Graphical Exploratory Data Analysis Exercises

- For this group of exercises We are going to use you will use a classic data set collected by botanist Edward Anderson and made famous by Ronald Fisher, one of the most prolific statisticians in history. Anderson carefully measured the anatomical properties of samples of three different species of iris, Iris setosa, Iris versicolor, and Iris virginica.
- The full data set is available as part of scikit-learn.
- To make it easy we downloaded the data set as data/iris.csv
- We also created numpy array from samples of the data existing in this csv file
- Here, you will work with his measurements of petal length

Exercise 1: Creating a histogram

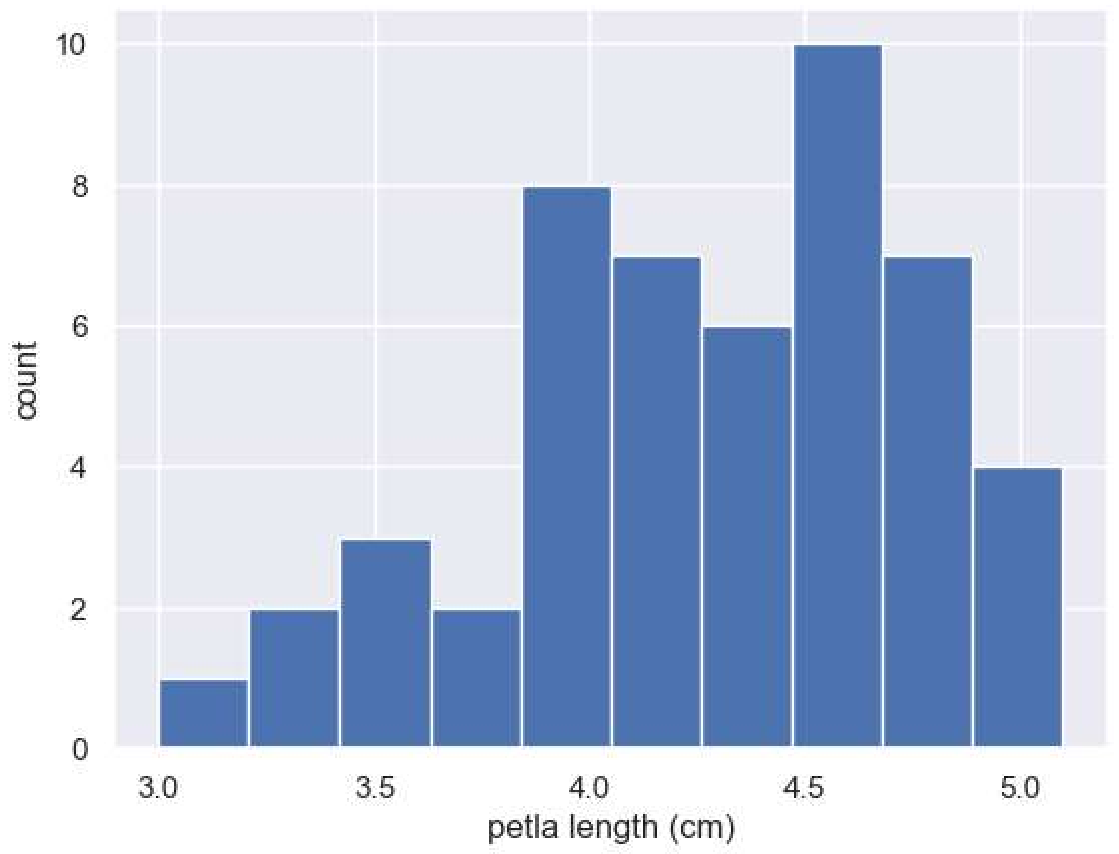
Instructions

- Load the data/iris.csv to a data frame df
- display the head of df to look at the columns
- Use the following numpy array which has been created by selecting a sample from the column. This is to make it easy. versicolor_petal_length = np.array([4.7, 4.5, 4.9, 4. , 4.6, 4.5, 4.7, 3.3, 4.6, 3.9, 3.5,

```
    4.2,  4. ,  4.7,  3.6,  4.4,  4.5,  4.1,  4.5,  3.9,  4.8,  4. ,
    4.9,  4.7,  4.3,  4.4,  4.8,  5. ,  4.5,  3.5,  3.8,  3.7,  3.9,
    5.1,  4.5,  4.5,  4.7,  4.4,  4.1,  4. ,  4.4,  4.6,  4. ,  3.3,
    4.2,  4.2,  4.2,  4.3,  3. ,  4.1])
```

- Import matplotlib.pyplot and seaborn as their usual aliases (plt and sns).
- Use seaborn to set the plotting defaults.
- Plot a histogram of the Iris versicolor petal lengths using plt.hist() and the provided NumPy array versicolor_petal_length.
- Label the axes. Don't forget that you should always include units in your axis labels.
- Your y-axis label is just 'count'.
- Your x-axis label is 'petal length (cm)'. The units are essential!
- Show the histogram using plt.show().

In [1]:
```
# Import plotting modules and pandas
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns


#use the numpy that we mentioned in the exercise
versicolor_petal_length = np.array([ 4.7,  4.5,  4.9,  4. ,  4.6,  4.5,  4.7,  3.
        4.2,  4. ,  4.7,  3.6,  4.4,  4.5,  4.1,  4.5,  3.9,  4.8,  4. ,
        4.9,  4.7,  4.3,  4.4,  4.8,  5. ,  4.5,  3.5,  3.8,  3.7,  3.9,
        5.1,  4.5,  4.5,  4.7,  4.4,  4.1,  4. ,  4.4,  4.6,  4. ,  3.3,
        4.2,  4.2,  4.2,  4.3,  3. ,  4.1])

# Set default Seaborn style
sns.set()


# Plot histogram of versicolor petal lengths
_=plt.hist(versicolor_petal_length)



# Label axes
_=plt.xlabel('petla length (cm)')
_=plt.ylabel('count')

# Show histogram
plt.show()
```



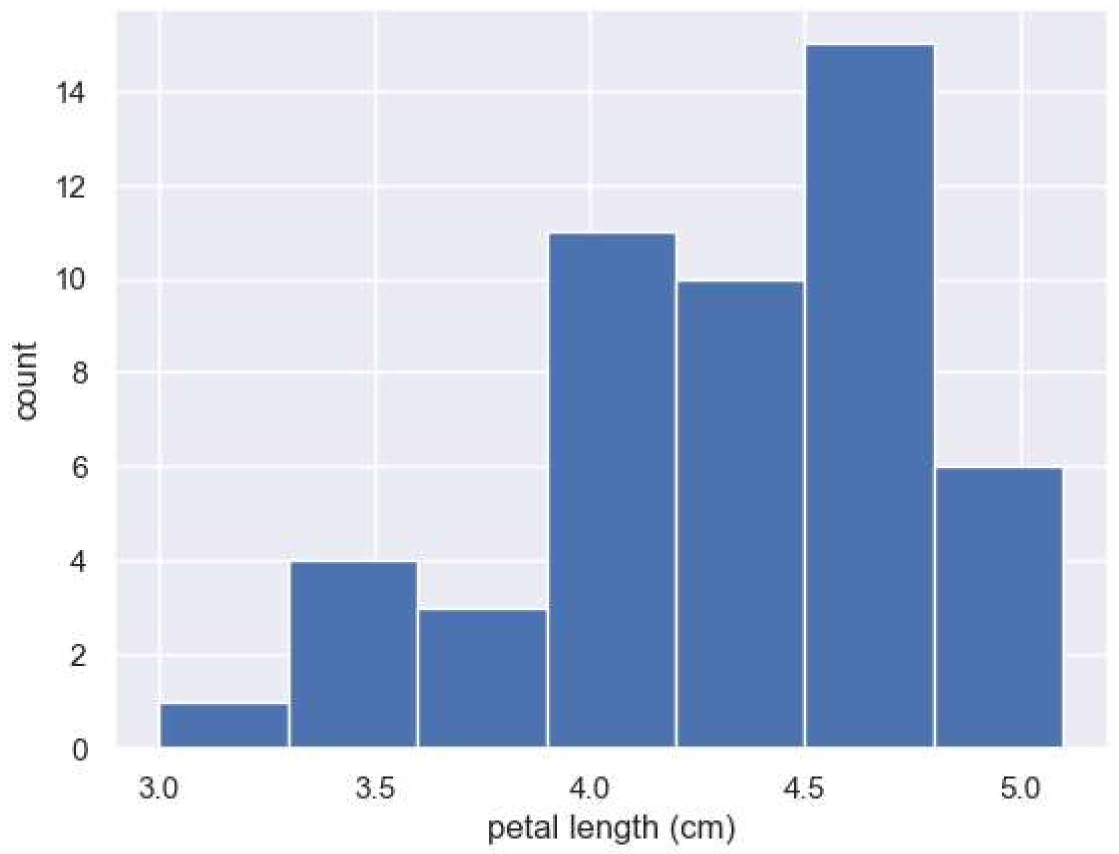
In [20]:
```
print('I am Preston')
```

```
I am Preston
```

Exericse 2: Adding bins to the histogram

- Instructions
- Continuation to the Exercise 1
- Import numpy as np. This gives access to the square root function, np.sqrt().
- Determine how many data points you have using len().
- Compute the number of bins using the square root rule.
- Convert the number of bins to an integer using the built in int() function.
- Generate the histogram and make sure to use the bins keyword argument.
- Show the plot

In [4]:
```
# Import numpy
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns


#use the numpy mentioned in the exercise
versicolor_petal_length = np.array([ 4.7,  4.5,  4.9,  4. ,  4.6,  4.5,  4.7,  3.
        4.2,  4. ,  4.7,  3.6,  4.4,  4.5,  4.1,  4.5,  3.9,  4.8,  4. ,
        4.9,  4.7,  4.3,  4.4,  4.8,  5. ,  4.5,  3.5,  3.8,  3.7,  3.9,
        5.1,  4.5,  4.5,  4.7,  4.4,  4.1,  4. ,  4.4,  4.6,  4. ,  3.3,
        4.2,  4.2,  4.2,  4.3,  3. ,  4.1])


# Compute number of data points: n_data
n_data=len(versicolor_petal_length)

# Number of bins is the square root of number of data points: n_bins
n_bins=np.sqrt(n_data)


# Convert number of bins to integer: n_bins
n_bins=int(n_bins)


# Plot the histogram

_=plt.hist(versicolor_petal_length,bins=n_bins)

# Label axes
_=plt.xlabel('petal length (cm)')
_=plt.ylabel('count')
# Show histogram
plt.show()
```



In [21]:
```
print('I am Preston')
```

```
I am Preston
```

Exercise 3: Bee Swarm Plot

- Make a bee swarm plot of the iris petal lengths. Your x-axis should contain each of the three species, and the y-axis the petal lengths. A data frame containing the data is in your namespace as df.
- Instructions
- inspect the DataFrame df using df.head(). This will let you identify which column names you need to pass as the x and y keyword arguments in your call to sns.swarmplot().
- Use sns.swarmplot() to make a bee swarm plot from the DataFrame containing the Fisher iris data set, df. The x-axis should contain each of the three species, and the y-axis should contain the petal lengths.
- Label the axes.
- Show your plot.

In [6]:
```
# Import plotting modules and pandas
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns


#Load the data/iris.csv to a data frame df
df=pd.read_csv('data/iris.csv')

#show the head to see the various columns

df.head()
```

Out[6]:

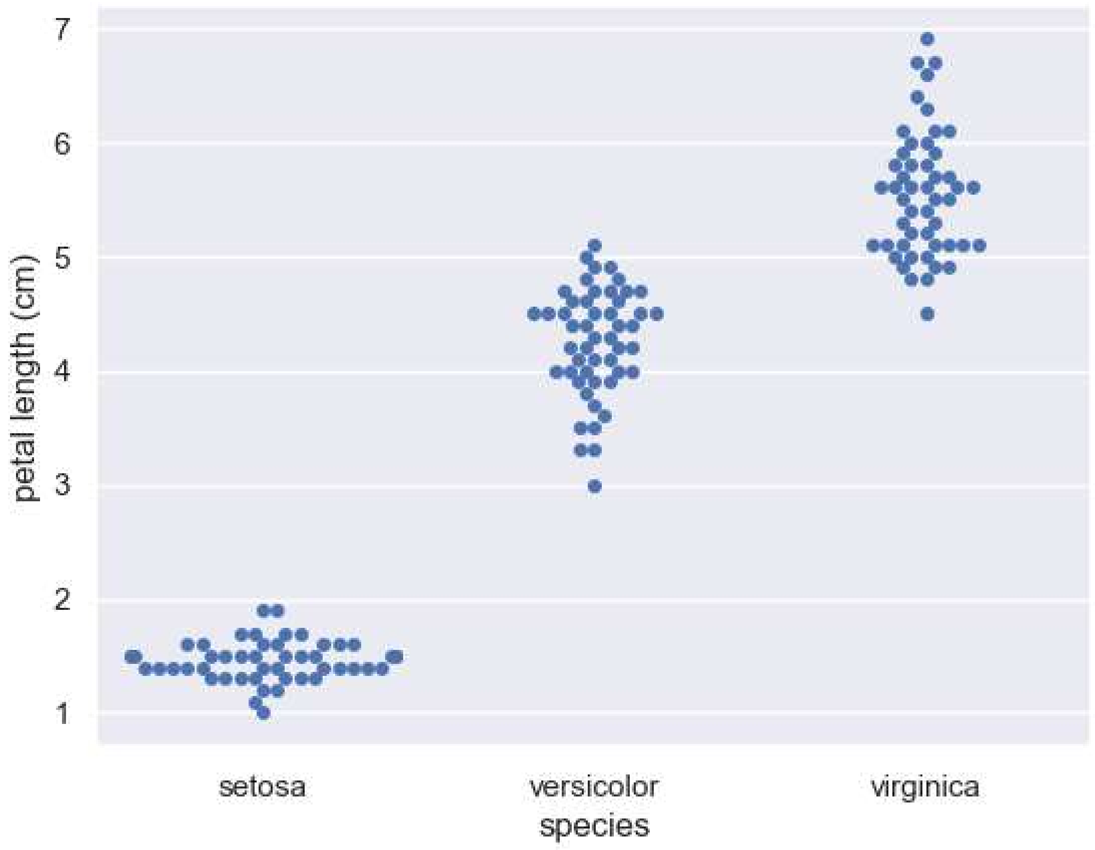
	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

In [22]:
```
print('I am Preston')
```

```
I am Preston
```

In [9]:
```
# Create bee swarm plot with Seaborn's default settings
sns.swarmplot(x='species',y='petal length (cm)',data=df)

# Label the axes
plt.xlabel('species')
plt.ylabel('petal length (cm)')


# Show the plot
plt.show()
```

```
C:\Users\prest\anaconda3\lib\site-packages\seaborn\categorical.py:3540: UserWar
ning: 10.0% of the points cannot be placed; you may want to decrease the size o
f the markers or use stripplot.
  warnings.warn(msg, UserWarning)
```



In [23]:
```
print('I am Preston')
```

```
I am Preston
```

Exercise 4: Computing the ECDF

- In this exercise, you will write a function that takes as input a 1D array of data and then returns the x and y values of the ECDF. You will use this function over and over again throughout this course and its sequel. ECDFs are among the most important plots in statistical analysis
- Instructions:
    - Define a function with the signature ecdf(data). Within the function definition,
        - Compute the number of data points, n, using the len() function.
        - The x-values are the sorted data. Use the np.sort() function to perform the sorting.
        - The y data of the ECDF go from 1/n to 1 in equally spaced increments. You can construct this using np.arange(). Remember, however, that the end value in np.arange() is not inclusive. Therefore, np.arange() will need to go from 1 to n+1. Be sure to divide this by n.
        - The function returns the values x and y.

In [10]:
```
#import numpy
import numpy as np

#define a function name it ecdf that will take data as argument
def ecdf(data):

    #add a comment using the triple quotes
    """Compute BSDF for a one-dimensional array of measurements."""

    # Number of data points: n
    n=len(data)

    # x-data for the ECDF: x
    x=np.sort(data)

    # y-data for the ECDF: y
    y=np.array(1,n+1) /n

#add a return statement that
    return x,y
```

In [24]:
```python
print('I am Preston')
```

```
I am Preston
```

Exercise 5: Comparison of ECDFs

- Instructions
  - Compute ECDFs for each of the three species using your ecdf() function. The variables setosa_petal_length, versicolor_petal_length, and virginica_petal_length are all in your namespace.
  - Unpack the ECDFs into x_set, y_set, x_vers, y_vers and x_virg, y_virg, respectively.
  - Plot all three ECDFs on the same plot as dots. To do this, you will need three plt.plot() commands.
  - Assign the result of each to _.
  - Specify 2% margins.
  - A legend and axis labels have been added for you!
- Use the following numpy arrays

setosa_petal_length = np.array([1.4, 1.4, 1.3, 1.5, 1.4, 1.7, 1.4, 1.5, 1.4, 1.5, 1.5, 1.6, 1.4, 1.1, 1.2, 1.5, 1.3, 1.4, 1.7, 1.5, 1.7, 1.5, 1. , 1.7, 1.9, 1.6, 1.6, 1.5, 1.4, 1.6, 1.6, 1.5, 1.5, 1.4, 1.5, 1.2, 1.3, 1.5, 1.3, 1.5, 1.3, 1.3, 1.3, 1.6, 1.9, 1.4, 1.6, 1.4, 1.5, 1.4])

versicolor_petal_length = np.array([4.7, 4.5, 4.9, 4. , 4.6, 4.5, 4.7, 3.3, 4.6, 3.9, 3.5, 4.2, 4. , 4.7, 3.6, 4.4, 4.5, 4.1, 4.5, 3.9, 4.8, 4. , 4.9, 4.7, 4.3, 4.4, 4.8, 5. , 4.5, 3.5, 3.8, 3.7, 3.9, 5.1, 4.5, 4.5, 4.7, 4.4, 4.1, 4. , 4.4, 4.6, 4. , 3.3, 4.2, 4.2, 4.2, 4.3, 3. , 4.1])

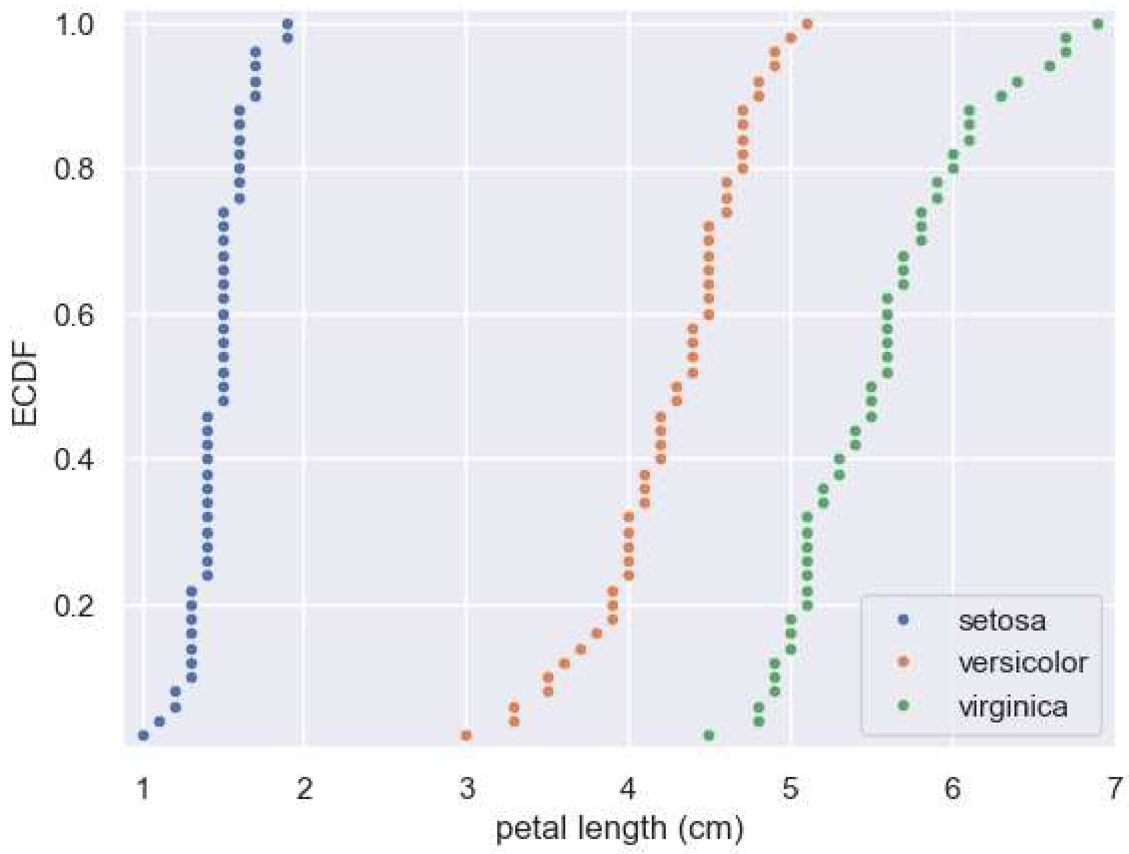
virginica_petal_length = np.array([6. , 5.1, 5.9, 5.6, 5.8, 6.6, 4.5, 6.3, 5.8, 6.1, 5.1, 5.3, 5.5, 5. , 5.1, 5.3, 5.5, 6.7, 6.9, 5. , 5.7, 4.9, 6.7, 4.9, 5.7, 6. , 4.8, 4.9, 5.6, 5.8, 6.1, 6.4, 5.6, 5.1, 5.6, 6.1, 5.6, 5.5, 4.8, 5.4, 5.6, 5.1, 5.1, 5.9, 5.7, 5.2, 5. , 5.2, 5.4, 5.1])

In [25]:
```python
print('I am Preston')
```

```
I am Preston
```

In [18]:
```
#import the required libraries
import numpy as np
import matplotlib.pyplot as plt


#use the numpy arrays mentioned in the exercise
setosa_petal_length = np.array([ 1.4,  1.4,  1.3,  1.5,  1.4,  1.7,  1.4,  1.5,
        1.6,  1.4,  1.1,  1.2,  1.5,  1.3,  1.4,  1.7,  1.5,  1.7,  1.5,
        1. ,  1.7,  1.9,  1.6,  1.6,  1.5,  1.4,  1.6,  1.6,  1.5,  1.5,
        1.4,  1.5,  1.2,  1.3,  1.5,  1.3,  1.5,  1.3,  1.3,  1.3,  1.6,
        1.9,  1.4,  1.6,  1.4,  1.5,  1.4])

versicolor_petal_length = np.array([ 4.7,  4.5,  4.9,  4. ,  4.6,  4.5,  4.7,  3.
        4.2,  4. ,  4.7,  3.6,  4.4,  4.5,  4.1,  4.5,  3.9,  4.8,  4. ,
        4.9,  4.7,  4.3,  4.4,  4.8,  5. ,  4.5,  3.5,  3.8,  3.7,  3.9,
        5.1,  4.5,  4.5,  4.7,  4.4,  4.1,  4. ,  4.4,  4.6,  4. ,  3.3,
        4.2,  4.2,  4.2,  4.3,  3. ,  4.1])

virginica_petal_length = np.array([ 6. ,  5.1,  5.9,  5.6,  5.8,  6.6,  4.5,  6.
        5.3,  5.5,  5. ,  5.1,  5.3,  5.5,  6.7,  6.9,  5. ,  5.7,  4.9,
        6.7,  4.9,  5.7,  6. ,  4.8,  4.9,  5.6,  5.8,  6.1,  6.4,  5.6,
        5.1,  5.6,  6.1,  5.6,  5.5,  4.8,  5.4,  5.6,  5.1,  5.1,  5.9,
        5.7,  5.2,  5. ,  5.2,  5.4,  5.1])



#define the function call it ecdf and it take an argument data
def ecdf(data):
    """Compute ECDF for a one-dimensional array of measurements."""

    # Number of data points: n
    n=len(data)

    # x-data for the ECDF: x
    x=np.sort(data)

    # y-data for the ECDF: y
    y=np.arange(1,n+1) /n

    #add a return statement
    return x,y

#Compute ECDFs
x_set,y_set=ecdf(setosa_petal_length)
x_vers,y_vers=ecdf(versicolor_petal_length)
x_virg,y_virg=ecdf(virginica_petal_length)


# Plot all ECDFs on the same plot
_=plt.plot(x_set,y_set,marker='.',linestyle='none')
_=plt.plot(x_vers,y_vers,marker='.',linestyle='none')
_=plt.plot(x_virg,y_virg,marker='.',linestyle='none')


# Make nice margins
plt.margins(0.02)
```

```
# Annotate the plot add a legend locate it at the lower right
plt.legend(('setosa','versicolor','virginica'),loc='lower right')
_=plt.xlabel('petal length (cm)')
_=plt.ylabel('ECDF')


# Display the plot
plt.show()
```



In [ ]:

In [19]:
```
print('I am Preston')
```
```
I am Preston
```

In [ ]: