
6	Connecticut	A-	45	4.91	3,563,077	82400	



	State	2019 Grade	Gun Death Rate	Gun Death Rate Per 100K	Pop	Guns	Gun Law Stren
0	Alabama	F	2	21.70	4908621	161641	
1	Alaska	F	7	20.74	734002	15824	
2	Arizona	F	18	15.29	7378494	179738	
3	Arkansas	F	8	18.96	3038999	79841	
4	California	A	44	7.45	39937489	344642	
5	Colorado	C+	19	15.14	5,845,526	92435	
6	Connecticut	A-	45	4.91	3,563,077	82400	

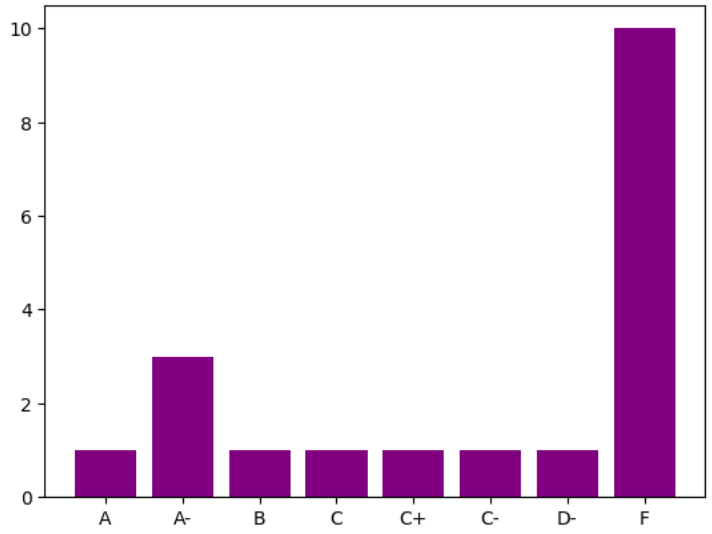
Focusing on visualization using matplotlib.pyplot



Today we will be focusing on creating a variety of basic visualization using matplotlib.pyplot

We start with a bar chart of the
2019 Gun Grades
Notice these are in column 1 data
(recall we start counting at 0)

To do : Create a value x to represent this column



The command

```
val, count = np.unique(x, return_counts=True)
```

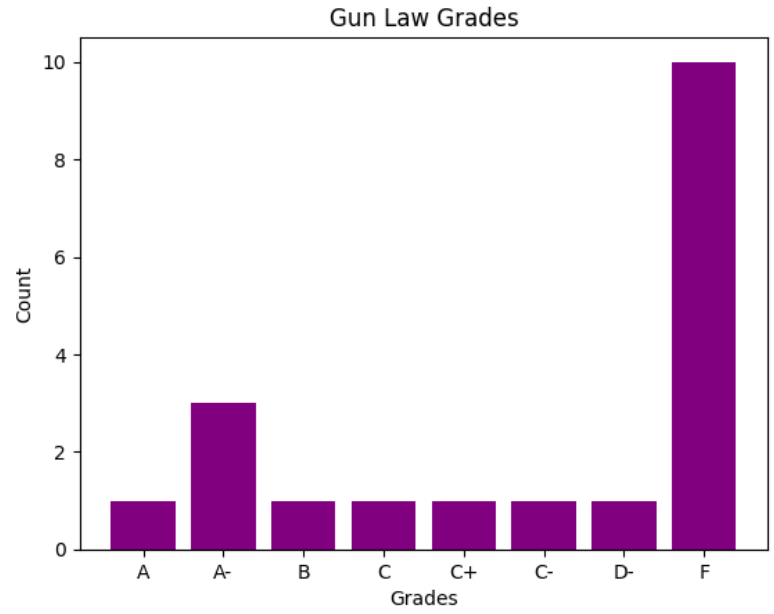
Creates a list of values and counts the unique number of each.

We will use plt.bar(val,count) to

plot the values versus the counts.

To Do: Create the bar graph on the left. Don't forget to finish with plt.show()

```
x=data[:,1]
val, count = np.unique(x, return_counts=True)
plt.bar(val,count,color='purple')
plt.title('Gun Law Grades')
plt.xlabel('Grades')
plt.ylabel('Count')
plt.show()
```



Add in a title, xaxis, and y axis label

If you wish to have horizontal bars instead use plt.barh()

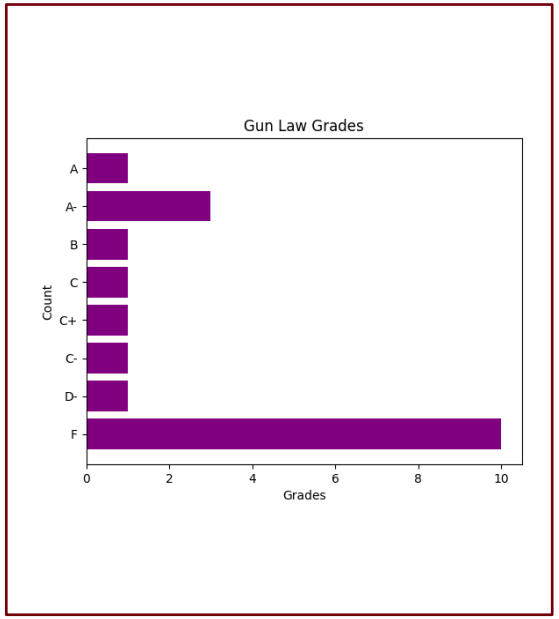
And use the command

plt.gca().invert_yaxis()

To invert the values on y axis.

To do: Create a horizontal bar graph with an inverted yaxis.

```
x=data[:,1]
val, count = np.unique(x, return_counts=True)
plt.barh(val,count,color='purple')
plt.title('Gun Law Grades')
plt.xlabel('Grades')
plt.ylabel('Count')
plt.gca().invert_yaxis()
plt.show()
```



Now let's try a pie chart with the same data.

If you don't want any labels you can just use

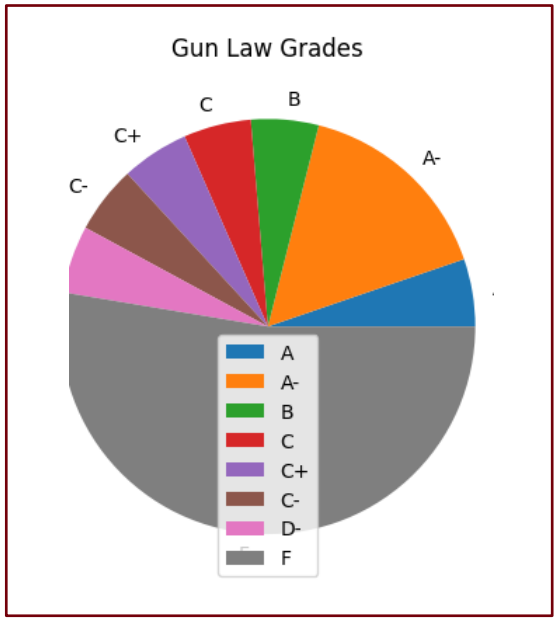
plt.pie(count)

You can add in some labels with

plt.pie(count,labels=val)

To do: Create a pie chart and include a title and legend with plt.legend()

```
plt.title('Gun Law Grades')
plt.pie(count,labels = val)
plt.legend()
plt.show()
```



Features you might be interested

(1) Changing the position of the legend

(2) Changing a color of a slice

(3) Adding percent labels

(4) Exploding a slice or changing the rotation angle

```
myexplode = [0, 0, 0, 0,0,0,0,.2]

plt.pie(count,labels = val,colors=plt.cm.Paired.colors,autopct='%.0f%%',startangle = 90,explode =myexplode,
textprops={'color':"w"})
```

Features you might be interested

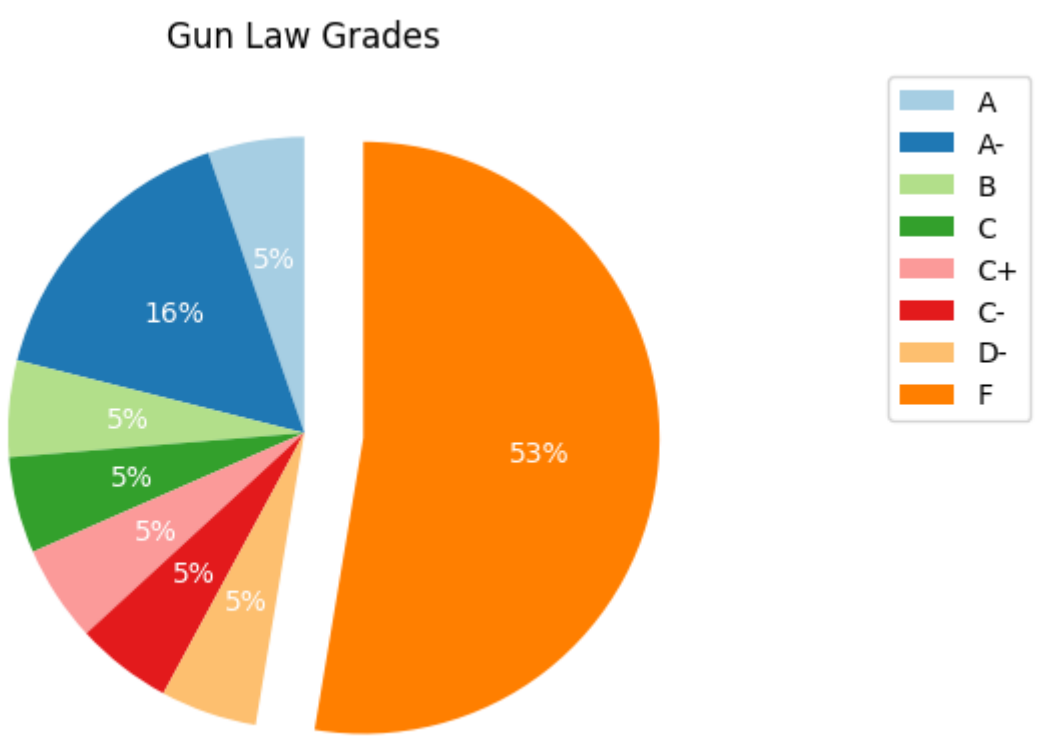
(1) Changing a color of a slice **(2) Adding percent labels** **(3) Changing the rotation angle** **(4) Exploding a slice**

Colors can be found at https://matplotlib.org/stable/users/explain/colors/colormaps.html

Or Changing the position of the legend

```
plt.legend(bbox_to_anchor=(1.5,1),loc="upper right")
```

To do: Create the pie chart below or be creative and update the feature to your preferences



For the second half of this session, we will focus on boxplots and histograms, both of which highlight quantitative data

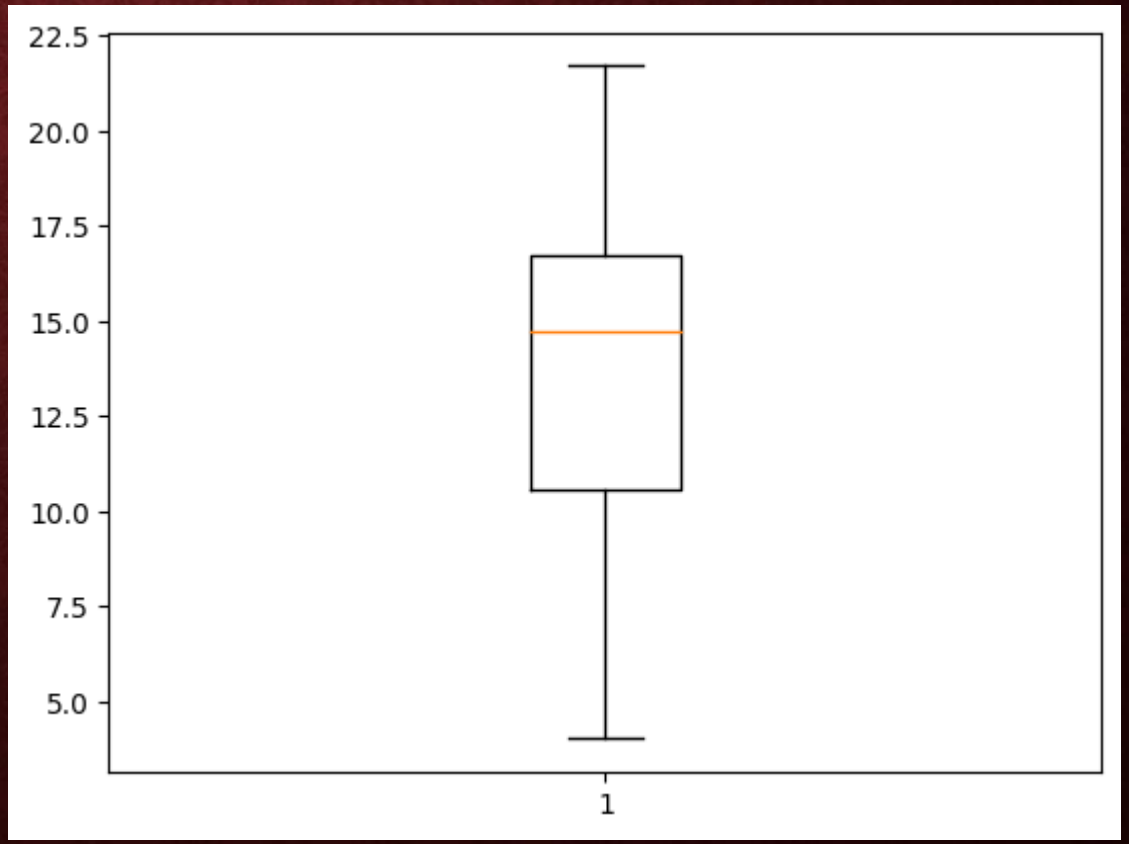
BOXPLOTS SHOW THE MAX, Q1, MEDIAN, Q3, AND MINIMUM AND ANY OUTLIERS

To fill the box
`patch_artist = True`

vert = 0 is horizontal

```
boxdata=data[:,3]
plt.boxplot(boxdata, vert = 1)
plt.show()
```

To do: Create a horizontal boxplot which is filled

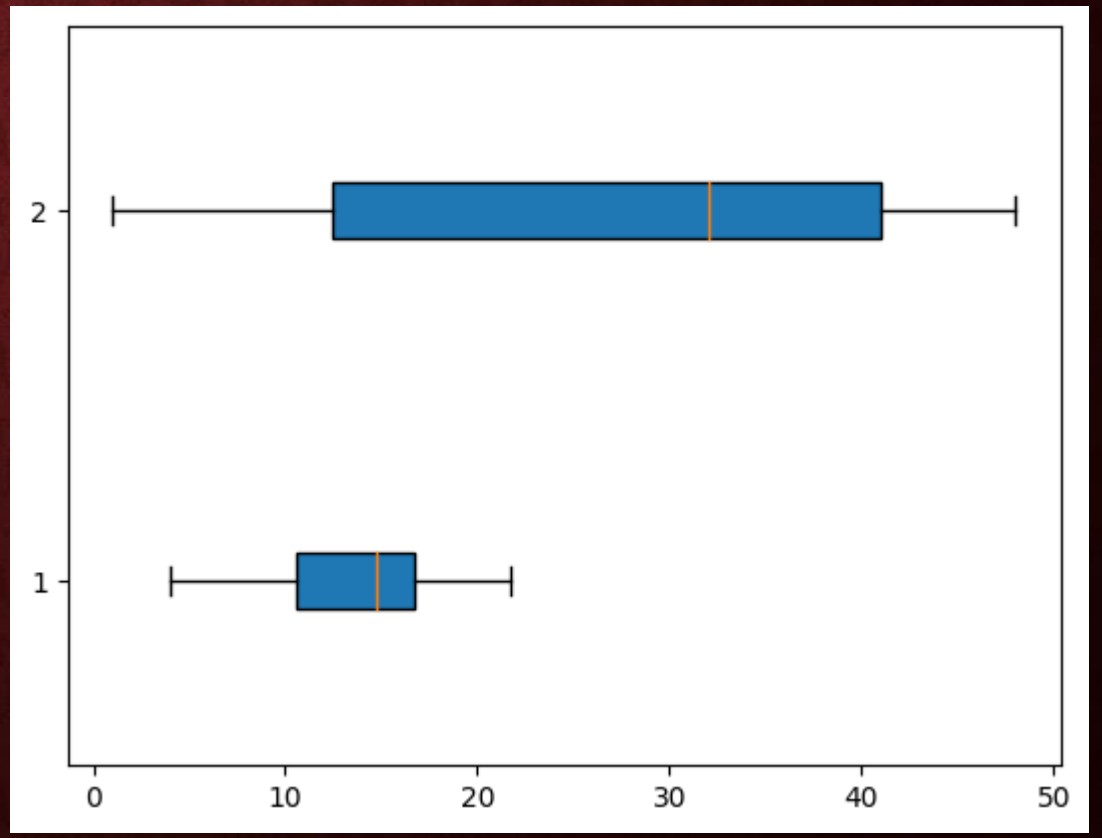


```
boxdata=[data[:,3],data[:,6]]
plt.boxplot(boxdata, patch_artist=True, vert = 0)
plt.show()
```

If you wish to compare data, you can add difference variables to the data list.

To change the colors of the box plots, first name your box plot, we have chosen the name bp.

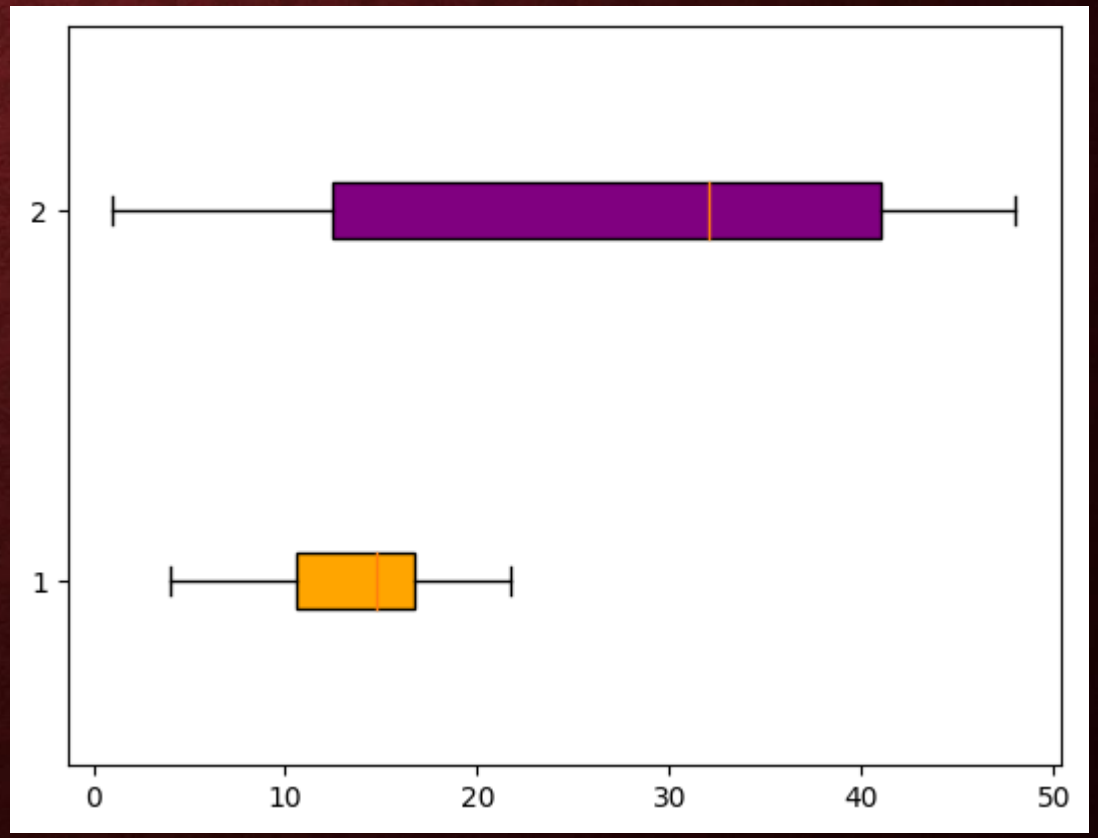
```
colors = ['orange', 'purple']

for patch, color in zip(bp['boxes'],
colors):
    patch.set_facecolor(color)
```



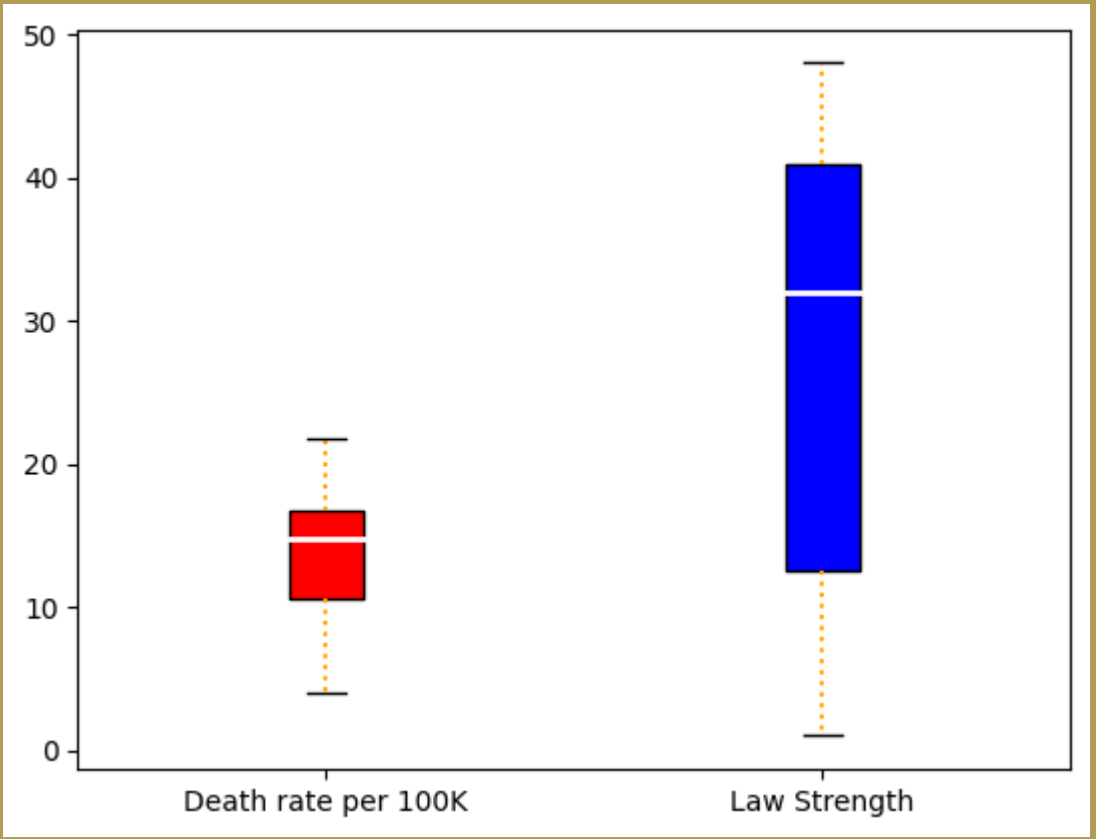
```
boxdata=[data[:,3],data[:,6]]

colors = ['orange', 'purple']

bp=plt.boxplot(boxdata, patch_artist=True, vert = 0)
for patch, color in zip(bp['boxes'], colors):
    patch.set_facecolor(color)

plt.show()
```

To change the colors of the medians and whiskers

```
# changing color and linewidth of
# whiskers
for whisker in bp['whiskers']:
    whisker.set(color ='blue',
                linewidth = 1.5,
                linestyle =":")

# changing color and linewidth of
# medians
for median in bp['medians']:
    median.set(color ='white',
               linewidth = 2)
```



To do: Create this box and whisker plot



Creating labels

```
plt.xticks([1, 2], ['Death rate per 100K', 'Law Strength'])
```

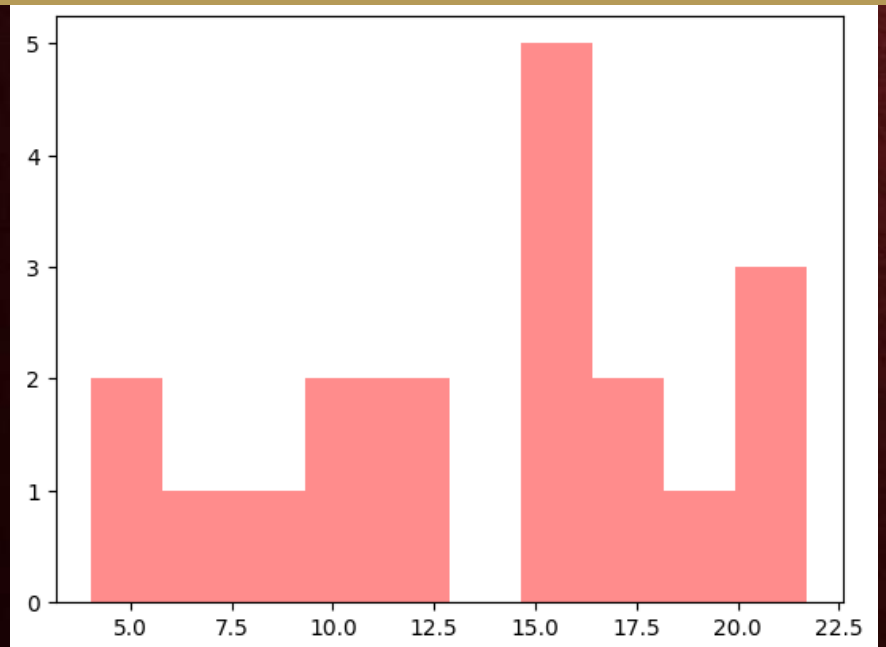
In order to create a histogram use

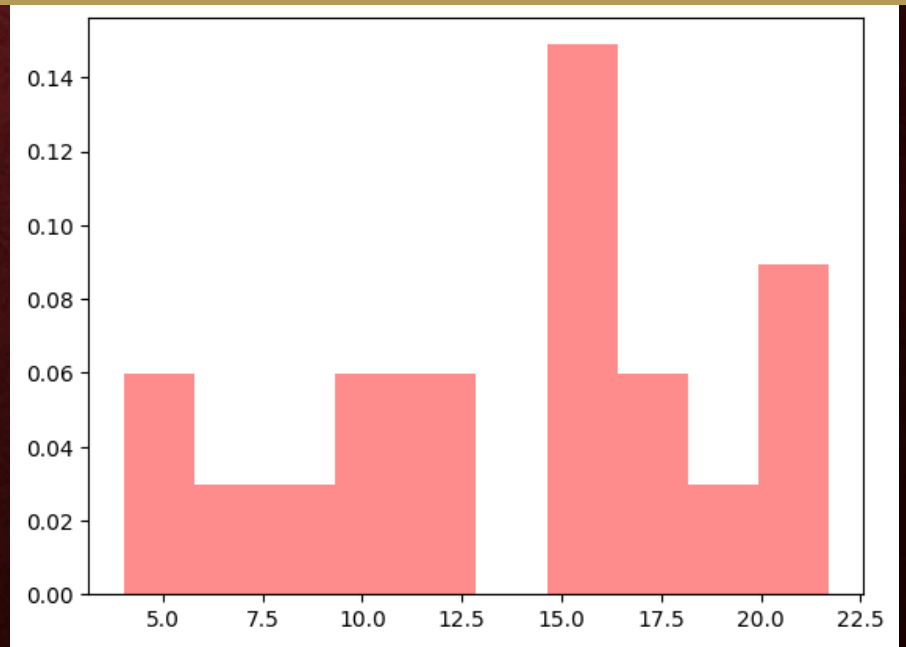
```
plt.hist(data[:,3],bins=5,histtype='bar',alpha=.45,color='red')
plt.show()
```

Change the bin size

Make this a probability density history by including density=True

To do: Create these histograms with data[:,3]





We can also create side-by side histograms by adding lists for each category.

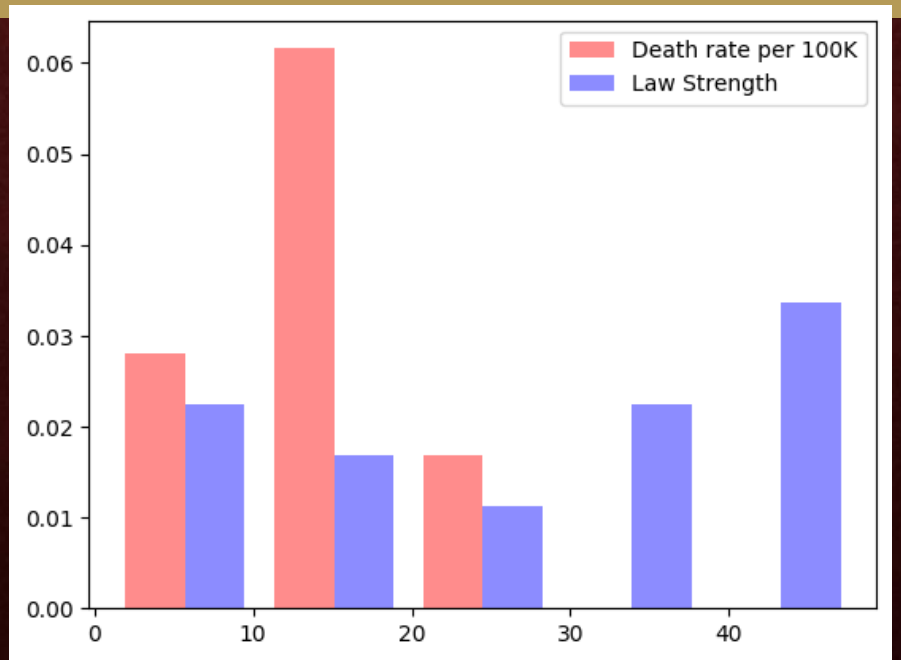
```
plt.hist(data[:,3],bins=5,histtype='bar',alpha=.45,color='red')
plt.show()
```

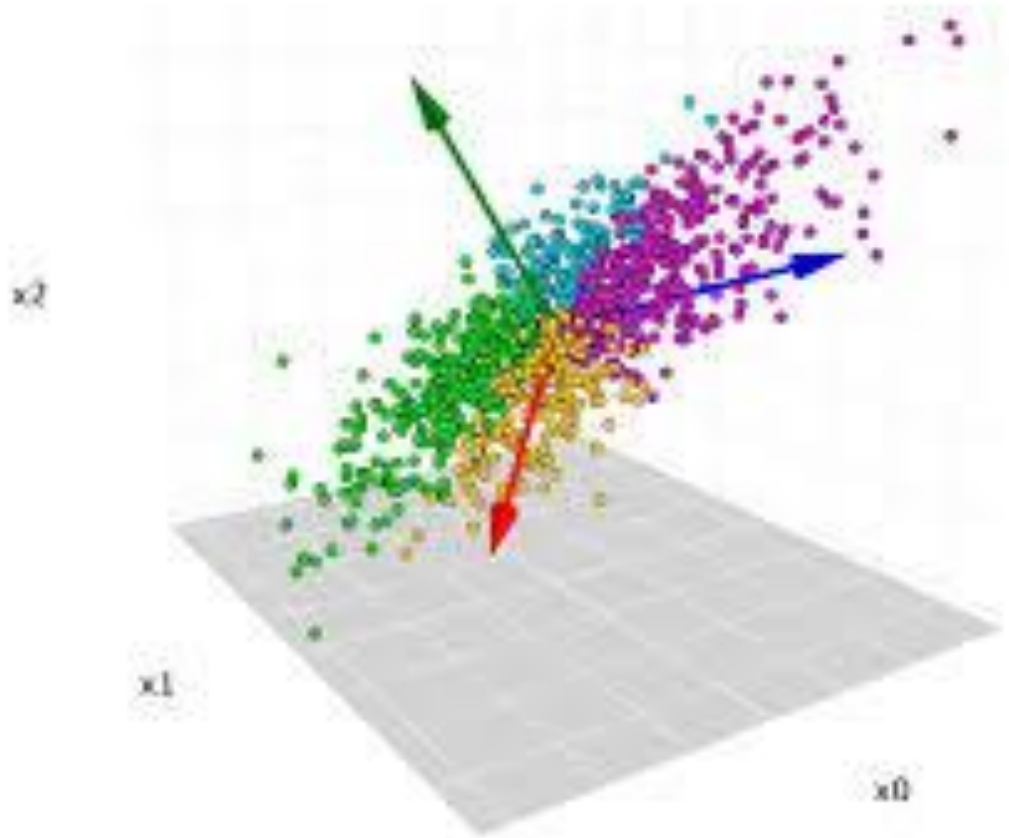
List of data such as [data:,3],data[:,6]]

List of colors such as color=['red','blue']

Include a list of labels such as label=['x',y']

To do: Create the following histogram with data[:,3] and data[:,6]. Don't forget to include a plt.legend





WATCH THE 4TH PYTHON VIDEO TO LEARN HOW USE CLUSTERING TECHNIQUES TO VISUALIZE DATA