Week 3 Assignment
Kevin Alessandro Bautista
00314-45064

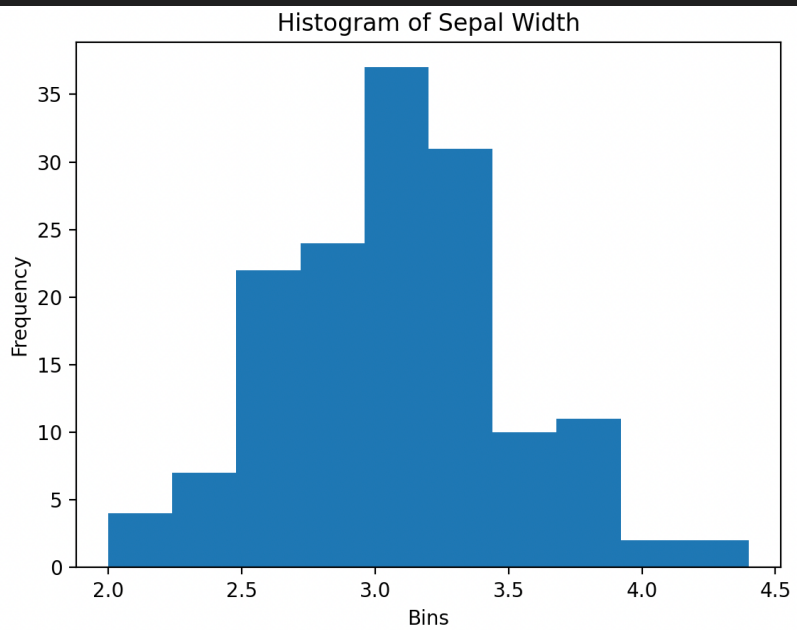
. Using the iris dataset...

```
from sklearn import datasets
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np

data = { "weight": [4.17, 5.58, 5.18, 6.11, 4.50, 4.61, 5.17, 4.53, 5.33, 5.14, 4.81,
4.17, 4.41, 3.59, 5.87, 3.83, 6.03, 4.89, 4.32, 4.69, 6.31, 5.12, 5.54, 5.50, 5.37,
5.29, 4.92, 6.15, 5.80, 5.26], "group": ["ctrl"] * 10 + ["trt1"] * 10 + ["trt2"] * 10}
PlantGrowth = pd.DataFrame(data)
iris = datasets.load_iris()
iris_df = pd.DataFrame(data=iris.data, columns=iris.feature_names)
iris_target = pd.DataFrame(data=iris.target, columns=['Species/target'])
```

a. Make a histogram of the variable Sepal.Width.

```
# Q1a. Histogram of Sepal Width
plt.hist(iris_df['sepal width (cm)'])
plt.xlabel('Bins')
plt.ylabel('Frequency')
plt.title('Histogram of Sepal Width')
plt.show()
```



b. Based on the histogram from #1a, which would you expect to be higher, the mean or the median? Why?

It is a relatively right-skewed histogram, so I'd expect the mean to be greater than the median.

c. Confirm your answer to #1b by actually finding these values.

```
#Q1c.
sepal_width = iris_df['sepal width (cm)']
sepal_width_mean = sepal_width.mean()
sepal_width_std = sepal_width.median()
print(f"Mean of Sepal Width: {sepal_width_mean}")
print(f"Median of Sepal Width: {sepal_width_std}")
```

Mean: 3.05733
Median: 3.0
So it was still right-skewed, but we can argue that it is about the same given how close these two values are.

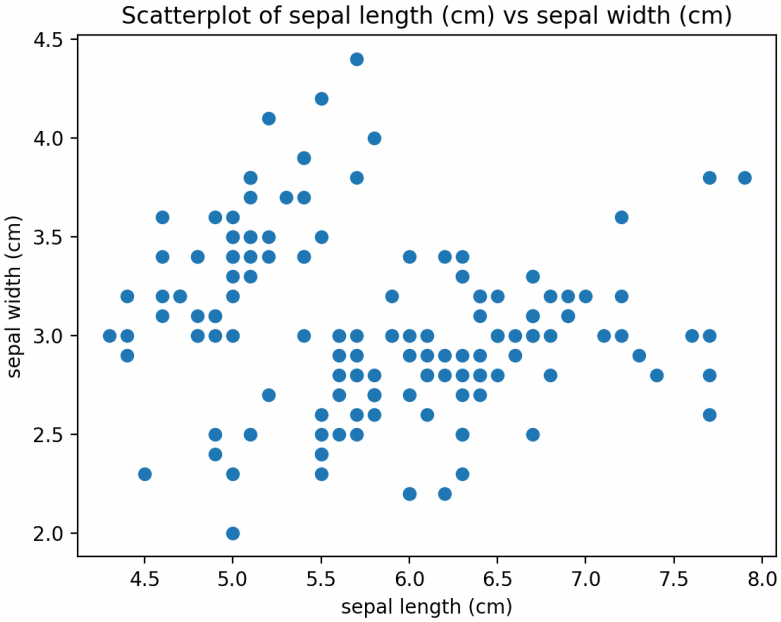
d. Only 27% of the flowers have a Sepal.Width higher than _____ cm.

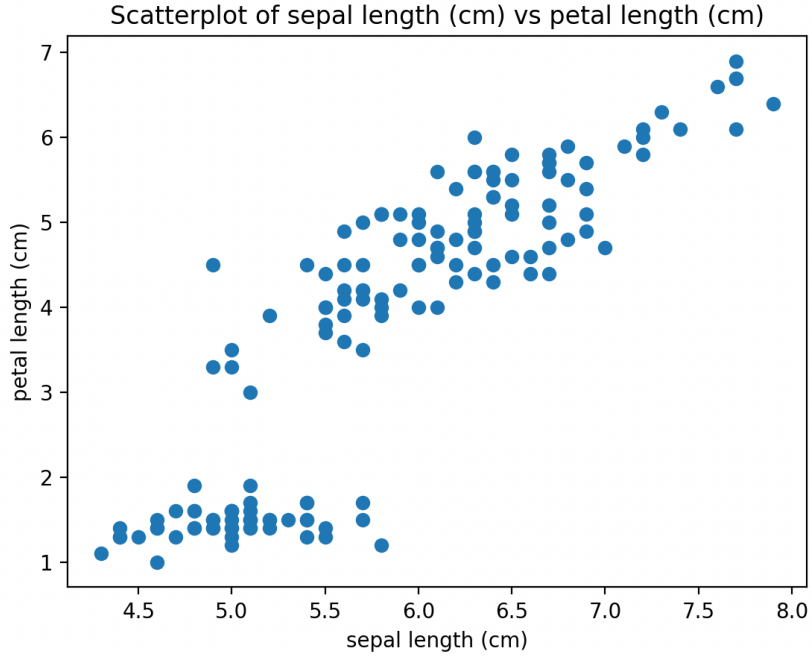
```
sepal_width_73 = sepal_width.quantile(0.73)
print(f"73rd Percentile of Sepal Width: {sepal_width_73}")
```

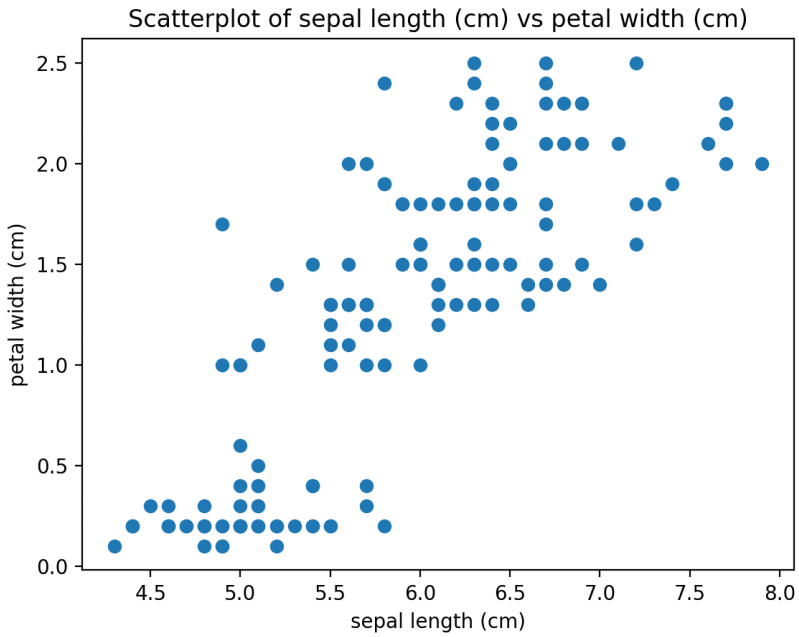
73rd percentile: 3.3 cm

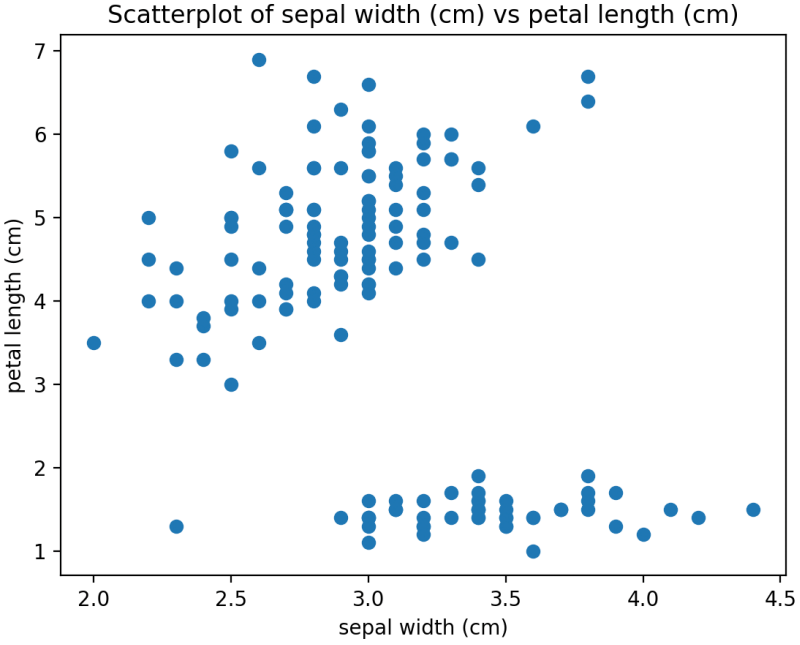
e. Make scatterplots of each pair of the numerical variables in iris (There should be 6 pairs/plots).

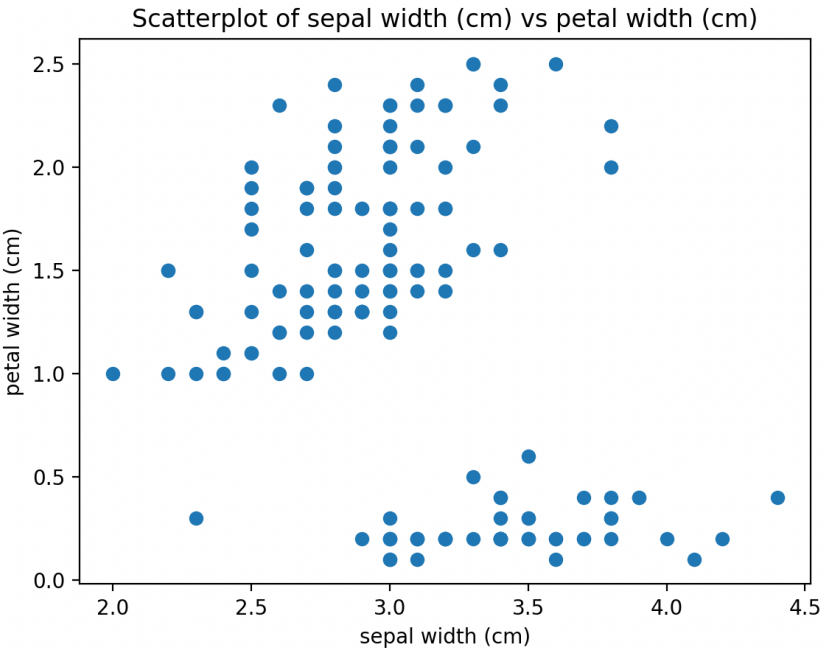
```
#Q1e. SNS Pairplot of Iris Dataset
iris_df_columns = ['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)',
'petal width (cm)']
for i in range(len(iris_df_columns)):
    for j in range(i+1, len(iris_df_columns)):
        plt.scatter(data=iris_df, x=iris_df_columns[i], y=iris_df_columns[j])
        plt.title(f'Scatterplot of {iris_df_columns[i]} vs {iris_df_columns[j]}')
        plt.xlabel(iris_df_columns[i])
        plt.ylabel(iris_df_columns[j])
        plt.show()
```

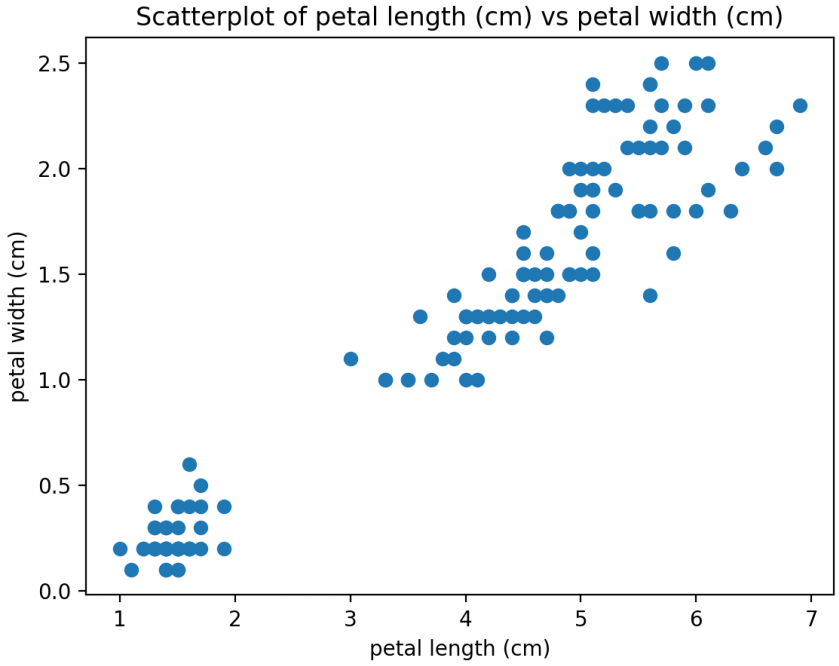

Scatterplot of sepal length (cm) vs sepal width (cm)
sepal width (cm)
sepal length (cm)
4.5
4.0
3.5
3.0
2.5
2.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0


Scatterplot of sepal length (cm) vs petal length (cm)
petal length (cm)
sepal length (cm)
7
6
5
4
3
2
1
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0


Scatterplot of sepal length (cm) vs petal width (cm)
petal width (cm)
sepal length (cm)
0.0
0.5
1.0
1.5
2.0
2.5
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0


Scatterplot of sepal width (cm) vs petal length (cm)
petal length (cm)
sepal width (cm)
1
2
3
4
5
6
7
2.0
2.5
3.0
3.5
4.0
4.5





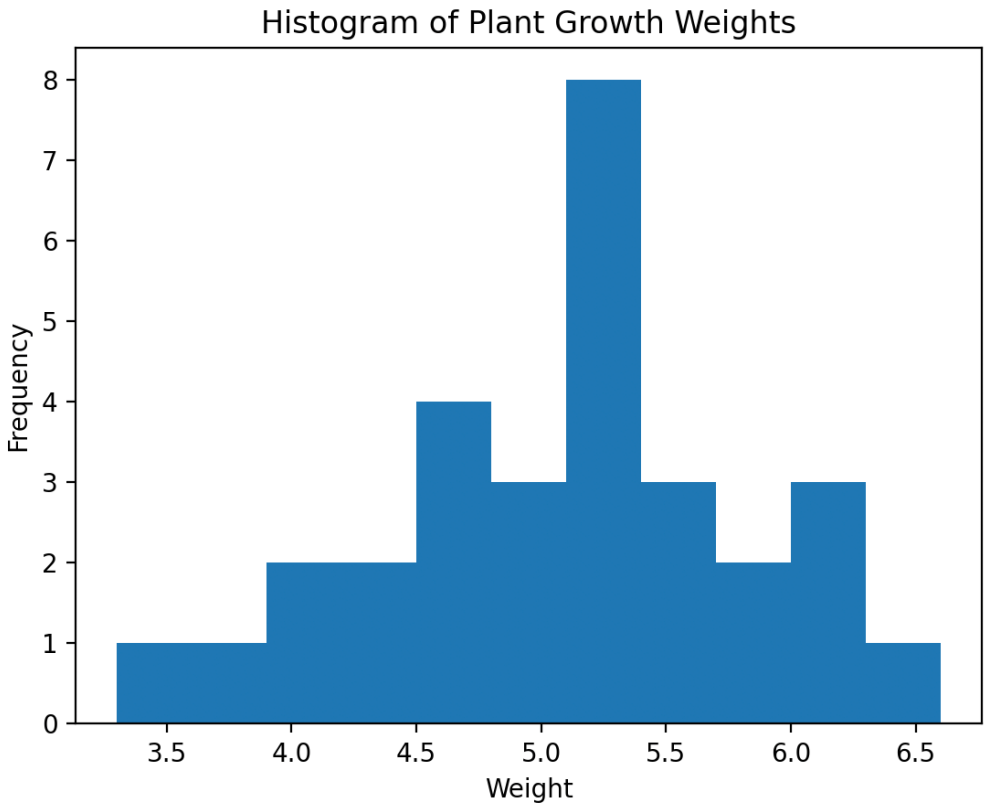
f. Based on #1e, which two variables appear to have the strongest relationship? And which two appear to have the weakest relationship?

Petal length vs petal width appear to have the strongest relationship due to the linear and grouped behavior they display.
Sepal length vs sepal width appear to be more random and exhibit no visual linearity, which makes it the weakest relationship.

2. Using the PlantGrowth dataset...

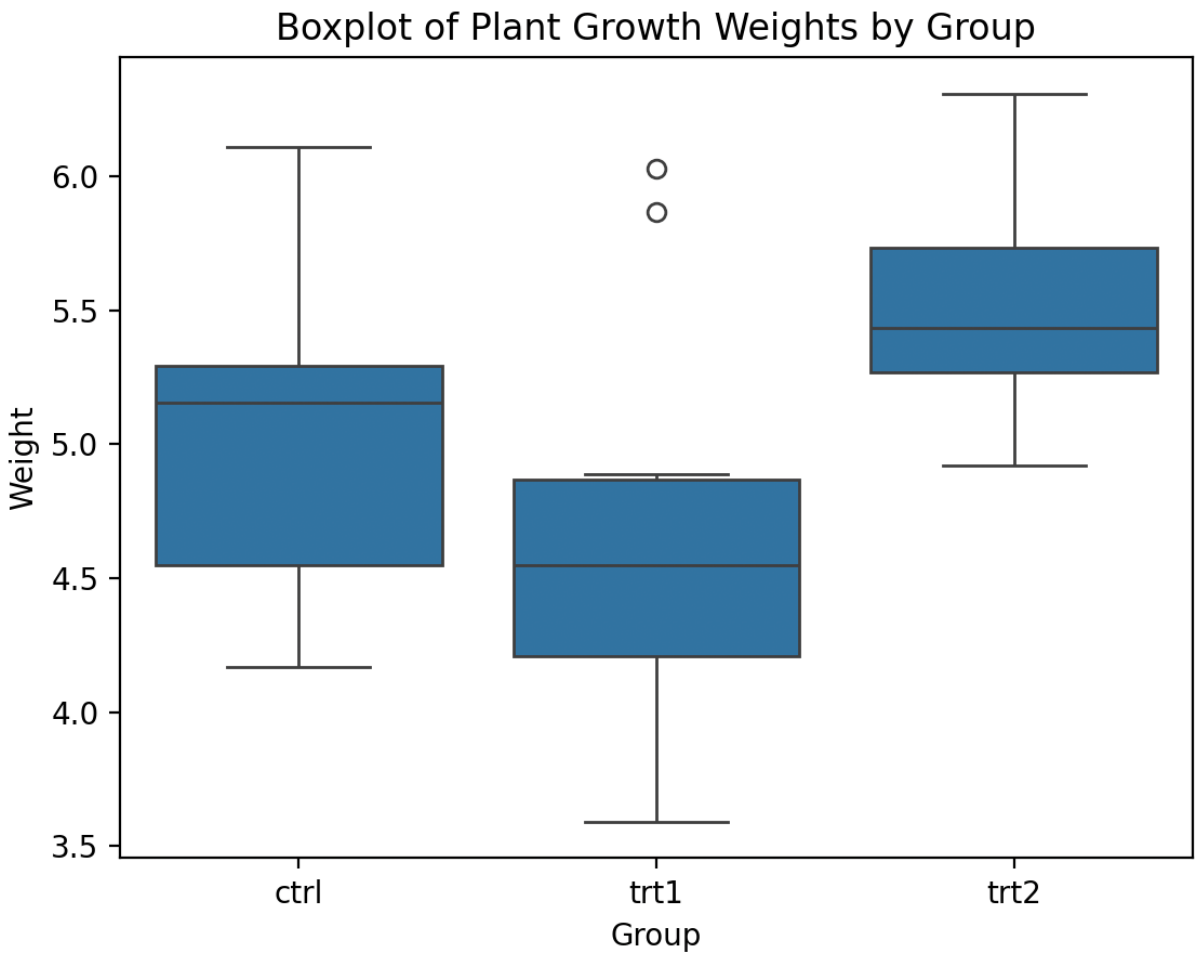
a. Make a histogram of the variable weight with breakpoints (bin edges) at every 0.3 units, starting at 3.3.

```
# Making bins for histogram
bins = np.arange(3.3, PlantGrowth['weight'].max() + 0.3, 0.3)
plt.hist(PlantGrowth['weight'], bins)
plt.xlabel('Weight')
plt.ylabel('Frequency')
plt.title('Histogram of Plant Growth Weights')
plt.show()
```



b. Make boxplots of weight separated by group in a single graph.

```
#Q2b: Making boxplots for Plant Growth for each group
sns.boxplot(x='group', y='weight', data=PlantGrowth)
plt.title('Boxplot of Plant Growth Weights by Group')
plt.xlabel('Group')
plt.ylabel('Weight')
plt.show()
```



c. Based on the boxplots in #2b, approximately what percentage of the "trt1" weights are below the minimum "trt2" weight?

```
#Q2c: Finding minimum and maximum weight for each group using groupby
group_min_max = PlantGrowth.groupby('group')['weight'].agg(['min', 'max', 'count'])
print(group_min_max)
```

group	min	max	count
ctrl	4.17	6.11	10
trt1	3.59	6.03	10
trt2	4.92	6.31	10

Given that the max whisker of trt1 is less than the min whisker of trt2, it is reasonable to assume that most of the trt1 weights are below the minimum trt2 weight. Given that there are two outliers in trt1 that are greater than the minimum whisker of trt2 and it has a count of 10, we assume that 80% of trt1 is below trt2's minimum weight.

d. Find the exact percentage of the "trt1" weights that are below the minimum "trt2" weight.

```
#d. Find the exact percentage of the "trt1" weights that are below the minimum "trt2"
weight.
trt2_min = PlantGrowth[PlantGrowth['group'] == 'trt2']['weight'].min()
trt1_below_trt2_min = PlantGrowth[(PlantGrowth['group'] == 'trt1') &
(PlantGrowth['weight'] < trt2_min)]
# Calculate the percentage of 'trt1' weights below the minimum 'trt2' weight
trt1_below_trt2_min_percent = (len(trt1_below_trt2_min) /
len(PlantGrowth[PlantGrowth['group'] == 'trt1'])) * 100
print(f"Percentage of 'trt1' weights below the minimum 'trt2' weight:
{trt1_below_trt2_min_percent:.2f}%")
```

Percentage of 'trt1' weights below the minimum 'trt2' weight: 80.00%

e. Only including plants with a weight above 5.5, make a barplot of the variable group. Make the barplot colorful using some color palette (in R, try running ?heat.colors and/or check out https://www.r-bloggers.com/palettes-in-r/).

```
#e. Barplot of plants with weight above 5.5
colors = ['yellow', 'red', 'blue']
filtered_plants = PlantGrowth[PlantGrowth['weight'] > 5.5]
freq_table = filtered_plants.groupby('group').size()
plt.bar(freq_table.index, freq_table.values, color=colors)
plt.title('Barplot of Plants with Weight Above 5.5')
plt.xlabel('Group')
plt.ylabel('Frequency')
plt.show()
```