In [1]:
```
square = lambda n: n ** 2
print(square(5))
#map function
def square(n):
    return n ** 2
squares = map(square, range(1, 10, 2))
squares
print(list(squares))
#filter function
nums = [1, 34, 23, 56, 89, 44, 92]
odds = list(filter(lambda x: x % 2 != 0, nums))
print(odds)
#reduce function
from functools import reduce
nums = [1, 2, 3, 4, 5]
summ = reduce(lambda x, y: x + y, nums)
print(summ)
```

```
25
[1, 9, 25, 49, 81]
[1, 23, 89]
15
```

In [2]:
```
import numpy as np
a=np.array([1,2,3,4,5])
print("a :",a)
sum=np.sum(a)
print("sum :",sum)
product=np.prod(a)
print("product :",product)
mean=np.mean(a)
print("mean :",mean)
standard_deviation=np.std(a)
print("standard_deviation :",standard_deviation)
variance=np.var(a)
print("variance :",variance)
minimum=np.min(a)
print("minimum value :",minimum)
maximum=np.max(a)
print("maximum value :",maximum)
minimum_index=np.argmin(a)
print("minimum index :",minimum_index)
maximum_index=np.argmax(a)
print("maximum-index :",maximum_index)
median=np.median(a)
print("median :",median)
```

```
a : [1 2 3 4 5]
sum : 15
product : 120
mean : 3.0
standard_deviation : 1.4142135623730951
variance : 2.0
minimum value : 1
maximum value : 5
minimum index : 0
maximum-index : 4
median : 3.0
```

In [3]:
```
# import module
import pandas as pd

# Creating our dataset
```

```
df = pd.DataFrame([[9, 4, 8, 9],
            [8, 10, 7, 6],
            [7, 6, 8, 5]],
            columns=['Maths', 'English',
                  'Science', 'History'])

# display dataset
print(df)
print(df.sum())
print(df.describe())
print(df.agg(['sum', 'min', 'max']))
print(df.groupby(by=['Maths']))
a = df.groupby('Maths')
print(a.first())
```

```
   Maths  English  Science  History
0      9        4        8        9
1      8       10        7        6
2      7        6        8        5
Maths      24
English    20
Science    23
History    20
dtype: int64
       Maths    English   Science   History
count    3.0   3.000000  3.000000  3.000000
mean     8.0   6.666667  7.666667  6.666667
std      1.0   3.055050  0.577350  2.081666
min      7.0   4.000000  7.000000  5.000000
25%      7.5   5.000000  7.500000  5.500000
50%      8.0   6.000000  8.000000  6.000000
75%      8.5   8.000000  8.000000  7.500000
max      9.0  10.000000  8.000000  9.000000
     Maths  English  Science  History
sum     24       20       23       20
min      7        4        7        5
max      9       10        8        9
<pandas.core.groupby.generic.DataFrameGroupBy object at 0x0000019040F217F0>
       English  Science  History
Maths
7            6        8        5
8           10        7        6
9            4        8        9
```

In [4]:
```
import pandas as pd
from datetime import datetime
import numpy as np

range_date = pd.date_range(start ='1/1/2019', end ='1/08/2019',
                                                  freq ='Min')

df = pd.DataFrame(range_date, columns =['date'])
df['data'] = np.random.randint(0, 100, size =(len(range_date)))

string_data = [str(x) for x in range_date]
print(string_data[1:11])
```

```
['2019-01-01 00:01:00', '2019-01-01 00:02:00', '2019-01-01 00:03:00', '2019-01-01
00:04:00', '2019-01-01 00:05:00', '2019-01-01 00:06:00', '2019-01-01 00:07:00', '2
019-01-01 00:08:00', '2019-01-01 00:09:00', '2019-01-01 00:10:00']
```

In [5]:
```
import pandas as pd

data = {"City": ["Pune", "Satara", "Solapur"], "ID": [1, 2, 3], "Fav": ["1", "3",
dataf = pd.DataFrame(data)
print("Before melting..")
print(dataf)

melt_df = pd.melt(dataf, id_vars=["ID"], value_vars=["City", "Fav"])
print("After melting..")
print(melt_df)
import pandas as pd

data = {"City": ["Pune", "Satara", "Solapur"], "ID": [1, 2, 3], "Fav": ["1", "3",

dataf = pd.DataFrame(data)
print("Before melting..")
print(dataf)

melt_df = pd.melt(dataf, id_vars=["ID"], value_vars=["City","Fav"], var_name="Expr
print("After melting..")
print(melt_df)

unmelt = melt_df.pivot(index='ID', columns='Expression')
print("Post unmelting..")
print(unmelt)
```

```
Before melting..
      City  ID Fav
0     Pune   1   1
1   Satara   2   3
2  Solapur   3  10
After melting..
   ID variable    value
0   1     City     Pune
1   2     City   Satara
2   3     City  Solapur
3   1      Fav        1
4   2      Fav        3
5   3      Fav       10
Before melting..
      City  ID Fav
0     Pune   1   1
1   Satara   2   3
2  Solapur   3  10
After melting..
   ID Expression    Value
0   1       City     Pune
1   2       City   Satara
2   3       City  Solapur
3   1        Fav        1
4   2        Fav        3
5   3        Fav       10
Post unmelting..
              Value
Expression     City Fav
ID
1              Pune   1
2            Satara   3
3           Solapur  10
```

In [12]:

```
import pandas as pd
import numpy as np
technologies= {
    'Fee' :[22000,25000,23000,np.NaN,26000],
    'Duration':['30days','50days','30days','35days','40days']
          }
df = pd.DataFrame(technologies)
```

```
print(df)
# Using Lambda Function
df['Fee'] = df['Fee'].map(lambda x: x - (x*10/100))
print(df)



import pandas as pd
df = pd.DataFrame(np.array(([2, 3, 4], [5, 6, 7])),
                  index=['cat', 'dog'],
                  columns=['one', 'two', 'three'])
print(df.filter(items=['one', 'two']))
# select columns by regular expression
print(df.filter(regex='e$', axis=1))
```

```
       Fee Duration
0  22000.0   30days
1  25000.0   50days
2  23000.0   30days
3      NaN   35days
4  26000.0   40days
       Fee Duration
0  19800.0   30days
1  22500.0   50days
2  20700.0   30days
3      NaN   35days
4  23400.0   40days
     one  two
cat    2    3
dog    5    6
     one  three
cat    2      4
dog    5      7
```

In [13]:
```
data = [11, 6, 7, 3, 28, 1]
series = pd.Series(data)
print(series)
# import functools module
import functools

# using reduce operation to apply function on the series
product = functools.reduce(lambda x,y : x*y,series)
print("Product:",product,sep="  ")
import pandas as pd

# creating and initializing a list
values = [['Rohan', 455], ['Elvish', 250], ['Deepak', 495],
          ['Soni', 400], ['Radhika', 350], ['Vansh', 450]]

# creating a pandas dataframe
df = pd.DataFrame(values, columns=['Name', 'Total_Marks'])

# Applying lambda function to find
# percentage of 'Total_Marks' column
# using df.assign()
df = df.assign(Percentage=lambda x: (x['Total_Marks'] / 500 * 100))

# displaying the data frame
print(df)
```

```
0    11
1     6
2     7
3     3
4    28
5     1
dtype: int64
Product:  38808
      Name  Total_Marks  Percentage
0    Rohan          455        91.0
1   Elvish          250        50.0
2   Deepak          495        99.0
3     Soni          400        80.0
4  Radhika          350        70.0
5    Vansh          450        90.0
```

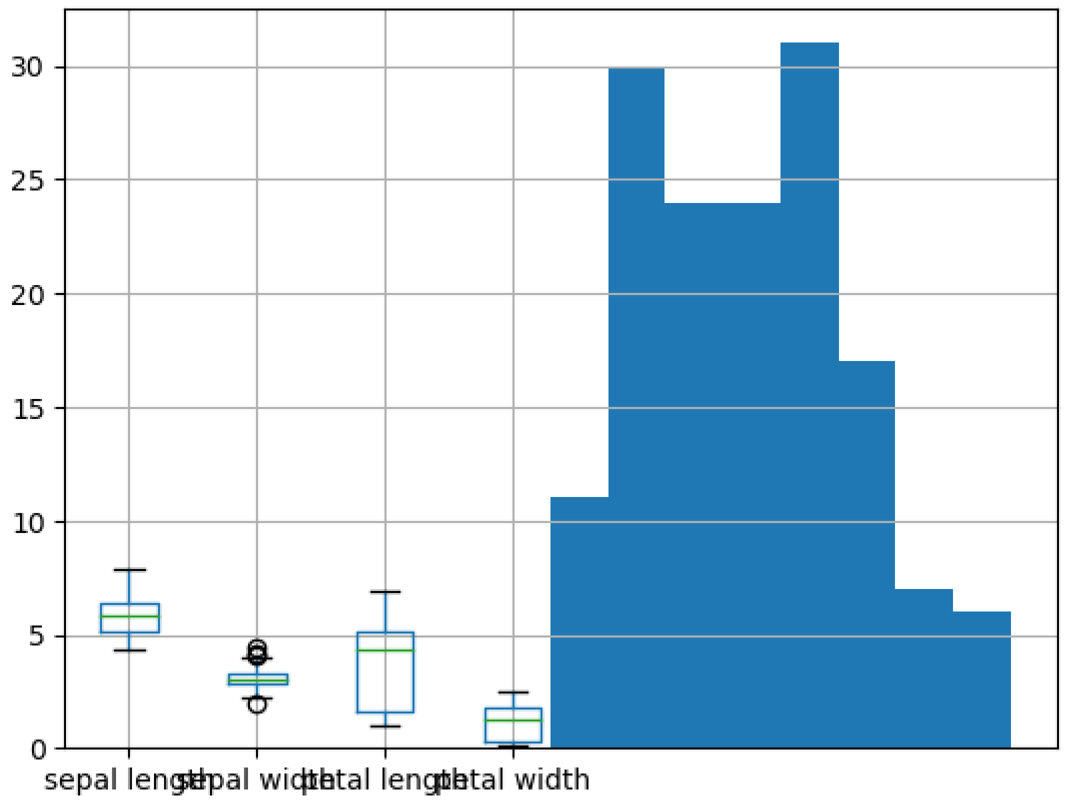
In [2]:

```
import pandas as pd
data = pd.read_csv('http://archive.ics.uci.edu/ml/machine-learning-databases/iris/i
data.columns = ['sepal length', 'sepal width', 'petal length', 'petal width', 'cla

data.head()

from pandas.api.types import is_numeric_dtype

for col in data.columns:
    if is_numeric_dtype(data[col]):
        print('%s:' % (col))
        print('\t Mean = %.2f' % data[col].mean())
        print('\t Mean = %.2f' % data[col].mean())
        print('\t Standard deviation = %.2f' % data[col].std())
        print('\t Minimum = %.2f' % data[col].min())
        print('\t Maximum = %.2f' % data[col].max())
data['class'].value_counts()

data.describe(include='all')

print('Covariance:')
data.cov()


%matplotlib inline

data['sepal length'].hist(bins=8)
data.boxplot()
```

```
sepal length:
         Mean = 5.84
         Mean = 5.84
         Standard deviation = 0.83
         Minimum = 4.30
         Maximum = 7.90
sepal width:
         Mean = 3.05
         Mean = 3.05
         Standard deviation = 0.43
         Minimum = 2.00
         Maximum = 4.40
petal length:
         Mean = 3.76
         Mean = 3.76
         Standard deviation = 1.76
         Minimum = 1.00
         Maximum = 6.90
petal width:
         Mean = 1.20
         Mean = 1.20
         Standard deviation = 0.76
         Minimum = 0.10
         Maximum = 2.50
Covariance:
```

Out[2]:

```
<AxesSubplot:>
```

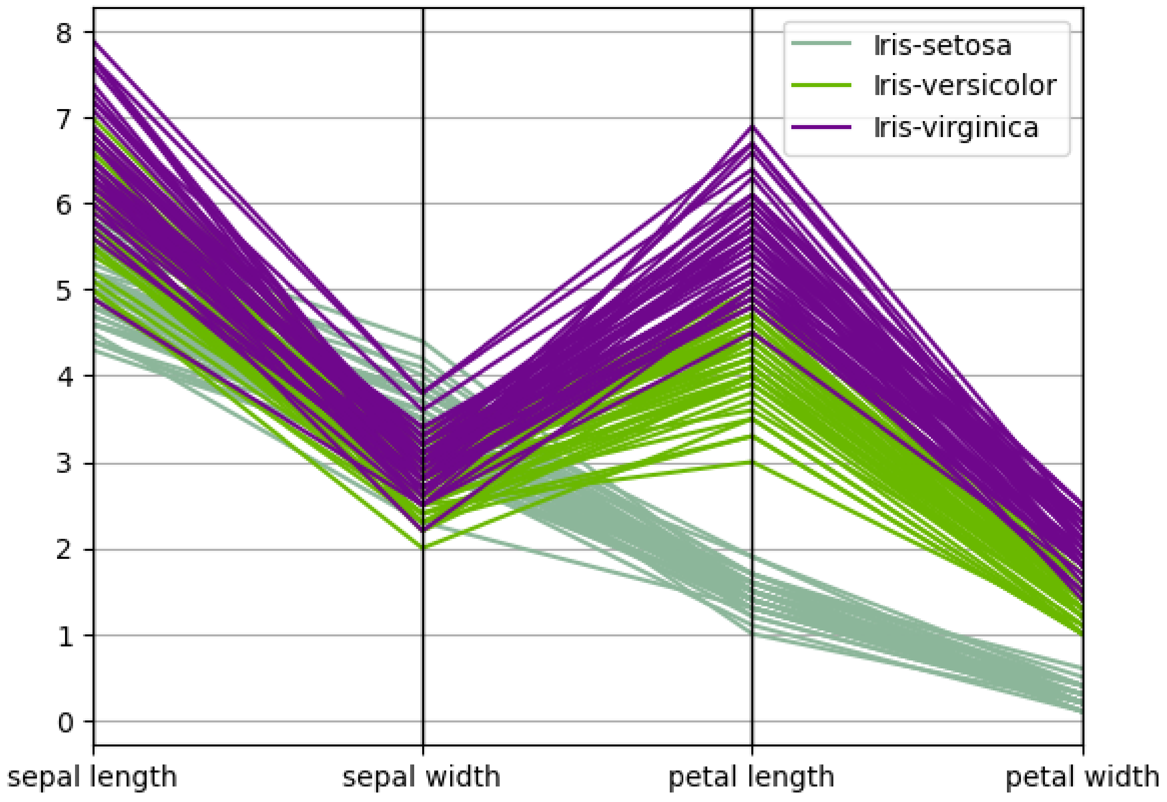


In [6]:

```python
import matplotlib.pyplot as plt

fig, axes = plt.subplots(3, 2, figsize=(12,12))
index = 0
for i in range(3):
    for j in range(i+1,4):
        ax1 = int(index/2)
        ax2 = index % 2
        axes[ax1][ax2].scatter(data[data.columns[i]], data[data.columns[j]], color
        axes[ax1][ax2].set_xlabel(data.columns[i])
        axes[ax1][ax2].set_ylabel(data.columns[j])
        index = index + 1
        from pandas.plotting import parallel_coordinates
%matplotlib inline

parallel_coordinates(data, 'class')
```

Out[6]:

```
<AxesSubplot:>
```



In [7]:

```
# importing pandas as pd
import pandas as pd

# Creating the series
sr = pd.Series([12, 5, None, 5, None, 11])

# Print the series
sr
# to detect the missing values
sr.isna()
```

Out[7]:

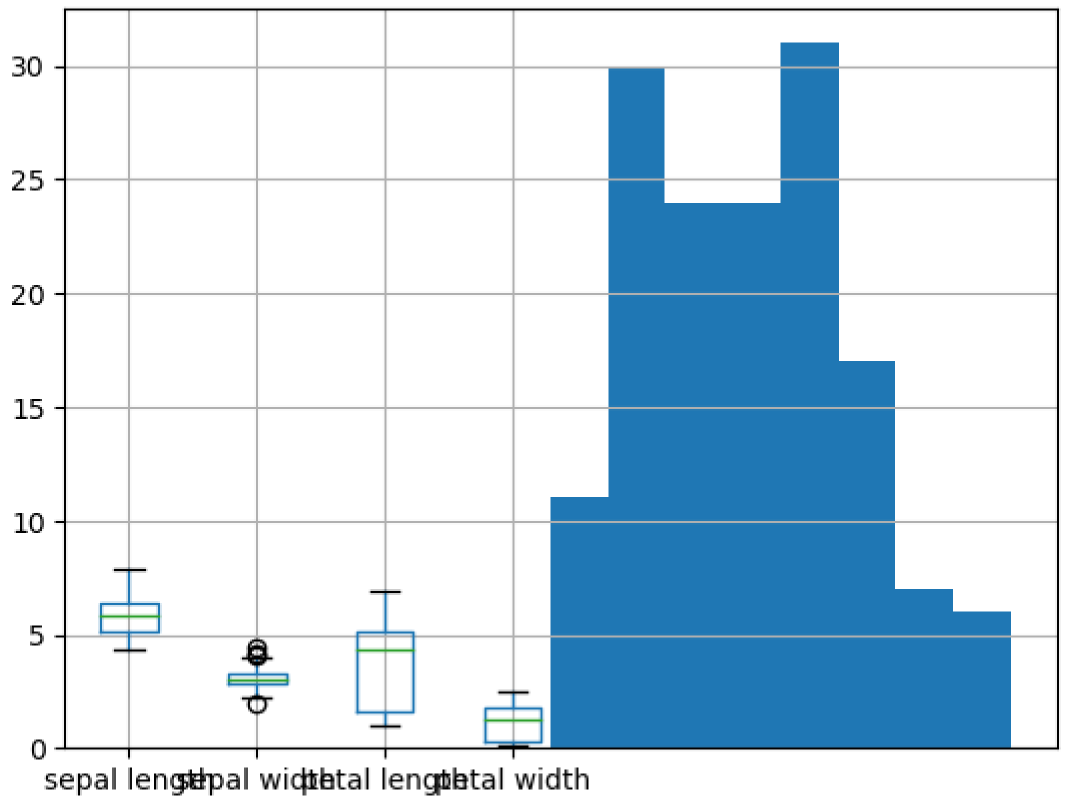
```
0    False
1    False
2     True
3    False
4     True
5    False
dtype: bool
```

In [8]:

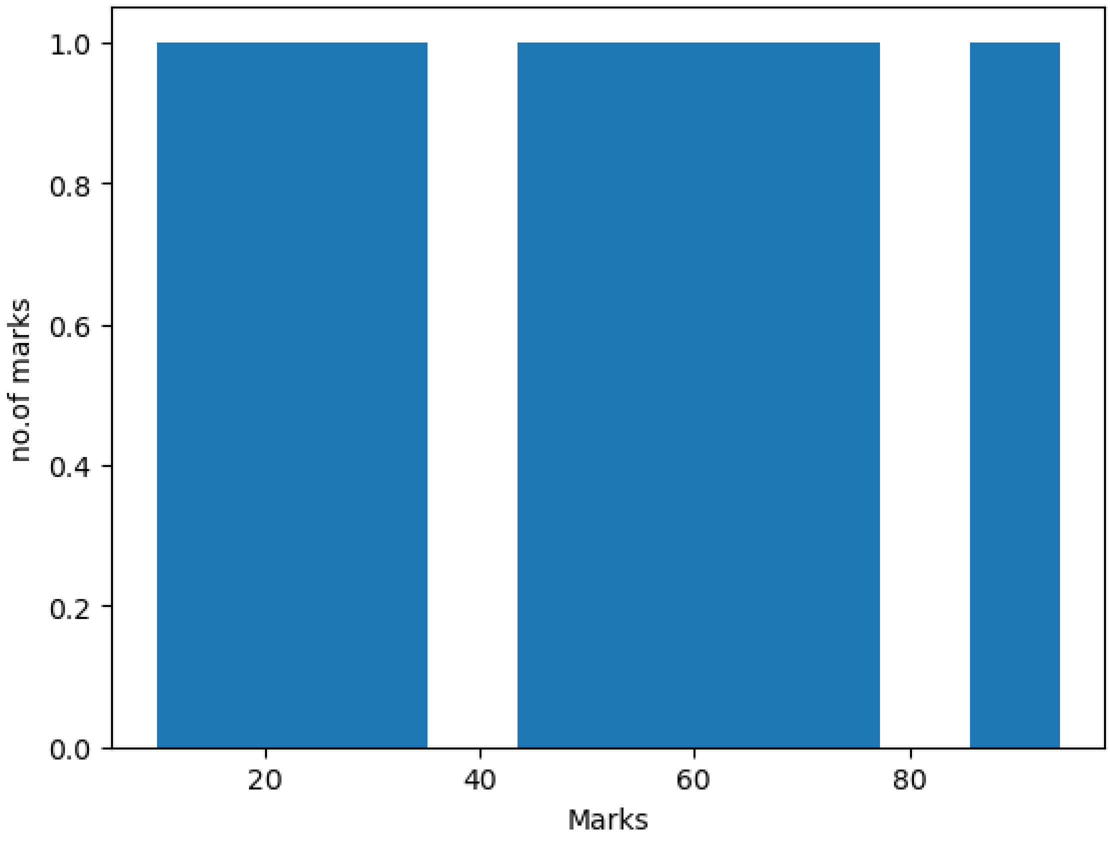
```
%matplotlib inline

data['sepal length'].hist(bins=8)
data.boxplot()
```

Out[8]:

```
<AxesSubplot:>
```



```
In [13]: import matplotlib.pyplot as plt
         import numpy as np
         fig,ax=plt.subplots(1,1)
         a=np.array([10,25,34,45,65,75,52,94])
         plt.hist(a)


         ax.set_xlabel('Marks')
         ax.set_ylabel('no.of marks')
         plt.show()
```



```
In [17]: import matplotlib.pyplot as plt
         import numpy as np
         fig,ax=plt.subplots(1,1)
         a=np.array=([10,20,30,40,40,50,60,70,80])
         plt.hist(a)
         ax.set_xlabel('Marks')
         ax.set_ylabel('No.of students')
         plt.show()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_9404\1458441211.py in <module>
      1 import matplotlib.pyplot as plt
      2 import numpy as np
----> 3 fig,ax=plt.subplots(1,1)
      4 a=np.array=([10,20,30,40,40,50,60,70,80])
      5 plt.hist(a)

~\anaconda3\lib\site-packages\matplotlib\pyplot.py in subplots(nrows, ncols, share
x, sharey, squeeze, subplot_kw, gridspec_kw, **fig_kw)
   1453
   1454     """
-> 1455     fig = figure(**fig_kw)
   1456     axs = fig.subplots(nrows=nrows, ncols=ncols, sharex=sharex, sharey=sha
rey,
   1457                        squeeze=squeeze, subplot_kw=subplot_kw,

~\anaconda3\lib\site-packages\matplotlib\pyplot.py in figure(num, figsize, dpi, fa
cecolor, edgecolor, frameon, FigureClass, clear, **kwargs)
    806                 RuntimeWarning)
    807
--> 808         manager = new_figure_manager(
    809             num, figsize=figsize, dpi=dpi,
    810             facecolor=facecolor, edgecolor=edgecolor, frameon=frameon,

~\anaconda3\lib\site-packages\matplotlib\pyplot.py in new_figure_manager(*args, **
kwargs)
    325     """Create a new figure manager instance."""
    326     _warn_if_gui_out_of_main_thread()
--> 327     return _get_backend_mod().new_figure_manager(*args, **kwargs)
    328
    329

~\anaconda3\lib\site-packages\matplotlib_inline\backend_inline.py in new_figure_ma
nager(num, FigureClass, *args, **kwargs)
     25     This function is part of the API expected by Matplotlib backends.
     26     """
---> 27     return new_figure_manager_given_figure(num, FigureClass(*args, **kwarg
s))
     28
     29

~\anaconda3\lib\site-packages\matplotlib\figure.py in __init__(self, figsize, dpi,
facecolor, edgecolor, linewidth, frameon, subplotpars, tight_layout, constrained_l
ayout, layout, **kwargs)
   2331             frameon = mpl.rcParams['figure.frameon']
   2332
-> 2333         if not np.isfinite(figsize).all() or (np.array(figsize) < 0).any
():
   2334             raise ValueError('figure size must be positive finite not '
   2335                              f'{figsize}')

TypeError: 'list' object is not callable
```