1. Using the iris dataset...

```
from sklearn import datasets
import pandas as pd
import matplotlib.pyplot as plt

# Loading iris dataset
iris = datasets.load_iris()

# Converting to DataFrame
df = pd.DataFrame(data=iris.data, columns=iris.feature_names)

print(df.head())  # peek at the first 5 rows
```

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

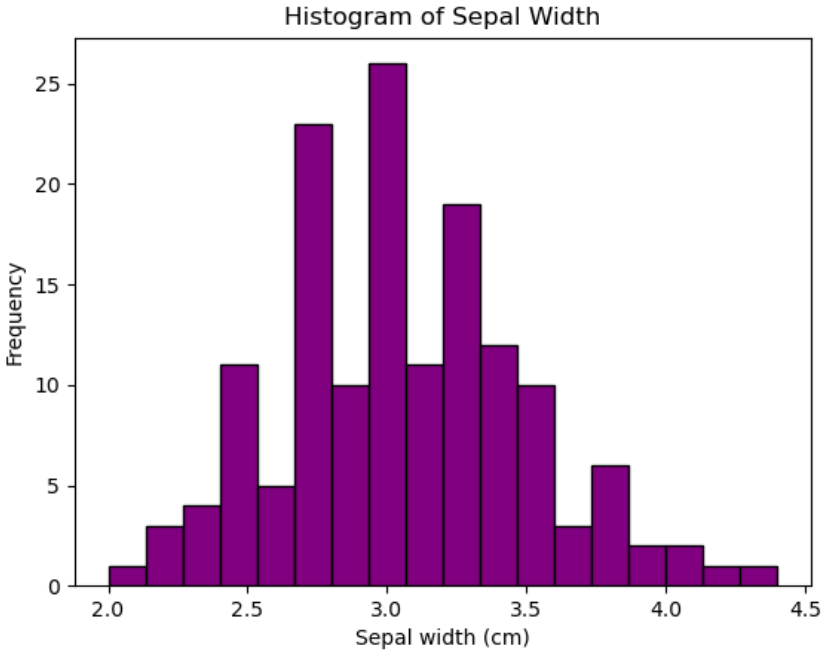
Code:

```
from sklearn import datasets
import pandas as pd
import matplotlib.pyplot as plt
iris = datasets.load_iris()

# Convert to DataFrame
df = pd.DataFrame(data=iris.data, columns=iris.feature_names)
print(df.head())  # peek at the first 5 rows
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

```
plt.hist(df['sepal width (cm)'], bins=18, color='purple', edgecolor='black'\
)
plt.xlabel('Sepal width (cm)')
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?
   - *Based on histogram 1a I believe that the mean will have a similar value to the median , because the median of the sepal's with is is also the most frequent value of the data (which is 3).*

c. Confirm your answer to #1b by actually finding these values.
   - Code:
   - ***import numpy as np***

mean = np.mean(df['sepal width (cm)'])
median = np.median(df['sepal width (cm)'])
(np.float64(3.0573333333333337), np.float64(3.0))

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

To find this value we must determined the 73% percentile . Because 73% are below it, 27% are above it.Therefore, to find that value we have to find the quantile of 73%. To find the value of 73%

- Code:
- ***threshold = np.percentile(df['sepal width (cm)'], 73)***

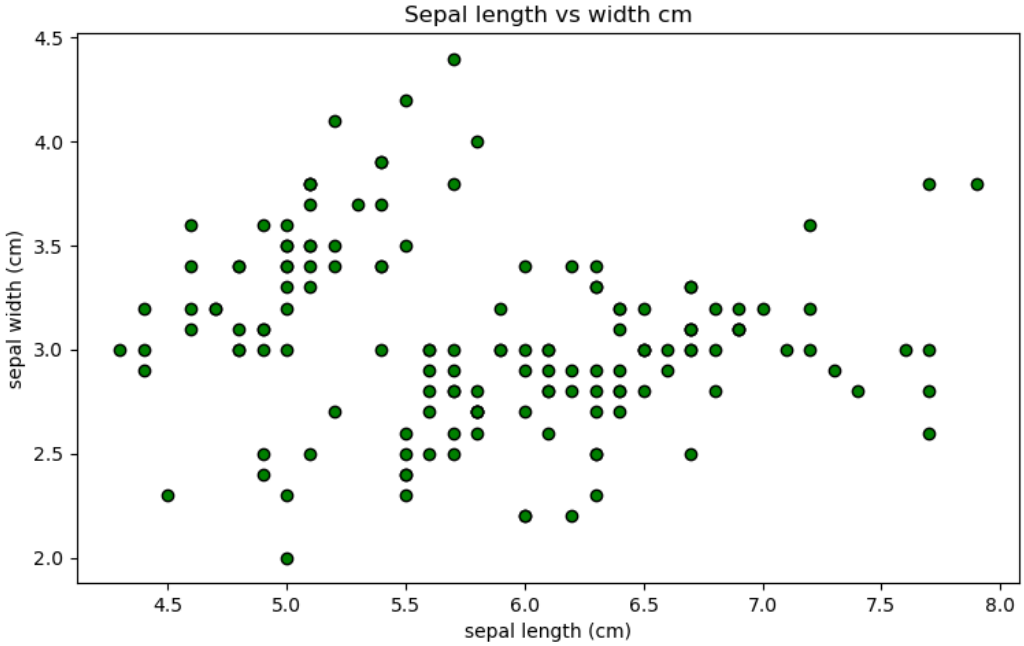
print("Only 27% of flowers have Sepal Width > {:.2f} cm".format(threshold))
Only 27% of flowers have Sepal Width > 3.30 cm

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

Code:

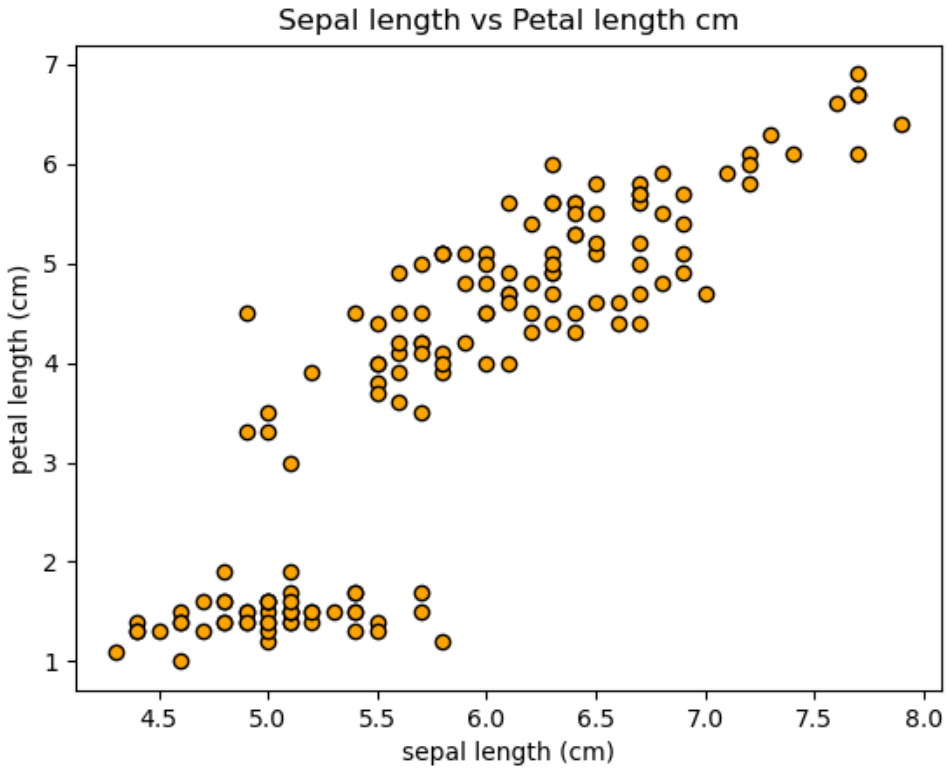
Sepal length vs width cm

```
plt.scatter(df['sepal length (cm)'], df['sepal width (cm)'], color='green',\
 edgecolor='black')
plt.xlabel('sepal length (cm)')
plt.ylabel('sepal width (cm)')
plt.title('Sepal length vs width cm')
plt.show()
```



