Exercises_Quantative_Exploratory_Data_Analysis

We are going to use data/iris.csv or some numpy arrays that we pulled out from this data set. Same data sets that we used for Exercises_Exploratory_Data_Analysis

Exercise 1: Calculating Mean

- Instructions:
  - Use the following numpy array 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])
```

  - Calculate the mean of those values

In [7]:
```
#import numpy
import numpy as np


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

# Compute the mean: mean_length_vers
mean_length_vers=np.mean(versicolor_petal_length)

# Print the result with some nice formatting
print('I. versicolor:', mean_length_vers, 'cm')
```

```
I. versicolor: 4.26 cm
```

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

```
I am Preston
```

Exercise 2: Computing percentiles

- Instructions:
  - Use the following numpy array 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])
```

- Create percentiles, a NumPy array of percentiles you want to compute. These are the 2.5th, 25th, 50th, 75th, and 97.5th. You can do so by creating a list containing these ints/floats and convert the list to a NumPy array using np.array().
- For example, np.array([30, 50]) would create an array consisting of the 30th and 50th percentiles.
- Use np.percentile() to compute the percentiles of the petal lengths from the Iris versicolor samples.
- The variable versicolor_petal_length is in your namespace.
- Print the percentiles.

In [9]:
```
#import required libraries
import numpy as np
import seaborn as sns

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


# Specify array of percentiles: percentiles
percentiles=np.array([2.5,25,50,75,97.5])


# Compute percentiles: ptiles_vers

ptiles_vers=np.percentile(versicolor_petal_length,percentiles)

# Print the result
print(ptiles_vers)
```

```
[3.3    4.     4.35   4.6    4.9775]
```

In [26]:
```
print('I am Preston')
```

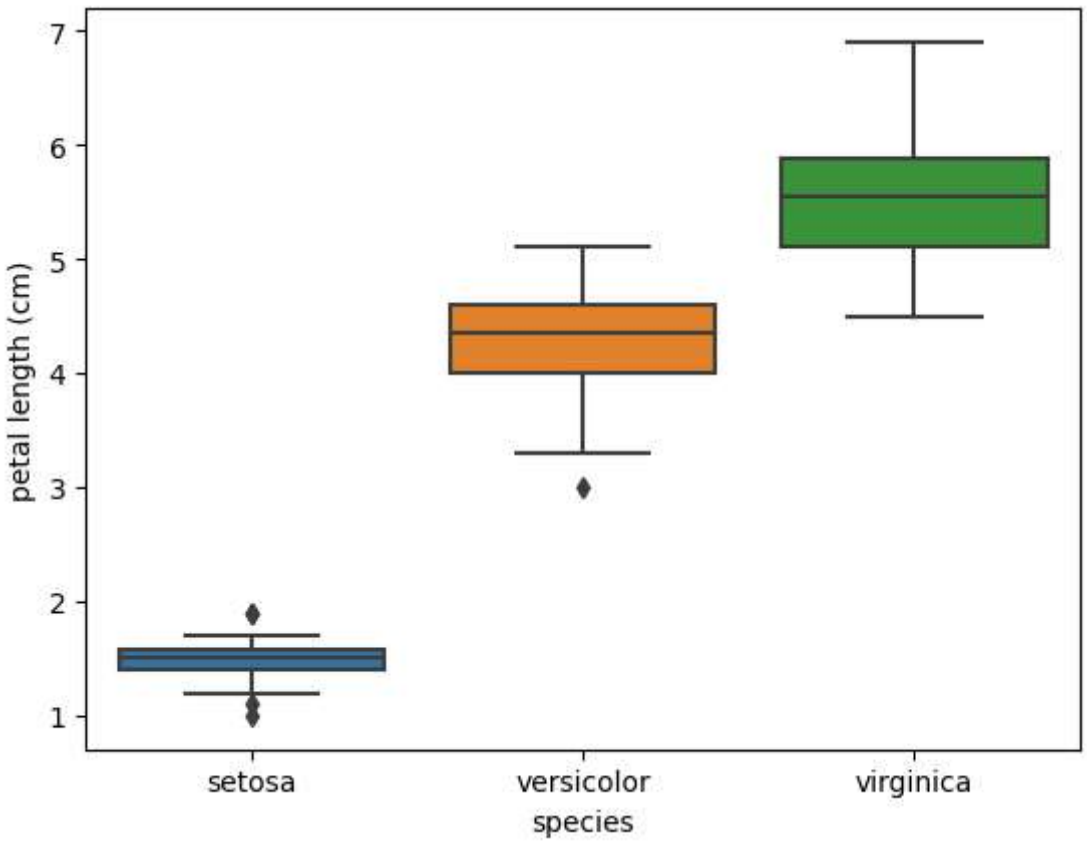
```
I am Preston
```

Exercise 3: BoxPlot and Whiskers

- Instructions:

- Make a box plot of the iris petal lengths. You have a pandas DataFrame, df, which contains the petal length data, in your namespace. Inspect the data frame df using df.head() to make sure you know what the pertinent columns are. Of course we can find the column names in another say.
- The set-up is exactly the same as for the bee swarm plot; you just call sns.boxplot() with the same keyword arguments as you would sns.swarmplot().
- The x-axis is 'species'
- The y-axis is 'petal length (cm)'.
- Don't forget to label your axes!
- Display the figure using the normal call.

In [12]:
```
#import required libraries
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt


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


# Create box plot with Seaborn's default settings
_=sns.boxplot(x='species',y='petal length (cm)', data=df)


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


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



In [27]:
```
print('I am Preston')
```

```
I am Preston
```

Exercise 4: Comparing percentiles to ECDF

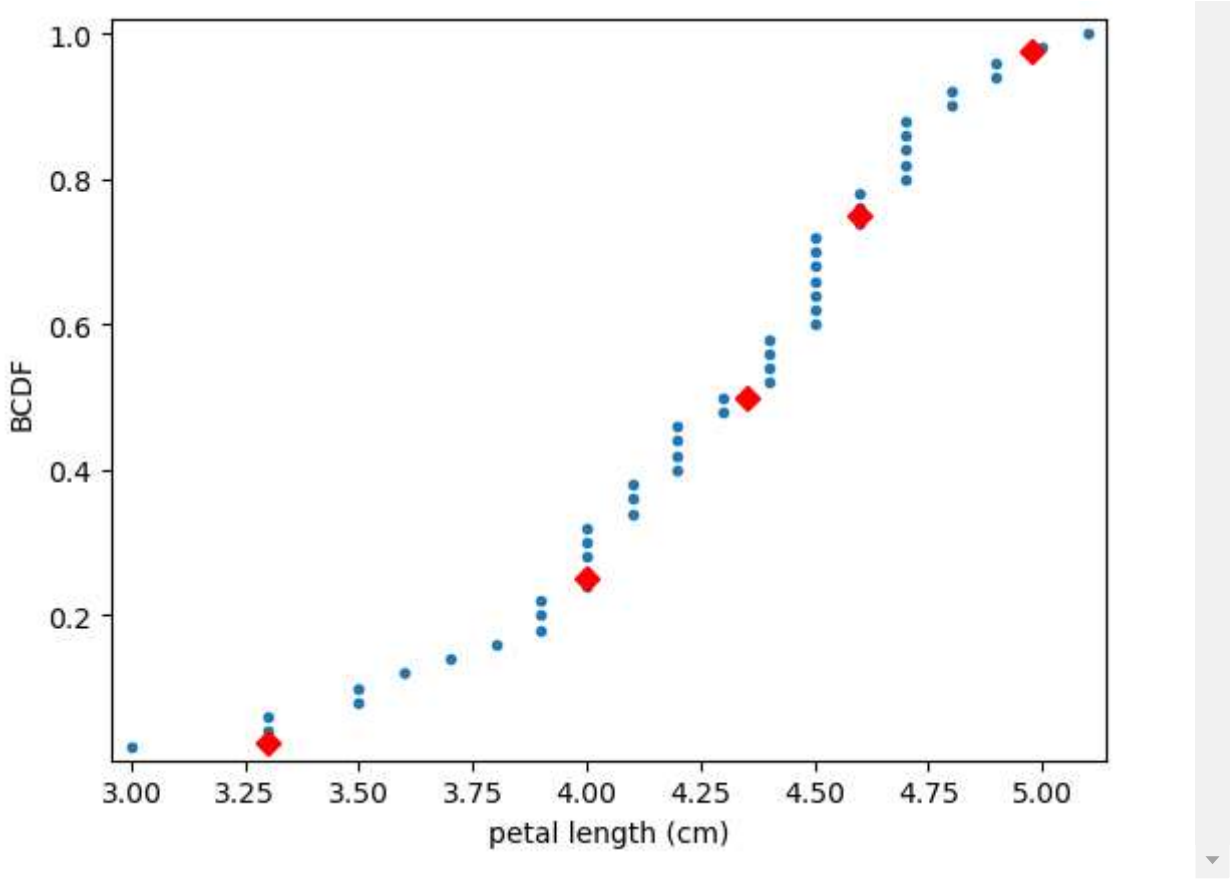
- Instructions
  - Plot the percentiles as red diamonds on the ECDF. Pass the x and y co-ordinates - ptiles_vers and percentiles/100 - as positional arguments and specify the marker='D', color='red' and linestyle='none' keyword arguments.
  - The argument for the y-axis - percentiles/100 has been specified for you.
  - Display the plot.
  - Use the following numpy arrays ptiles_vers = np.array([3.3, 4., 4.35, 4.6, 4.9775])

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

y_vers = np.array([0.02, 0.04, 0.06, 0.08, 0.1, 0.12, 0.14, 0.16, 0.18, 0.2, 0.22, 0.24, 0.26, 0.28, 0.3, 0.32, 0.34, 0.36, 0.38, 0.4, 0.42, 0.44, 0.46, 0.48, 0.5, 0.52, 0.54, 0.56, 0.58, 0.6, 0.62, 0.64, 0.66, 0.68, 0.7, 0.72, 0.74, 0.76, 0.78, 0.8, 0.82, 0.84, 0.86, 0.88, 0.9, 0.92, 0.94, 0.96, 0.98, 1.])

percentiles = np.array([2.5, 25., 50., 75., 97.5])

In [13]:
```
#import the required librarie
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np


#use the numpy arrays mentioned in the exercise
ptiles_vers = np.array([3.3, 4., 4.35, 4.6, 4.9775])

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

y_vers = np.array([0.02,  0.04,  0.06,  0.08,  0.1,  0.12,  0.14,  0.16,  0.18,
                   0.2,  0.22,  0.24,  0.26,  0.28,  0.3,  0.32,  0.34,  0.36,
                   0.38,  0.4,  0.42,  0.44,  0.46,  0.48,  0.5,  0.52,  0.54,
                   0.56,  0.58,  0.6,  0.62,  0.64,  0.66,  0.68,  0.7,  0.72,
                   0.74,  0.76,  0.78,  0.8,  0.82,  0.84,  0.86,  0.88,  0.9,
                   0.92,  0.94,  0.96,  0.98,  1.])

percentiles = np.array([2.5,  25.,  50.,  75.,  97.5])




# Plot the ECDF
_=plt.plot(x_vers,y_vers,'.')
plt.margins(0.02)
_=plt.xlabel('petal length (cm)')
_=plt.ylabel('BCDF')
# Overlay percentiles as red diamonds.
_=plt.plot(ptiles_vers,percentiles/100,marker='D',color='red',
          linestyle='none')



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



In [28]:
```
print('I am Preston')
```

```
I am Preston
```

Exercise 5: The standard deviation and the variance

- Instructions
  - Use the following numpy array 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])
```

  - Compute the variance of the data in the versicolor_petal_length array using np.var().
  - Print the square root of this value.
  - Compute the standard deviation of the data in the versicolor_petal_length array using np.std() and print the result.

In [14]:
```
#import required libraries
import numpy as np

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


# Compute the variance: variance

variance=np.var(versicolor_petal_length)

# Print the square root of the variance
print(np.sqrt(variance))


# Print the standard deviation
print(np.std(versicolor_petal_length))
```

```
0.4651881339845203
0.4651881339845203
```

In [29]:
```
print('I am Preston')
```

```
I am Preston
```

Exercise 6: Scattered Plot

- Instructions
  - Use the following numpy arrays 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,

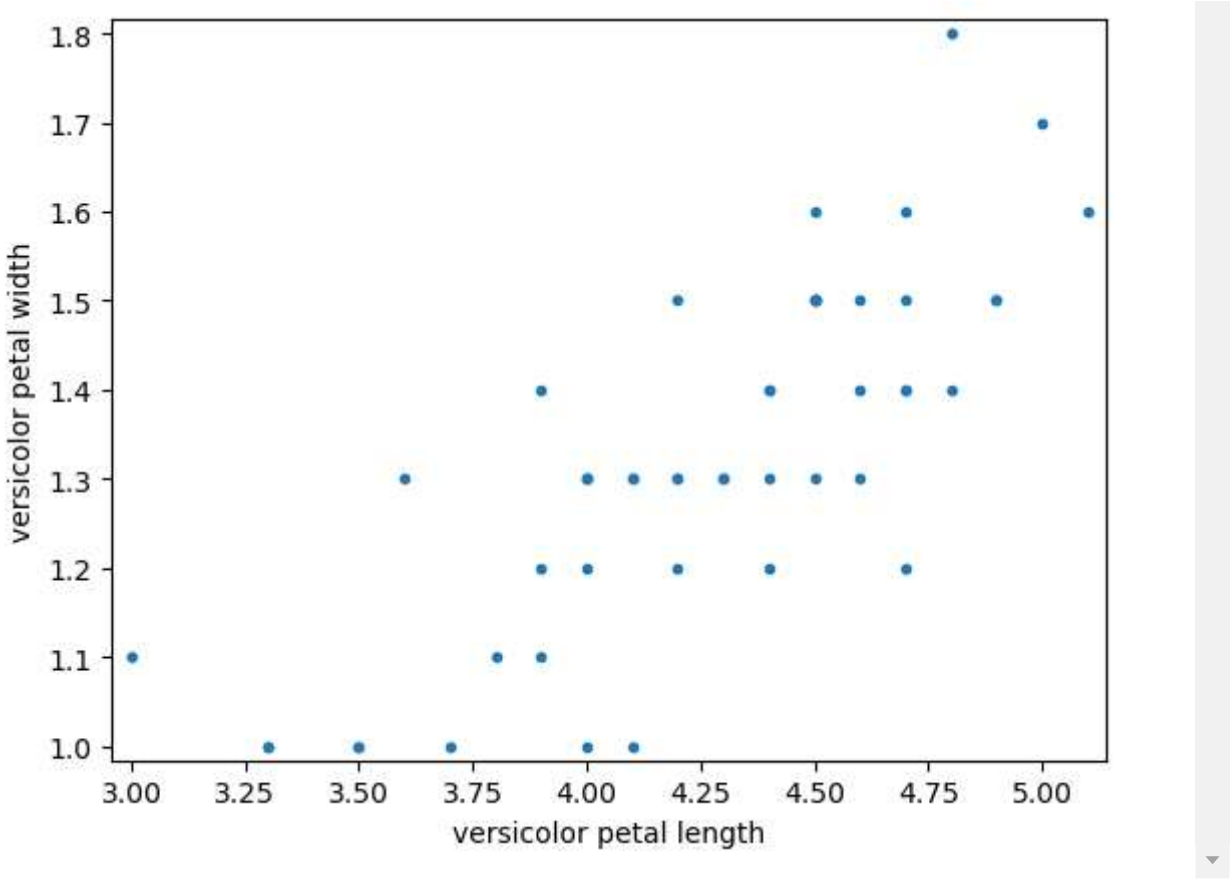
```
                                4.2,  4.,  4.7,  3.6,  4.4,  4.5,  4.
    1,  4.5,  3.9,  4.8,  4.,
                                4.9,  4.7,  4.3,  4.4,  4.8,  5.,  4.
    5,  3.5,  3.8,  3.7,  3.9,
                                5.1,  4.5,  4.5,  4.7,  4.4,  4.1,
     4.,  4.4,  4.6,  4.,  3.3,
                                4.2,  4.2,  4.2,  4.3,  3.,  4.1])
```

versicolor_petal_width = np.array([1.4, 1.5, 1.5, 1.3, 1.5, 1.3, 1.6, 1., 1.3, 1.4, 1., 1.5, 1., 1.4, 1.3, 1.4, 1.5, 1., 1.5, 1.1, 1.8, 1.3, 1.5, 1.2, 1.3, 1.4, 1.4, 1.7, 1.5, 1., 1.1, 1., 1.2, 1.6, 1.5, 1.6, 1.5, 1.3, 1.3, 1.3, 1.2, 1.4, 1.2, 1., 1.3, 1.2, 1.3, 1.3, 1.1, 1.3])

- Use plt.plot() with the appropriate keyword arguments to make a scatter plot of versicolor petal length (x-axis) versus petal width (y-axis).
- The variables versicolor_petal_length and versicolor_petal_width are already in your namespace.

- Do not forget to use the marker='.' and linestyle='none' keyword arguments.
- Specify 2% margins so no data are cut off.
- Label the axes.
- Display the plot.

In [15]:
```
#import required libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt


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

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



# Make a scatter plot
_=plt.plot(versicolor_petal_length,versicolor_petal_width,
          marker='.',linestyle='none')


# Set margins
_=plt.margins(0.02)


# Label the axes
_=plt.xlabel('versicolor petal length')
_=plt.ylabel('versicolor petal width')



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



In [30]:
```
print('I am Preston')
```

```
I am Preston
```

Exercise 7: Computing the covariance

- Instructions
  - The covariance may be computed using the Numpy function np.cov().
  - Use the following numpy arrays 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])
```

versicolor_petal_width = np.array([1.4, 1.5, 1.5, 1.3, 1.5, 1.3, 1.6, 1., 1.3, 1.4, 1., 1.5, 1., 1.4, 1.3, 1.4, 1.5, 1., 1.5, 1.1, 1.8, 1.3, 1.5, 1.2, 1.3, 1.4, 1.4, 1.7, 1.5, 1., 1.1, 1., 1.2, 1.6, 1.5, 1.6, 1.5, 1.3, 1.3, 1.3, 1.2, 1.4, 1.2, 1., 1.3, 1.2, 1.3, 1.3, 1.1, 1.3])

- Use np.cov() to compute the covariance matrix for the petal length (versicolor_petal_length) and width (versicolor_petal_width) of I. versicolor.
- Print the covariance matrix.
- Extract the covariance from entry [0,1] of the covariance matrix. Note that by symmetry, entry [1,0] is the same as entry [0,1].

- Print the covariance.

In [20]:
```
#import required libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt


#use the following numpy arrays
versicolor_petal_length = np.array([4.7,  4.5,  4.9,  4.,  4.6,  4.5,  4.7,  3.3,
                                    4.2,  4.,  4.7,  3.6,  4.4,  4.5,  4.1,  4.5,
                                    4.9,  4.7,  4.3,  4.4,  4.8,  5.,  4.5,  3.5,
                                    5.1,  4.5,  4.5,  4.7,  4.4,  4.1,  4.,  4.4,
                                    4.2,  4.2,  4.2,  4.3,  3.,  4.1])

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



# Compute the covariance matrix: covariance_matrix
covariance_matrix=np.cov(versicolor_petal_length,versicolor_petal_width)


# Print covariance matrix
print(covariance_matrix)


# Extract covariance of length and width of petals: petal_cov
petal_cov=covariance_matrix[0,1]


# Print the length/width covariance
print(petal_cov)
```

```
[[0.22081633 0.07310204]
 [0.07310204 0.03910612]]
0.07310204081632653
```

In [31]:
```
print('I am Preston')
```

```
I am Preston
```

Exercise 8: Computing the Pearson correlation coefficient

- Instructions
- In this exercise, you will write a function, pearson_r(x, y) that takes in two arrays and returns the Pearson correlation coefficient.
- You will then use this function to compute it for the petal lengths and widths of I. versicolor.

- Use the following numpy arrays 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])
```

versicolor_petal_width = np.array([1.4, 1.5, 1.5, 1.3, 1.5, 1.3, 1.6, 1., 1.3, 1.4, 1., 1.5, 1., 1.4, 1.3, 1.4, 1.5, 1., 1.5, 1.1, 1.8, 1.3, 1.5, 1.2, 1.3, 1.4, 1.4, 1.7, 1.5, 1., 1.1, 1., 1.2, 1.6, 1.5, 1.6, 1.5, 1.3, 1.3, 1.3, 1.2, 1.4, 1.2, 1., 1.3, 1.2, 1.3, 1.3, 1.1, 1.3])

- Note the Pearson correlation coefficient, also called the Pearson r, is often easier to interpret than the covariance. It is computed using the np.corrcoef() function. Like np.cov(), it takes two arrays as arguments and returns a 2D array.
- Steps:
  - Define a function with signature pearson_r(x, y).
  - Use np.corrcoef() to compute the correlation matrix of x and y (pass them to np.corrcoef() in that order).
  - The function returns entry [0,1] of the correlation matrix.
  - Compute the Pearson correlation between the data in the arrays versicolor_petal_length and versicolor_petal_width. Assign the result to r.
  - Print the result.

In [23]:
```
#import the required libraries
import numpy as np


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

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



#define a function name it pearson_r that will take two arguments x and y
def pearson_r(x, y):
    """Compute Pearson correlation coefficient between two arrays."""
    # Compute correlation matrix: corr_mat
    corr_mat=np.corrcoef(x,y)


    # Return entry [0,1]
    return corr_mat[0,1]



# Compute Pearson correlation coefficient for I. versicolor: r
r=pearson_r(versicolor_petal_length,versicolor_petal_width)


# Print the result
print(r)
```

```
0.7866680885228169
```

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

```
I am Preston
```

In [ ]:
```

```