#19

# Using Matplotlib for Visualization (cont.)

MIS561 Data Visualization
Original Author: Lusi Yang

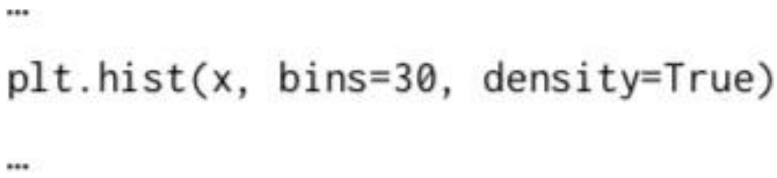
# Histogram

**plt.hist(x)** creates a histogram.

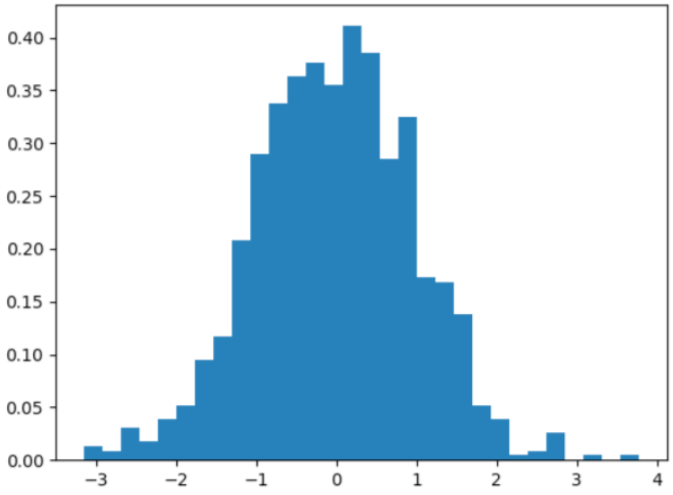
**Parameters:**

- **x** – Specifies the input values
- **bins** (optional) – Either specifies the number of bins as an integer
- **range** (optional) – Specifies the lower and upper range of the bins as a tuple
- **density** (optional) – If true, the histogram represents a probability density

**Example:**



```
...
plt.hist(x, bins=30, density=True)
...
```



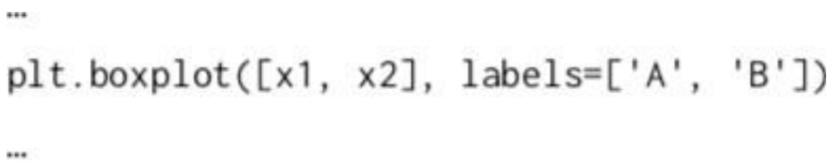
# Box Plot

**plt.boxplot(x)** creates a box plot.

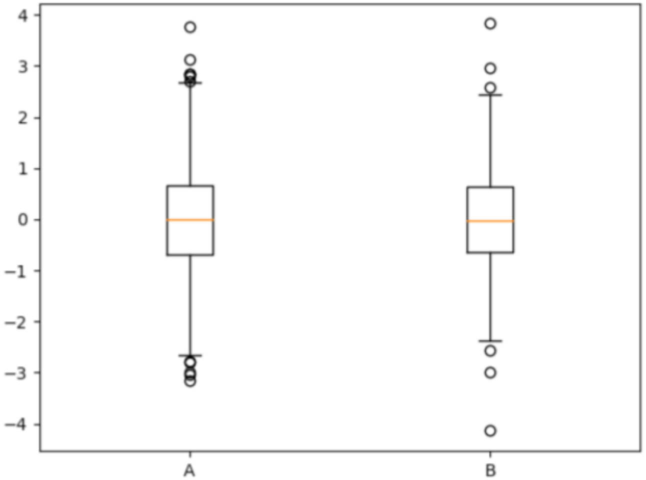
**Parameters:**

- **x** – Specifies the input data. It specifies either a 1D array for a single box or a sequence of arrays for multiple boxes.
- **notch** (optional) – If true, notches will be added to the plot to indicate the confidence interval around the median.
- **labels** (optional) – Specifies the labels as a sequence.
- **showfliers** (optional) – By default, it is true, and outliers are plotted beyond the caps.
- **showmeans** (optional) – If true, arithmetic means are shown.

**Example:**



```
...
plt.boxplot([x1, x2], labels=['A', 'B'])
...
```

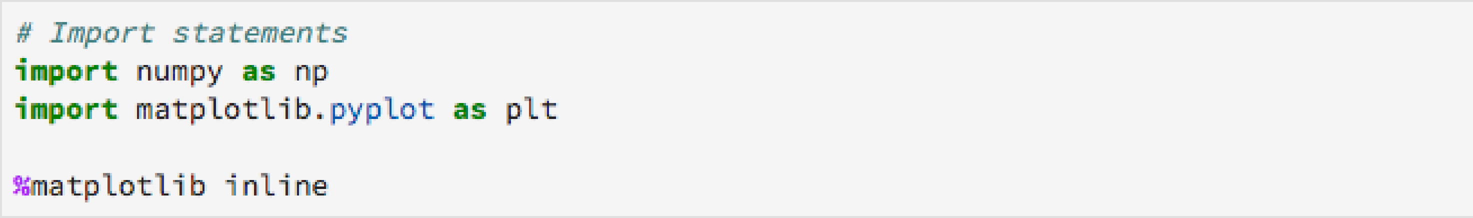


Activity 5

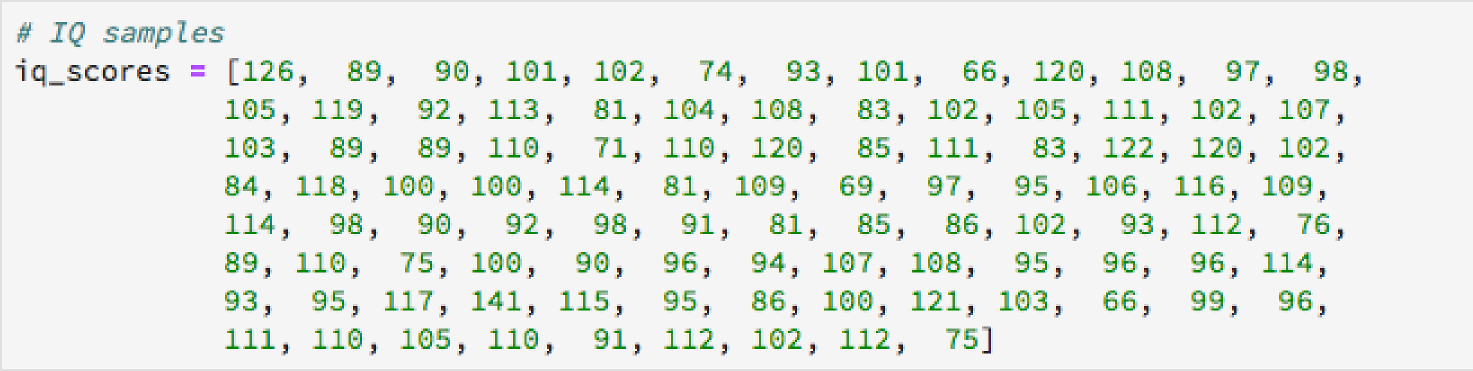
# Using a Histogram and a Box Plot to Visualize IQ

## Objective

We are given the intelligent quotient scores of 100 adults. We want to visualize the distribution of the IQ scores.We will create a histogram and box plots to visualize this information.

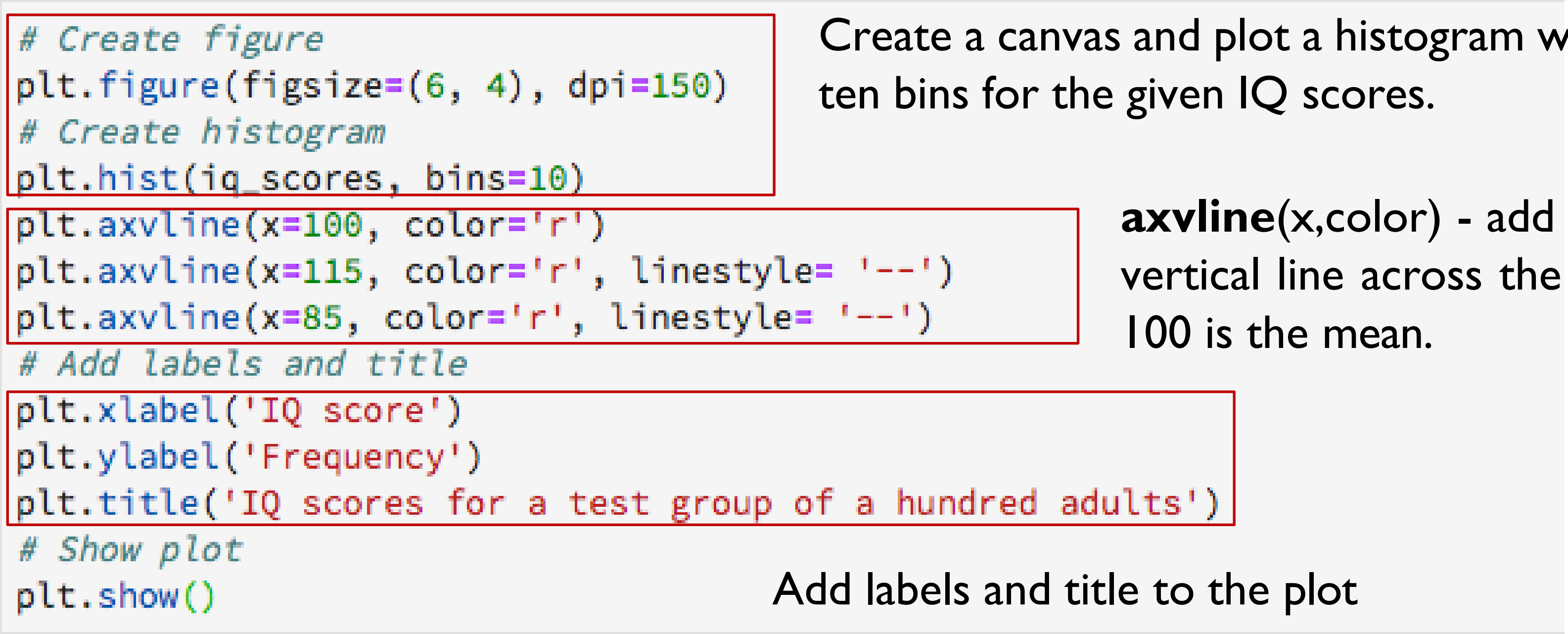


```
# Import statements
import numpy as np
import matplotlib.pyplot as plt

%matplotlib inline
```



```
# IQ samples
iq_scores = [126,  89,  90, 101, 102,  74,  93, 101,  66, 120, 108,  97,  98,
            105, 119,  92, 113,  81, 104, 108,  83, 102, 105, 111, 102, 107,
            103,  89,  89, 110,  71, 110, 120,  85, 111,  83, 122, 120, 102,
            84, 118, 100, 100, 114,  81, 109,  69,  97,  95, 106, 116, 109,
            114,  98,  90,  92,  98,  91,  81,  85,  86, 102,  93, 112,  76,
            89, 110,  75, 100,  90,  96,  94, 107, 108,  95,  96,  96, 114,
            93,  95, 117, 141, 115,  95,  86, 100, 121, 103,  66,  99,  96,
            111, 110, 105, 110,  91, 112, 102, 112,  75]
```

Create an array that includes the IQ scores of the 100 adults



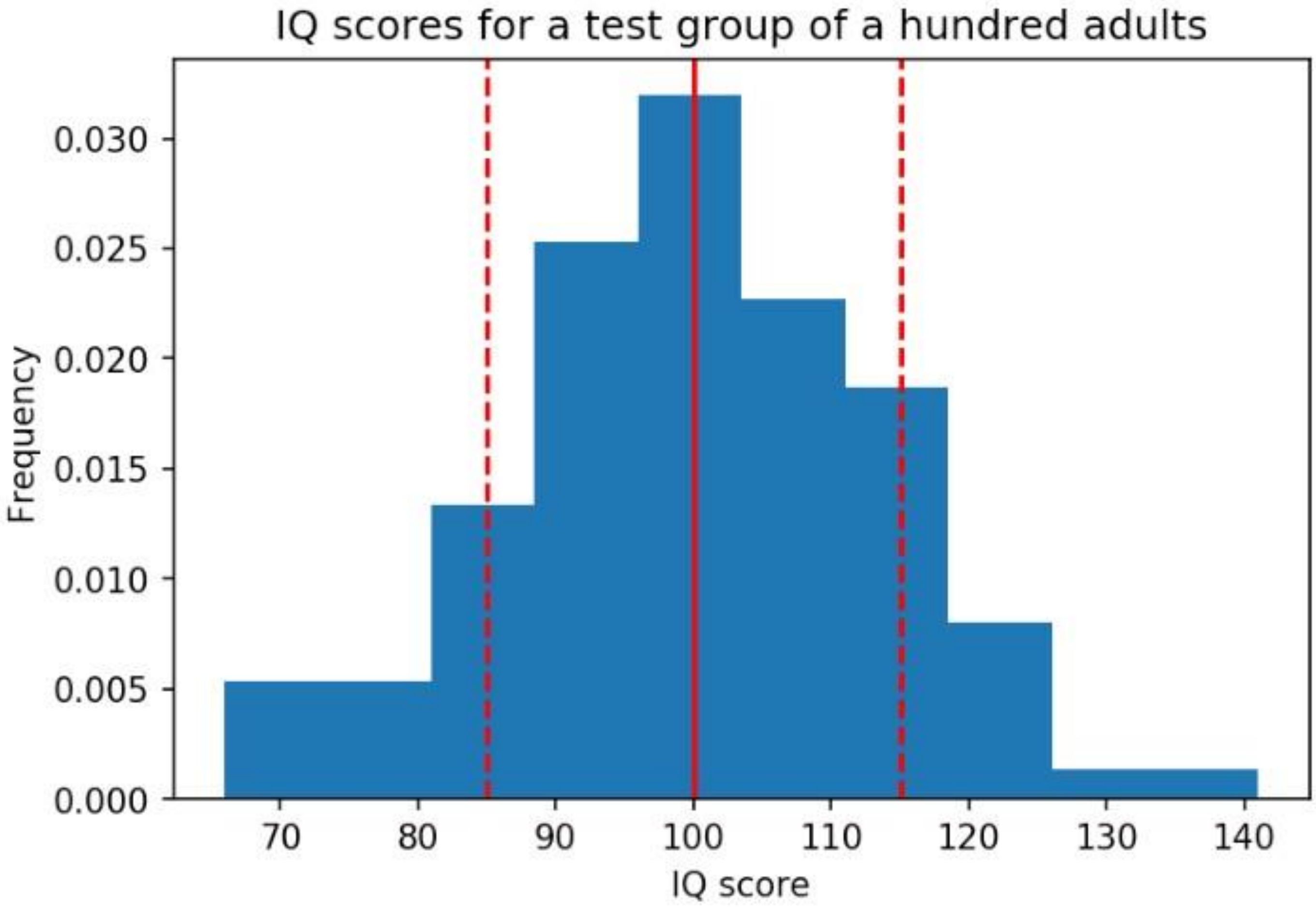
```
# Create figure
plt.figure(figsize=(6, 4), dpi=150)
# Create histogram
plt.hist(iq_scores, bins=10)
plt.axvline(x=100, color='r')
plt.axvline(x=115, color='r', linestyle= '--')
plt.axvline(x=85, color='r', linestyle= '--')
# Add labels and title
plt.xlabel('IQ score')
plt.ylabel('Frequency')
plt.title('IQ scores for a test group of a hundred adults')
# Show plot
plt.show()
```

Create a canvas and plot a histogram with ten bins for the given IQ scores.

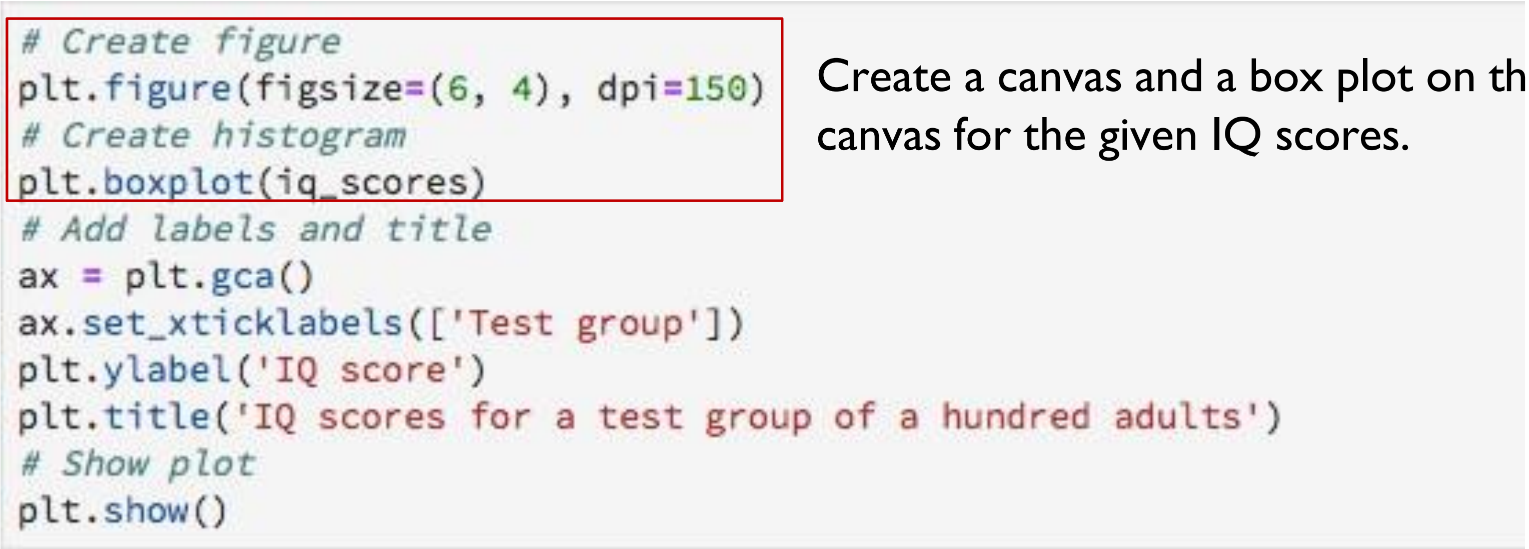
**axvline**(x,color) - add three vertical line across the axes. 100 is the mean.

Add labels and title to the plot

## Expected view



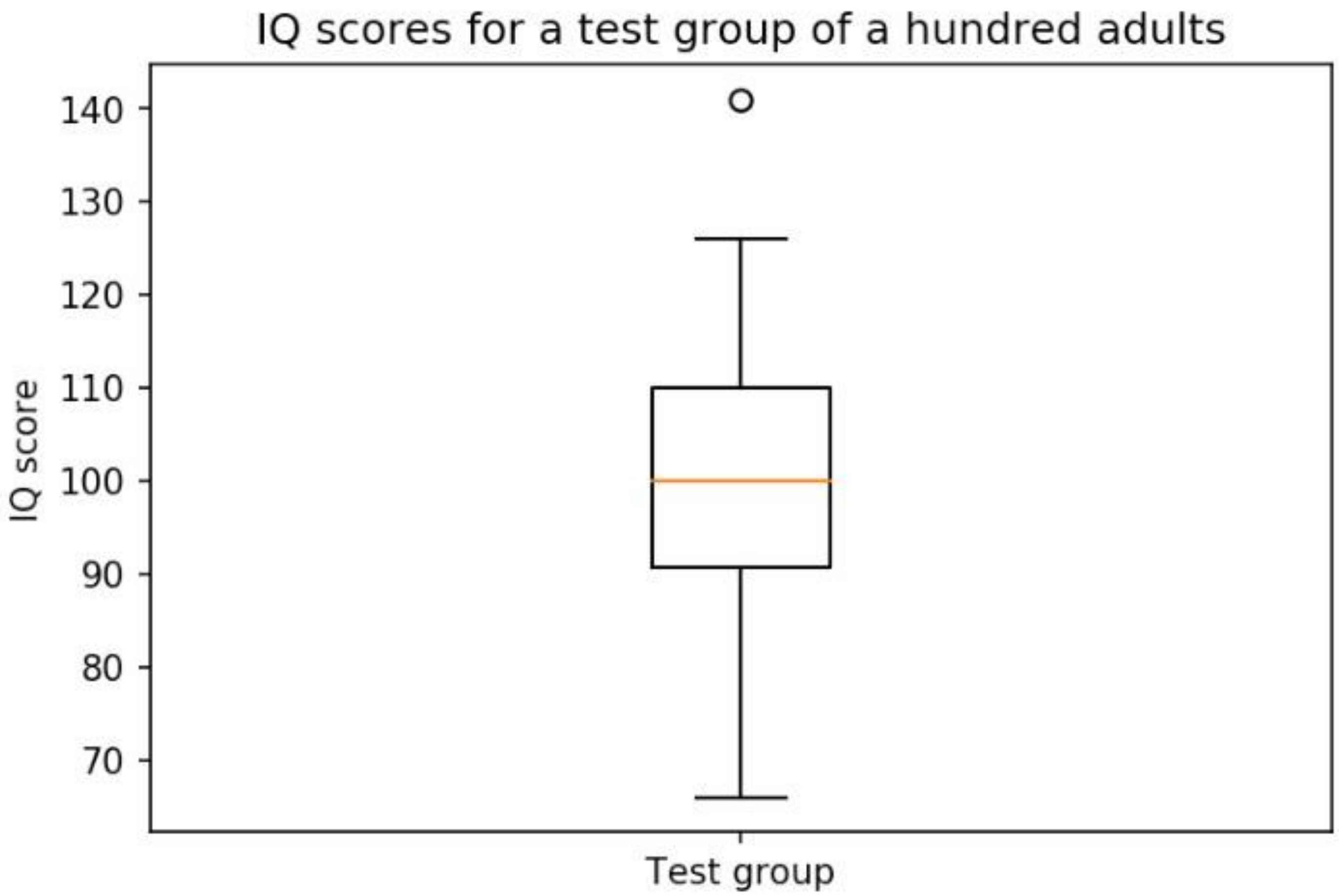
- What would the histogram look like if it has 20 bins?
- What if we change the y-axis to number of adults in each bin (rather than probability density)?



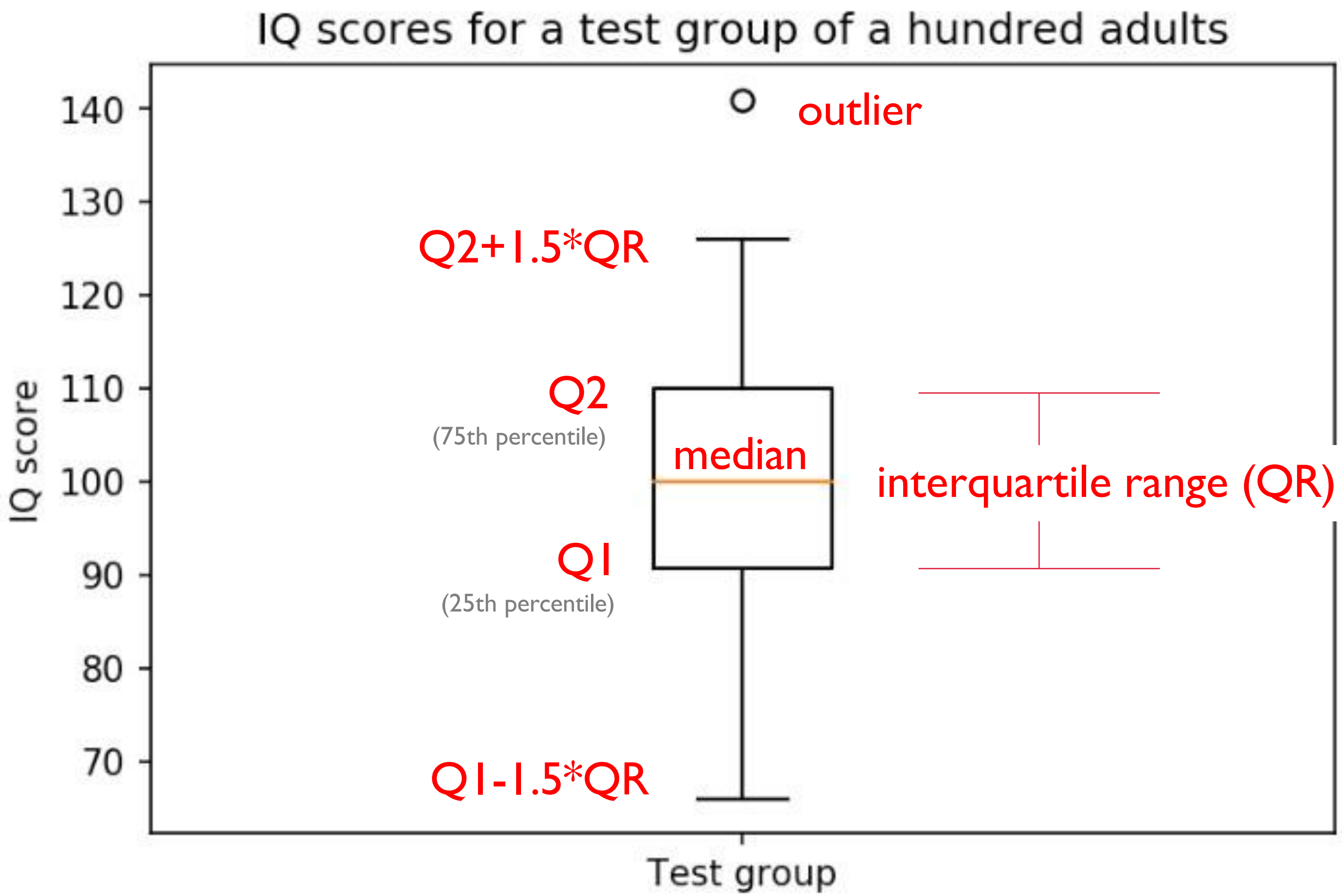
```
# Create figure
plt.figure(figsize=(6, 4), dpi=150)
# Create histogram
plt.boxplot(iq_scores)
# Add labels and title
ax = plt.gca()
ax.set_xticklabels(['Test group'])
plt.ylabel('IQ score')
plt.title('IQ scores for a test group of a hundred adults')
# Show plot
plt.show()
```

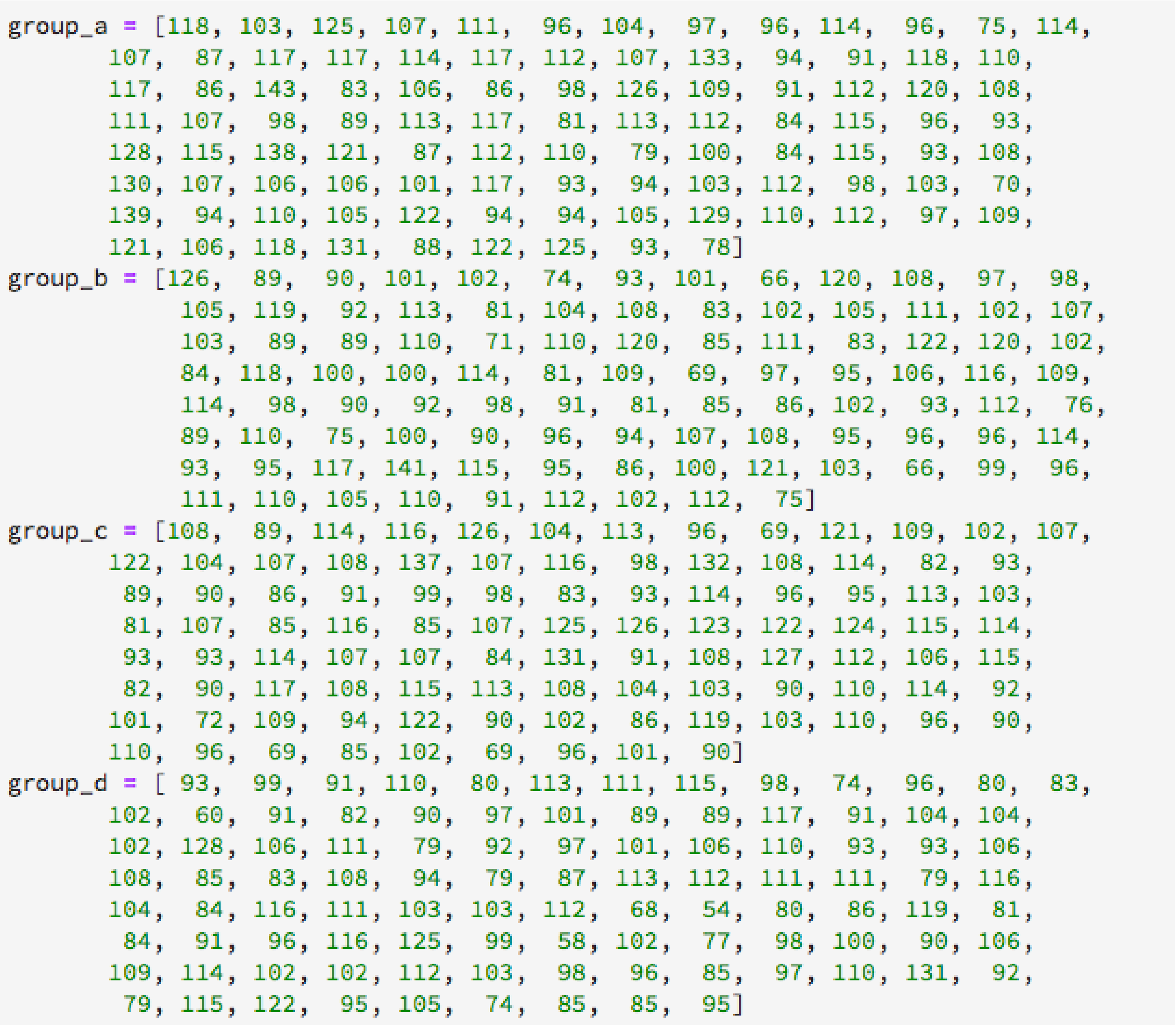
Create a canvas and a box plot on the canvas for the given IQ scores.

# Expected view



## Interpretation of box plot

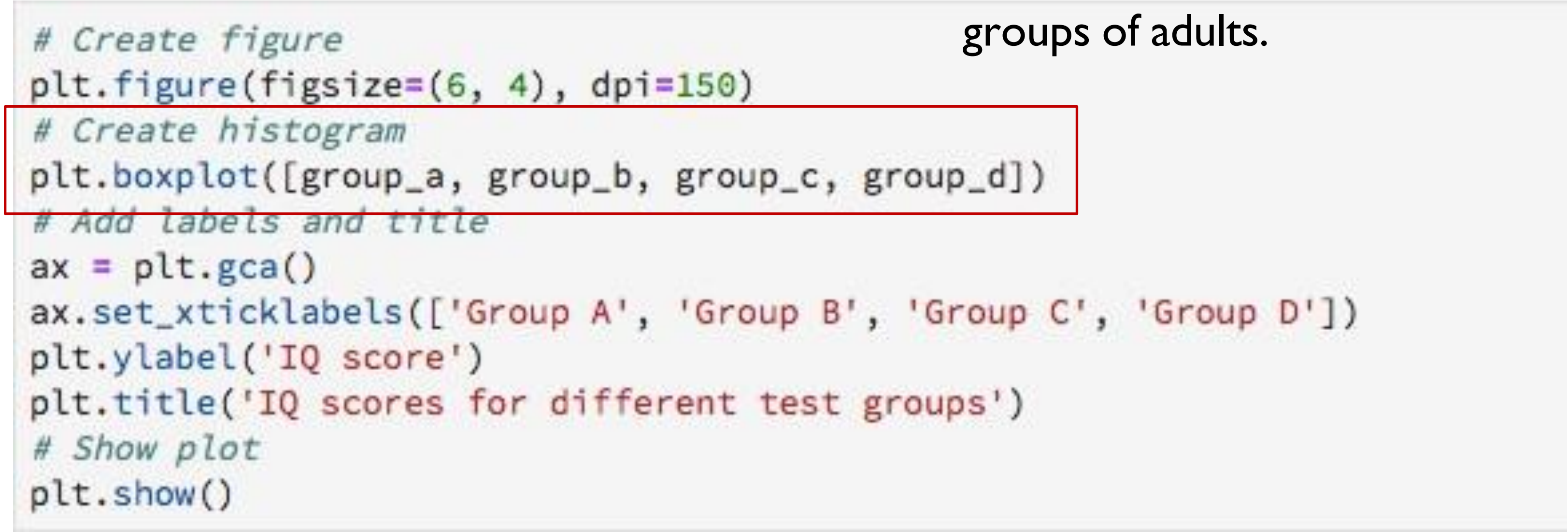




```
group_a = [118, 103, 125, 107, 111,  96, 104,  97,  96, 114,  96,  75, 114,
       107,  87, 117, 117, 114, 117, 112, 107, 133,  94,  91, 118, 110,
       117,  86, 143,  83, 106,  86,  98, 126, 109,  91, 112, 120, 108,
       111, 107,  98,  89, 113, 117,  81, 113, 112,  84, 115,  96,  93,
       128, 115, 138, 121,  87, 112, 110,  79, 100,  84, 115,  93, 108,
       130, 107, 106, 106, 101, 117,  93,  94, 103, 112,  98, 103,  70,
       139,  94, 110, 105, 122,  94,  94, 105, 129, 110, 112,  97, 109,
       121, 106, 118, 131,  88, 122, 125,  93,  78]
group_b = [126,  89,  90, 101, 102,  74,  93, 101,  66, 120, 108,  97,  98,
            105, 119,  92, 113,  81, 104, 108,  83, 102, 105, 111, 102, 107,
            103,  89,  89, 110,  71, 110, 120,  85, 111,  83, 122, 120, 102,
            84, 118, 100, 100, 114,  81, 109,  69,  97,  95, 106, 116, 109,
            114,  98,  90,  92,  98,  91,  81,  85,  86, 102,  93, 112,  76,
            89, 110,  75, 100,  90,  96,  94, 107, 108,  95,  96,  96, 114,
            93,  95, 117, 141, 115,  95,  86, 100, 121, 103,  66,  99,  96,
            111, 110, 105, 110,  91, 112, 102, 112,  75]
group_c = [108,  89, 114, 116, 126, 104, 113,  96,  69, 121, 109, 102, 107,
       122, 104, 107, 108, 137, 107, 116,  98, 132, 108, 114,  82,  93,
        89,  90,  86,  91,  99,  98,  83,  93, 114,  96,  95, 113, 103,
        81, 107,  85, 116,  85, 107, 125, 126, 123, 122, 124, 115, 114,
        93,  93, 114, 107, 107,  84, 131,  91, 108, 127, 112, 106, 115,
        82,  90, 117, 108, 115, 113, 108, 104, 103,  90, 110, 114,  92,
       101,  72, 109,  94, 122,  90, 102,  86, 119, 103, 110,  96,  90,
       110,  96,  69,  85, 102,  69,  96, 101,  90]
group_d = [ 93,  99,  91, 110,  80, 113, 111, 115,  98,  74,  96,  80,  83,
       102,  60,  91,  82,  90,  97, 101,  89,  89, 117,  91, 104, 104,
       102, 128, 106, 111,  79,  92,  97, 101, 106, 110,  93,  93, 106,
       108,  85,  83, 108,  94,  79,  87, 113, 112, 111, 111,  79, 116,
       104,  84, 116, 111, 103, 103, 112,  68,  54,  80,  86, 119,  81,
        84,  91,  96, 116, 125,  99,  58, 102,  77,  98, 100,  90, 106,
       109, 114, 102, 102, 112, 103,  98,  96,  85,  97, 110, 131,  92,
        79, 115, 122,  95, 105,  74,  85,  85,  95]
```

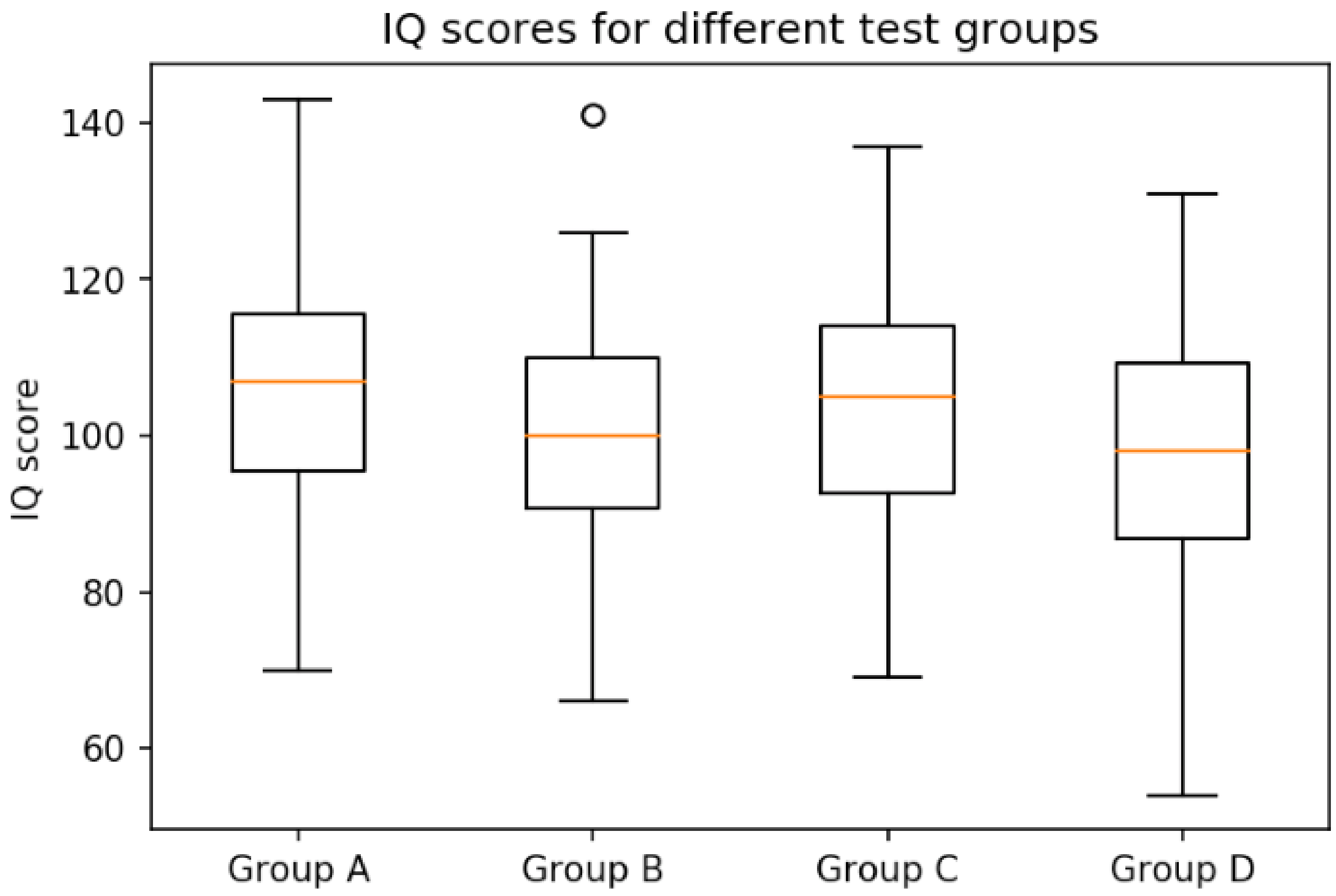
Create four arrays that include the IQ scores of four groups of adults

Create a box plot for the four groups of adults.



```
# Create figure
plt.figure(figsize=(6, 4), dpi=150)
# Create histogram
plt.boxplot([group_a, group_b, group_c, group_d])
# Add labels and title
ax = plt.gca()
ax.set_xticklabels(['Group A', 'Group B', 'Group C', 'Group D'])
plt.ylabel('IQ score')
plt.title('IQ scores for different test groups')
# Show plot
plt.show()
```

## Expected view



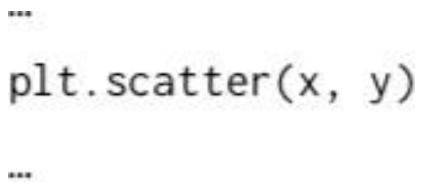
# Scatter Plot

**plt.scatter(x, y)** creates a scatter plot of y versus x.

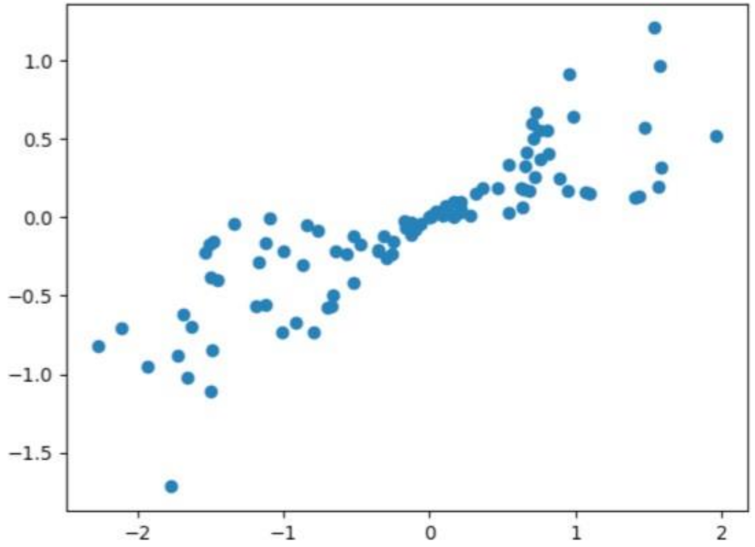
**Parameters:**

- **x**, **y** – Specifies the data positions.
- **s** (optional) – Specifies the marker size in points squared.
- **c** (optional) – Specifies the marker color. If a sequence of numbers is specified, the numbers will be mapped to colors of the color map.

**Example:**



```
…
plt.scatter(x, y)
…
```

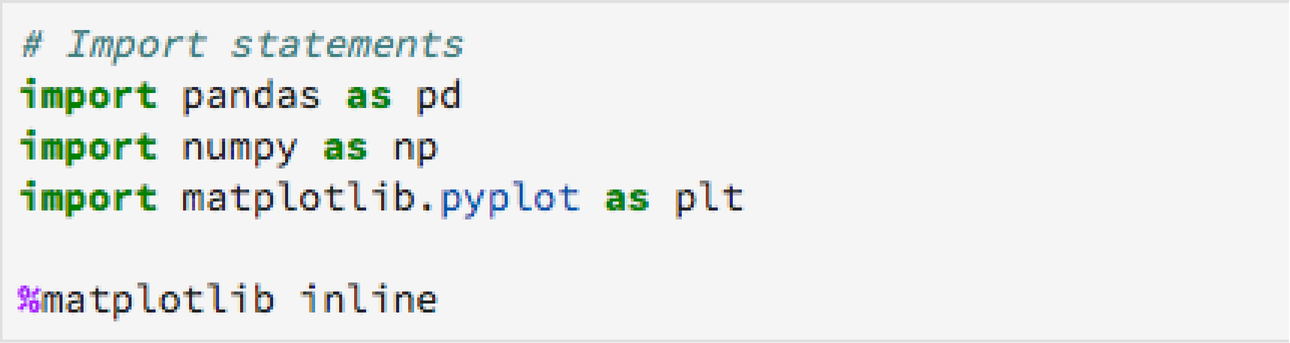


Activity 6

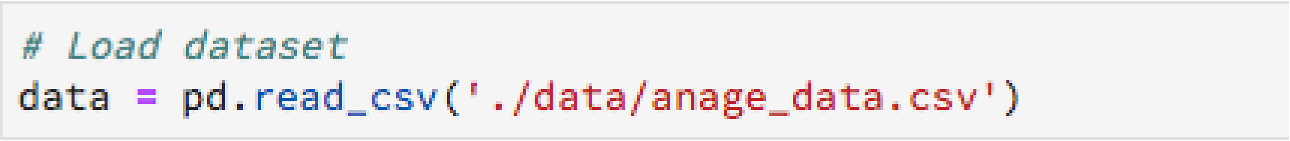
# Using a Scatter Plot to Visualize Correlation Between Various Animals

## Objective

We are given a dataset containing information about various animals. To show correlation between animal attributes within the dataset, we will create a scatter plot.



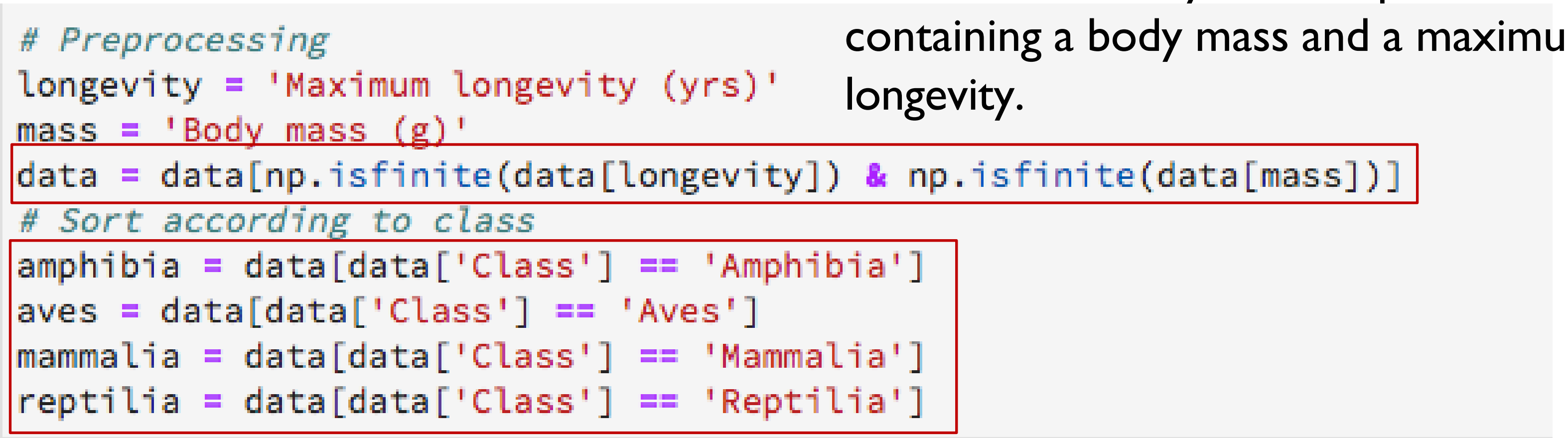
```
# Import statements
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

%matplotlib inline
```



```
# Load dataset
data = pd.read_csv('./data/anage_data.csv')
```

Load data from the the CVS file we downloaded

Filter the data so you end up with samples containing a body mass and a maximum longevity.



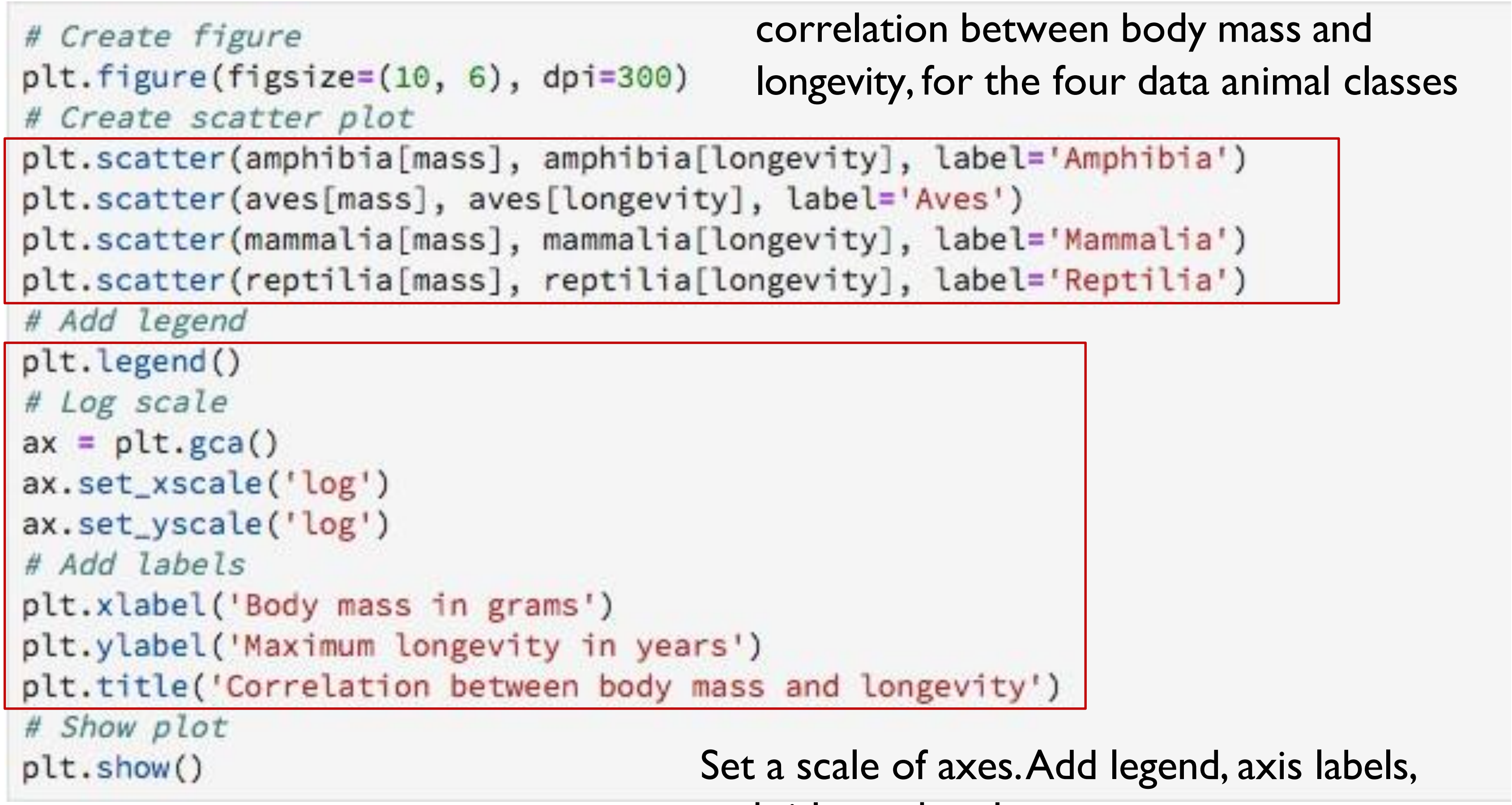
```
# Preprocessing
longevity = 'Maximum longevity (yrs)'
mass = 'Body mass (g)'
data = data[np.isfinite(data[longevity]) & np.isfinite(data[mass])]
# Sort according to class
amphibia = data[data['Class'] == 'Amphibia']
aves = data[data['Class'] == 'Aves']
mammalia = data[data['Class'] == 'Mammalia']
reptilia = data[data['Class'] == 'Reptilia']
```

Sort the data into four arrays based on animal class

isfinite() returns a Boolean, showing whether a value is finite

If a value is missing, infinite() returns False.

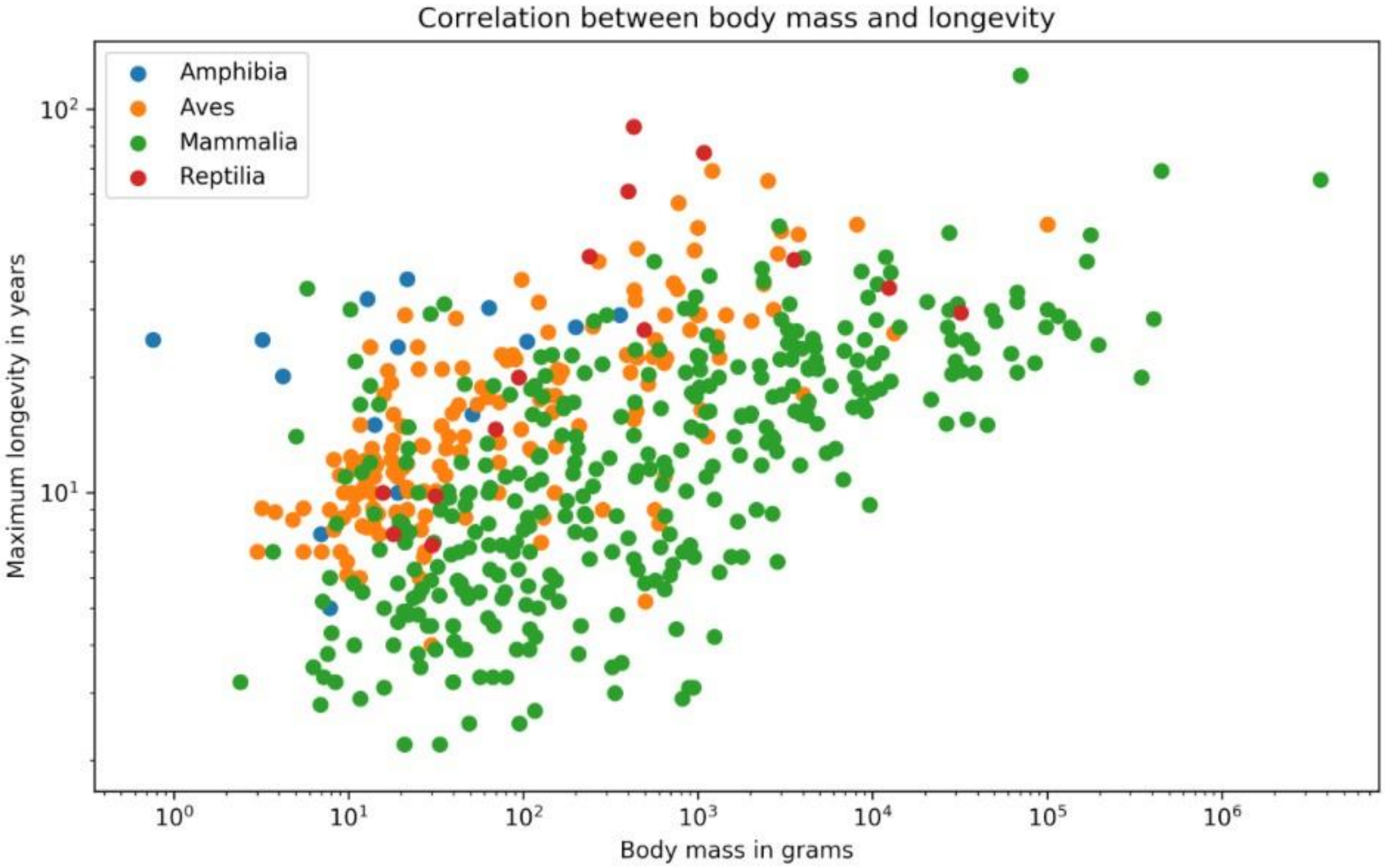
Create a scatter plot, visualizing the correlation between body mass and longevity, for the four data animal classes



```
# Create figure
plt.figure(figsize=(10, 6), dpi=300)
# Create scatter plot
plt.scatter(amphibia[mass], amphibia[longevity], label='Amphibia')
plt.scatter(aves[mass], aves[longevity], label='Aves')
plt.scatter(mammalia[mass], mammalia[longevity], label='Mammalia')
plt.scatter(reptilia[mass], reptilia[longevity], label='Reptilia')
# Add legend
plt.legend()
# Log scale
ax = plt.gca()
ax.set_xscale('log')
ax.set_yscale('log')
# Add labels
plt.xlabel('Body mass in grams')
plt.ylabel('Maximum longevity in years')
plt.title('Correlation between body mass and longevity')
# Show plot
plt.show()
```

Set a scale of axes. Add legend, axis labels, and title to the plot.

# Expected view



# Layouts: subplots

Matplotlib offers the concept of subplots, which are multiple Axes within a Figure.These plots can be grids of plots, nested plots, and so forth.

The functions to create subplots:

- **plt.subplots(nrows, ncols)** creates a Figure and a set of subplots.
- **plt.subplot(nrows, ncols, index)** adds a subplot to the current Figure.The index starts at 1.
- **Figure.subplots(nrows, ncols)** adds a set of subplots to the specified Figure.
- **Figure.add_subplot(nrows, ncols, index)** adds a subplot to the specified Figure.

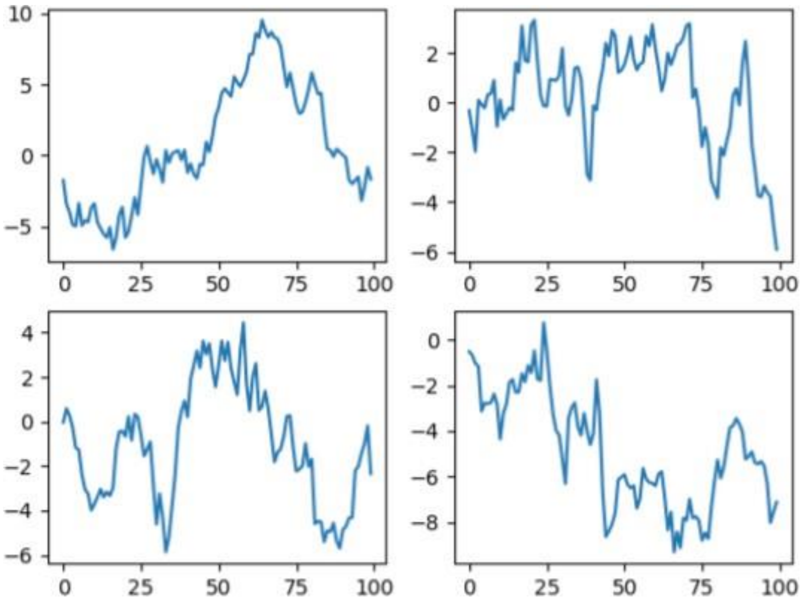
For sharing the x or y axis, the parameters **sharex** and **sharey** must be set, respectively. The axis will have the same limits, ticks, and scale.

# Layouts: subplots

## Example 1:

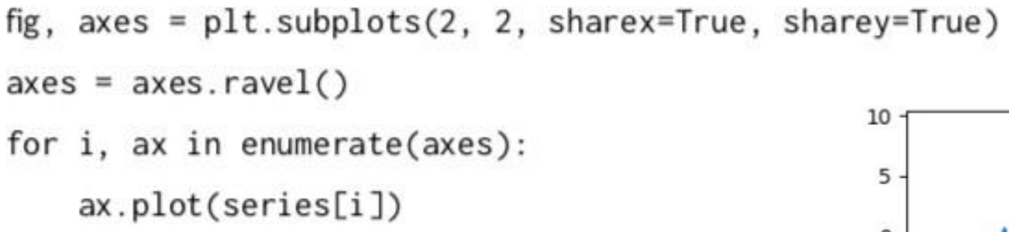
```
…
fig, axes = plt.subplots(2, 2)
axes = axes.ravel()
for i, ax in enumerate(axes):
    ax.plot(series[i])
…
```

```
…
for i in range(4):
    plt.subplot(2, 2, i+1)
    plt.plot(series[i])
…
```

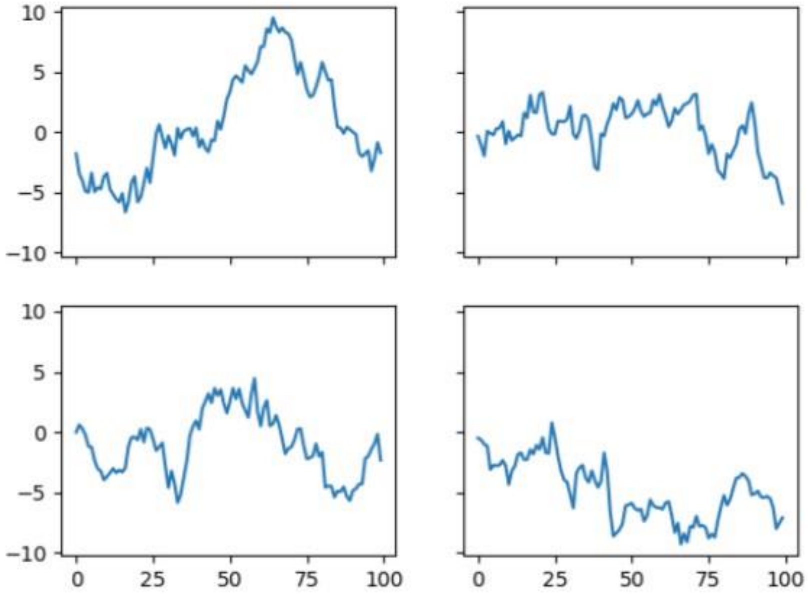


# Layouts: subplots

## Example 2:



```
fig, axes = plt.subplots(2, 2, sharex=True, sharey=True)
axes = axes.ravel()
for i, ax in enumerate(axes):
    ax.plot(series[i])
```

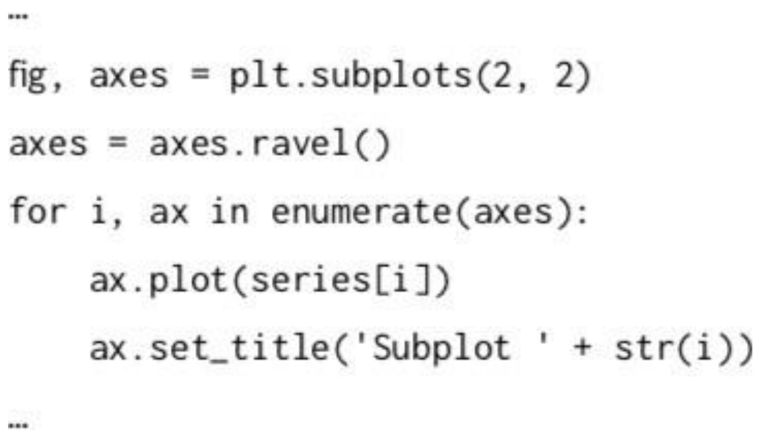


Setting **sharex** and **sharey** to true

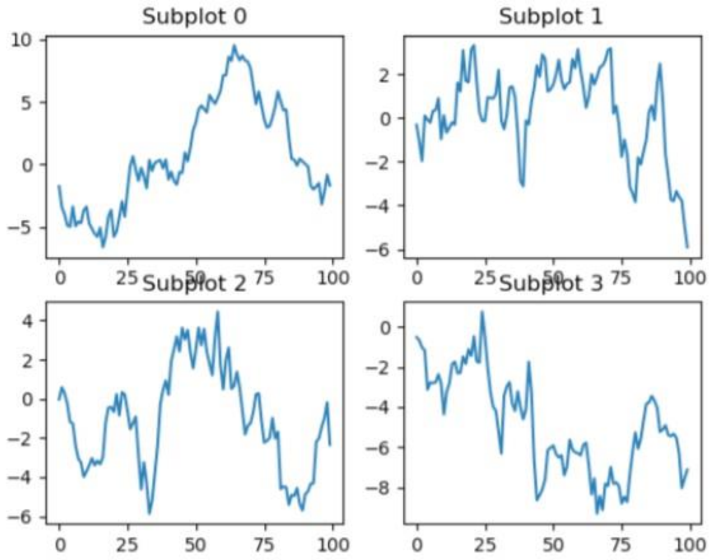
# Layouts: tight layouts

**plt.tight_layout()** adjusts subplot parameters so that the subplots fit well in the Figure.

**Example:**
If you do not use **plt.tight_layout()**, subplots might overlap:

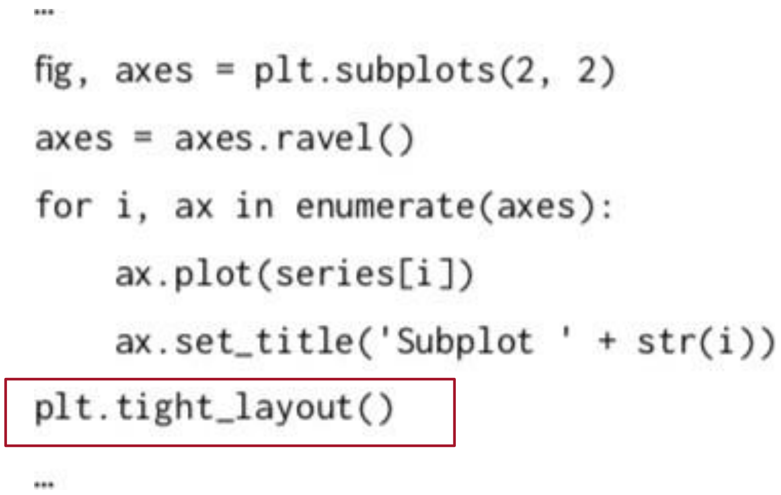


```
…

fig, axes = plt.subplots(2, 2)

axes = axes.ravel()

for i, ax in enumerate(axes):

    ax.plot(series[i])

    ax.set_title('Subplot ' + str(i))

…
```

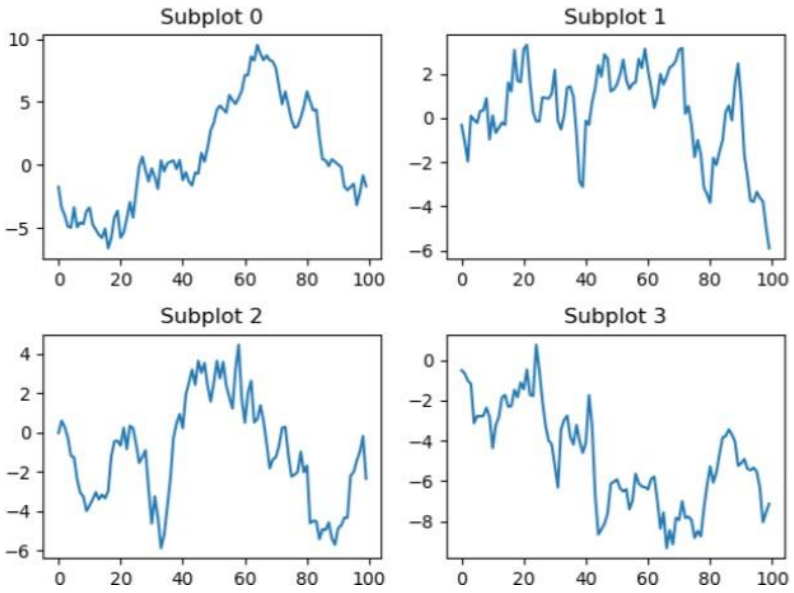


# Layouts: tight layouts

Using **plt.tight_layout()** results in no overlapping of the subplots.



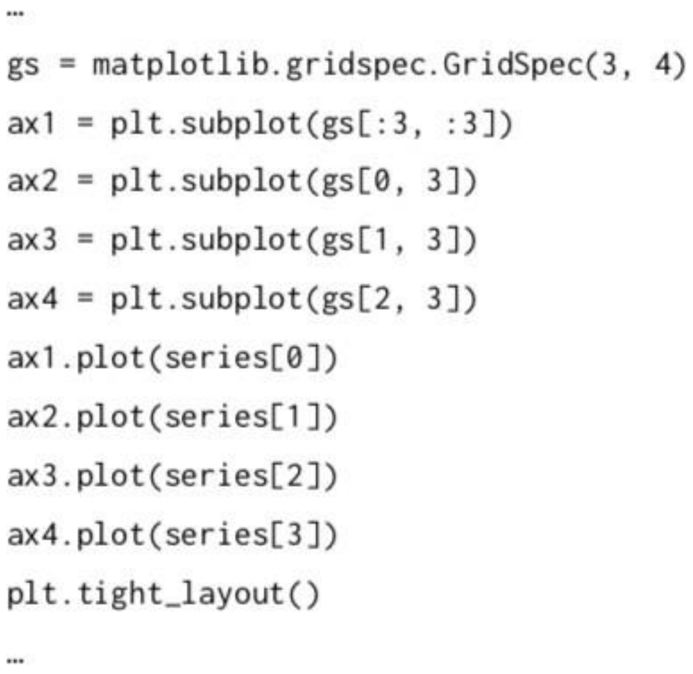
```
...
fig, axes = plt.subplots(2, 2)
axes = axes.ravel()
for i, ax in enumerate(axes):
    ax.plot(series[i])
    ax.set_title('Subplot ' + str(i))
plt.tight_layout()
...
```



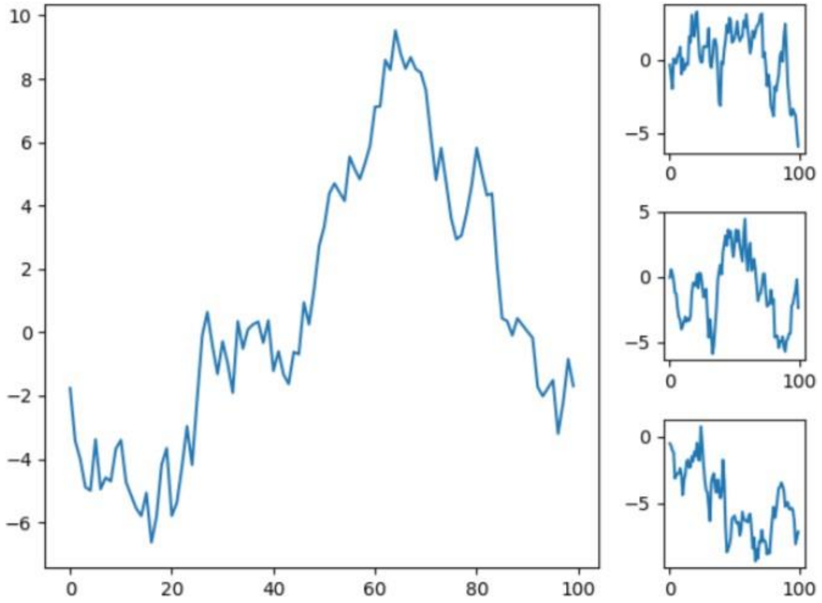
# Layouts: GridSpec

**matplotlib.gridspec.GridSpec(nrows, ncols)** specifies the geometry of the grid in which a subplot will be placed.

**Example:**



```
...
gs = matplotlib.gridspec.GridSpec(3, 4)
ax1 = plt.subplot(gs[:3, :3])
ax2 = plt.subplot(gs[0, 3])
ax3 = plt.subplot(gs[1, 3])
ax4 = plt.subplot(gs[2, 3])
ax1.plot(series[0])
ax2.plot(series[1])
ax3.plot(series[2])
ax4.plot(series[3])
plt.tight_layout()
...
```

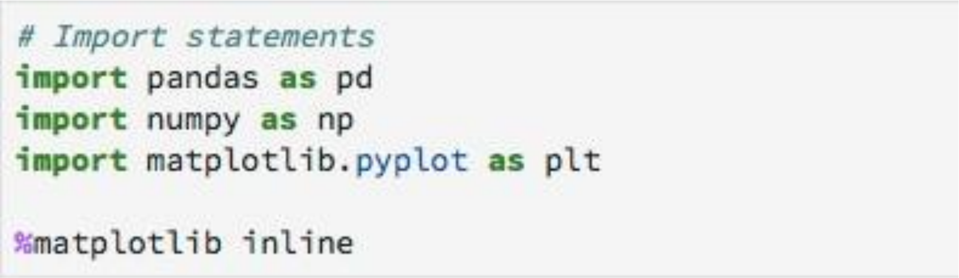


## Activity 7

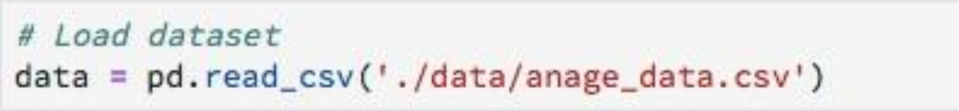
# Creating a Scatter Plot with Marginal Histograms

### Objective

In this activity, we will make use of **GridSpec** to visualize a **scatter plot** with **marginal histograms** on the same figure.

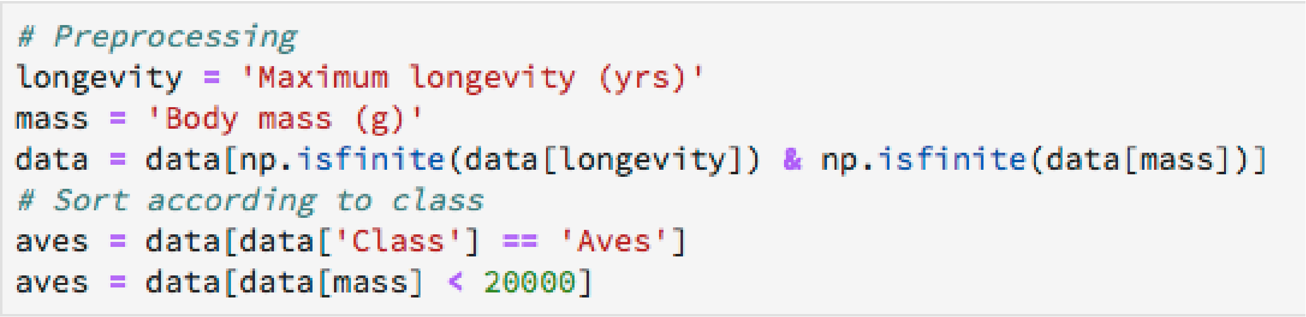


```
# Import statements
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

%matplotlib inline
```

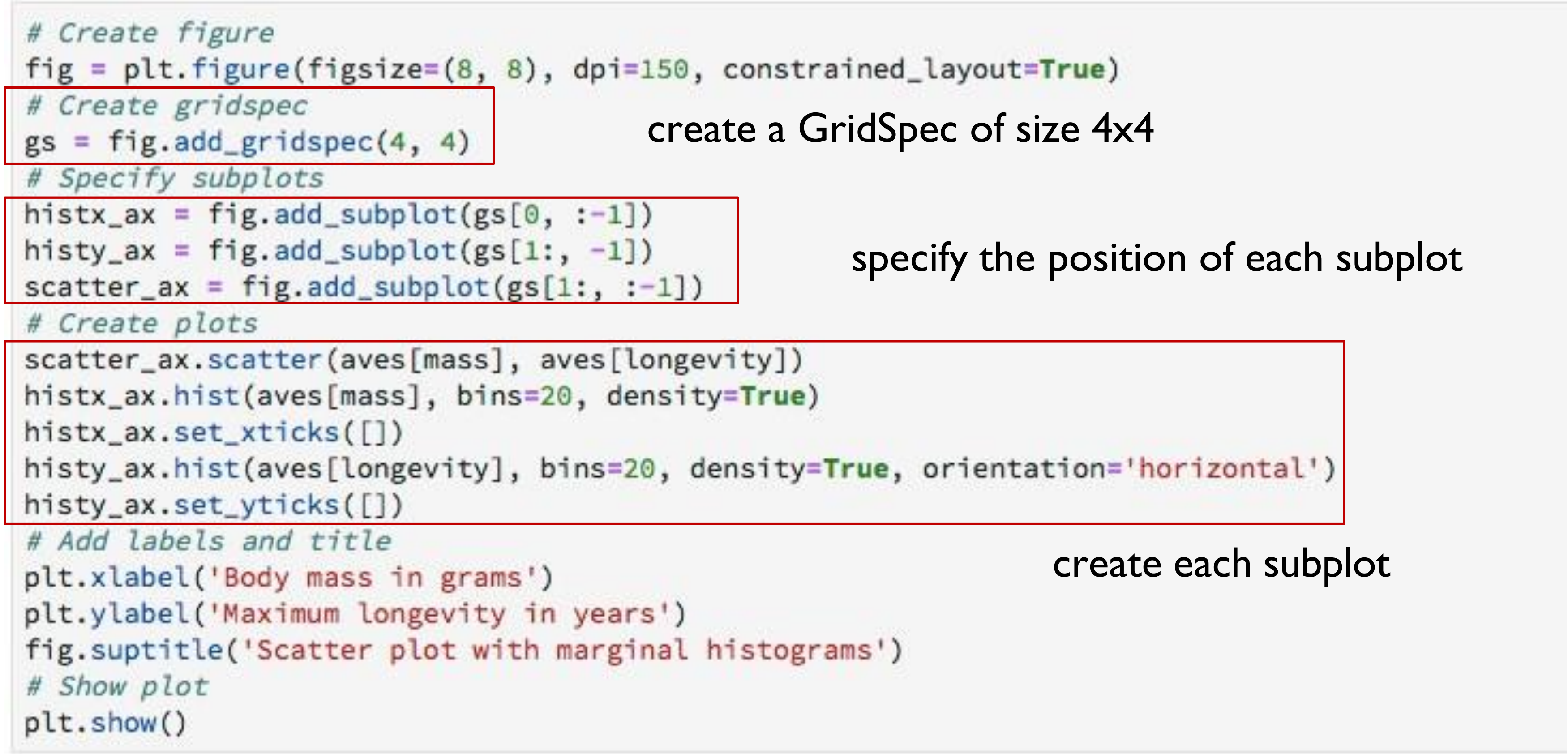


```
# Load dataset
data = pd.read_csv('./data/anage_data.csv')
```

Load data and filter the data



```
# Preprocessing
longevity = 'Maximum longevity (yrs)'
mass = 'Body mass (g)'
data = data[np.isfinite(data[longevity]) & np.isfinite(data[mass])]
# Sort according to class
aves = data[data['Class'] == 'Aves']
aves = data[data[mass] < 20000]
```



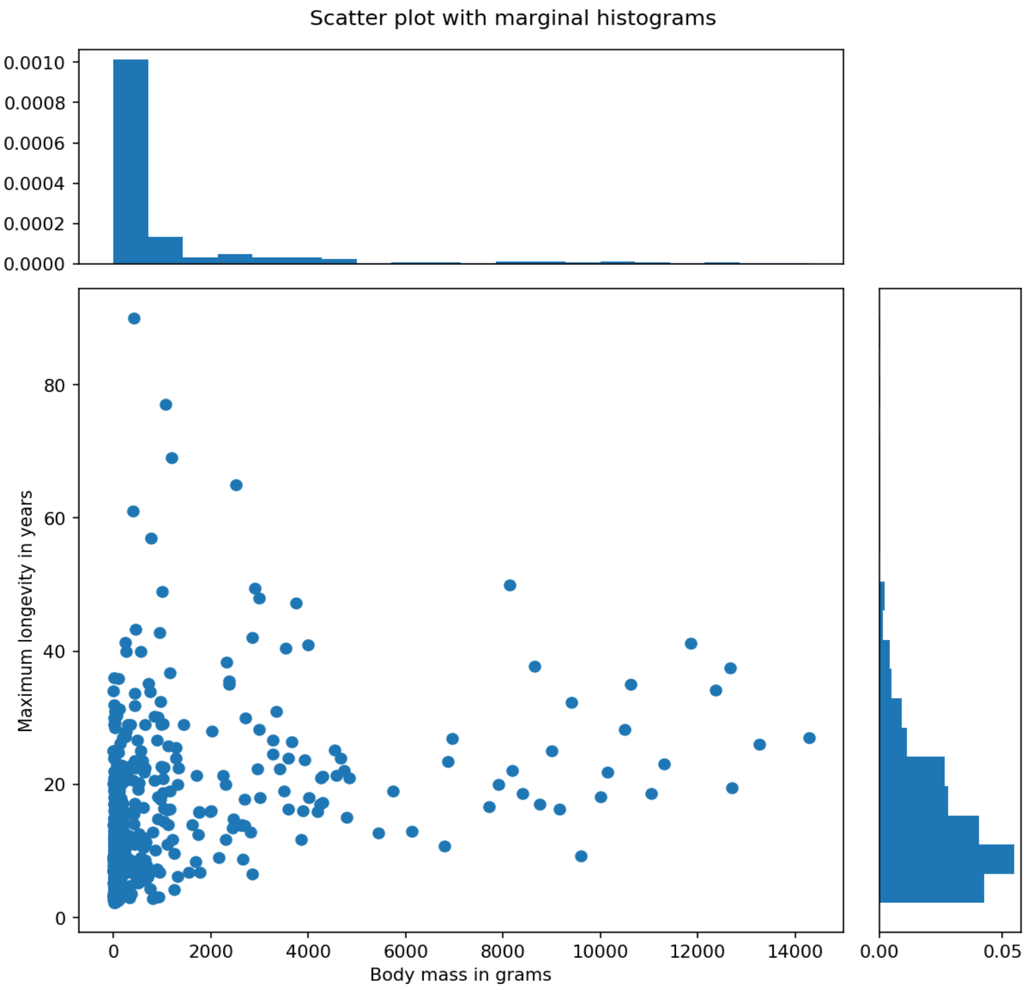
```
# Create figure
fig = plt.figure(figsize=(8, 8), dpi=150, constrained_layout=True)
# Create gridspec
gs = fig.add_gridspec(4, 4)
# Specify subplots
histx_ax = fig.add_subplot(gs[0, :-1])
histy_ax = fig.add_subplot(gs[1:, -1])
scatter_ax = fig.add_subplot(gs[1:, :-1])
# Create plots
scatter_ax.scatter(aves[mass], aves[longevity])
histx_ax.hist(aves[mass], bins=20, density=True)
histx_ax.set_xticks([])
histy_ax.hist(aves[longevity], bins=20, density=True, orientation='horizontal')
histy_ax.set_yticks([])
# Add labels and title
plt.xlabel('Body mass in grams')
plt.ylabel('Maximum longevity in years')
fig.suptitle('Scatter plot with marginal histograms')
# Show plot
plt.show()
```

create a GridSpec of size 4x4

specify the position of each subplot

create each subplot

# Expected view



How to change the position of subplots?

# Hands-on time

1. Using the code in Activity 5, understand how to create histogram and box plots to show the distribution of a variable. Execute the code and recreate the chart.

2. Using the code in Activity 6, understand how to create a scatter plot with multiple data series. Execute the code and recreate the chart.

3. Using the code in Activity 7, understand how to create a grid on a figure, and create subplot in the grid. Execute the code and recreate the chart.

Check here for the detailed descriptions of functions:

https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.html

# Summary

# What we have learnt today?

A detailed introduction to Matplotlib – one of the most popular visualization libraries for Python.

Techniques we have learnt:

- Line chart
- Bar chart
- Stacked bar chart
- Stacked area chart
- Histogram
- Box plot
- Scatterplot
- Setting layout