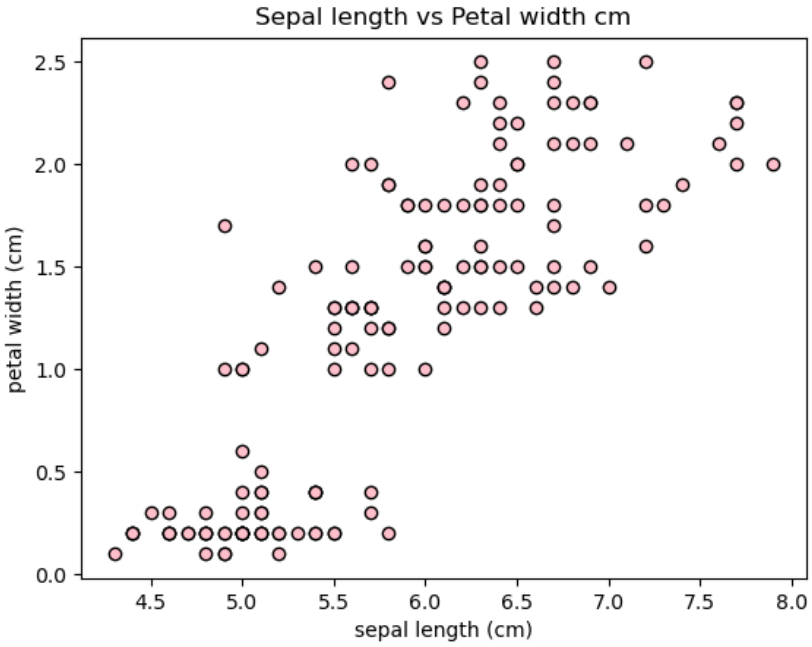
Sepal length vs Petal length cm

```
plt.scatter(df['sepal length (cm)'], df['petal length (cm)'], color='orange\
', edgecolor='black')
plt.xlabel('sepal length (cm)')
plt.ylabel('petal length (cm)')
plt.title('Sepal length vs Petal length cm')
plt.show()
```



