In [1]:

```
from sklearn.datasets import load_iris
```

In [14]:

```
import pandas as pd
import numpy as np
```

In [26]:

```
import matplotlib.pyplot as pl
```

In [2]:

```
iris = load_iris()
```

In [3]:

```
from sklearn.cluster import DBSCAN
```

In [12]:

```
iris.DESCR
```

Out[12]:

```
'Iris Plants Database\n====================\n\nNotes\n-----\nData Set Char
acteristics:\n    :Number of Instances: 150 (50 in each of three classes)
\n    :Number of Attributes: 4 numeric, predictive attributes and the clas
s\n    :Attribute Information:\n        - sepal length in cm\n        - se
pal width in cm\n        - petal length in cm\n        - petal width in cm
\n        - class:\n                - Iris-Setosa\n                - Iris-
Versicolour\n                - Iris-Virginica\n    :Summary Statistics:\n
\n    ============== ==== ==== ======= ===== ====================\n
            Min  Max   Mean    SD   Class Correlation\n    ==============
==== ==== ======= ===== ====================\n    sepal length:   4.3  7.9
   5.84   0.83    0.7826\n    sepal width:    2.0  4.4   3.05   0.43   -0.
4194\n    petal length:   1.0  6.9   3.76   1.76    0.9490  (high!)\n    p
etal width:    0.1  2.5   1.20  0.76     0.9565  (high!)\n    ============
== ==== ==== ======= ===== ====================\n\n    :Missing Attribute
Values: None\n    :Class Distribution: 33.3% for each of 3 classes.\n    :
Creator: R.A. Fisher\n    :Donor: Michael Marshall (MARSHALL%PLU@io.arc.na
sa.gov)\n    :Date: July, 1988\n\nThis is a copy of UCI ML iris dataset
s.\nhttp://archive.ics.uci.edu/ml/datasets/Iris\n\nThe famous Iris databas
e, first used by Sir R.A Fisher\n\nThis is perhaps the best known database
to be found in the\npattern recognition literature.  Fisher\'s paper is a
classic in the field and\nis referenced frequently to this day.  (See Duda
& Hart, for example.)  The\ndata set contains 3 classes of 50 instances ea
ch, where each class refers to a\ntype of iris plant.  One class is linear
ly separable from the other 2; the\nlatter are NOT linearly separable from
each other.\n\nReferences\n----------\n   - Fisher,R.A. "The use of multip
le measurements in taxonomic problems"\n     Annual Eugenics, 7, Part II,
179-188 (1936); also in "Contributions to\n     Mathematical Statistics"
(John Wiley, NY, 1950).\n   - Duda,R.O., & Hart,P.E. (1973) Pattern Classi
fication and Scene Analysis.\n     (Q327.D83) John Wiley & Sons.  ISBN 0-4
71-22361-1.  See page 218.\n   - Dasarathy, B.V. (1980) "Nosing Around the
Neighborhood: A New System\n     Structure and Classification Rule for Rec
ognition in Partially Exposed\n     Environments".  IEEE Transactions on P
attern Analysis and Machine\n     Intelligence, Vol. PAMI-2, No. 1, 67-7
1.\n   - Gates, G.W. (1972) "The Reduced Nearest Neighbor Rule".  IEEE Tra
nsactions\n     on Information Theory, May 1972, 431-433.\n   - See also:
1988 MLC Proceedings, 54-64.  Cheeseman et al"s AUTOCLASS II\n     concept
ual clustering system finds 3 classes in the data.\n   - Many, many more
...\n'
```

In [15]:

```
b=pd.DataFrame(iris.data)
```

In [16]:

```
b.describe()
```

Out[16]:

	0	1	2	3
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.054000	3.758667	1.198667
std	0.828066	0.433594	1.764420	0.763161
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

In [17]:

```
b.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 4 columns):
0    150 non-null float64
1    150 non-null float64
2    150 non-null float64
3    150 non-null float64
dtypes: float64(4)
memory usage: 4.8 KB
```

In [32]:

```
b.columns=iris.feature_names
```

In [34]:

```
b.head()
```

Out[34]:

	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

In [41]:

```
b.shape
```

Out[41]:

```
(150, 4)
```

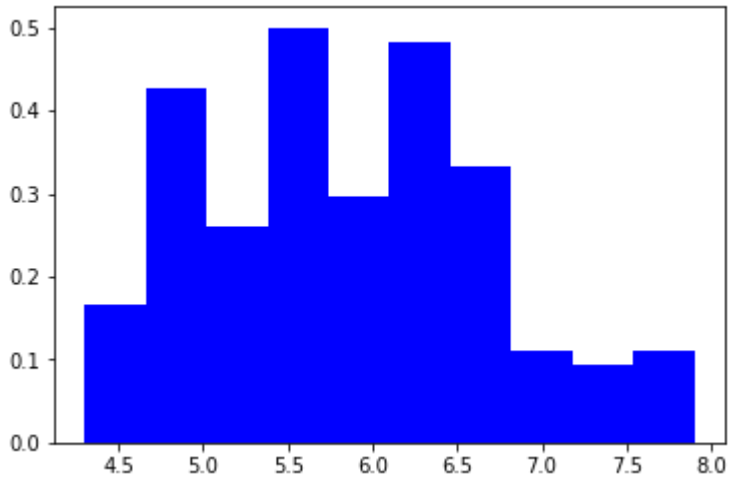
In [37]:

```
num_bins =10
pl.hist(b['sepal length (cm)'],num_bins,normed=1,facecolor='blue')
```

```
C:\Users\Pavi\Anaconda3\lib\site-packages\matplotlib\axes\_axes.py:6462: U
serWarning: The 'normed' kwarg is deprecated, and has been replaced by the
'density' kwarg.
  warnings.warn("The 'normed' kwarg is deprecated, and has been "
```

Out[37]:

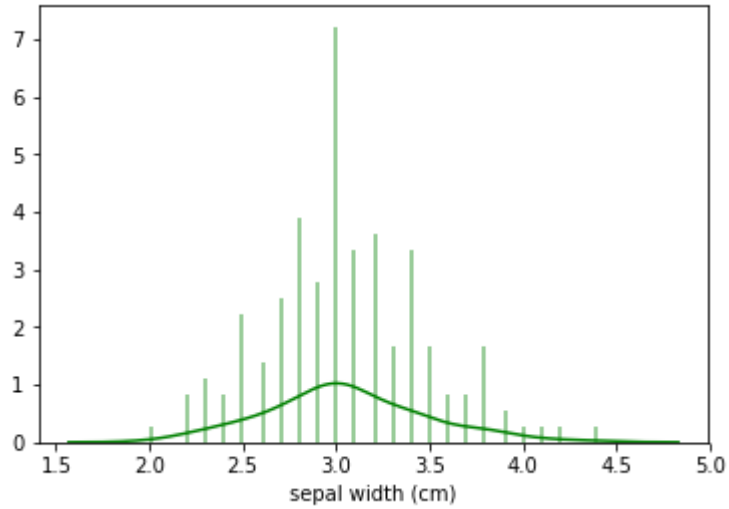
```
(array([0.16666667, 0.42592593, 0.25925926, 0.5       , 0.2962963 ,
        0.48148148, 0.33333333, 0.11111111, 0.09259259, 0.11111111]),
 array([4.3 , 4.66, 5.02, 5.38, 5.74, 6.1 , 6.46, 6.82, 7.18, 7.54, 7.9
]),
 <a list of 10 Patch objects>)
```



In [39]:

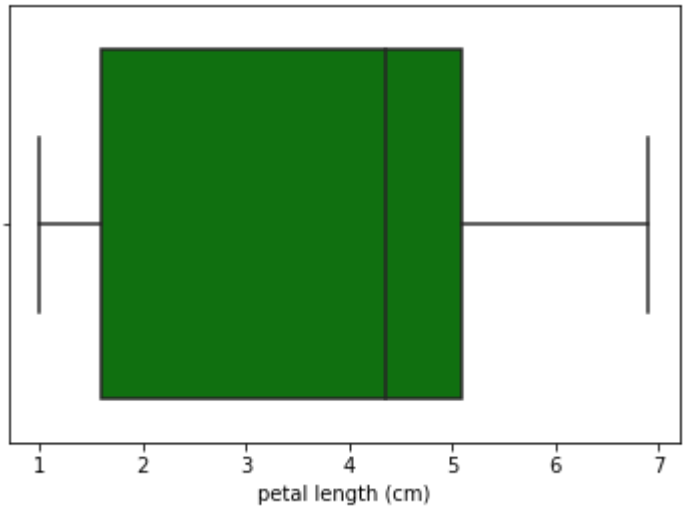
```
import seaborn as snb
snb.distplot(b['sepal width (cm)'],color='green',bins=100,hist_kws={'alpha':0.4});
```

```
C:\Users\Pavi\Anaconda3\lib\site-packages\matplotlib\axes\_axes.py:6462: U
serWarning: The 'normed' kwarg is deprecated, and has been replaced by the
'density' kwarg.
  warnings.warn("The 'normed' kwarg is deprecated, and has been "
```



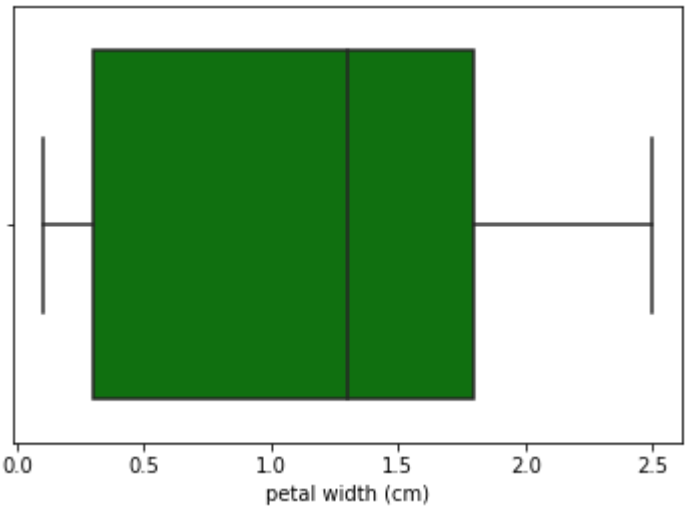
In [44]:

```
snb.boxplot(b['petal length (cm)'],color='green');
```



In [45]:

```
snb.boxplot(b['petal width (cm)'],color='green');
```



In [18]:

```
dbscan = DBSCAN(eps=0.5, metric='euclidean', min_samples=5).fit(b)
```

In [19]:

```
dbscan.labels_
```

Out[19]:

```
array([ 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,  0,  0,  0,  0,  0,  0,  0,  0,  0,
        0,  0,  0,  0,  0,  0,  0, -1,  0,  0,  0,  0,  0,  0,  0,  0,  1,
        1,  1,  1,  1,  1,  1, -1,  1,  1, -1,  1,  1,  1,  1,  1,  1,  1,
       -1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,
        1,  1, -1,  1,  1,  1,  1,  1, -1,  1,  1,  1,  1, -1,  1,  1,  1,
        1,  1,  1, -1, -1,  1, -1, -1,  1,  1,  1,  1,  1,  1,  1, -1, -1,
        1,  1,  1, -1,  1,  1,  1,  1,  1,  1,  1,  1, -1,  1,  1, -1, -1,
        1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1,  1],
      dtype=int64)
```

In [20]:

```
from sklearn.decomposition import PCA
```

In [27]:

```
pca = PCA(n_components=2).fit(b)
```

In [28]:

```
pca_2d = pca.transform(b)
```

In [29]:

```
from sklearn.decomposition import PCA
pca = PCA(n_components=2).fit(iris.data)
pca_2d = pca.transform(iris.data)
for i in range(0, pca_2d.shape[0]):
 if dbscan.labels_[i] == 0:
  c1 = pl.scatter(pca_2d[i,0],pca_2d[i,1],c='r',
   marker='+')
 elif dbscan.labels_[i] == 1:
  c2 = pl.scatter(pca_2d[i,0],pca_2d[i,1],c='g',
   marker='o')
 elif dbscan.labels_[i] == -1:
  c3 = pl.scatter(pca_2d[i,0],pca_2d[i,1],c='b',
   marker='*')
pl.legend([c1, c2, c3], ['Cluster 1', 'Cluster 2',
   'Noise'])
pl.title('DBSCAN finds 2 clusters and noise')
pl.show()
```