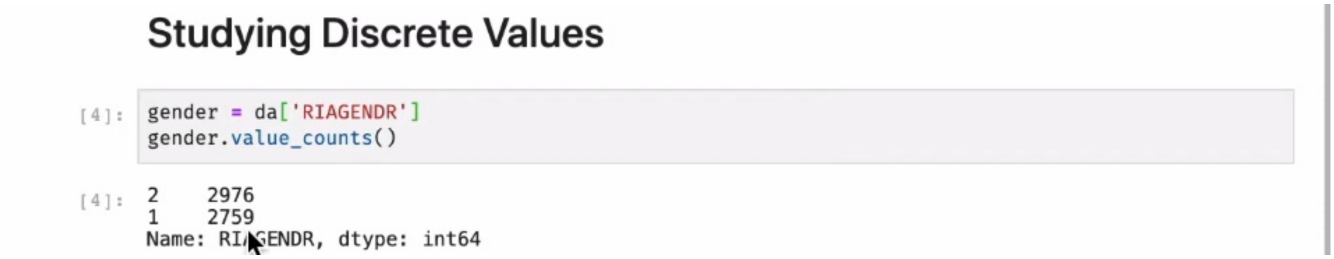
LECTURE 4

Analysis on the Nhanes dataset
in gender column the values are in discrete form 2 for male 1 for female

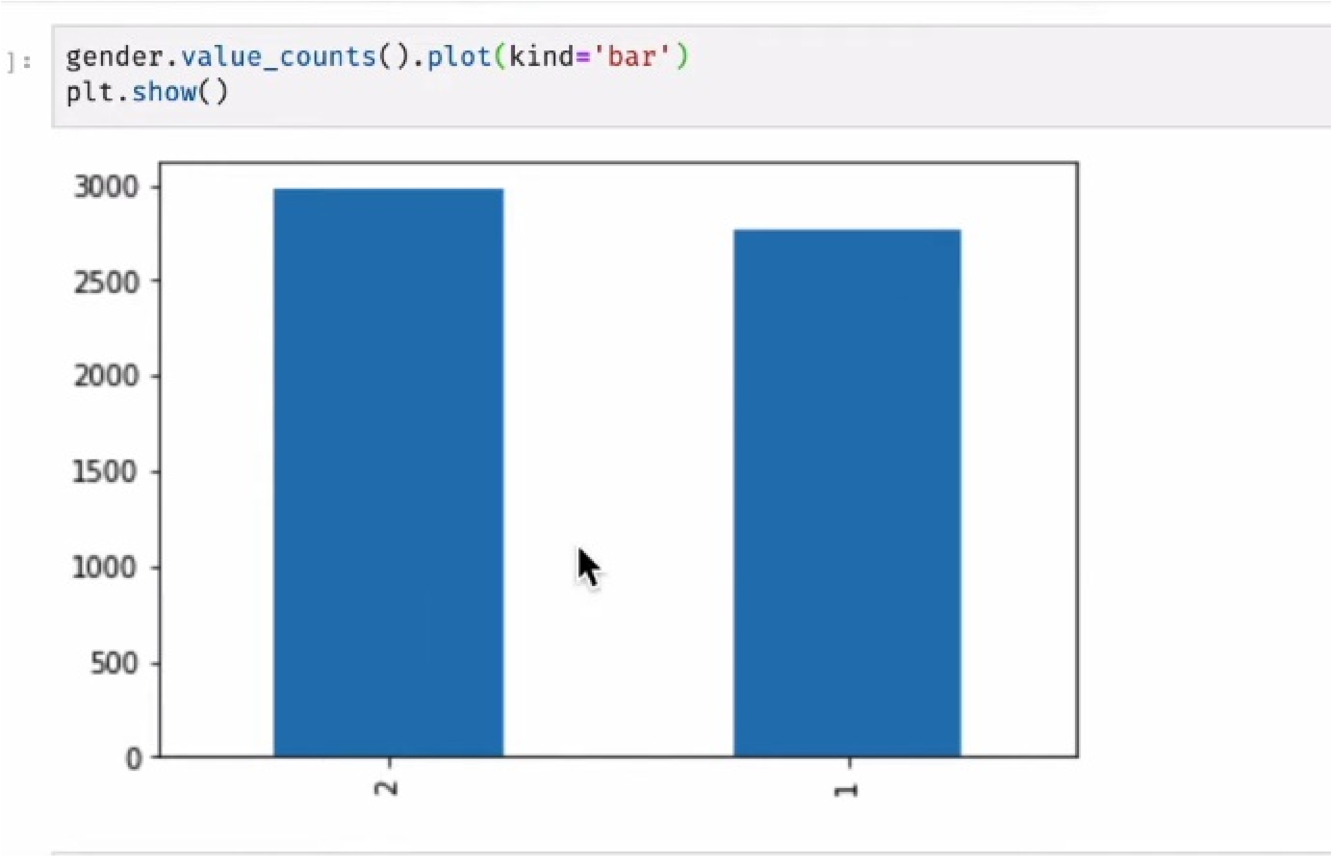


Studying Discrete Values

```
[4]: gender = da['RIAGENDR']
     gender.value_counts()
```

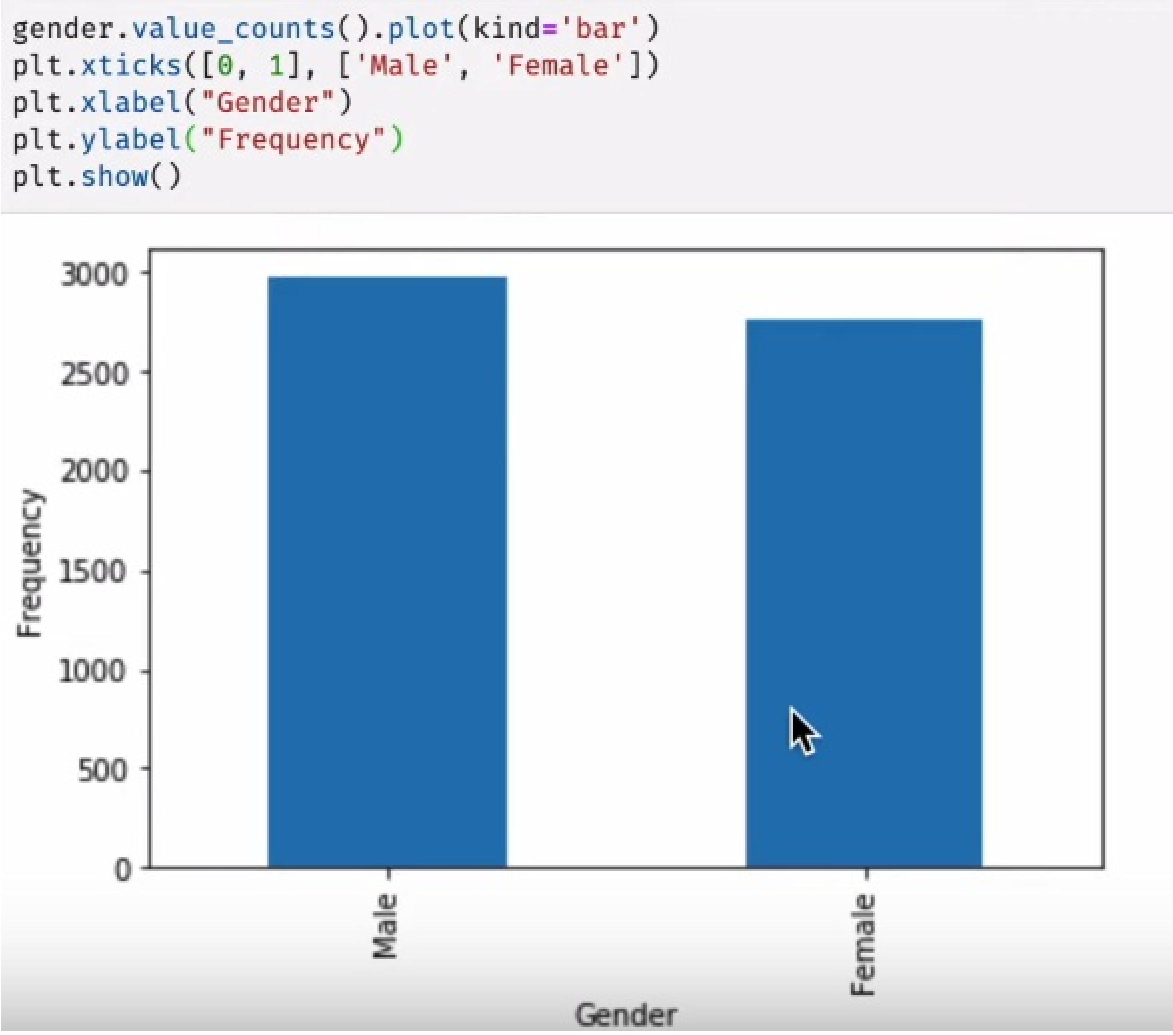
```
[4]: 2    2976
     1    2759
     Name: RIAGENDR, dtype: int64
```

plot the bar graph for better understanding



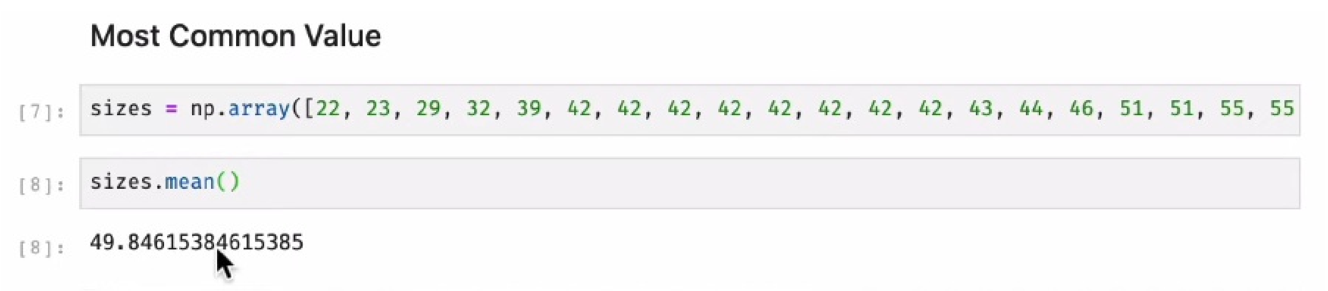
```
]: gender.value_counts().plot(kind='bar')
   plt.show()
```

Now there are some issues with the above graph if u represent your graph no one will understand from it by looking at it (as only u know that 1 represent male and 2 female) also x and y axis are not labeled.



```
gender.value_counts().plot(kind='bar')
plt.xticks([0, 1], ['Male', 'Female'])
plt.xlabel("Gender")
plt.ylabel("Frequency")
plt.show()
```

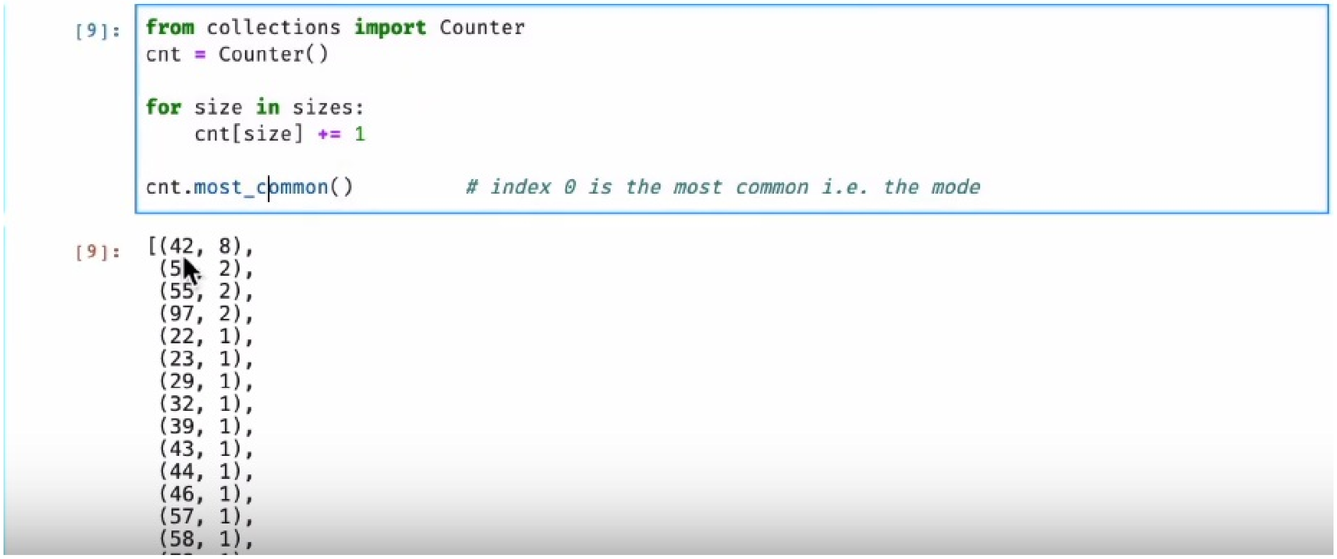
Suppose the array below represents the sizes of the T-shirts. Now, if someone wants to determine the most commonly sold T-shirt size, using the mean is not a good approach. The mean is 49, but the most sold size is 42



Most Common Value

```
[7]: sizes = np.array([22, 23, 29, 32, 39, 42, 42, 42, 42, 42, 42, 42, 42, 43, 44, 46, 51, 51, 55, 55

[8]: sizes.mean()

[8]: 49.84615384615385
```

So we need to find the most common value for this we use **Counter Method .**



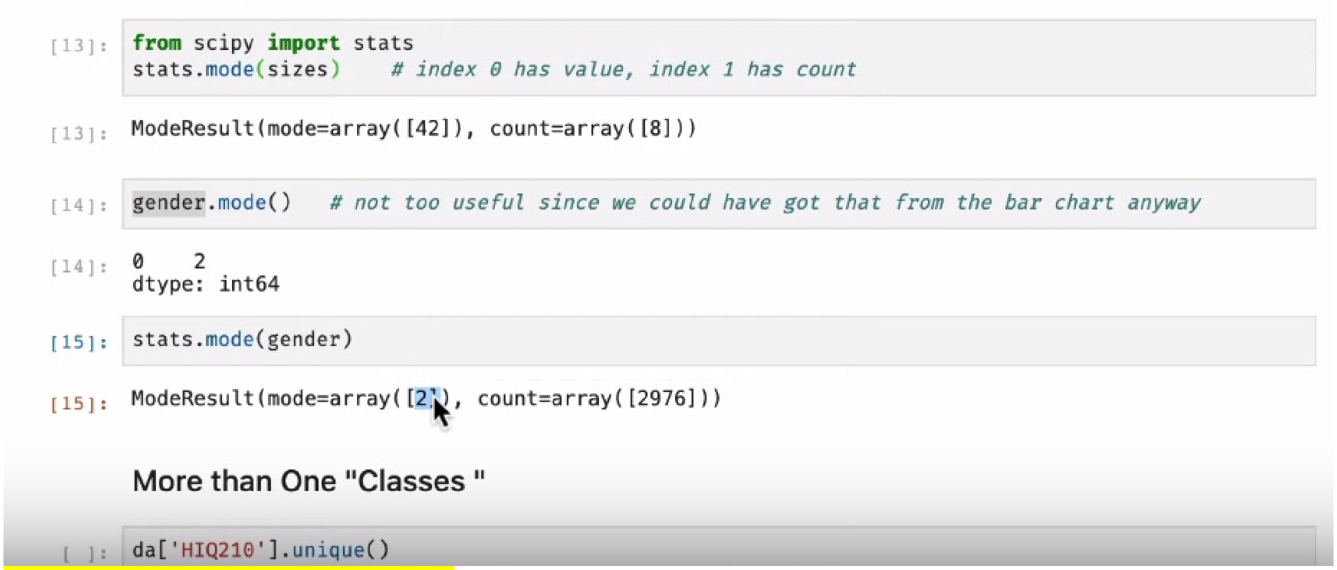
```
[9]: from collections import Counter
     cnt = Counter()

     for size in sizes:
         cnt[size] += 1

     cnt.most_common()          # index 0 is the most common i.e. the mode

[9]: [(42, 8),
      (5[illegible], 2),
      (55, 2),
      (97, 2),
      (22, 1),
      (23, 1),
      (29, 1),
      (32, 1),
      (39, 1),
      (43, 1),
      (44, 1),
      (46, 1),
      (57, 1),
      (58, 1),
```

and in the most_common the value which is most occuring will appear on the zero index .U can do the same thing with dictionary but it is the easier than that.

Now there is another approach to do this.

Using scipy.

Now mode is the most common value in the data and avg is the arithmetic mean(you can calculate mode without the scipy.

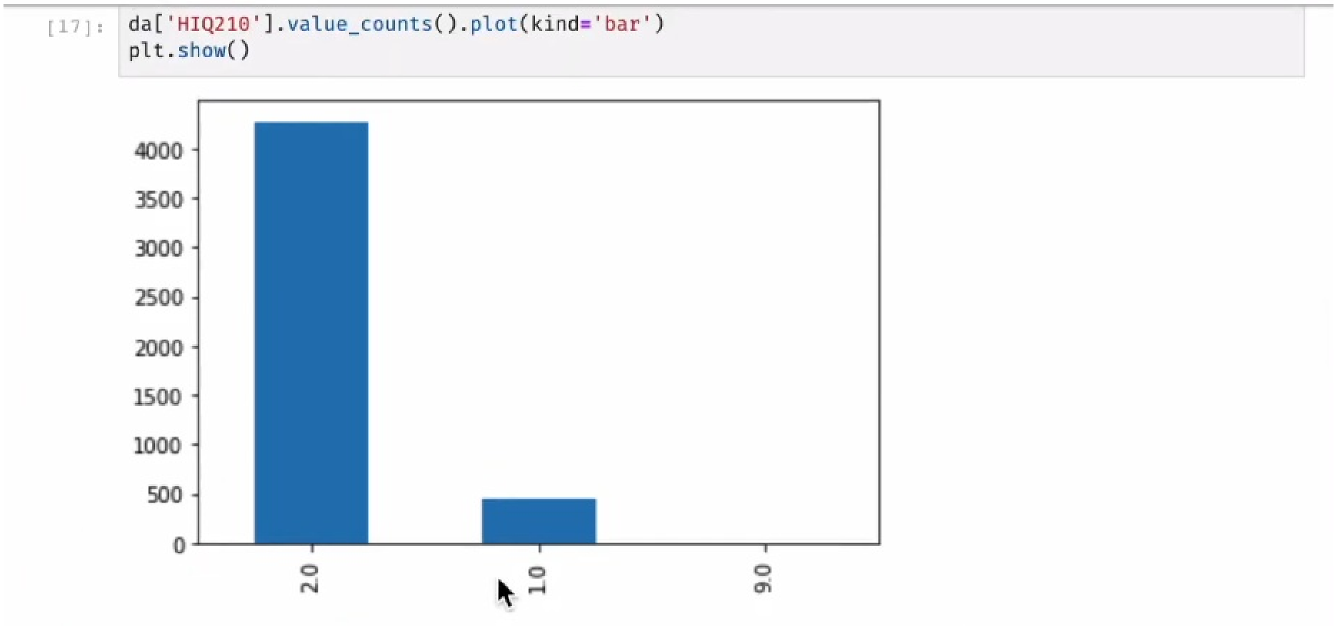


```
[13]: from scipy import stats
      stats.mode(sizes)    # index 0 has value, index 1 has count

[13]: ModeResult(mode=array([42]), count=array([8]))

[14]: gender.mode()   # not too useful since we could have got that from the bar chart anyway

[14]: 0    2
      dtype: int64

[15]: stats.mode(gender)

[15]: ModeResult(mode=array([2]), count=array([2976]))
```

More than One "Classes "

```
[ ]: da['HIQ210'].unique()
```

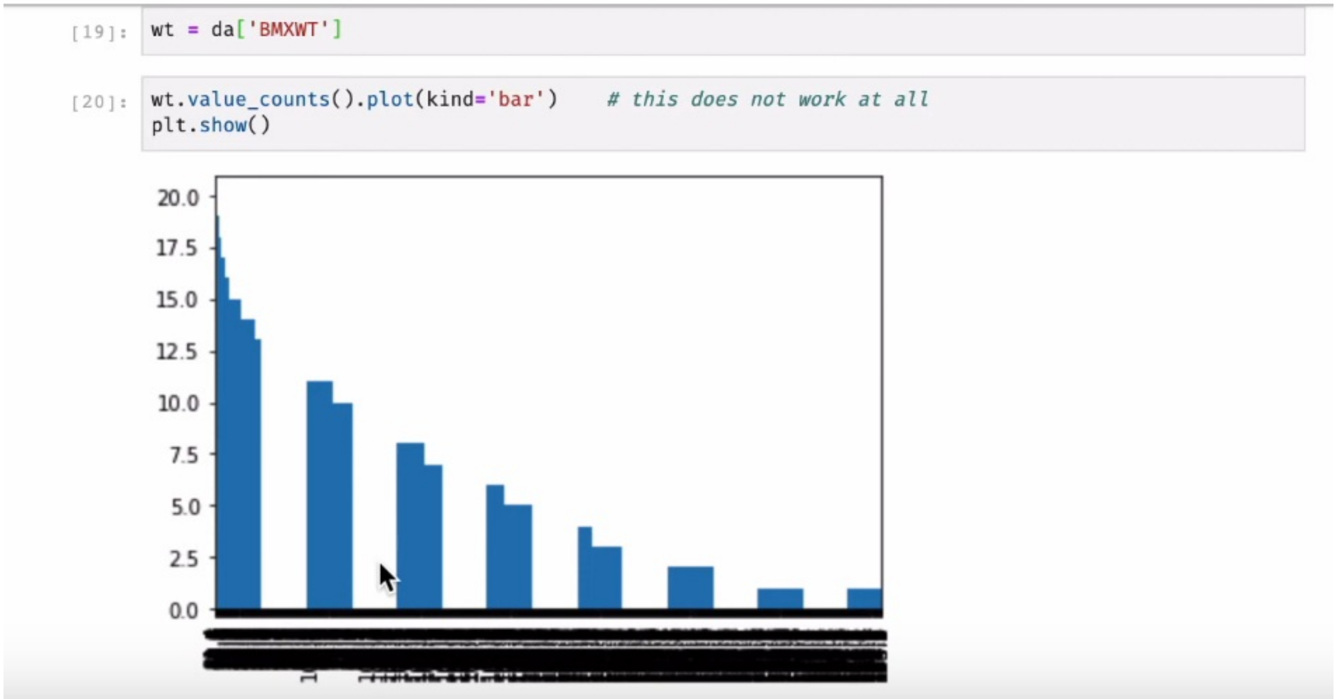
FOR MORE THAN ONE VALUE



```
[17]: da['HIQ210'].value_counts().plot(kind='bar')
      plt.show()
```

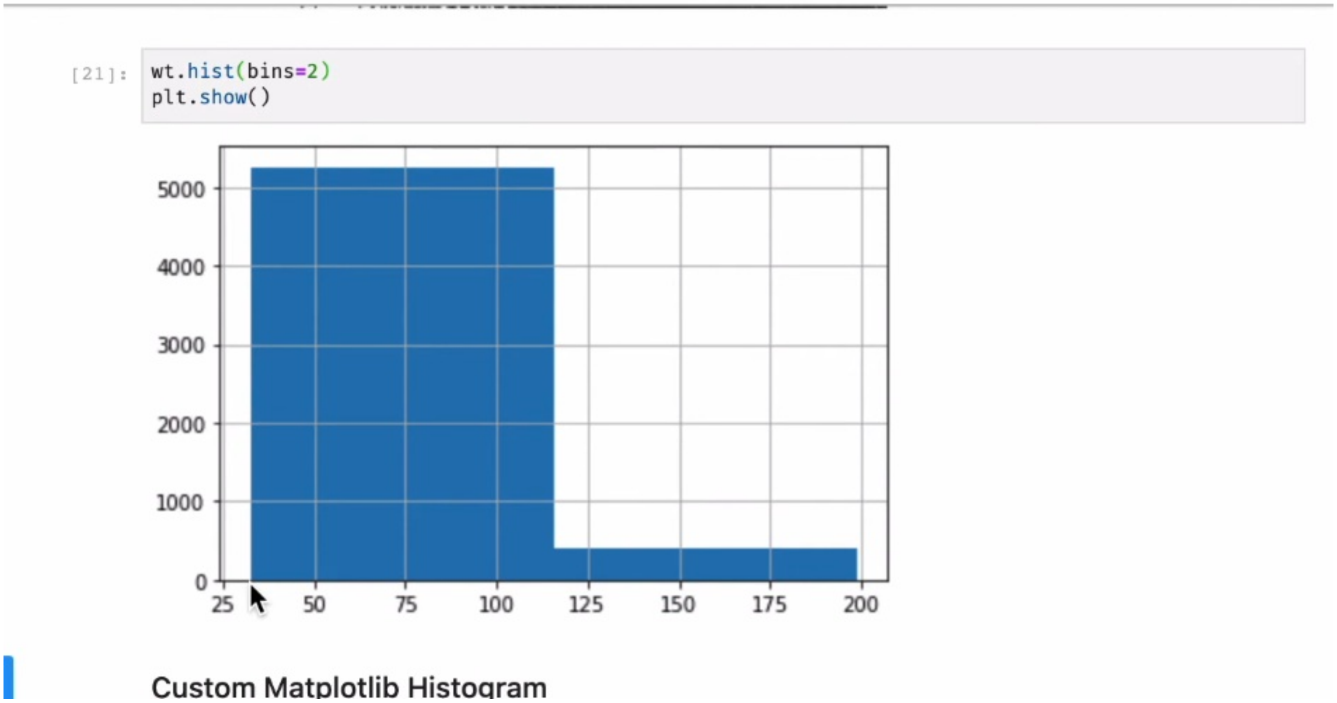
AS SHOWN 2 IS THE MOST COMMON
U CAN USE PIE CHART BUT IT IS NOT **RECOMMENDED**

REAL VALUES:
SUPPOSE CONSIDER WEIGHT :AS EACH PERSON HAS DIFFERENT WEIGHTS SO FOR THIS FREQUENCY IS NOT A GOOD APPROACH.



```
[19]: wt = da['BMXWT']

[20]: wt.value_counts().plot(kind='bar')     # this does not work at all
      plt.show()
```

SO FOR THIS HISTOGRAMS ARE BEST DIVIDE THE DATA INTO BINS BUT THERE IS A PROBLEM

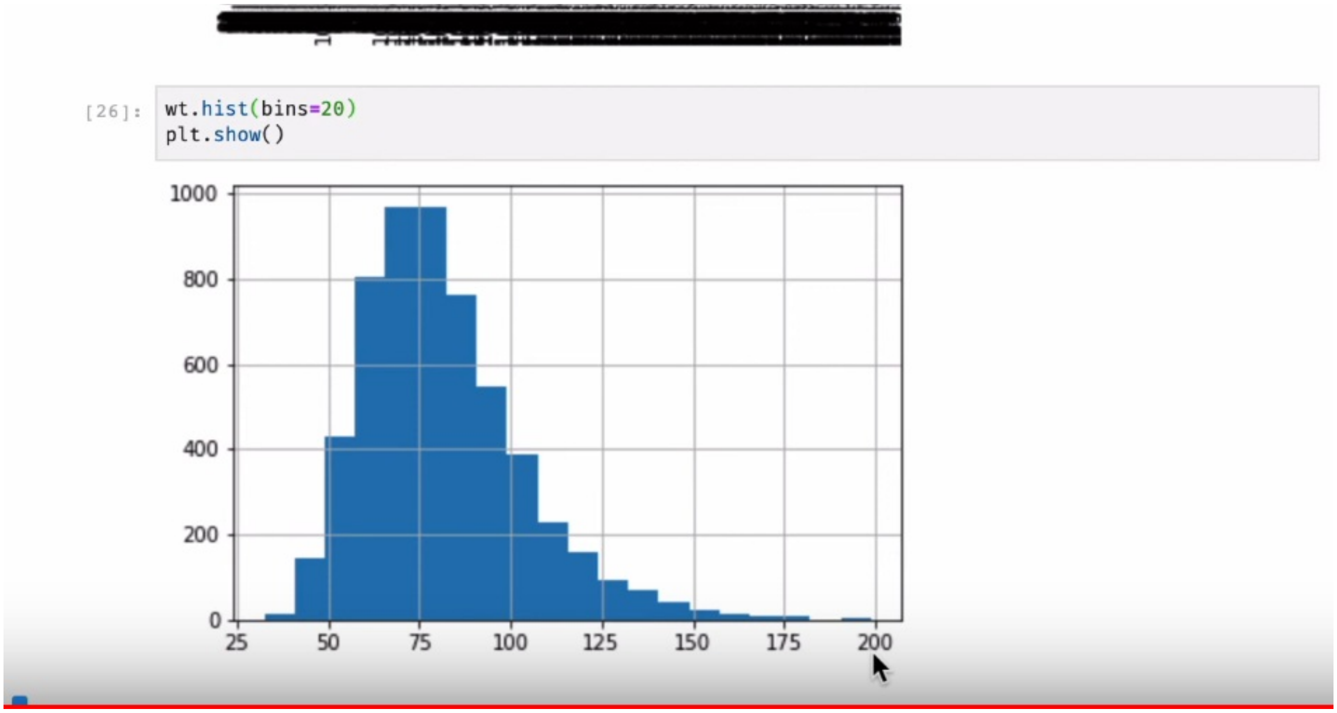


```
[21]: wt.hist(bins=2)
      plt.show()
```

Custom Matplotlib Histogram

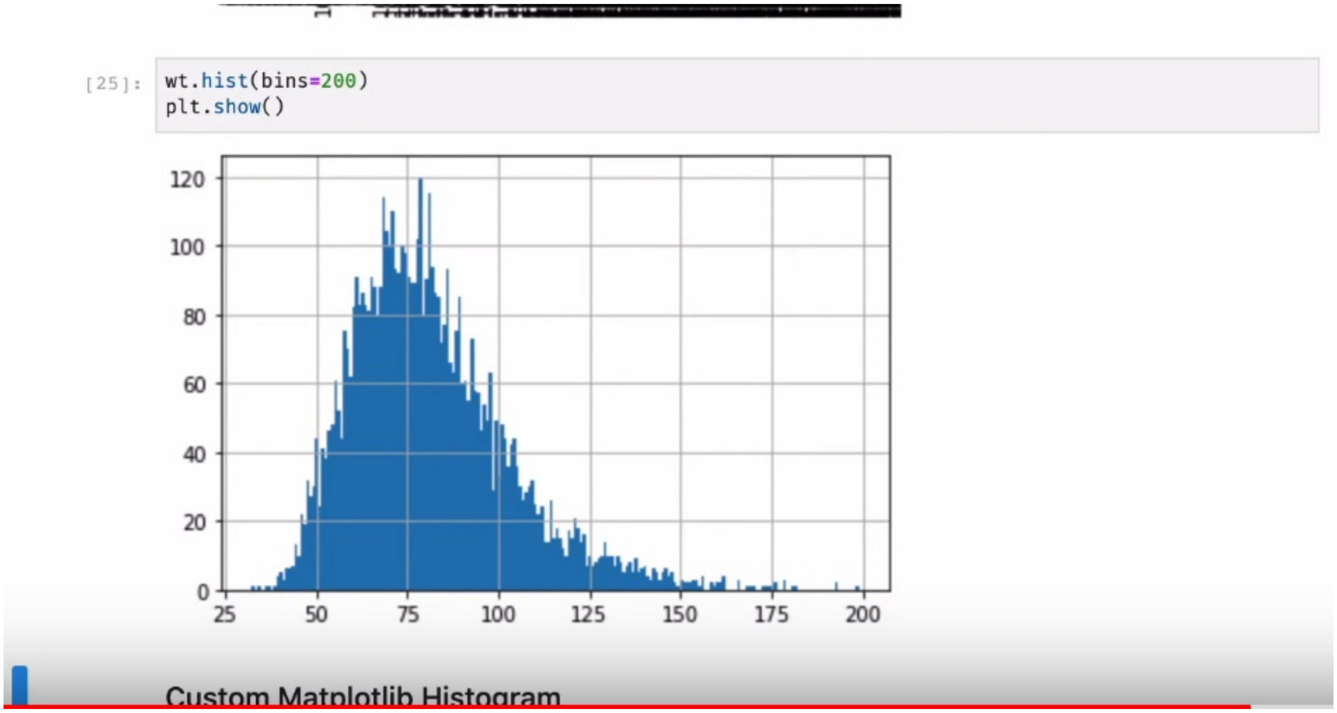
AS SEE U LOSS THE SPECIFIC INFORMATION SO
FOR THIS INCREASE THE BIN NUMBERS

[redacted]



```
[26]: wt.hist(bins=20)
      plt.show()
```

BUT IF THE BINS ARE MORE THEN IT IS DIFFICULT TO UNDERSTAND IT.

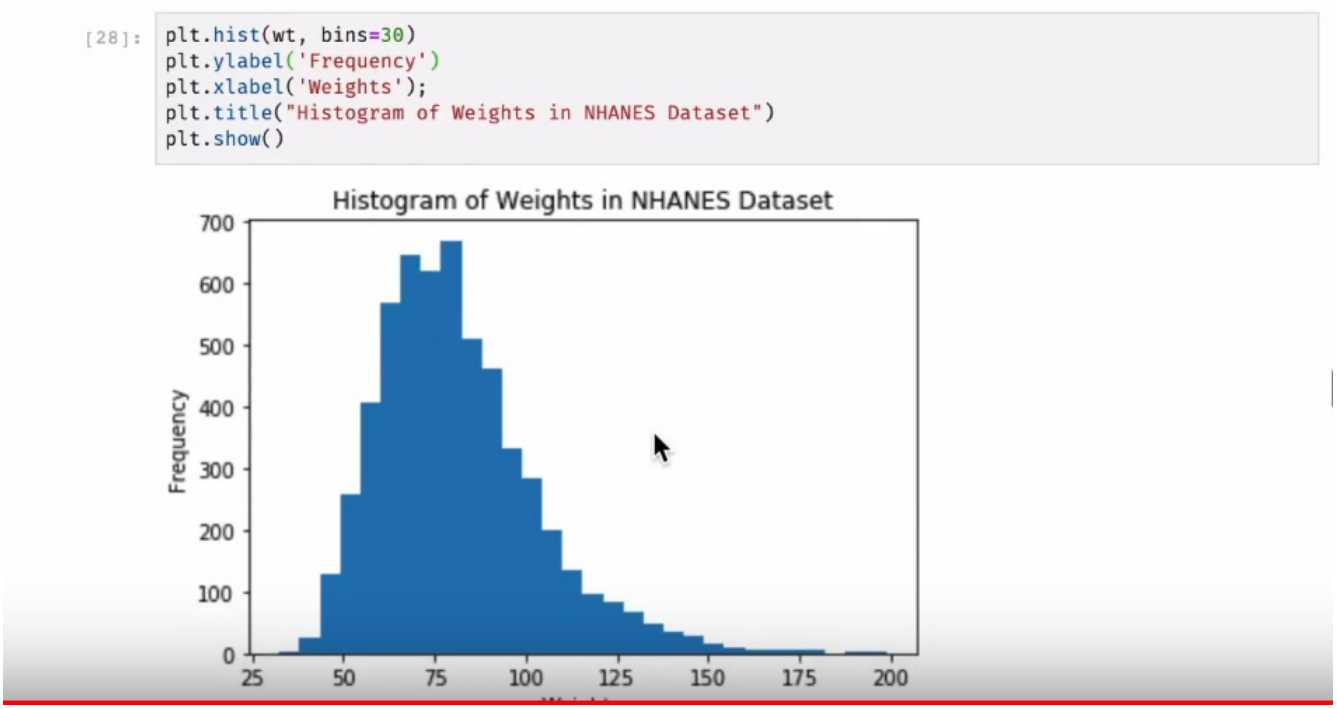


```
[25]: wt.hist(bins=200)
      plt.show()
```

Custom Matplotlib Histogram

SO ADJUST THE NO OF BINS ACCORDING TO YOUR NEED

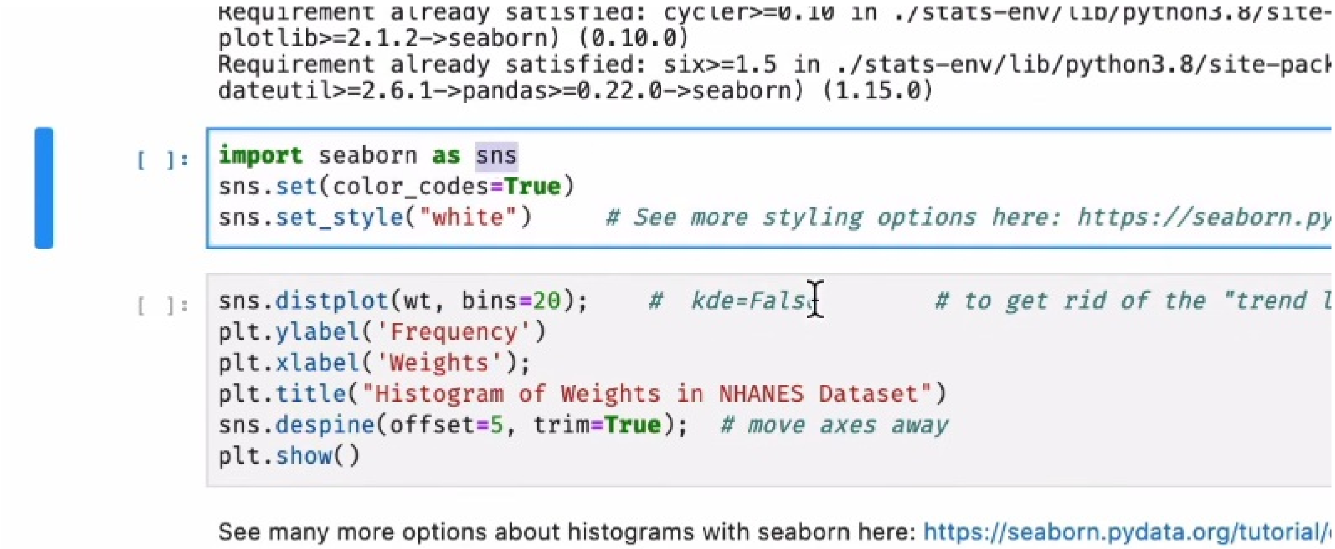
USING MATPLOTLIB



```
[28]: plt.hist(wt, bins=30)
      plt.ylabel('Frequency')
      plt.xlabel('Weights');
      plt.title("Histogram of Weights in NHANES Dataset")
      plt.show()
```

USING SEABORN

IN SEABORN HISTOGRAMS ARE CALLED DISTPLOT



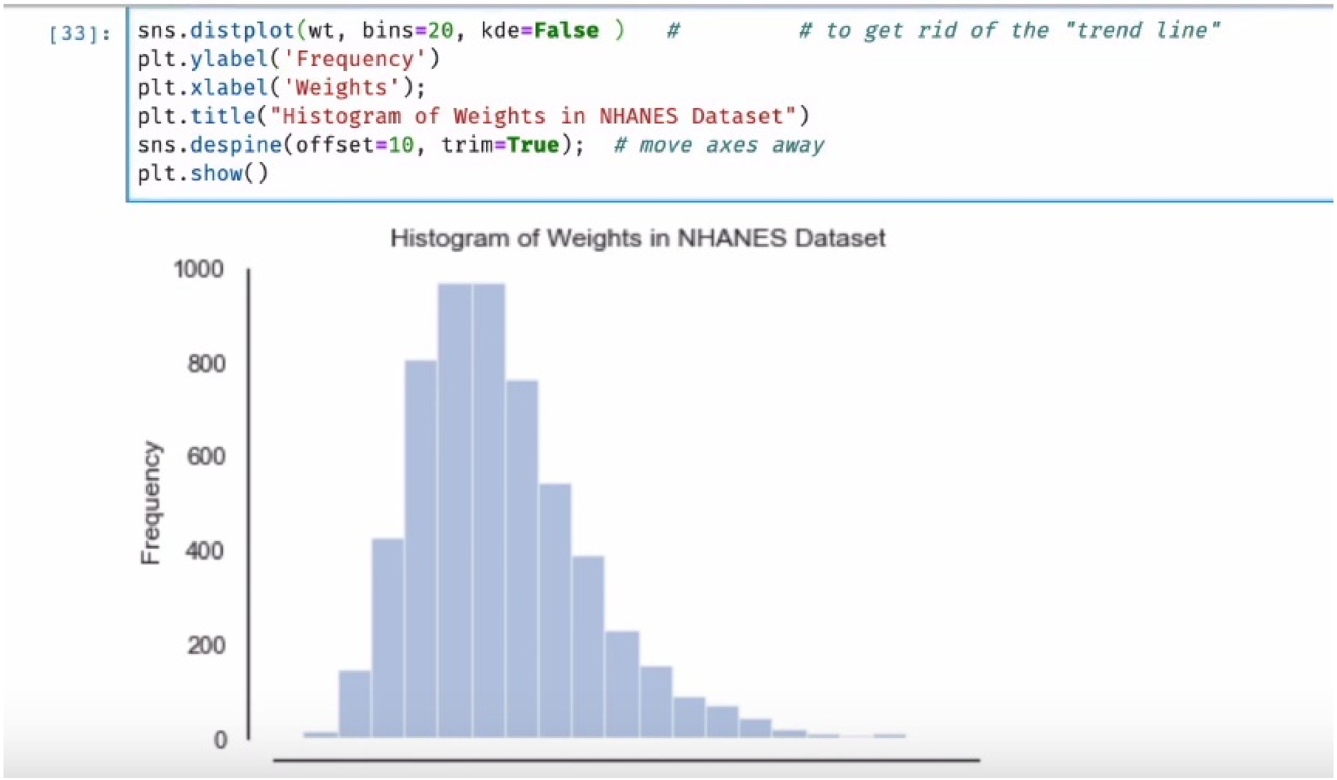
```
Requirement already satisfied: cycler>=0.10 in ./stats-env/lib/python3.8/site-
plotlib>=2.1.2->seaborn) (0.10.0)
Requirement already satisfied: six>=1.5 in ./stats-env/lib/python3.8/site-pack
dateutil>=2.6.1->pandas>=0.22.0->seaborn) (1.15.0)
```

```
[ ]: import seaborn as sns
     sns.set(color_codes=True)
     sns.set_style("white")     # See more styling options here: https://seaborn.py
```

```
[ ]: sns.distplot(wt, bins=20);    #  kde=False        # to get rid of the "trend l
     plt.ylabel('Frequency')
     plt.xlabel('Weights');
     plt.title("Histogram of Weights in NHANES Dataset")
     sns.despine(offset=5, trim=True);  # move axes away
     plt.show()
```

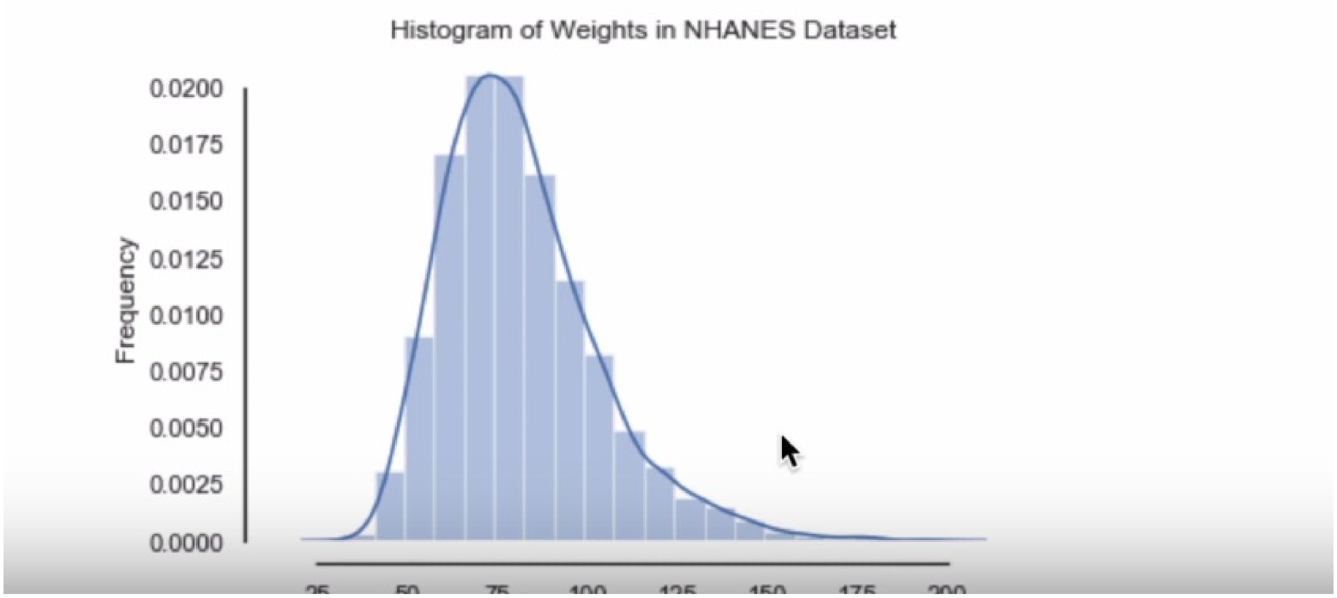
See many more options about histograms with seaborn here: https://seaborn.pydata.org/tutorial/

DESPINE IS USED TO SEPERATE X-AXIS FROM Y-AXIS



```
[33]: sns.distplot(wt, bins=20, kde=False )   #        # to get rid of the "trend line"
      plt.ylabel('Frequency')
      plt.xlabel('Weights');
      plt.title("Histogram of Weights in NHANES Dataset")
      sns.despine(offset=10, trim=True);  # move axes away
      plt.show()
```

KDE IS TRUE :



WHERE ARE MOST OF THE PEOPLE THIS IDEA IS CALLED CENTRAL TENDENCY HOW MANY VALUES ARE NEAR THE CENTER.