INTRODUCTION

The Flower iris is used by R.A fisher in his research paper in 1936,to use for the multiple classification problem. Dataset is also available on Kaggele and UCI.

The major purpose of this project is to show some graphs i.e. data visulization like histograms and scatter plots. Where Histograms are similar to bar charts shows counts of data in bins and scatter plots show relationship between two data points, one can use third variable for the radius of circle of plot as well.

The total count of data is 150 rows, 50 rows for each class.

There are four features consists on these columns.

```
SepalLength
SepalWidth
PetalLength
PetalWidth
Species
```

The class variale is "Species". It has 3 number of classes.

Classes are below:

```
Iris-setosa
Iris-versicolor
Iris-virginica
```

Necessary Imports

In [0]:
```
#imports

import os
import matplotlib.pyplot as plt
import pandas as pd
```

Read Dataset

In [5]:
```
#read dataset

data = pd.read_csv('drive/My Drive/iris.csv')
data.head()
```

Out[5]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

In [7]:
```
#data describe
data.describe()
```

Out[7]:

	sepal_length	sepal_width	petal_length	petal_width
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 [8]:
```
data['sepal_length'].describe()
```

Out[8]:
```
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 [9]:
```
data['sepal_width'].describe()
```

Out[9]:
```
count    150.000000
mean       3.054000
std        0.433594
min        2.000000
25%        2.800000
50%        3.000000
75%        3.300000
max        4.400000
Name: sepal_width, dtype: float64
```

In [10]:
```
data['petal_length'].describe()
```

Out[10]:
```
count    150.000000
mean       3.758667
std        1.764420
min        1.000000
25%        1.600000
50%        4.350000
75%        5.100000
max        6.900000
Name: petal_length, dtype: float64
```

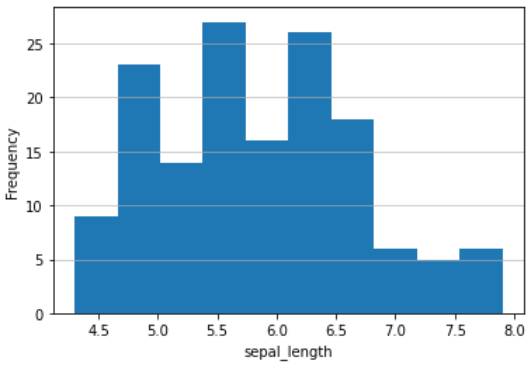
In [11]:
```
data['petal_width'].describe()
```

Out[11]:
```
count    150.000000
mean       1.198667
std        0.763161
min        0.100000
25%        0.300000
50%        1.300000
75%        1.800000
max        2.500000
Name: petal_width, dtype: float64
```

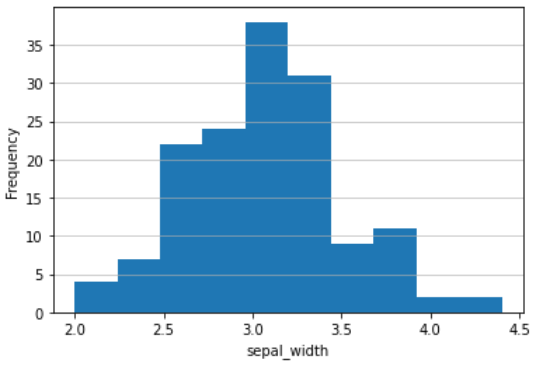
HISTOGRAM

A histogram is a graphical representation that organizes a group of data points into user-specified ranges.

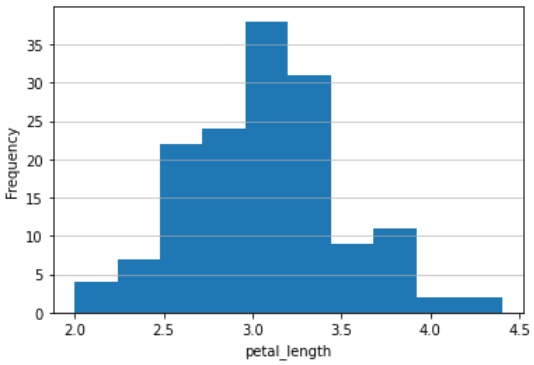
In [12]:
```
#histogram

col = data.columns
commutes0 = pd.Series(data[col[0]])
commutes0.plot.hist()
plt.xlabel(col[0])
plt.grid(axis='y', alpha=0.75)
plt.savefig(col[0]+'.png')
plt.show()
```



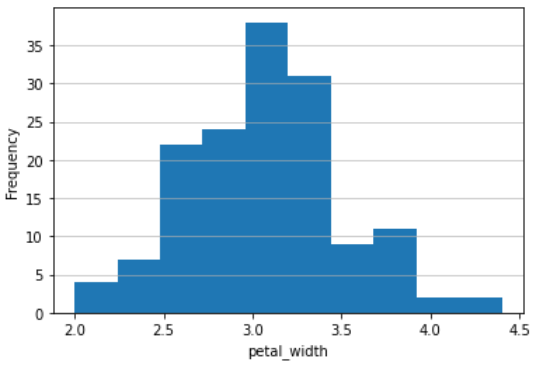
In [13]:
```
commutes1 = pd.Series(data[col[1]])
commutes1.plot.hist()
plt.xlabel(col[1])
plt.grid(axis='y', alpha=0.75)
plt.savefig(col[1]+'.png')
plt.show()
```



In [14]:
```
commutes1 = pd.Series(data[col[1]])
commutes1.plot.hist()
plt.xlabel(col[2])
plt.grid(axis='y', alpha=0.75)
plt.savefig(col[2]+'.png')
plt.show()
```



In [15]:
```
commutes1 = pd.Series(data[col[1]])
commutes1.plot.hist()
plt.xlabel(col[3])
plt.grid(axis='y', alpha=0.75)
plt.savefig(col[3]+'.png')
plt.show()
```

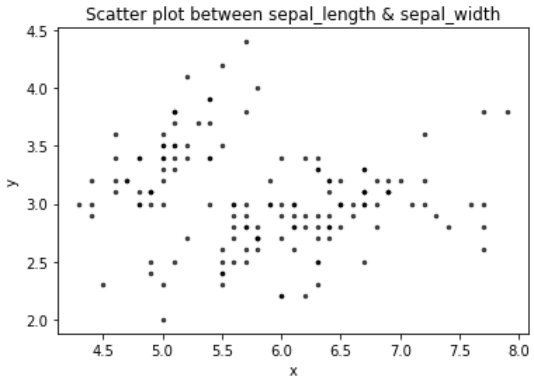


SCATTER PLOT

A graph of plotted points that show the relationship between two sets of data.

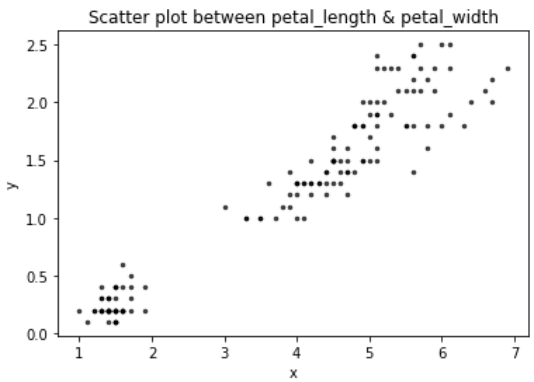
In [16]:
```
colors = (0,0,0)
area = 3.14*2
plt.scatter(data[col[0]], data[col[1]], s=area, c=colors, alpha=0.7)
plt.title('Scatter plot between '+ col[0]+ ' & ' + col[1])
plt.xlabel('x')
plt.ylabel('y')
plt.savefig('Scatter plot between '+ col[0]+ '  ' + col[1]+'.png')
plt.show()
```

```
'c' argument looks like a single numeric RGB or RGBA sequence, which should be avoided as val
ue-mapping will have precedence in case its length matches with 'x' & 'y'.  Please use a 2-D
array with a single row if you really want to specify the same RGB or RGBA value for all poin
ts.
```



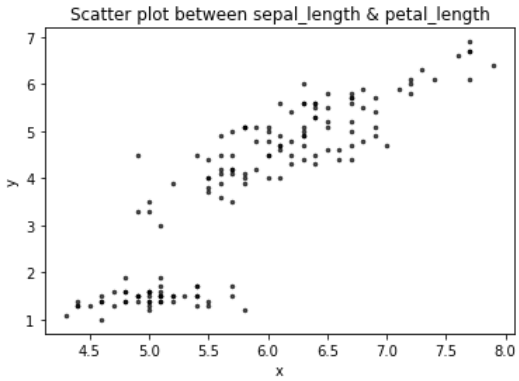
In [17]:
```

plt.scatter(data[col[2]], data[col[3]], s=area, c=colors, alpha=0.7)
plt.title('Scatter plot between '+ col[2]+ ' & ' + col[3])
plt.xlabel('x')
plt.ylabel('y')
plt.savefig('Scatter plot between '+ col[2]+ '  ' + col[3]+'.png')
plt.show()
```

```
'c' argument looks like a single numeric RGB or RGBA sequence, which should be avoided as val
ue-mapping will have precedence in case its length matches with 'x' & 'y'.  Please use a 2-D
array with a single row if you really want to specify the same RGB or RGBA value for all poin
ts.
```



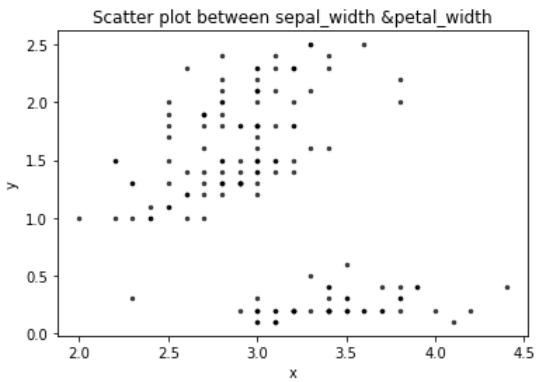
In [18]:
```
plt.scatter(data[col[0]], data[col[2]], s=area, c=colors, alpha=0.7)
plt.title('Scatter plot between '+ col[0]+ ' & ' + col[2])
plt.xlabel('x')
plt.ylabel('y')
plt.savefig('Scatter plot between '+ col[0]+ ' ' + col[2]+'.png')
plt.show()
```

```
'c' argument looks like a single numeric RGB or RGBA sequence, which should be avoided as val
ue-mapping will have precedence in case its length matches with 'x' & 'y'.  Please use a 2-D
array with a single row if you really want to specify the same RGB or RGBA value for all poin
ts.
```



In [19]:
```
plt.scatter(data[col[1]], data[col[3]], s=area, c=colors, alpha=0.7)
plt.title('Scatter plot between '+ col[1]+ ' &' + col[3] )
plt.xlabel('x')
plt.ylabel('y')
plt.savefig('Scatter plot between '+ col[1]+ ' ' + col[3]+'.png')
plt.show()
```

```
'c' argument looks like a single numeric RGB or RGBA sequence, which should be avoided as val
ue-mapping will have precedence in case its length matches with 'x' & 'y'.  Please use a 2-D
array with a single row if you really want to specify the same RGB or RGBA value for all poin
ts.
```



In [0]: