In [2]:
```
import pandas as pd
```

In [19]:
```
import matplotlib.pyplot as plt
import seaborn as sns
plt.rcParams['figure.figsize']=(9,6)
plt.rcParams['figure.dpi']=80
%matplotlib inline

import warnings
warnings.filterwarnings('ignore')
import plotly.express as px
```

In [7]:
```
df = pd.read_csv("C:\\Users\\Akshay\\OneDrive\\Pictures\\Documents\\data.csv")
df
```

Out[7]:

	Unnamed: 0	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
0	1	5.1	3.5	1.4	0.2	setosa
1	2	4.9	3.0	1.4	0.2	setosa
2	3	4.7	3.2	1.3	0.2	setosa
3	4	4.6	3.1	1.5	0.2	setosa
4	5	5.0	3.6	1.4	0.2	setosa
...	...	...	...	...	...	...
145	146	6.7	3.0	5.2	2.3	virginica
146	147	6.3	2.5	5.0	1.9	virginica
147	148	6.5	3.0	5.2	2.0	virginica
148	149	6.2	3.4	5.4	2.3	virginica
149	150	5.9	3.0	5.1	1.8	virginica

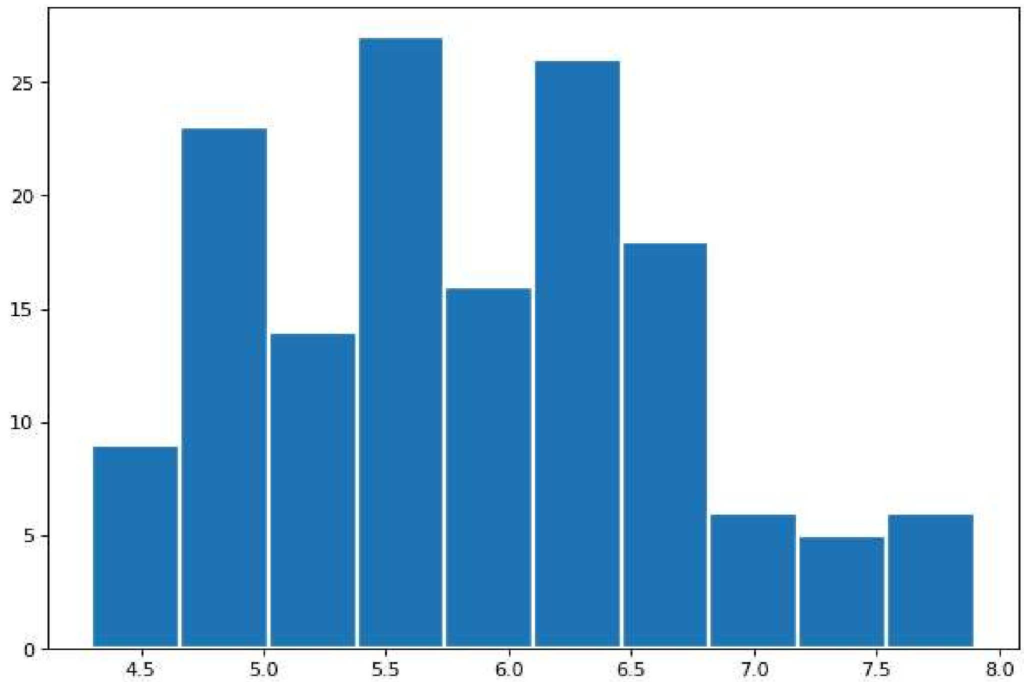
150 rows × 6 columns

DATA VISUALIZATION

We will first draw the histogram chart for each column.

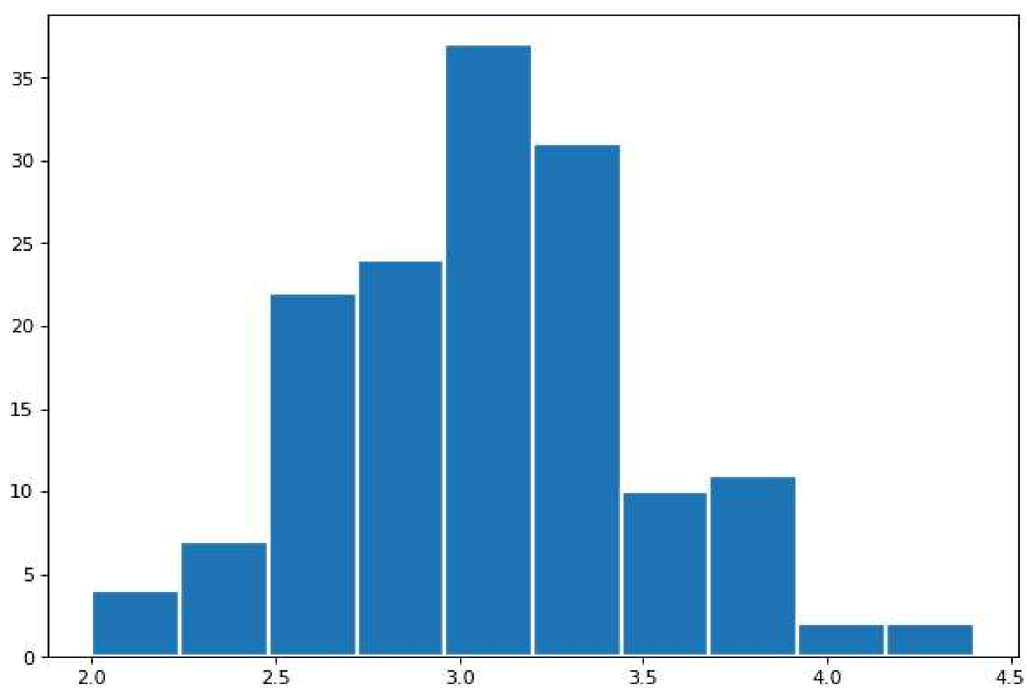
In [18]:
```
plt.hist(df['Sepal.Length'], bins=10, linewidth=2, edgecolor="white")
```

Out[18]:
```
(array([ 9., 23., 14., 27., 16., 26., 18.,  6.,  5.,  6.]),
 array([4.3 , 4.66, 5.02, 5.38, 5.74, 6.1 , 6.46, 6.82, 7.18, 7.54, 7.9 ]),
 <BarContainer object of 10 artists>)
```



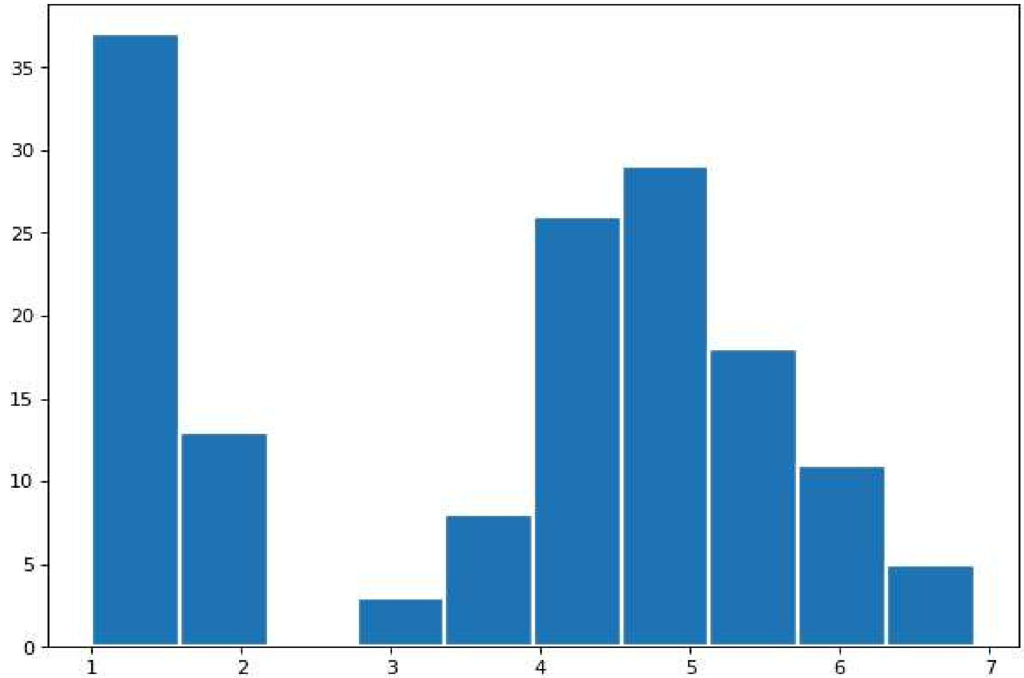
In [20]:
```
plt.hist(df['Sepal.Width'], bins=10, linewidth=2, edgecolor="white")
```

Out[20]:
```
(array([ 4.,  7., 22., 24., 37., 31., 10., 11.,  2.,  2.]),
 array([2.  , 2.24, 2.48, 2.72, 2.96, 3.2 , 3.44, 3.68, 3.92, 4.16, 4.4 ]),
 <BarContainer object of 10 artists>)
```



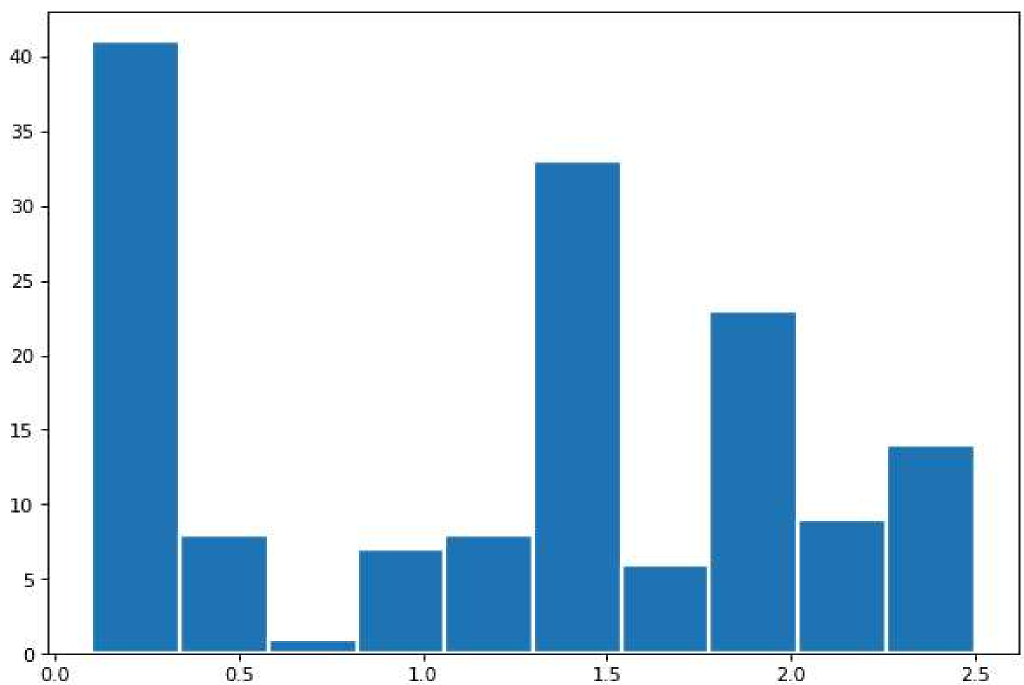
In [21]:
```
plt.hist(df['Petal.Length'], bins=10, linewidth=2, edgecolor="white")
```

Out[21]:
```
(array([37., 13.,  0.,  3.,  8., 26., 29., 18., 11.,  5.]),
 array([1.  , 1.59, 2.18, 2.77, 3.36, 3.95, 4.54, 5.13, 5.72, 6.31, 6.9 ]),
 <BarContainer object of 10 artists>)
```



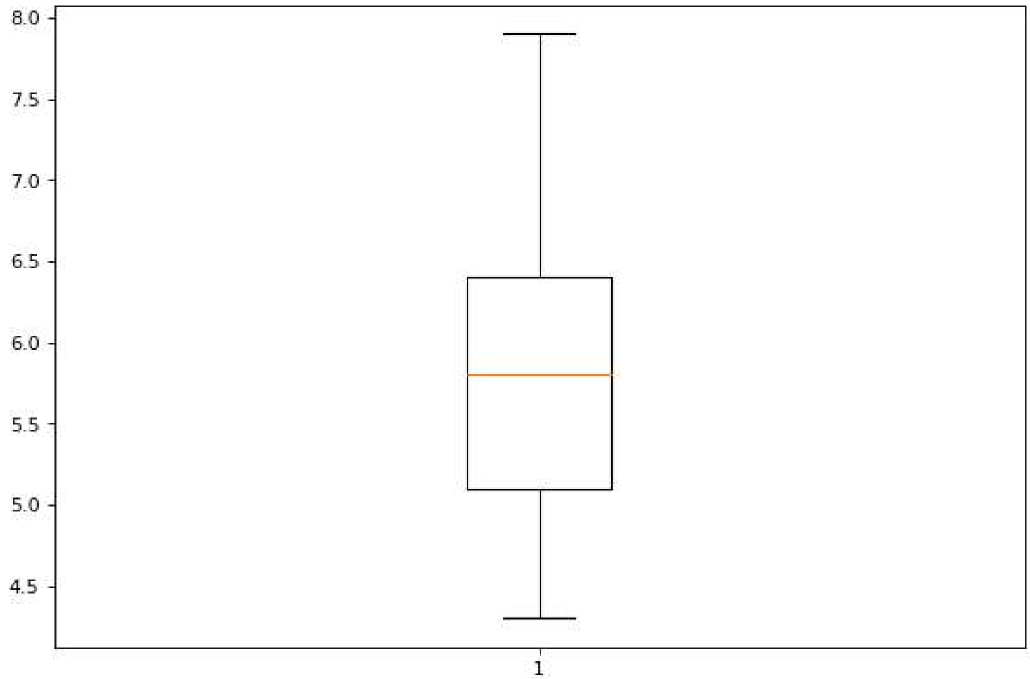
In [22]:
```
plt.hist(df['Petal.Width'], bins=10, linewidth=2, edgecolor="white")
```

Out[22]:
```
(array([41.,  8.,  1.,  7.,  8., 33.,  6., 23.,  9., 14.]),
 array([0.1 , 0.34, 0.58, 0.82, 1.06, 1.3 , 1.54, 1.78, 2.02, 2.26, 2.5 ]),
 <BarContainer object of 10 artists>)
```

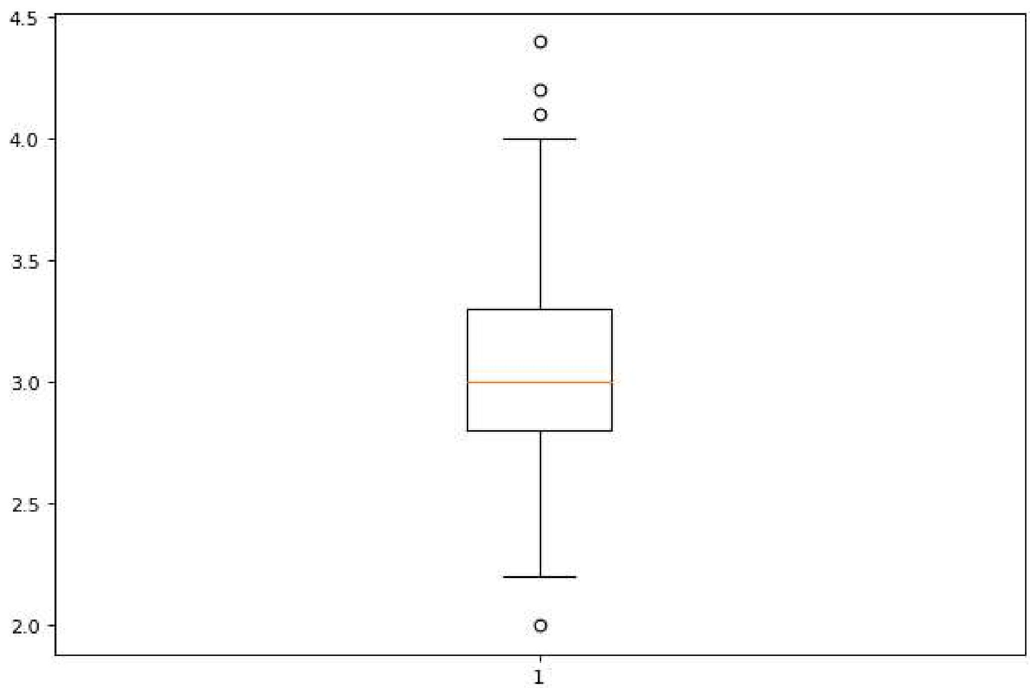


Boxplot for each column of given data.

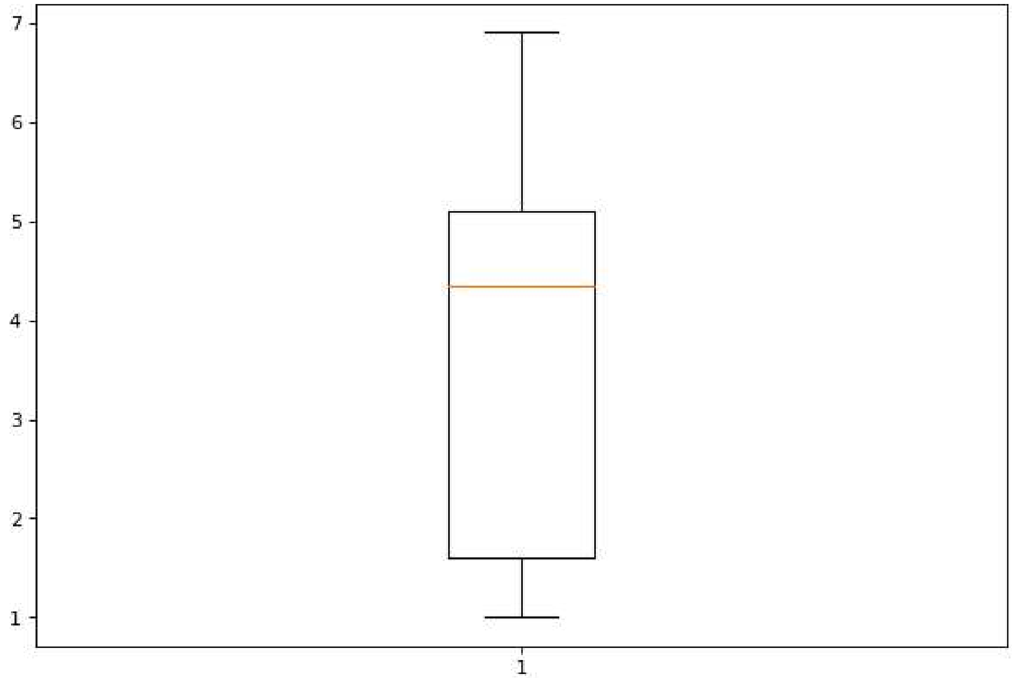
In [23]:
```
plt.boxplot(df['Sepal.Length'])
plt.show()
```



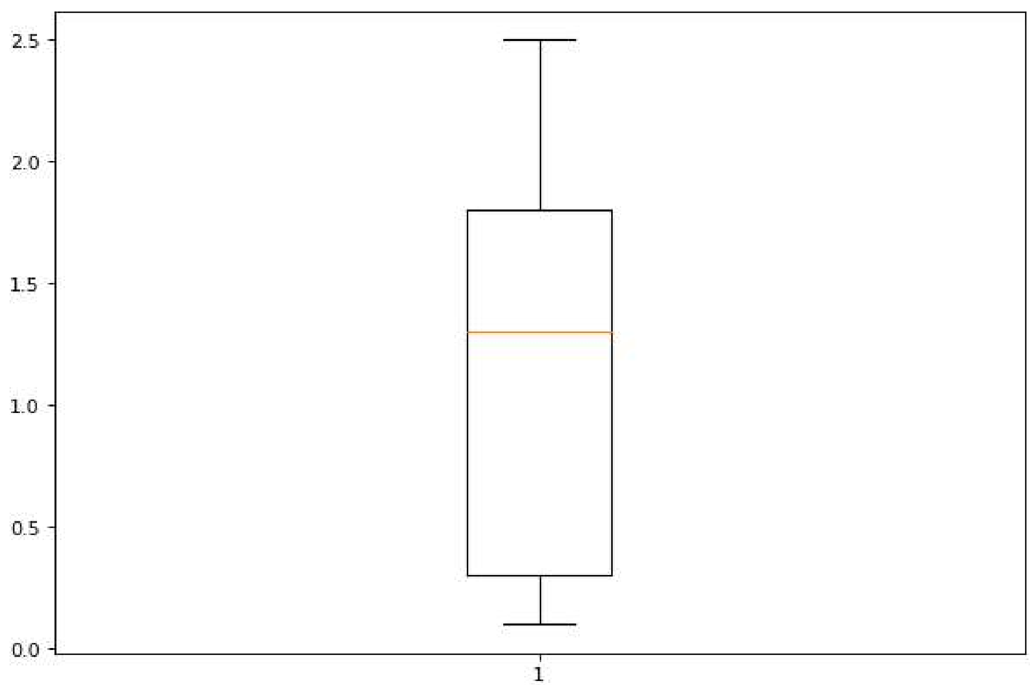
In [24]:
```
plt.boxplot(df['Sepal.Width'])
plt.show()
```



In [25]:
```
plt.boxplot(df['Petal.Length'])
plt.show()
```

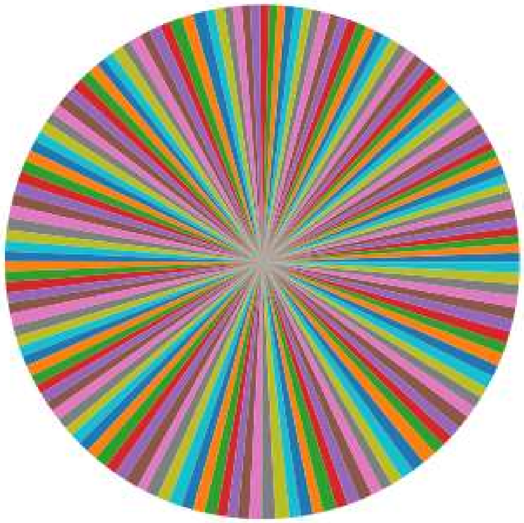


In [26]:
```
plt.boxplot(df['Petal.Width'])
plt.show()
```

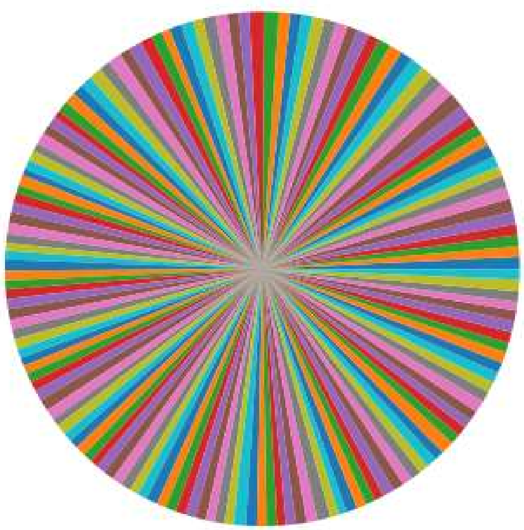


Pie plot for each column of given data.