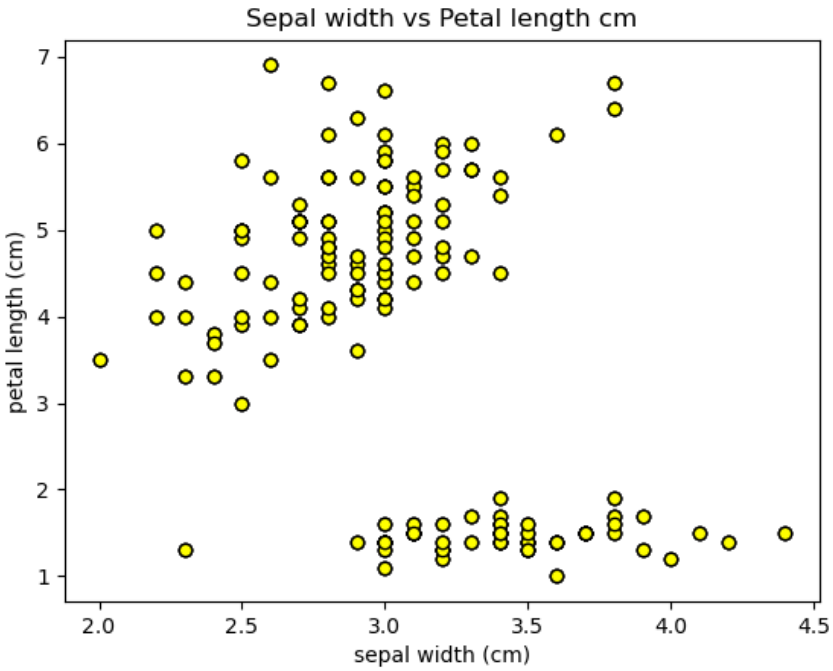
Sepal length vs Petal width cm

plt.scatter(df['sepal length (cm)'], df['petal width (cm)'], color='pink', edgecolor='black')
plt.xlabel('sepal length (cm)')
plt.ylabel('petal width (cm)')
plt.title('Sepal length vs Petal width cm')
plt.show()



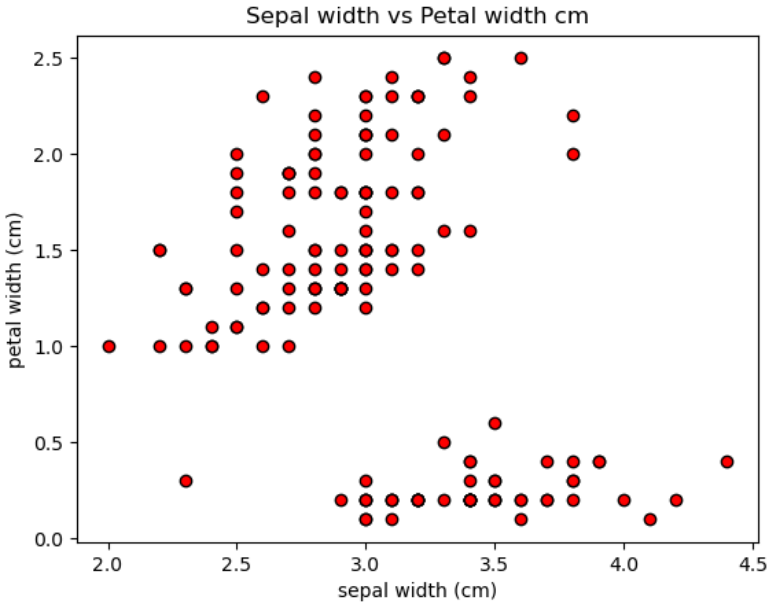
Sepal width vs Petal length cm

plt.scatter(df['sepal width (cm)'], df['petal length (cm)'], color='yellow', edgecolor='black')
plt.xlabel('sepal width (cm)')
plt.ylabel('petal length (cm)')
plt.title('Sepal width vs Petal length cm')
plt.show()



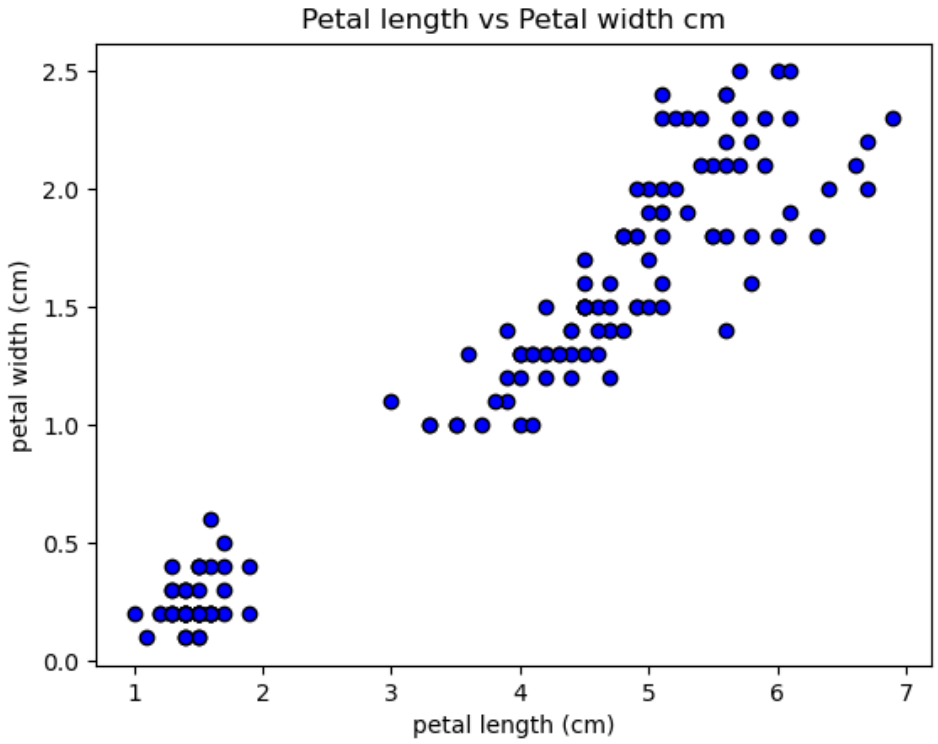
Sepal width vs Petal width

plt.scatter(df['sepal width (cm)'], df['petal width (cm)'], color='red', edgecolor='black')
plt.xlabel('sepal width (cm)')
plt.ylabel('petal width (cm)')
plt.title('Sepal width vs Petal width cm')
plt.show()



Petal length vs Petal width

```
plt.scatter(df['petal length (cm)'], df['petal width (cm)'], color='blue',
edgecolor='black')
plt.xlabel('petal length (cm)')
plt.ylabel('petal width (cm)')
plt.title('Petal length vs Petal width cm')
plt.show()
```



Summary Scatter Plot

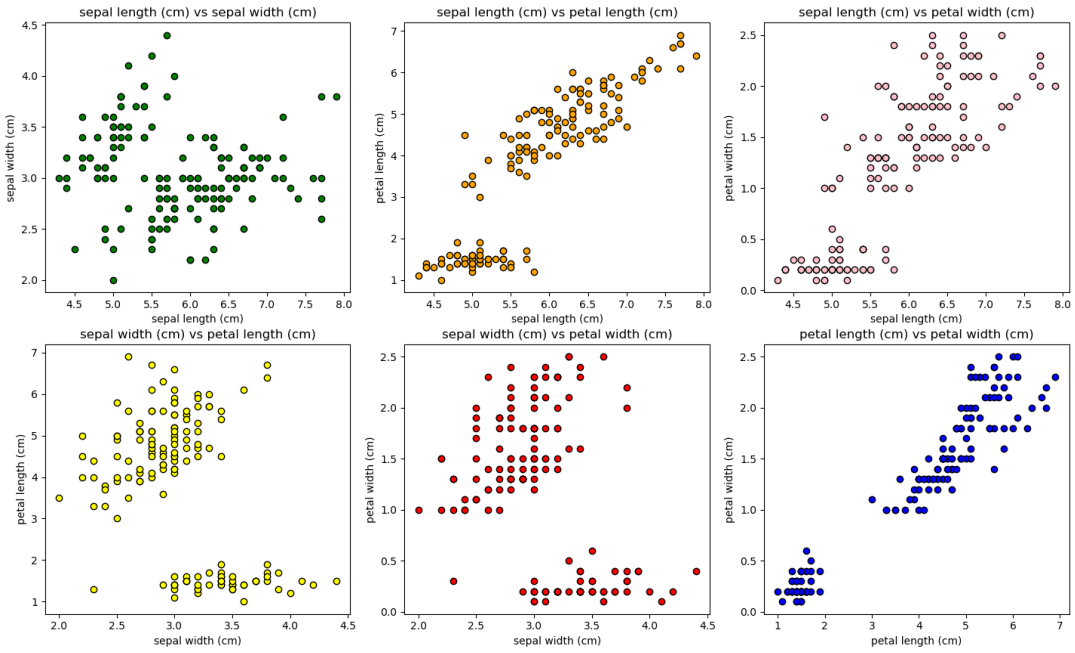
```
# Define the 6 variable pairs
pairs = [
    ('sepal length (cm)', 'sepal width (cm)'),
    ('sepal length (cm)', 'petal length (cm)'),
    ('sepal length (cm)', 'petal width (cm)'),
    ('sepal width (cm)',  'petal length (cm)'),
    ('sepal width (cm)',  'petal width (cm)'),
    ('petal length (cm)', 'petal width (cm)')
]

colors = ['green', 'orange', 'pink', 'yellow', 'red', 'blue']

# Make a 2x3 grid of plots
fig, axes = plt.subplots(2, 3, figsize=(15, 10))
axes = axes.ravel()  # flatten to a 1D list

for ax, (x, y), color in zip(axes, pairs, colors):
    ax.scatter(df[x], df[y], color=color, edgecolor='black')
    ax.set_xlabel(x)
    ax.set_ylabel(y)
    ax.set_title(f"{x} vs {y}")

plt.tight_layout()
plt.show()
```



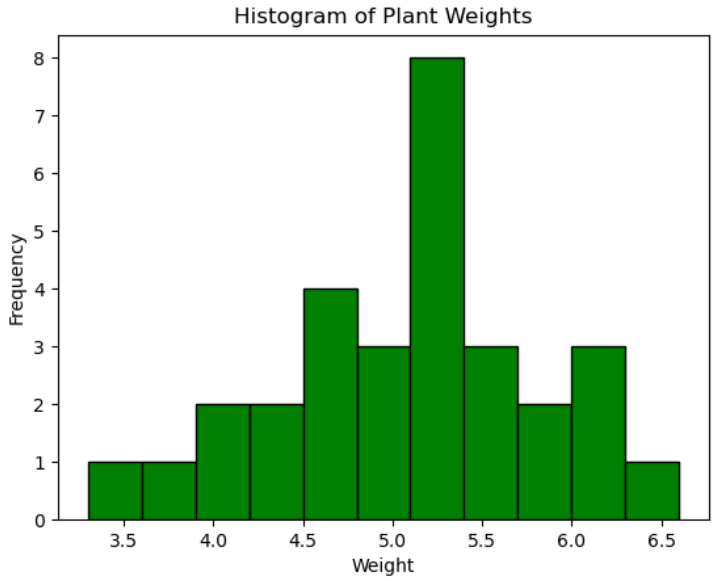
1f. Based on #1e, which two variables appear to have the strongest relationship? And which two appear to have the weakest relationship?
Based on the scatter plots that were made. The variables that have the strongest relationship are the Petal length and petal width.They provide a strong positive correlation.

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.

- Code:

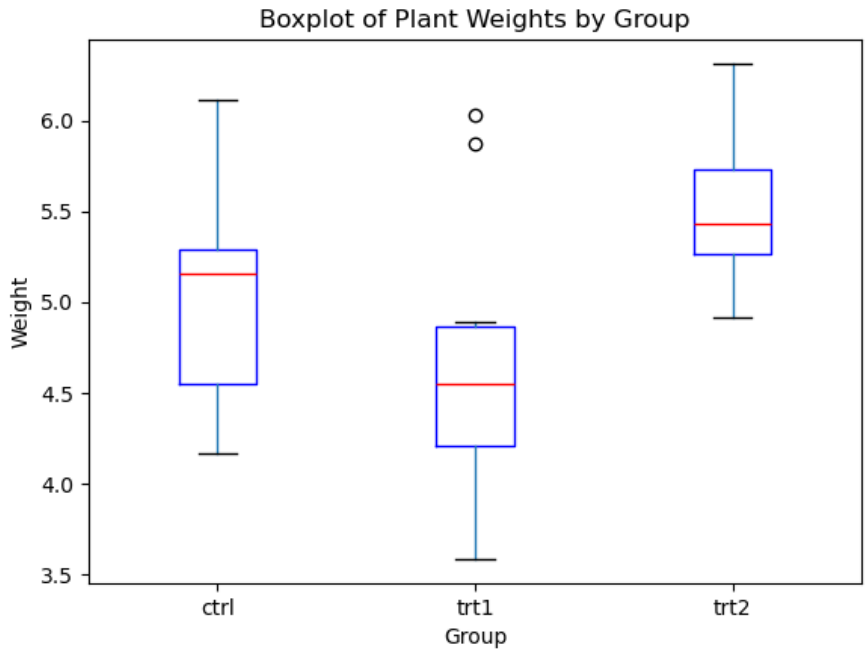
```
import pandas as pd
data = { "weight": [4.17, 5.58, 5.18, 6.11, 4.50, 4.61, 5.17, 4.53, 5.33, 5.14, 4.81, 4.17, 4.41, 3.59, 5.87, 3.83, 6.03, 4.89, 4.32, 4.69, 6.31, 5.12, 5.54, 5.50, 5.37, 5.29, 4.92, 6.15, 5.80, 5.26], "group": ["ctrl"] * 10 + ["trt1"] * 10 + ["trt2"] * 10}
PlantGrowth = pd.DataFrame(data)
bins = np.arange(3.3, PlantGrowth['weight'].max() + 0.3, 0.3)
import matplotlib.pyplot as plt
plt.hist(PlantGrowth['weight'], bins=bins, color='green', edgecolor='black')
plt.xlabel("Weight")
plt.ylabel("Frequency")
plt.title("Histogram of Plant Weights")
plt.show()
```



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

```
import pandas as pd
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)
import matplotlib.pyplot as plt
plt.figure(figsize=(6,4))
PlantGrowth.boxplot(column="weight", by="group", grid=False,
        boxprops=dict(color="blue"), medianprops=dict(color="red"))

plt.title("Boxplot of Plant Weights by Group")
plt.suptitle("")  # removes default pandas title
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?

Based on PlantGrowth data trt 1 weights are 4.17, 3.58, 2.48, 3.91, 2.20, 3.16, 3.22, 3.77, 2.87, 3.03 and trt2 weights are:4.81, 4.17, 4.41, 3.59, 5.87, 3.83, 6.03, 4.89, 4.32, 4.69. Range of trt1 is 2.20-4.17 and Range of trt2 3.59-6.03
Minimum value of trt2 is 3.59. Based on that information we can see that 7 of the 10 values trt1 are below 3.59. Therefore, approximately 70% of the values of trt1 are below the minimum of trt2.

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

```
import pandas as pd
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)
trt1 = PlantGrowth[PlantGrowth['group'] == 'trt1']['weight']
trt2 = PlantGrowth[PlantGrowth['group'] == 'trt2']['weight']
min_trt2 = trt2.min()
print("Minimum trt2 weight:", min_trt2)
count_below = (trt1 < min_trt2).sum()   # number of trt1 values less than min of trt2
total_trt1 = len(trt1)
percentage = (count_below / total_trt1) * 100
print(f"{percentage:.2f}% of trt1 weights are below the minimum trt2 weight.")
```

80.00% of trt1 weights are below the minimum trt2 weight.

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.

Code:

```
import pandas as pd
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)
plants_above_5_5 = PlantGrowth[PlantGrowth['weight'] > 5.5]
group_counts = plants_above_5_5['group'].value_counts()
print(group_counts)
group
trt2    4
ctrl    2
trt1    2
Name: count, dtype: int64
import seaborn as sns
plt.figure(figsize=(6,4))
sns.countplot(x='group', data=plants_above_5_5, palette="Set2")
plt.xlabel("Group")
plt.ylabel("Count")
plt.title("Number of Plants with Weight > 5.5 by Group")
plt.show()
```