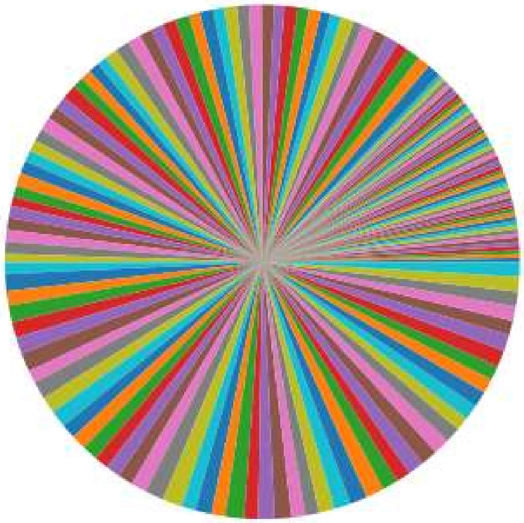
In [28]:
```
plt.pie(df['Sepal.Length'])
plt.show()
```



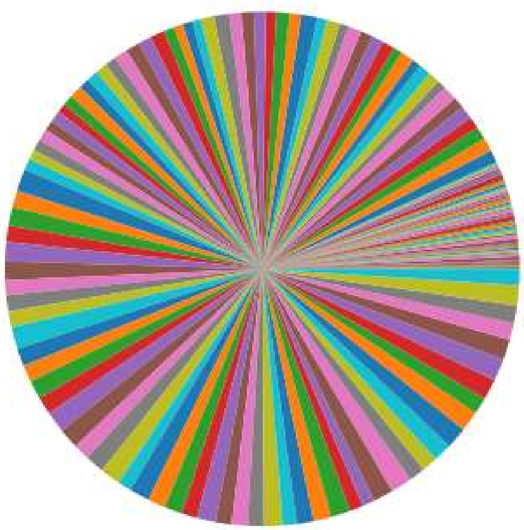
In [29]:
```
plt.pie(df['Sepal.Width'])
plt.show()
```



In [30]:
```
plt.pie(df['Petal.Length'])
plt.show()
```



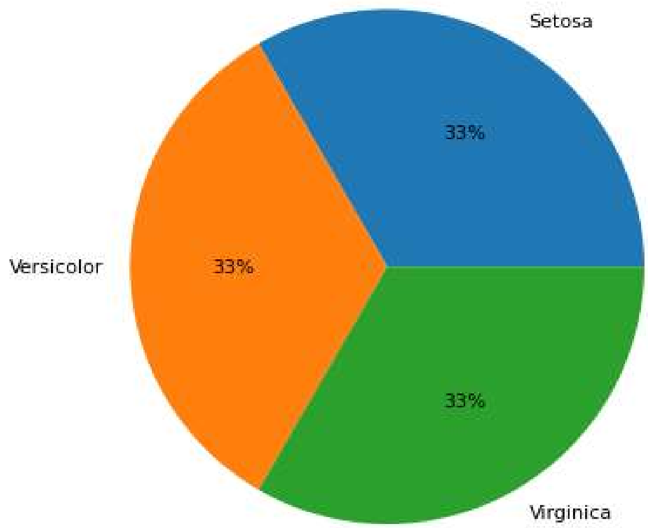
In [31]:
```
plt.pie(df['Petal.Width'])
plt.show()
```



In [32]:
```
df['Species'].value_counts()
```

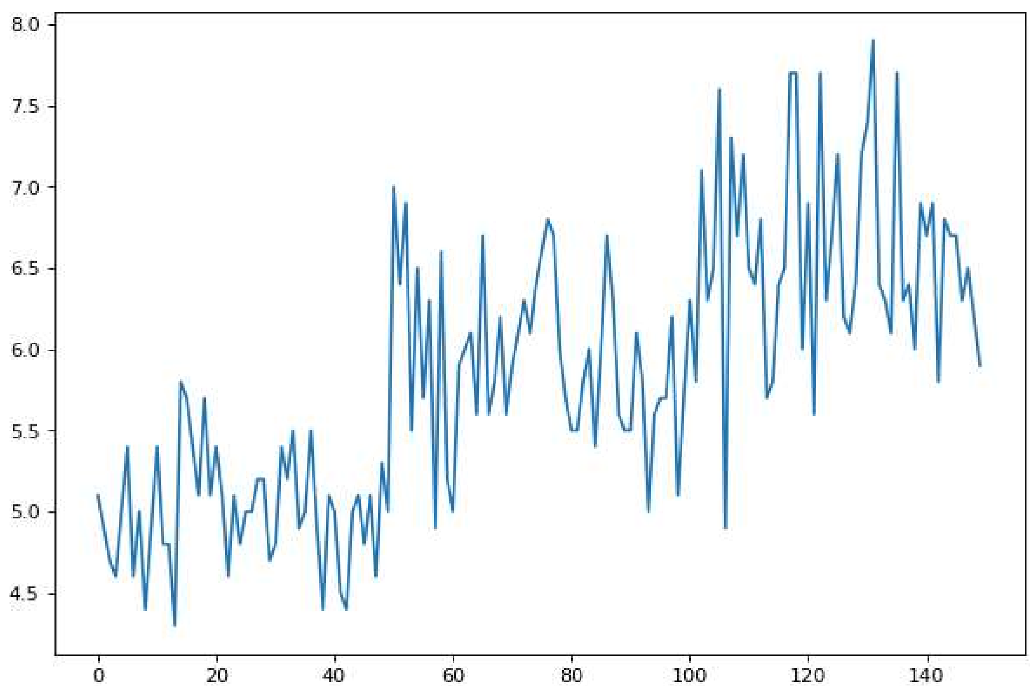
Out[32]:
```
Species
setosa        50
versicolor    50
virginica     50
Name: count, dtype: int64
```

In [34]:
```
plt.pie(df['Species'].value_counts(), labels=['Setosa', 'Versicolor', 'Virginica'], autopct='%.(
plt.show()
```

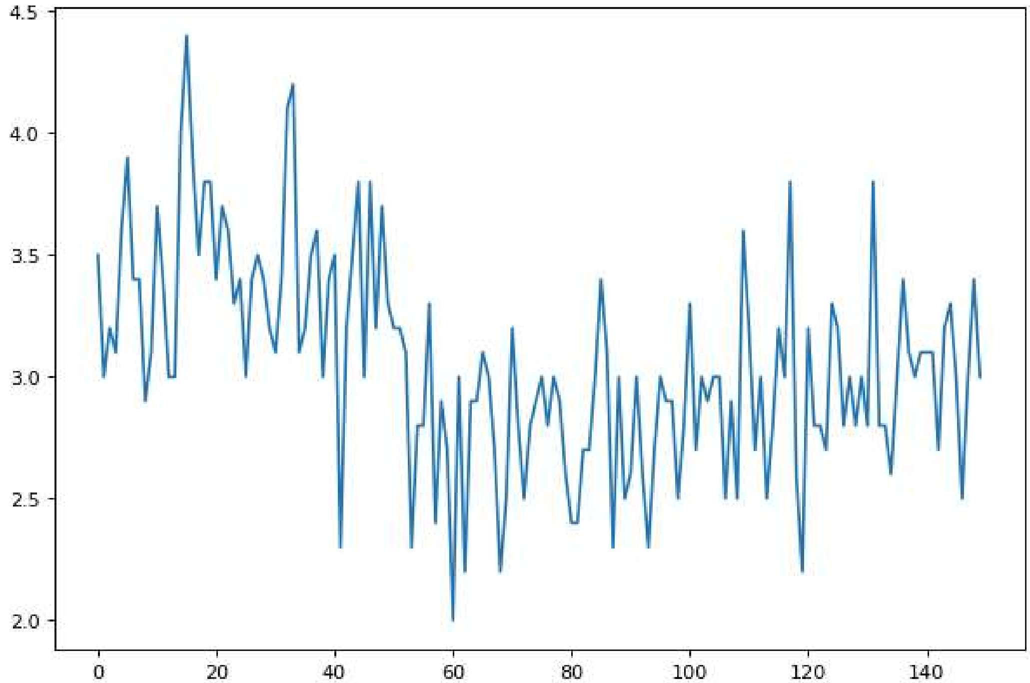


Line plot for each column of given data.

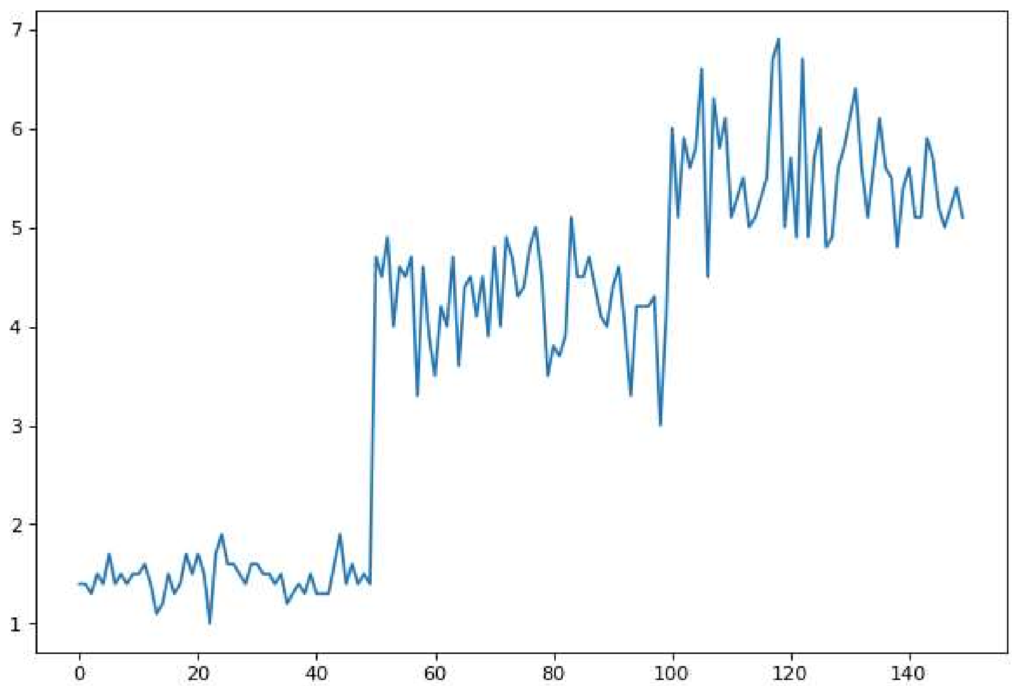
In [35]:
```
plt.plot(df['Sepal.Length'])
plt.show()
```



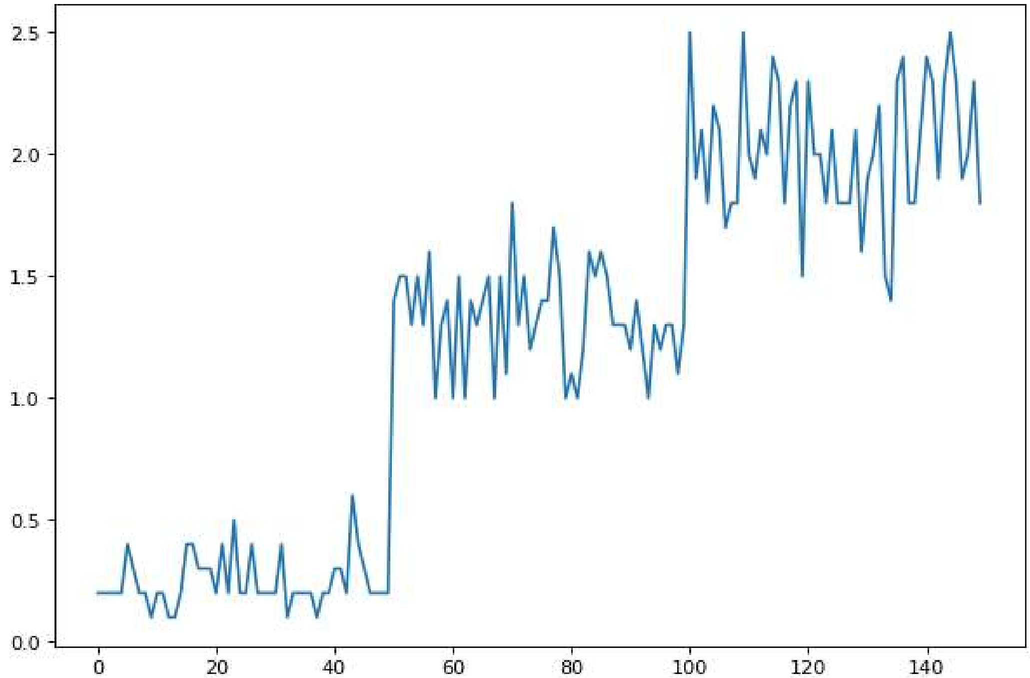
In [36]:
```
plt.plot(df['Sepal.Width'])
plt.show()
```



In [37]:
```
plt.plot(df['Petal.Length'])
plt.show()
```



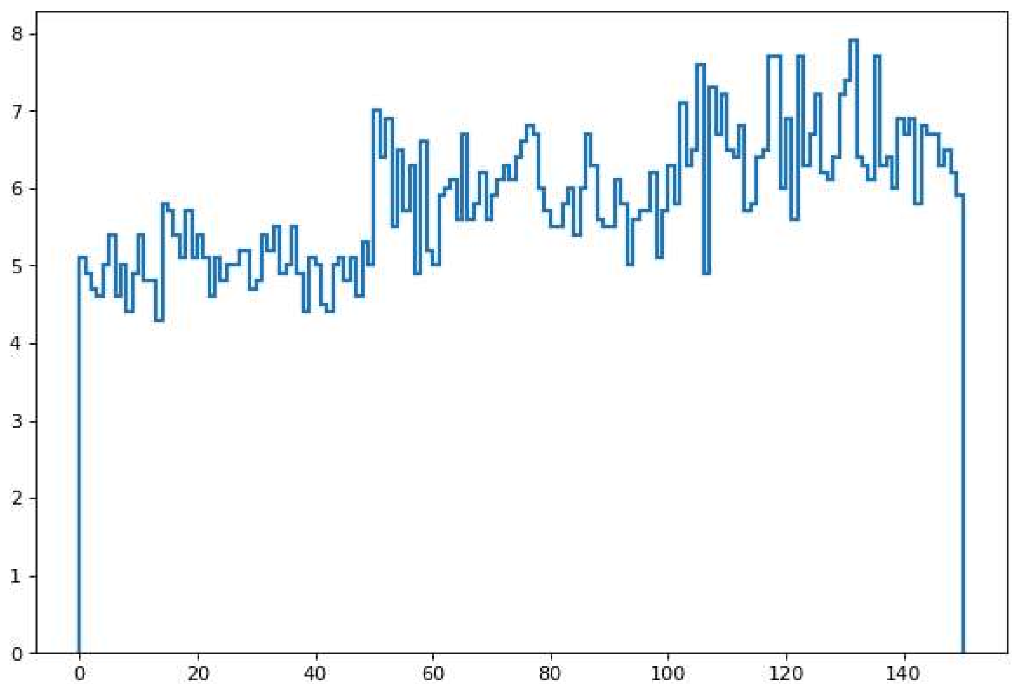
In [38]:
```
plt.plot(df['Petal.Width'])
plt.show()
```



Stair Plot for each column of given data.

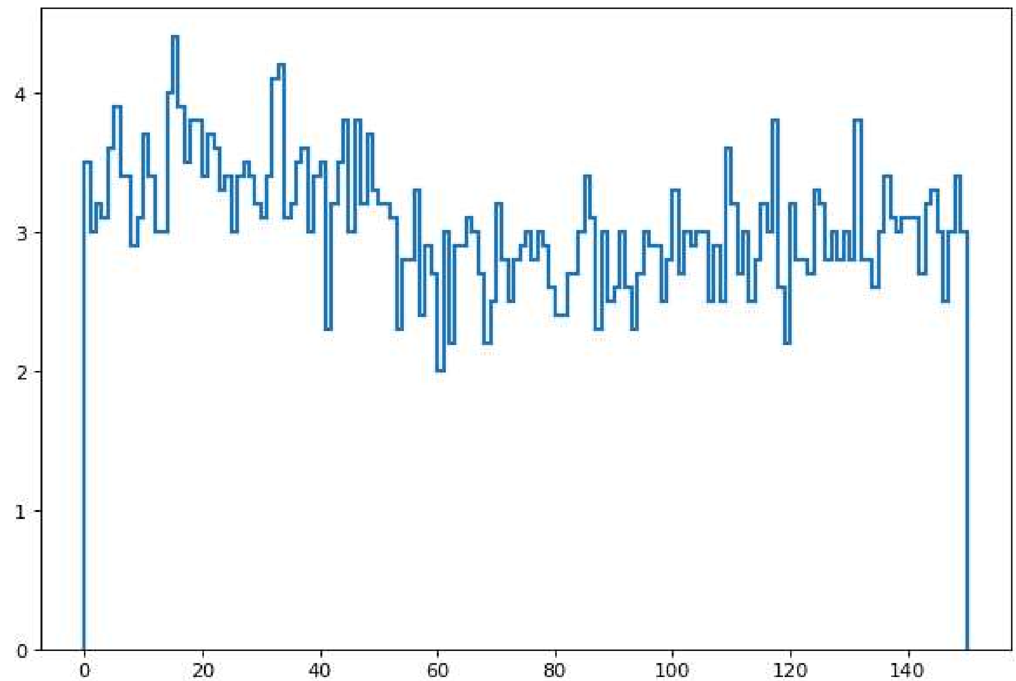
In [39]:
```
plt.stairs(df['Sepal.Length'], linewidth=2)
```

Out[39]:
```
<matplotlib.patches.StepPatch at 0x1f877e7b710>
```



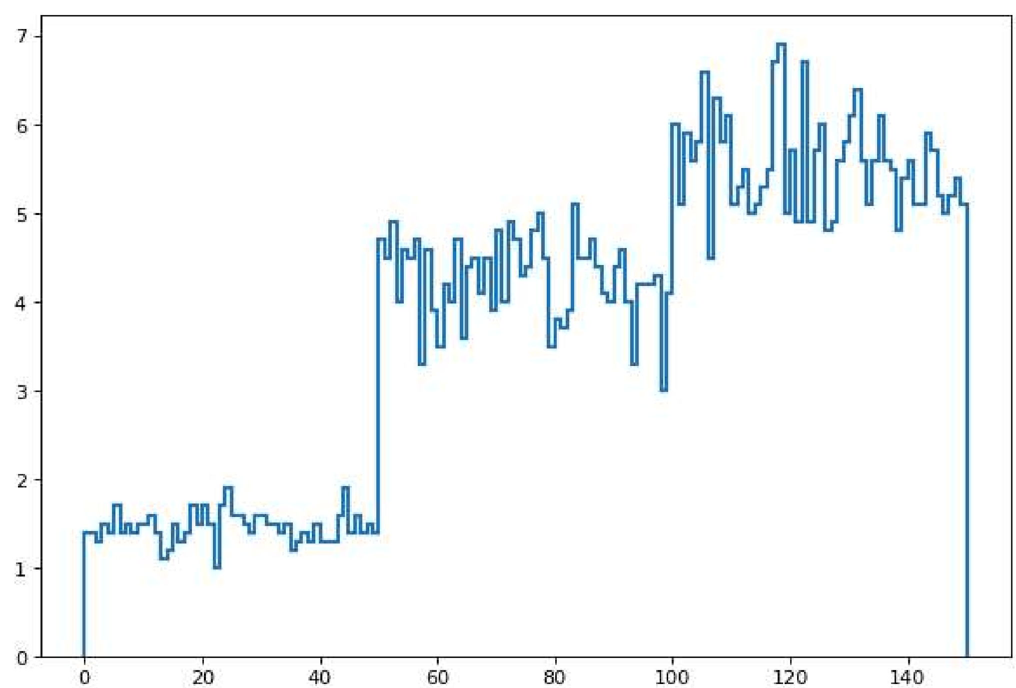
In [40]:
```
plt.stairs(df['Sepal.Width'], linewidth=2)
```

Out[40]: <matplotlib.patches.StepPatch at 0x1f877e132d0>



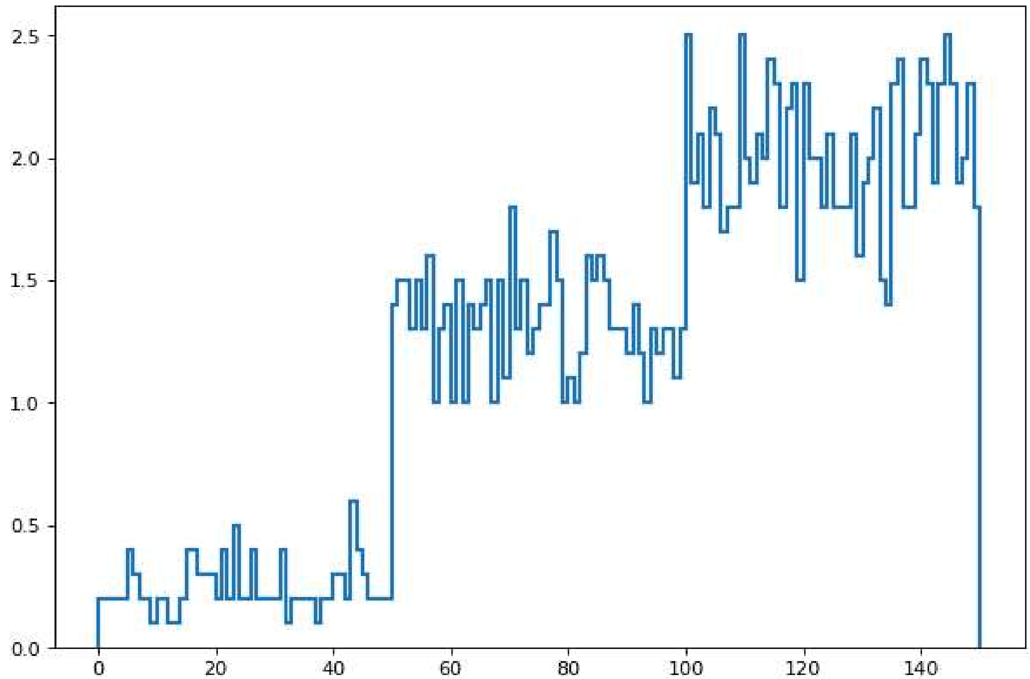
In [41]:
```
plt.stairs(df['Petal.Length'], linewidth=2)
```

Out[41]: <matplotlib.patches.StepPatch at 0x1f878164f10>



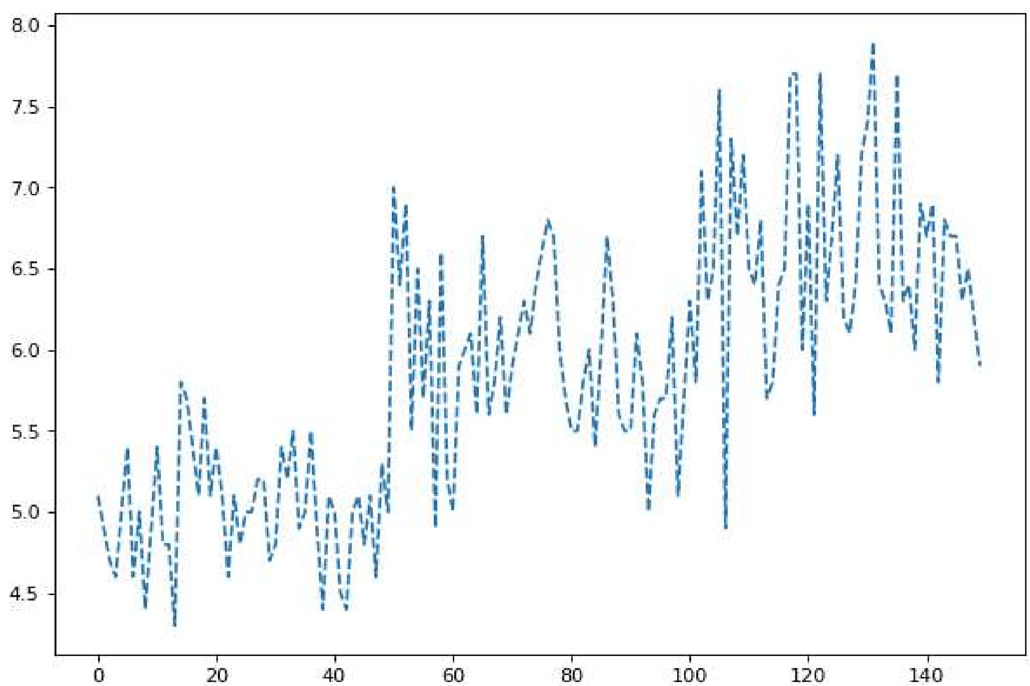
In [42]:
```
plt.stairs(df['Petal.Width'], linewidth=2)
```

Out[42]: <matplotlib.patches.StepPatch at 0x1f878258550>

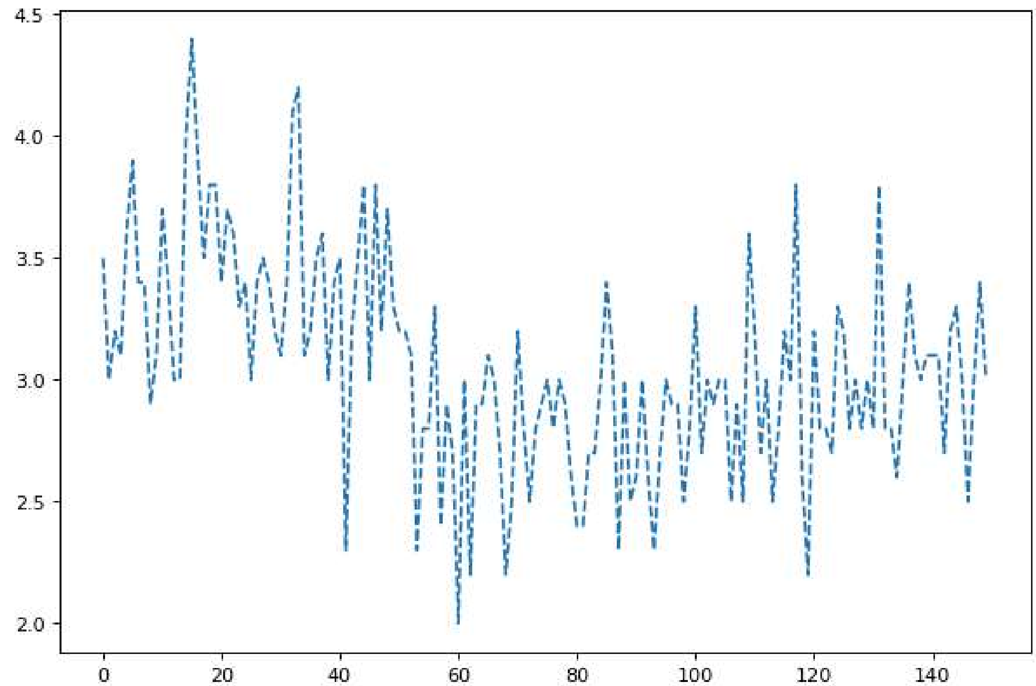


Plotting of each column of given data by "-------" style.

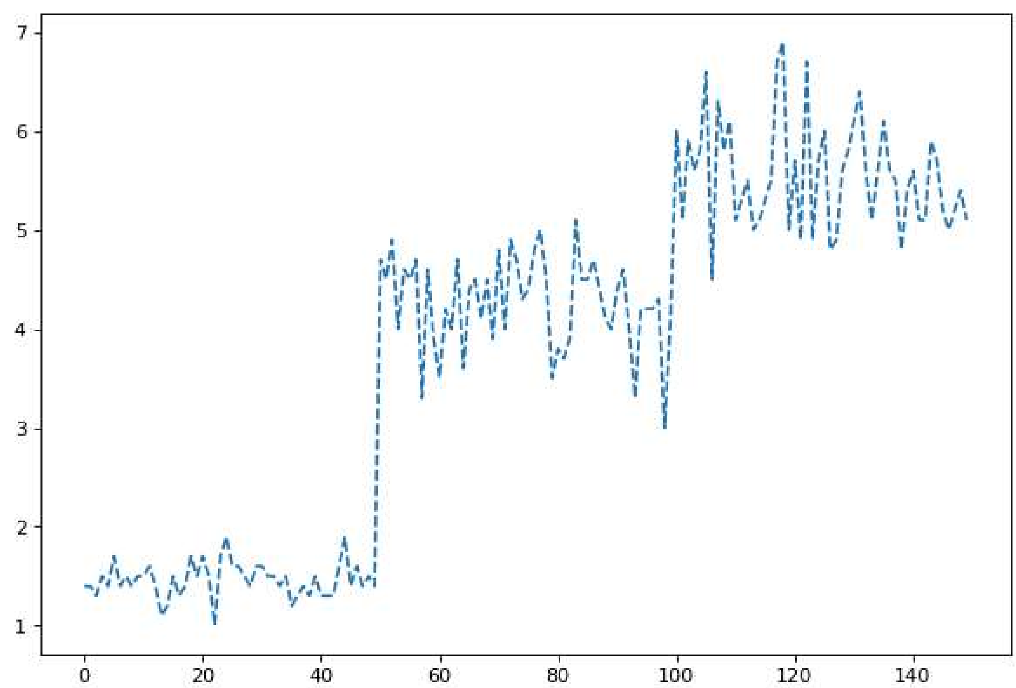
In [44]:
```
plt.plot(df['Sepal.Length'], linestyle='--')
plt.show()
```



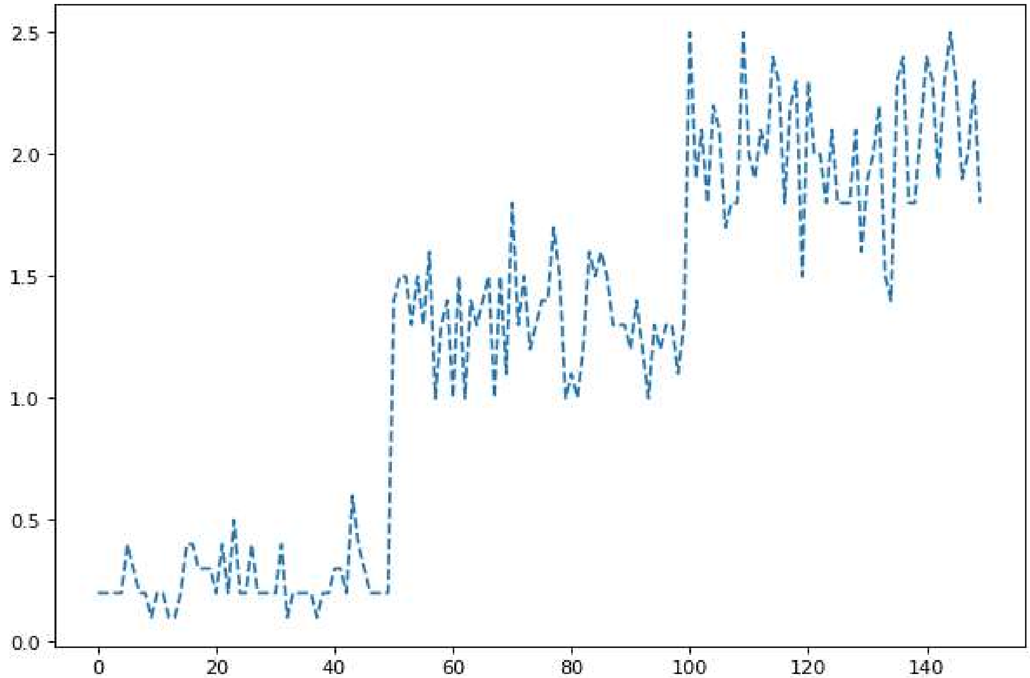
In [45]:
```
plt.plot(df['Sepal.Width'], linestyle='--')
plt.show()
```



In [46]:
```
plt.plot(df['Petal.Length'], linestyle='--')
plt.show()
```



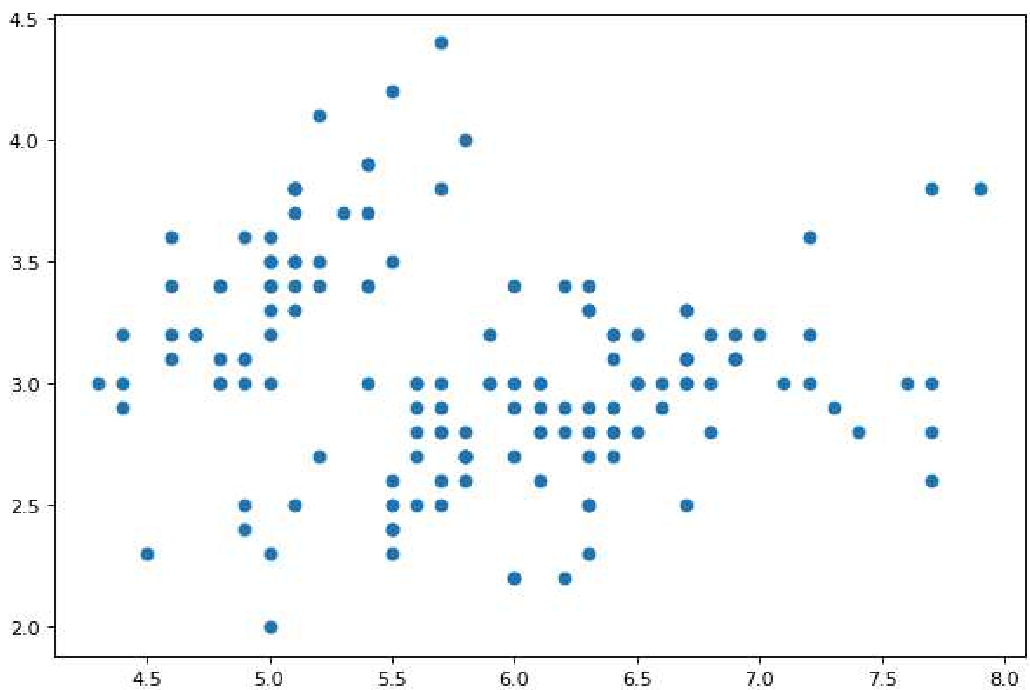
In [47]:
```
plt.plot(df['Petal.Width'], linestyle='--')
plt.show()
```



Scatter Plot

In [49]:
```
plt.scatter(df['Sepal.Length'], df['Sepal.Width'])
```

Out[49]: <matplotlib.collections.PathCollection at 0x1f879e27650>



In [50]:
```
df[['Sepal.Length', 'Sepal.Width']].corr()
```

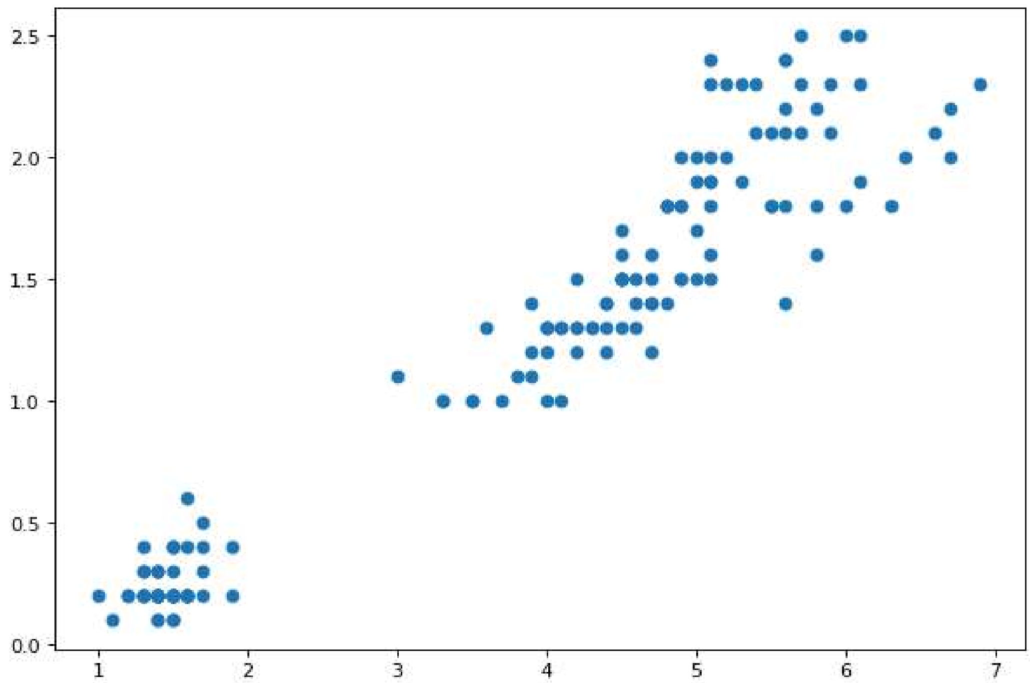
Out[50]:

	Sepal.Length	Sepal.Width
Sepal.Length	1.00000	-0.11757
Sepal.Width	-0.11757	1.00000

Sepal length and Sepal width are rarely correlated

In [51]:
```
plt.scatter(df['Petal.Length'], df['Petal.Width'])
```

Out[51]:
```
<matplotlib.collections.PathCollection at 0x1f87b6a8290>
```



In [52]:
```
df[['Petal.Length', 'Petal.Width']].corr()
```

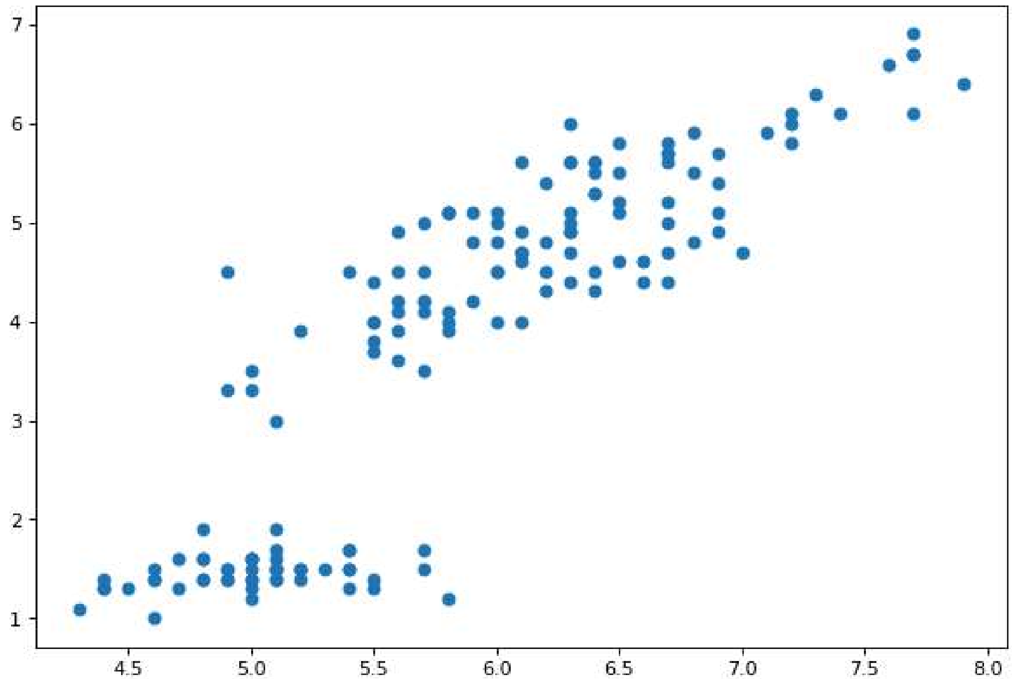
Out[52]:

	Petal.Length	Petal.Width
Petal.Length	1.000000	0.962865
Petal.Width	0.962865	1.000000

Petal length and petal width are highly correlated.

In [53]:
```
plt.scatter(df['Sepal.Length'], df['Petal.Length'])
```

Out[53]: <matplotlib.collections.PathCollection at 0x1f87cb43090>



In [54]:
```
df[['Sepal.Length', 'Petal.Length']].corr()
```

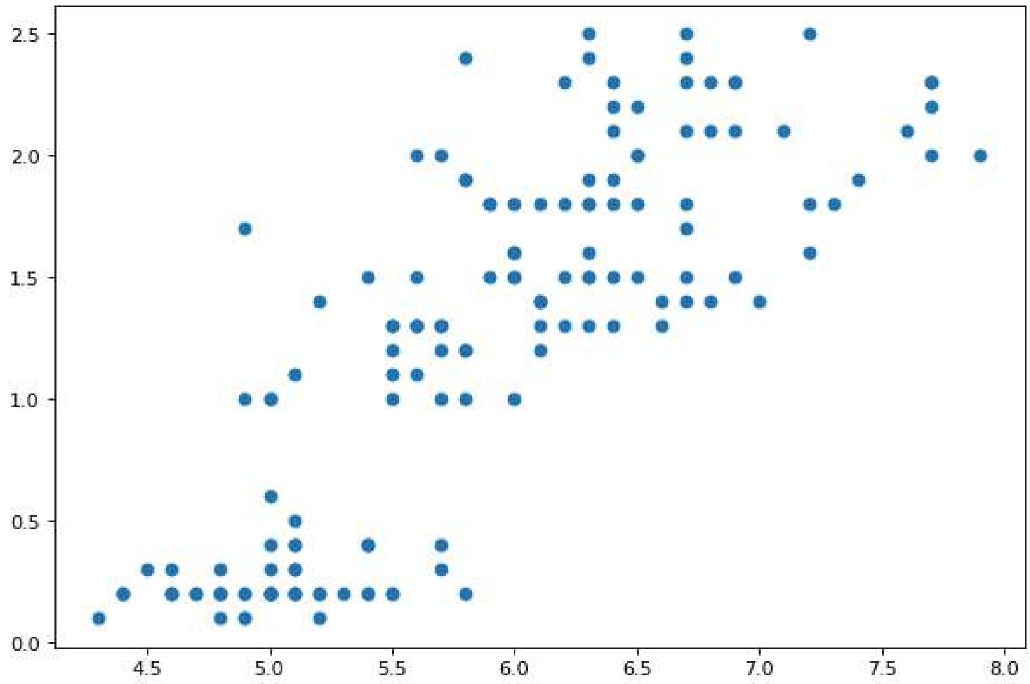
Out[54]:

	Sepal.Length	Petal.Length
Sepal.Length	1.000000	0.871754
Petal.Length	0.871754	1.000000

Sepal length and petal length are positively correlated.

In [55]:
```
plt.scatter(df['Sepal.Length'], df['Petal.Width'])
```

Out[55]: <matplotlib.collections.PathCollection at 0x1f87cbc4290>



In [56]:
```
df[['Sepal.Length', 'Petal.Width']].corr()
```

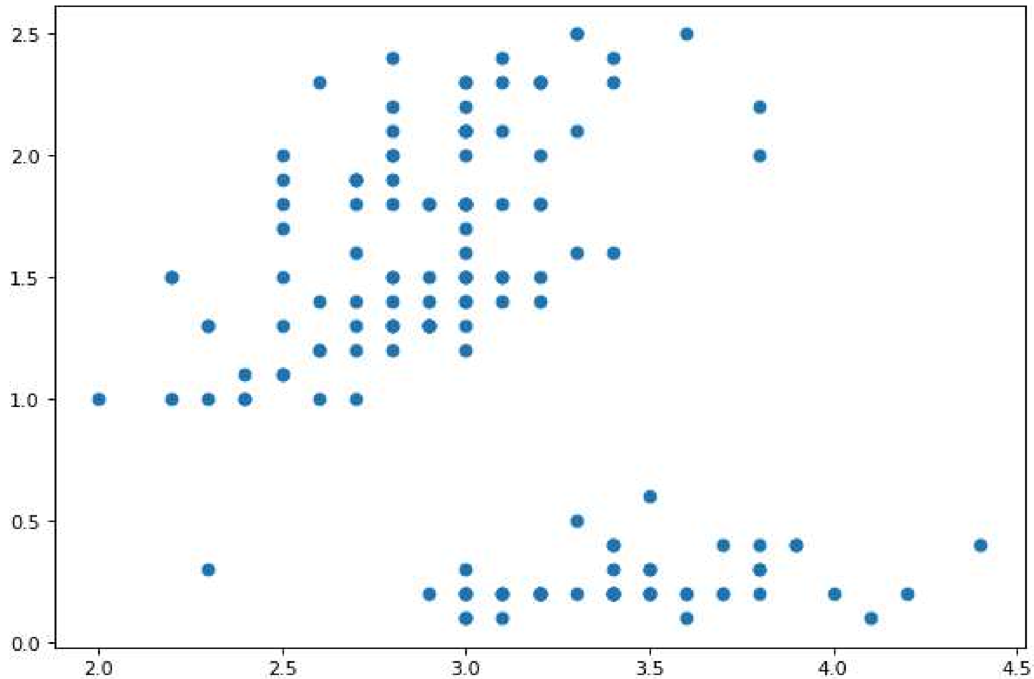
Out[56]:

	Sepal.Length	Petal.Width
Sepal.Length	1.000000	0.817941
Petal.Width	0.817941	1.000000

Sepal length and petal width are positively correlated.

In [57]:
```
plt.scatter(df['Sepal.Width'], df['Petal.Width'])
```

Out[57]:
```
<matplotlib.collections.PathCollection at 0x1f87cc219d0>
```



In [58]:
```
df[['Sepal.Width', 'Petal.Width']].corr()
```

Out[58]:

	Sepal.Width	Petal.Width
Sepal.Width	1.000000	-0.366126
Petal.Width	-0.366126	1.000000

Sepal width and petal width are negativly correlated.

STATISTICAL MEASURES

In [59]:
```
df.dtypes
```

Out[59]:
```
Unnamed: 0        int64
Sepal.Length    float64
Sepal.Width     float64
Petal.Length    float64
Petal.Width     float64
Species          object
dtype: object
```

In [60]:
```
df.drop_duplicates()
```

Out[60]:

	Unnamed: 0	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
0	1	5.1	3.5	1.4	0.2	setosa
1	2	4.9	3.0	1.4	0.2	setosa
2	3	4.7	3.2	1.3	0.2	setosa
3	4	4.6	3.1	1.5	0.2	setosa
4	5	5.0	3.6	1.4	0.2	setosa
...	...	...	...	...	...	...
145	146	6.7	3.0	5.2	2.3	virginica
146	147	6.3	2.5	5.0	1.9	virginica
147	148	6.5	3.0	5.2	2.0	virginica
148	149	6.2	3.4	5.4	2.3	virginica
149	150	5.9	3.0	5.1	1.8	virginica

150 rows × 6 columns

In [62]:
```
df1=df['Sepal.Length']
df1
```

Out[62]:
```
0      5.1
1      4.9
2      4.7
3      4.6
4      5.0
      ...
145    6.7
146    6.3
147    6.5
148    6.2
149    5.9
Name: Sepal.Length, Length: 150, dtype: float64
```

In [64]:
```
df1.sum()
```

Out[64]:
```
876.5
```

In [65]:
```
df1.count()
```

Out[65]:
```
150
```

In [66]:
```
df1.max()
```

Out[66]:
```
7.9
```

In [67]:
```
df1.min()
```

Out[67]:
```
4.3
```

In [69]:
```
df1.mean()
```

Out[69]:
```
5.843333333333334
```

In [70]:
```
df1.median()
```

Out[70]:
```
5.8
```

In [71]:
```
df1.mode()
```

Out[71]:
```
0    5.0
Name: Sepal.Length, dtype: float64
```

In [72]:
```
df1.std()
```

Out[72]:
```
0.8280661279778629
```

In [73]:
```
df1.skew()
```

Out[73]:
```
0.3149109566369728
```

In [74]:
```
df1.kurt()
```

Out[74]:
```
-0.5520640413156395
```

In [75]:
```
df1.describe()
```

Out[75]:
```
count    150.000000
mean       5.843333
std        0.828066
min        4.300000
25%        5.100000
50%        5.800000
75%        6.400000
max        7.900000
Name: Sepal.Length, dtype: float64
```

In [76]:
```
df2=df['Sepal.Width']
df2
```

Out[76]:
```
0      3.5
1      3.0
2      3.2
3      3.1
4      3.6
      ...
145    3.0
146    2.5
147    3.0
148    3.4
149    3.0
Name: Sepal.Width, Length: 150, dtype: float64
```

In [80]:
```
df2.describe()
```

Out[80]:
```
count    150.000000
mean       3.057333
std        0.435866
min        2.000000
25%        2.800000
50%        3.000000
75%        3.300000
max        4.400000
Name: Sepal.Width, dtype: float64
```

In [81]:
```
df2.median()
```

Out[81]:
```
3.0
```

In [82]:
```
df2.mode()
```

Out[82]:
```
0    3.0
Name: Sepal.Width, dtype: float64
```

In [83]:
```
df2.skew()
```

Out[83]:
```
0.31896566471359966
```

In [84]:
```
df2.kurt()
```

Out[84]:
```
0.2282490424681929
```

In [85]:
```
df3=df['Petal.Length']
df3
```

Out[85]:
```
0      1.4
1      1.4
2      1.3
3      1.5
4      1.4
      ...
145    5.2
146    5.0
147    5.2
148    5.4
149    5.1
Name: Petal.Length, Length: 150, dtype: float64
```

In [86]:
```
df3.describe()
```

Out[86]:
```
count    150.000000
mean       3.758000
std        1.765298
min        1.000000
25%        1.600000
50%        4.350000
75%        5.100000
max        6.900000
Name: Petal.Length, dtype: float64
```

In [87]:
```
df3.median()
```

Out[87]:
```
4.35
```

In [88]:
```
df3.mode()
```

Out[88]:
```
0    1.4
1    1.5
Name: Petal.Length, dtype: float64
```

In [89]:
```
df3.skew()
```

Out[89]:
```
-0.27488417975101276
```

In [90]:
```
df3.kurt()
```
Out[90]:
```
-1.4021034155217518
```

In [91]:
```
df4=df['Petal.Width']
df4
```
Out[91]:
```
0      0.2
1      0.2
2      0.2
3      0.2
4      0.2
      ...
145    2.3
146    1.9
147    2.0
148    2.3
149    1.8
Name: Petal.Width, Length: 150, dtype: float64
```

In [92]:
```
df4.median()
```
Out[92]:
```
1.3
```

In [93]:
```
df4.mode()
```
Out[93]:
```
0    0.2
Name: Petal.Width, dtype: float64
```

In [94]:
```
df4.skew()
```
Out[94]:
```
-0.10296674764898116
```

In [96]:
```
df4.kurt()
```
Out[96]:
```
-1.340603996612646
```

In [97]:
```
df.describe()
```
Out[97]:

	Unnamed: 0	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	75.500000	5.843333	3.057333	3.758000	1.199333
std	43.445368	0.828066	0.435866	1.765298	0.762238
min	1.000000	4.300000	2.000000	1.000000	0.100000
25%	38.250000	5.100000	2.800000	1.600000	0.300000
50%	75.500000	5.800000	3.000000	4.350000	1.300000
75%	112.750000	6.400000	3.300000	5.100000	1.800000
max	150.000000	7.900000	4.400000	6.900000	2.500000

In [98]:
```
df.describe(include=object)
```

Out[98]:

	Species
count	150
unique	3
top	setosa
freq	50

ROW ECHELON FORM

In [100]:
```
import numpy as np
```

In [101]:
```
import sympy as sp
```

In [102]:
```
import scipy
```

In [128]:
```
np.random.seed(56)
A=np.random.randint(0,10,(5,5))
A
```

Out[128]:
```
array([[5, 4, 0, 2, 9],
       [7, 6, 4, 9, 7],
       [1, 8, 2, 0, 5],
       [6, 1, 9, 5, 5],
       [2, 9, 3, 5, 9]])
```

In [129]:
```
A[0]=(1/5)*A[0]
A
```

Out[129]:
```
array([[1, 0, 0, 0, 1],
       [7, 6, 4, 9, 7],
       [1, 8, 2, 0, 5],
       [6, 1, 9, 5, 5],
       [2, 9, 3, 5, 9]])
```

In [130]:
```
A[1]=A[1]-7*A[0]
A[2]=A[2]-A[0]
A[3]=A[3]-6*A[0]
A[4]=A[4]-2*A[0]

A
```

Out[130]:
```
array([[ 1,  0,  0,  0,  1],
       [ 0,  6,  4,  9,  0],
       [ 0,  8,  2,  0,  4],
       [ 0,  1,  9,  5, -1],
       [ 0,  9,  3,  5,  7]])
```

In [131]:
```
A[1]=(1/6)*A[1]
A
```

Out[131]:
```
array([[ 1,  0,  0,  0,  1],
       [ 0,  1,  0,  1,  0],
       [ 0,  8,  2,  0,  4],
       [ 0,  1,  9,  5, -1],
       [ 0,  9,  3,  5,  7]])
```

In [132]:
```
A[2]=A[2]-8*A[1]
A[3]=A[3]-A[1]
A[4]=A[4]-9*A[1]

A
```

Out[132]:
```
array([[ 1,  0,  0,  0,  1],
       [ 0,  1,  0,  1,  0],
       [ 0,  0,  2, -8,  4],
       [ 0,  0,  9,  4, -1],
       [ 0,  0,  3, -4,  7]])
```

In [133]:
```
A[2]=(1/2)*A[2]
A
```

Out[133]:
```
array([[ 1,  0,  0,  0,  1],
       [ 0,  1,  0,  1,  0],
       [ 0,  0,  1, -4,  2],
       [ 0,  0,  9,  4, -1],
       [ 0,  0,  3, -4,  7]])
```

In [134]:
```
A[3]=A[3]-9*A[2]
A[4]=A[4]-3*A[2]

A
```

Out[134]:
```
array([[  1,   0,   0,   0,   1],
       [  0,   1,   0,   1,   0],
       [  0,   0,   1,  -4,   2],
       [  0,   0,   0,  40, -19],
       [  0,   0,   0,   8,   1]])
```

In [135]:
```
A[3]=(1/40)*A[3]
A
```

Out[135]:
```
array([[ 1,  0,  0,  0,  1],
       [ 0,  1,  0,  1,  0],
       [ 0,  0,  1, -4,  2],
       [ 0,  0,  0,  1,  0],
       [ 0,  0,  0,  8,  1]])
```

In [136]:
```
A[4]=A[4]-8*A[3]
A
```

Out[136]:
```
array([[ 1,  0,  0,  0,  1],
       [ 0,  1,  0,  1,  0],
       [ 0,  0,  1, -4,  2],
       [ 0,  0,  0,  1,  0],
       [ 0,  0,  0,  0,  1]])
```

In [137]:
```
sp.Matrix(A)
```

Out[137]:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & -4 & 2 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

is required row echelon form of matrix A

SINGULAR VALUE DECOMPOSITION (SVD)

In [138]:
```
print("Matrix A =")
sp.Matrix(A)
```

```
Matrix A =
```

Out[138]:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & -4 & 2 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

In [139]:
```
U,s,Vt=np.linalg.svd(A)
```

In [140]:
```
sp.Matrix(np.round(U))
```

Out[140]:

$$\begin{bmatrix} 0 & -1.0 & 0 & 0 & 0 \\ 0 & 0 & -1.0 & 0 & 0 \\ 1.0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1.0 \\ 0 & 0 & 0 & -1.0 & 0 \end{bmatrix}$$

In [141]:
```
sp.Matrix(np.round(s))
```

Out[141]:

$$\begin{bmatrix} 5.0 \\ 2.0 \\ 1.0 \\ 1.0 \\ 0 \end{bmatrix}$$

In [142]:
```
sp.Matrix(np.round(Vt))
```

Out[142]:

$$\begin{bmatrix} 0 & 0 & 0 & -1.0 & 0 \\ 0 & 0 & 0 & 0 & -1.0 \\ 0 & -1.0 & 0 & 0 & 0 \\ 1.0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1.0 & 0 & 0 \end{bmatrix}$$

In [147]:
```
Sigma=np.zeros((A.shape[0], A.shape[1]))
sp.Matrix(Sigma)
```

Out[147]:
$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

In [148]:
```
Sigma[:A.shape[1], :A.shape[0]]=np.diag(s)
sp.Matrix(Sigma)
```

Out[148]:
$$\begin{bmatrix} 4.79128784747792 & 0 & 0 & 0 & 0 \\ 0 & 1.61803398874989 & 0 & 0 & 0 \\ 0 & 0 & 1.0 & 0 & 0 \\ 0 & 0 & 0 & 0.618033988749895 & 0 \\ 0 & 0 & 0 & 0 & 0.20871215252208 \end{bmatrix}$$

In [154]:
```
C=np.round(U@Sigma@Vt)
C
```

Out[154]:
```
array([[ 1.,  0.,  0.,  0.,  1.],
       [-0.,  1.,  0.,  1.,  0.],
       [ 0.,  0.,  1., -4.,  2.],
       [-0.,  0.,  0.,  1.,  0.],
       [-0.,  0.,  0.,  0.,  1.]])
```

In [153]:
```
A
```

Out[153]:
```
array([[ 1,  0,  0,  0,  1],
       [ 0,  1,  0,  1,  0],
       [ 0,  0,  1, -4,  2],
       [ 0,  0,  0,  1,  0],
       [ 0,  0,  0,  0,  1]])
```

In [155]:
```
C==A
```

Out[155]:
```
array([[ True,  True,  True,  True,  True],
       [ True,  True,  True,  True,  True],
       [ True,  True,  True,  True,  True],
       [ True,  True,  True,  True,  True],
       [ True,  True,  True,  True,  True]])
```

Hence, verified!!

In [156]:
```
#RANK 2 APPROXIMATION
```

In [157]:
```
U2=U[:,:2]
sp.Matrix(np.round(U2,3))
```

Out[157]:
$$\begin{bmatrix} 0.096 & -0.761 \\ -0.191 & -0.38 \\ 0.955 & 0 \\ -0.183 & -0.235 \\ 0.091 & -0.47 \end{bmatrix}$$

In [158]:
```
Sigma2=Sigma[:2,:2]
sp.Matrix(np.round(Sigma2,3))
```

Out[158]:
$$\begin{bmatrix} 4.791 & 0 \\ 0 & 1.618 \end{bmatrix}$$

In [159]:
```
Vt2=Vt[:2,:]
sp.Matrix(np.round(Vt2,3))
```

Out[159]:
$$\begin{bmatrix} 0.02 & -0.04 & 0.199 & -0.876 & 0.438 \\ -0.47 & -0.235 & 0 & -0.38 & -0.761 \end{bmatrix}$$

In [160]:
```
A2=U2@Sigma2@Vt2
sp.Matrix(np.round(A2,3))
```

Out[160]:
$$\begin{bmatrix} 0.588 & 0.271 & 0.091 & 0.068 & 1.137 \\ 0.271 & 0.181 & -0.183 & 1.036 & 0.068 \\ 0.091 & -0.183 & 0.913 & -4.008 & 2.004 \\ 0.161 & 0.124 & -0.175 & 0.911 & -0.094 \\ 0.366 & 0.161 & 0.087 & -0.094 & 0.771 \end{bmatrix}$$

In [161]:
```
#RANK 3 APPROXIMATION
```

In [162]:
```
U3=U[:,:3]
sp.Matrix(np.round(U3,3))
```

Out[162]:
$$\begin{bmatrix} 0.096 & -0.761 & 0.436 \\ -0.191 & -0.38 & -0.873 \\ 0.955 & 0 & -0.218 \\ -0.183 & -0.235 & 0 \\ 0.091 & -0.47 & 0 \end{bmatrix}$$

In [163]:
```
Sigma3=Sigma[:3,:3]
sp.Matrix(np.round(Sigma3,3))
```

Out[163]:
$$\begin{bmatrix} 4.791 & 0 & 0 \\ 0 & 1.618 & 0 \\ 0 & 0 & 1.0 \end{bmatrix}$$

In [164]:
```
Vt3=Vt[:3,:]
sp.Matrix(np.round(Vt3,3))
```

Out[164]:
$$\begin{bmatrix} 0.02 & -0.04 & 0.199 & -0.876 & 0.438 \\ -0.47 & -0.235 & 0 & -0.38 & -0.761 \\ 0.436 & -0.873 & -0.218 & 0 & 0 \end{bmatrix}$$

In [165]:
```
A3=U3@Sigma3@Vt3
sp.Matrix(np.round(A3,3))
```

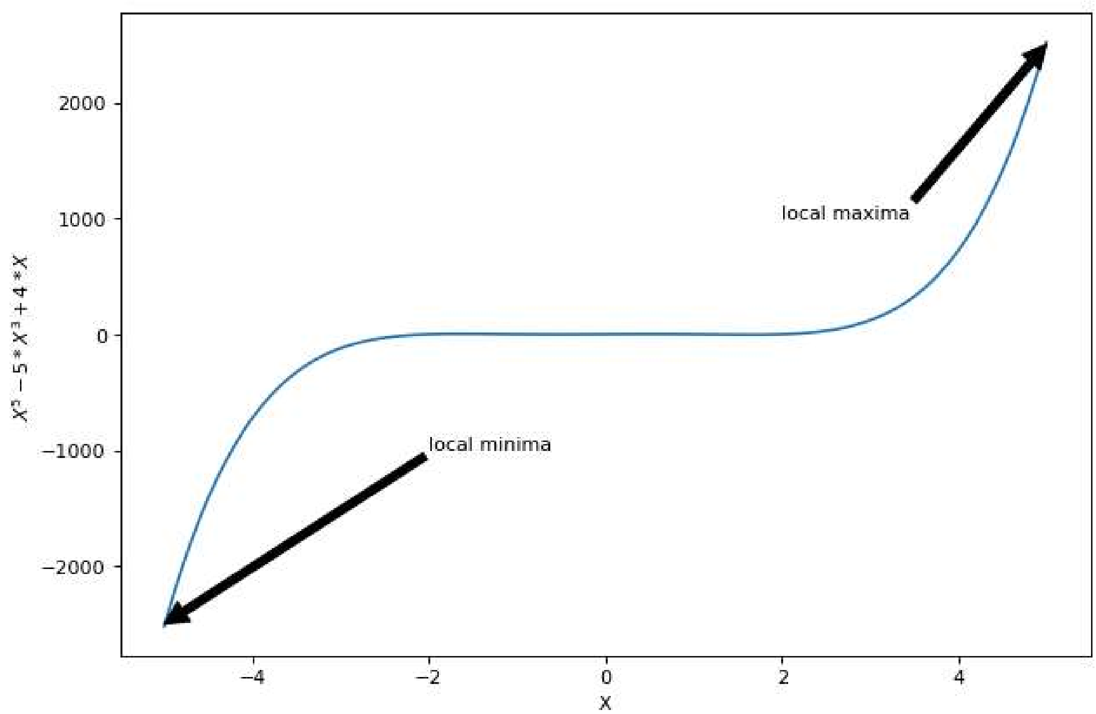
Out[165]:

$$\begin{bmatrix} 0.778 & -0.11 & -0.004 & 0.068 & 1.137 \\ -0.11 & 0.943 & 0.008 & 1.036 & 0.068 \\ -0.004 & 0.008 & 0.96 & -4.008 & 2.004 \\ 0.161 & 0.124 & -0.175 & 0.911 & -0.094 \\ 0.366 & 0.161 & 0.087 & -0.094 & 0.771 \end{bmatrix}$$

POLYNOMIAL PLOTTING WITH ANNOTATIONS

In [183]:
```
X=np.arange(-5,5+0.1,0.1)
plt.plot(X,X**5 -5*X**3 + 4*X)
plt.xlabel('X')
plt.ylabel(r'$X^5 -5*X^3 + 4*X$')
plt.annotate('local maxima', xy=(5,2500), xytext=(2,1000), arrowprops=dict(facecolor='black', sl
plt.annotate('local minima', xy=(-5,-2500), xytext=(-2,-1000), arrowprops=dict(facecolor='black
```

Out[183]:
```
Text(-2, -1000, 'local minima')
```



For the given graph, in particular range we can visualize the maxima and minima. But in Real line, Graph is not bounded. Hence no global maxima and global minima exists.

In [ ]: