importing matplotlib

just as we use the np shorthand for numpy and pd as pandas, we will use standard shorthands fro matplotlib import

```
import matplotlib. pyplot as plt
```

we import the pyplot interface of matplotlib with shorthand of plt and we will be using it like this in the entire notebook.

matplotlib for jupyter notebook

you can directly use matplolib with this notebook to create diffrent visualizations in the notebook itself. in order to do that, the followin command si used

```
%matplotlib inline
```

In [1]:

```
#importing required libraries

import numpy as np

import pandas as pd
#importing matplotlib
```

```
import matplotlib.pyplot as plt

#display plots in the notebook itself
%matplotlib inline
```

Matplotlib basics

make a simple plot

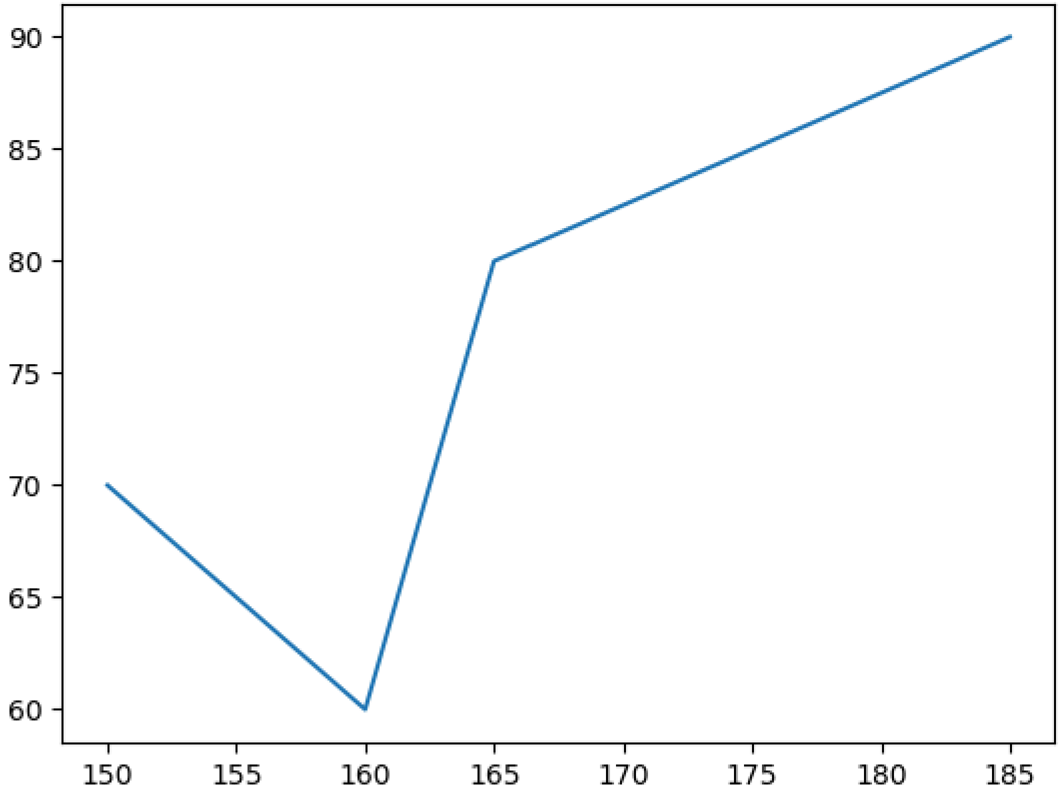
let's create a basic plot to start working with !

In [2]:

```
height=[150,160,165,185]
weight=[70,60,80,90]

#draw the plot

plt.plot(height,weight)
```

Out[2]:

```
[<matplotlib.lines.Line2D at 0x1db5970de50>]
```



we pass two arrays as our input arguments to plot() method and invoke the required plot. Here note the first array appears on the x-axis and second array appears on the y- axis of the plot

Time ,Lebles, and Legends

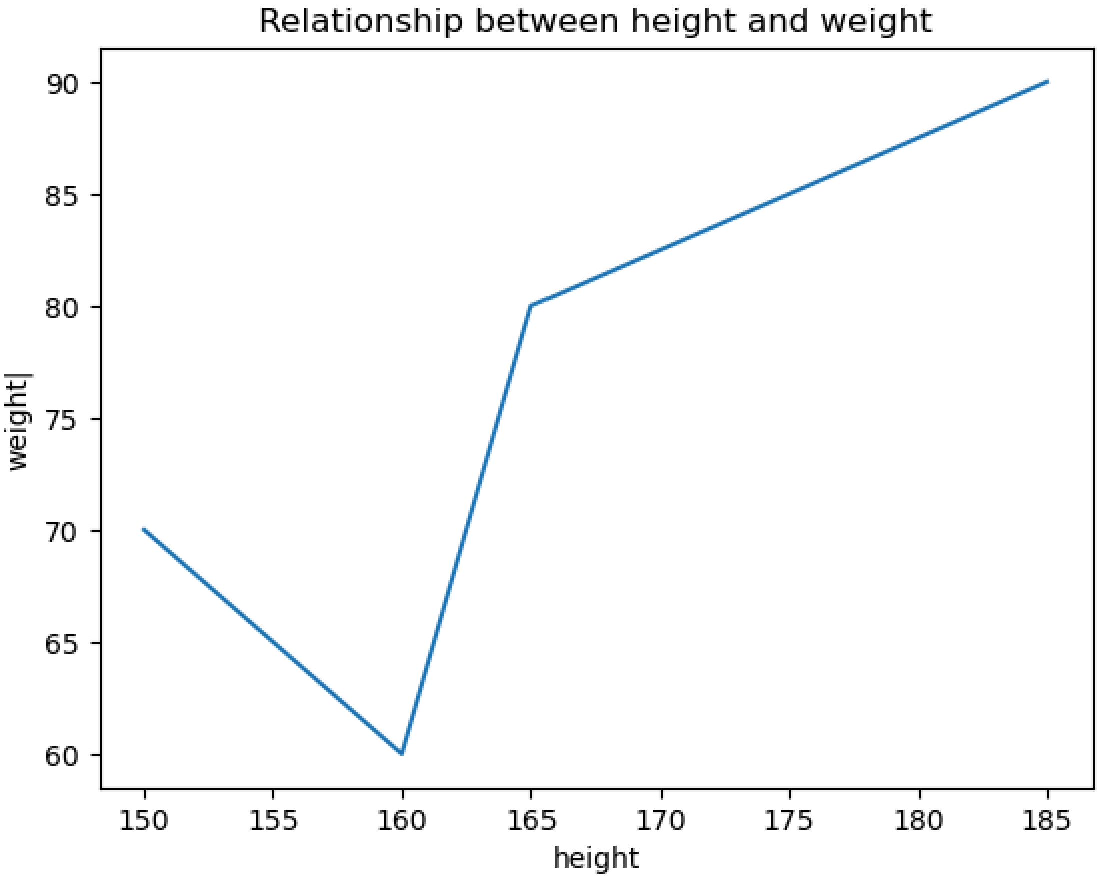
. now that our first plot is ready, let us add the title, and name x-axis and y-axis using method title(),xlabel() and ylabel() respectively

In [3]:

```
# draw the plot
plt.plot(height,weight)
#add title
plt.title(" Relationship between height and weight ")

#label x axis
plt.xlabel("height")

#label y axis

plt.ylabel("weight|")
```

Out[3]:

```
Text(0, 0.5, 'weight|')
```

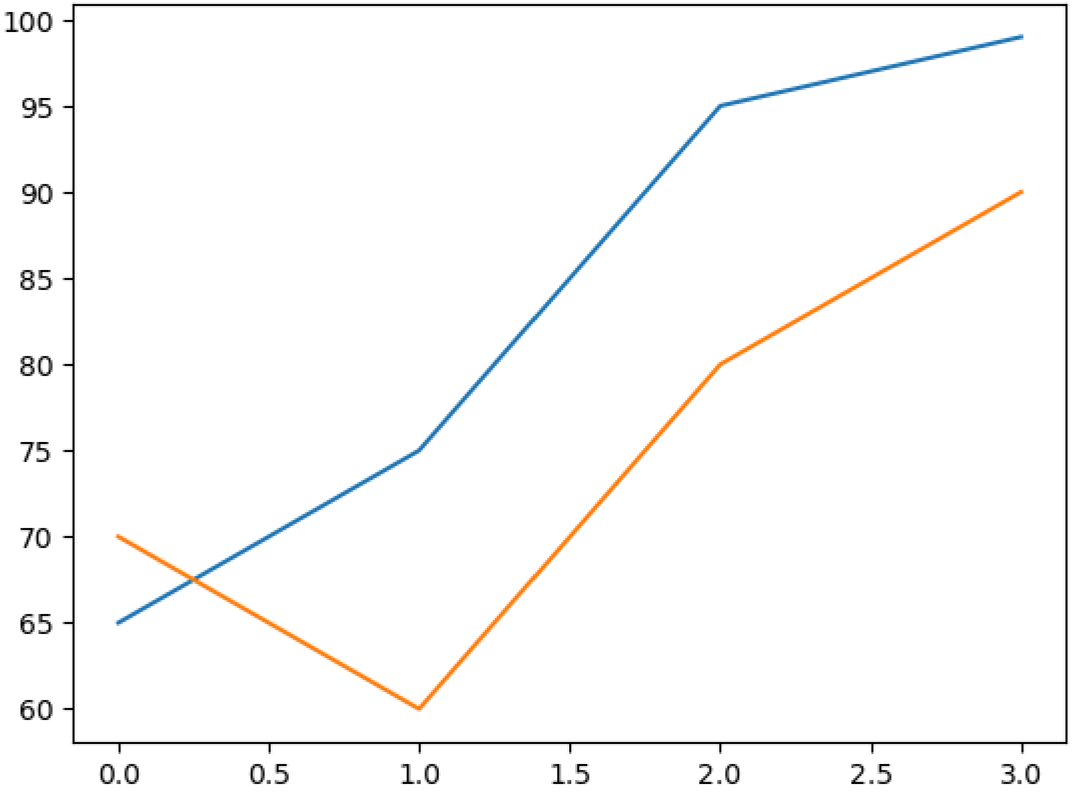


In [4]:

```
calories_brunt=[65,75,95,99]
#draw the plot for calories brunt
```

```python

plt.plot(calories_brunt)
# draw the plot for weight
plt.plot(weight)
```

```
Out[4]:  [<matplotlib.lines.Line2D at 0x1db598b6690>]
```



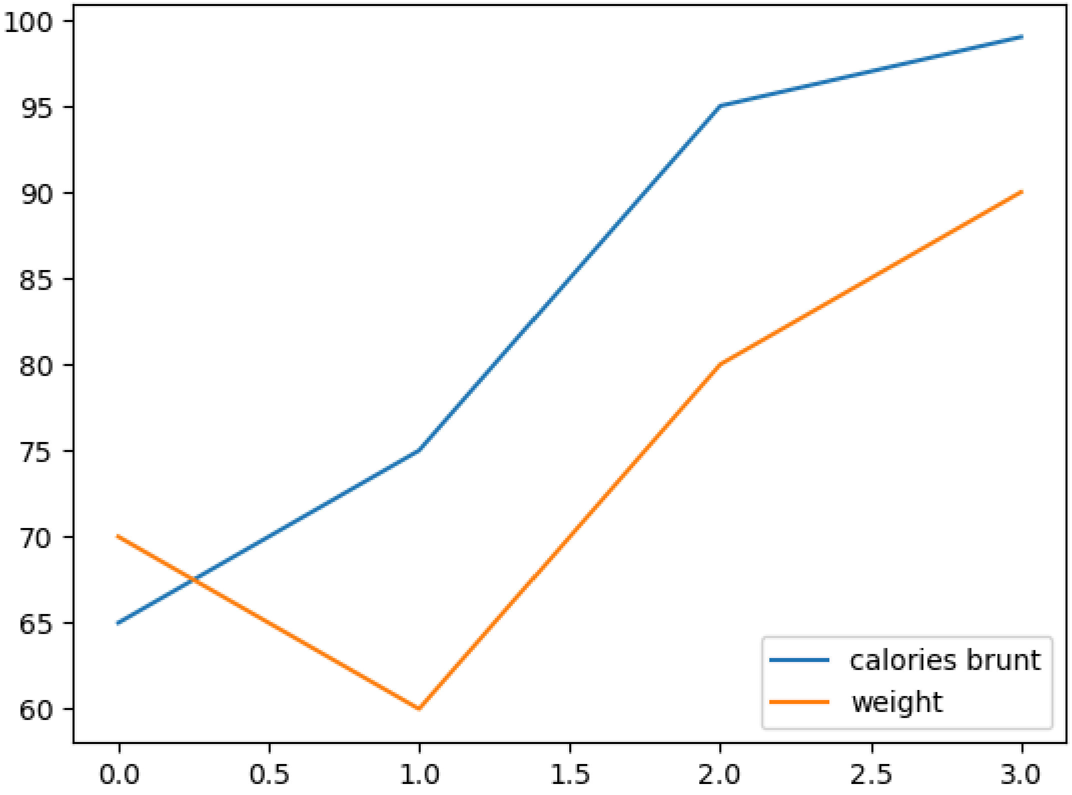
. Adding legends is also simple in matplotlib yopu can use the legend() whioch takes lasbels and loc as label names and location of legends in the figure as parameter

In [5]:

```
#draw the plot for calories brunt
plt.plot(calories_brunt)

#draw the plot for weight
plt.plot(weight)

# adding legends in the lower right part of the figure
plt.legend(labels=["calories brunt"  ,"weight"],loc="lower right" )
```

Out[5]:

```
<matplotlib.legend.Legend at 0x1db5993ea90>
```



. Notice that in the previous plot,we are not able to understand that each of these values belong to different personn.

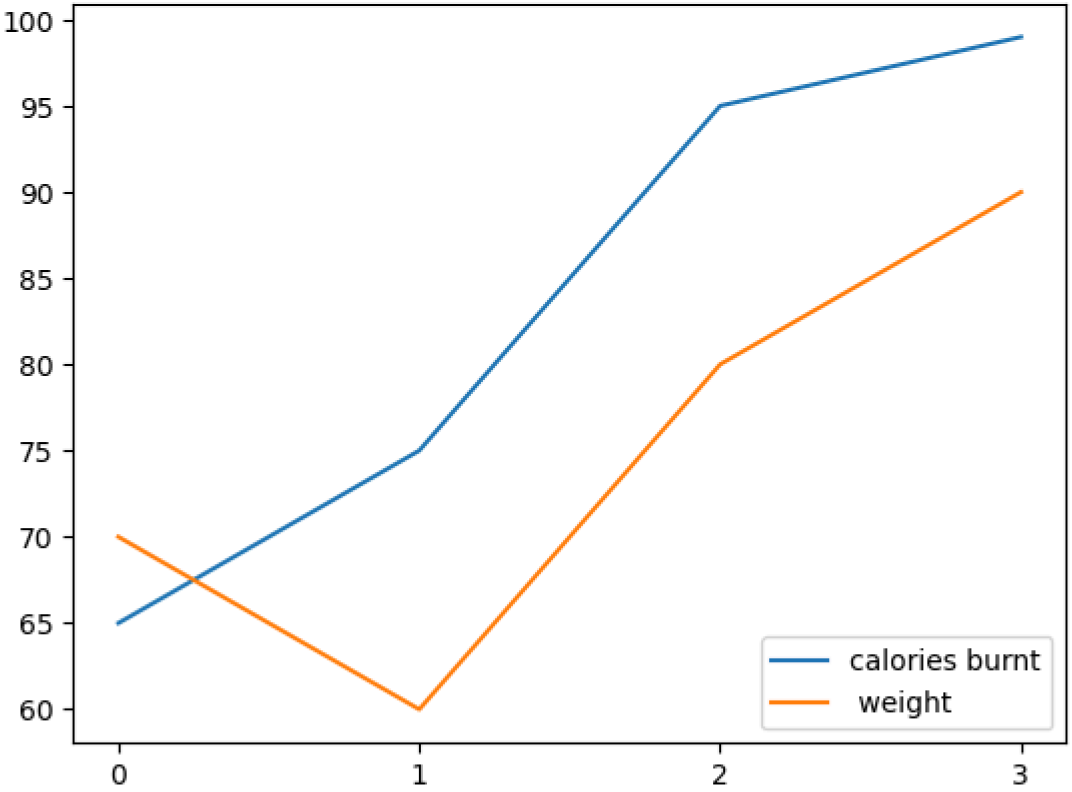
. Look at the x axis, can we add labels to show that each belong to different persons?

. The labeled values on any axis is known as a tick.

. you can use the xticks to change both the location of each tick and it's label. lets see

In [6]:

```
# draw the plot
plt.plot(calories_brunt)
plt.plot(weight)

# add legend in the lower right part of the future
plt.legend(labels=["calories burnt" ," weight"],loc="lower right" )

# set labels for each of these persons
plt.xticks(ticks=[0,1,2,3],label=["p1" ,"p2" ," p3" ," p4" ]);
```



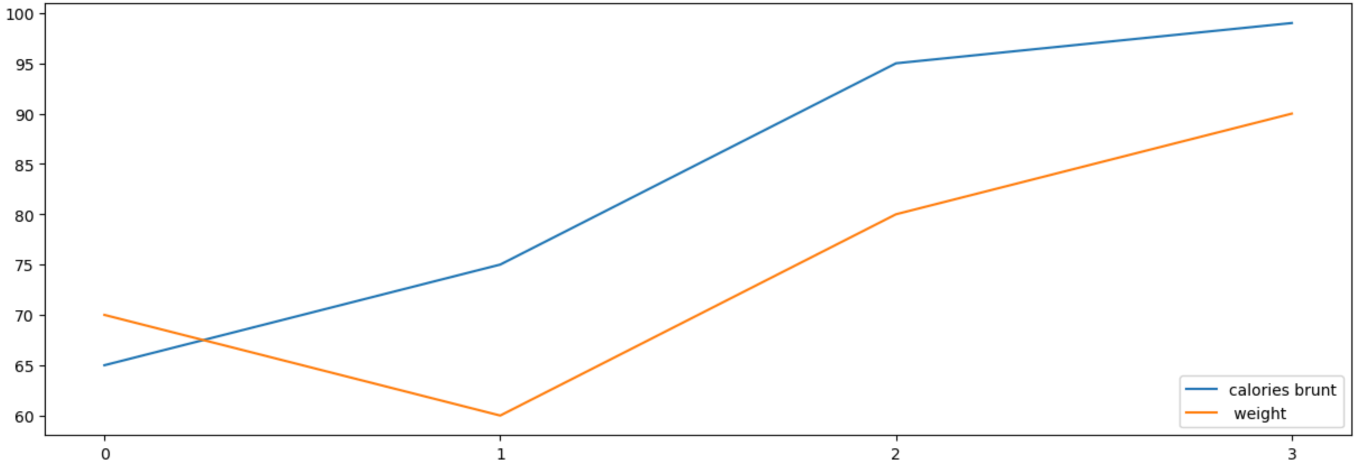
Size ,Colors,Makers,and Line styles

. you can also specify the size of the figure using method figure() and passing the values as a tuple of the length of rows and columns to the arguments figsize

. the values of the length are considered to be in inches.

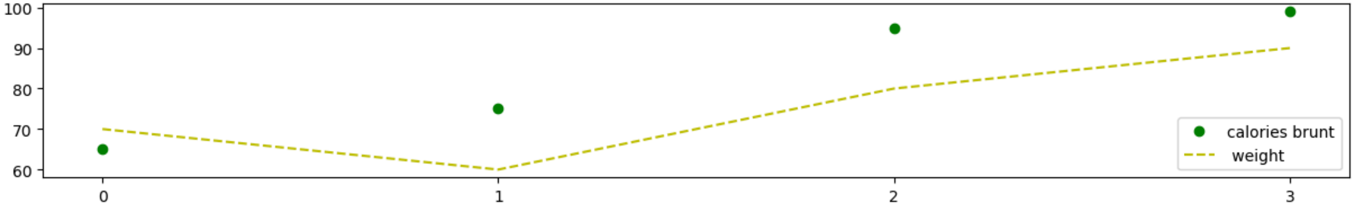
In [7]:

```
#figure size in inches

plt.figure(figsize=(15,5))
#draw the plot

plt.plot(calories_brunt)
plt.plot(weight)

#add legend in the lower right part of the figure

plt.legend(labels=["calories brunt" ," weight" ], loc="lower right" )

#set labels for each of these persons
plt.xticks(ticks=[0,1,2,3],label=["p1" ," p2" ,"p3","p4"]);
```



In [8]:

```
#figure size in inches

plt.figure(figsize=(15,2))
#draw the plot

plt.plot(calories_brunt," go" )
plt.plot(weight,"y--")

#add legend in the lower right part of the figure

plt.legend(labels=["calories brunt" ," weight" ], loc="lower right" )

#set labels for each of these persons
plt.xticks(ticks=[0,1,2,3],label=["p1" ," p2" ,"p3","p4"]);
```



. we can also plot multiple sets of arguments of x and y axis in the in the plot() method as shown.

Figure and subplots

. we can use subplots() method to add more thanone plots in one figure

. the subplots() method takes two arguments they are nrows,n cols.The indicate the numbers of rows, number of columns respectively.

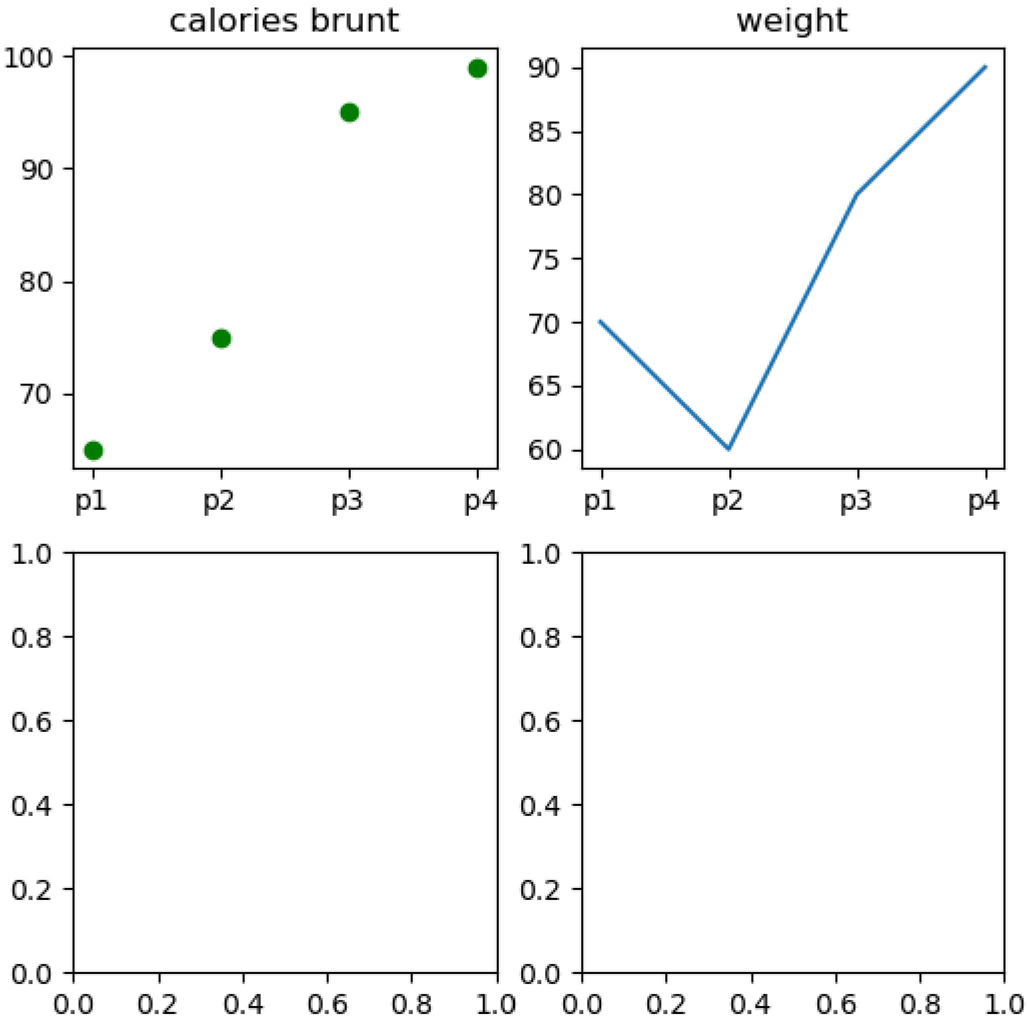
. this method creates two objects: figure and axes which we store in variable fig and ax.

. you plot each figure by specifying its posting using row inmdex and columns index. Lets have a look at the belopw example.

In [9]:

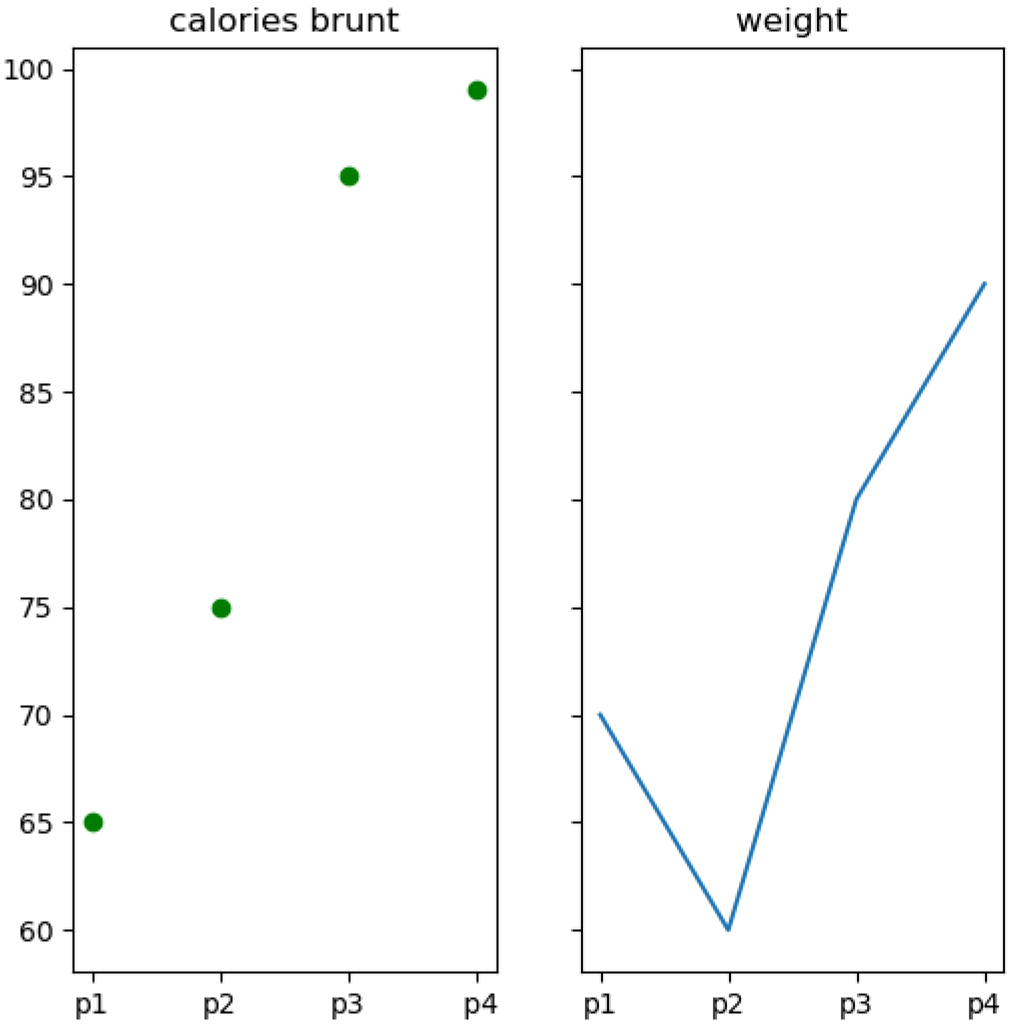
```
#create 2 plot
fig,ax=plt.subplots(nrows=2,ncols=2,figsize=(6,6))
#plot on 0 rows and 0 column
ax[0,0].plot(calories_brunt," go" )
#plot on 0 row and 1 columns
ax[0,1].plot(weight)
#set titles for subplopts
```

```
ax[0,0].set_title(" calories brunt ")
ax[0,1].set_title(" weight " )

#set ticks for each of these persons
ax[0,0].set_xticks(ticks=[0,1,2,3]);
ax[0,1].set_xticks(ticks=[0,1,2,3]);
#set labels for each of these persons

ax[0,0].set_xticklabels(labels=["p1" ,"p2" ,"p3" ," p4"]);
ax[0,1].set_xticklabels(labels=["p1" ,"p2" ,"p3" ,"p4" ]);
```


calories brunt
100
90
80
70
p1
p2
p3
p4
weight
90
85
80
75
70
65
60
p1
p2
p3
p4
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

In [10]:

```
#create 2 plot
fig,ax=plt.subplots(nrows=1,ncols=2,figsize=(6,6),sharey=True)
#plot on 0 rows and 0 column
ax[0].plot(calories_brunt," go" )
#plot on 0 row and 1 columns
ax[1].plot(weight)
#set titles for subplopts
ax[0].set_title(" calories brunt ")
ax[1].set_title(" weight " )

#set ticks for each of these persons
ax[0].set_xticks(ticks=[0,1,2,3]);
ax[1].set_xticks(ticks=[0,1,2,3]);
#set labels for each of these persons

ax[0].set_xticklabels(labels=["p1" ,"p2" ,"p3" ," p4"]);
ax[1].set_xticklabels(labels=["p1" ,"p2" ,"p3" ,"p4" ]);
```


calories brunt
weight
100
95
90
85
80
75
70
65
60
p1
p2
p3
p4
p1
p2
p3
p4

.notice in the above plot,now both x and y axes are only labelled once for each of the outer plots. this is because the inner plots "share" both the axes

also, there are only two plots since we decreased the number of rows to 1 and columns to 2 in the subplots().

load dataset

let's load a dataset and have a look at first five 5 rows

In [11]:

```
#read the dataset
data_bm=pd.read_csv("bigmart_data.csv")
#drop the null values
data_bm=data_bm.dropna(how="any")
#view the top results
data_bm.head()
```

Out[11]:

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identi
0	FDA15	9.300	Low Fat	0.016047	Dairy	249.8092	OUT
1	DRC01	5.920	Regular	0.019278	Soft Drinks	48.2692	OUT
2	FDN15	17.500	Low Fat	0.016760	Meat	141.6180	OUT
4	NCD19	8.930	Low Fat	0.000000	Household	53.8614	OUT
5	FDP36	10.395	Regular	0.000000	Baking Goods	51.4008	OUT

LINE CHART

We will create a line chart to denote the mean price per item. let's have a look at the code

with some dataset, you may want to understand changes inone variable as a function of time, or a simply continue vaariable.

. in matplotlib, line chart is the default plot when using the plot()

In [12]:

```
price_by_item=data_bm.groupby("Item_Type").Item_MRP.mean()[:10]
price_by_item
```

Out[12]:

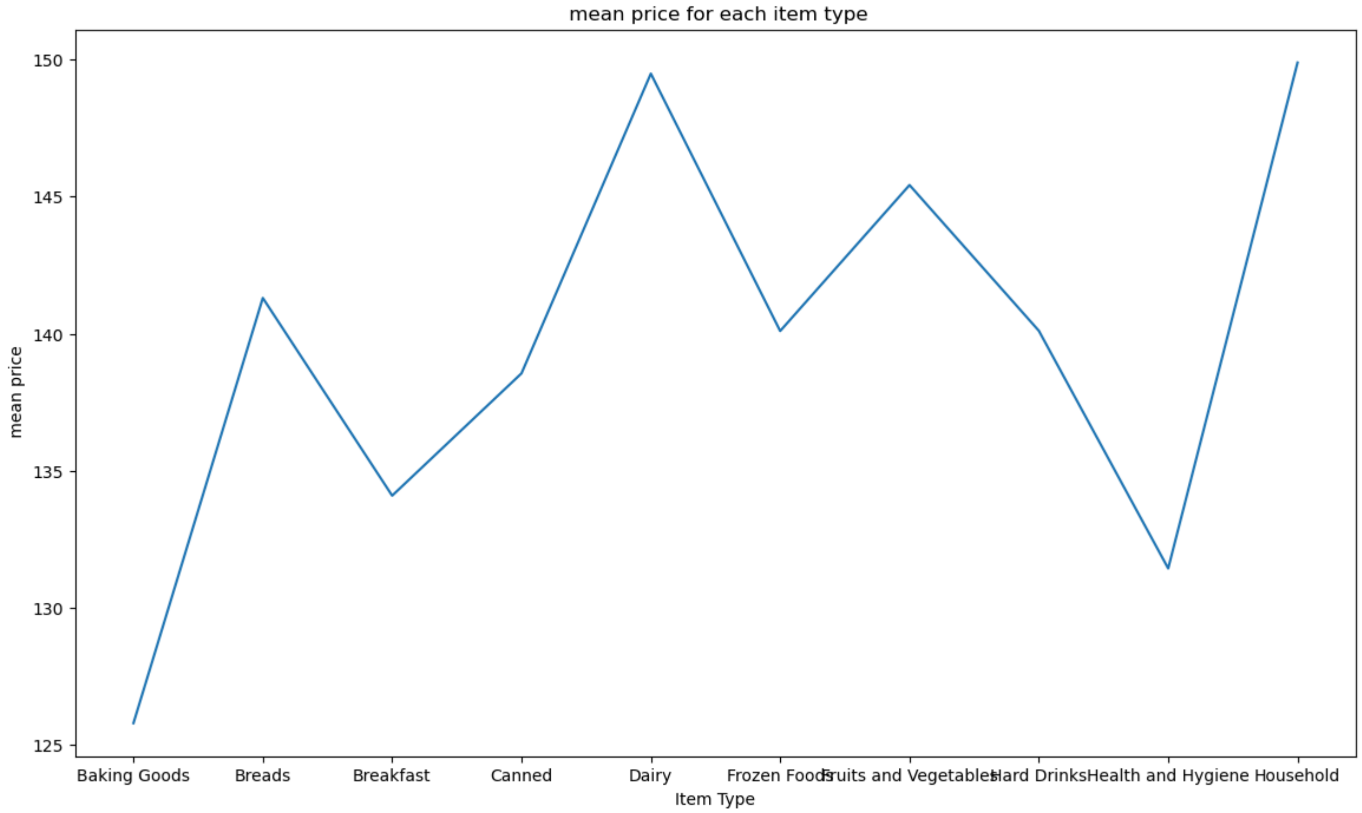
```
Item_Type
Baking Goods             125.795653
Breads                   141.300639
Breakfast                134.090683
Canned                   138.551179
Dairy                    149.481471
Frozen Foods             140.095830
Fruits and Vegetables    145.418257
Hard Drinks              140.102908
Health and Hygiene       131.437324
Household                149.884244
Name: Item_MRP, dtype: float64
```

In [13]:

```
#mean price based on item type
price_by_item=data_bm.groupby("Item_Type" ).Item_MRP.mean()[:10]

x=price_by_item.index.tolist()
y=price_by_item.values.tolist()

#set figure size
plt.figure(figsize=(14,8))

#title
plt.title(" mean price for each item type" )
#set axis labels
plt.xlabel("Item Type")
plt.ylabel(" mean price" )
```

```
#set xticks
plt.xticks(labels=x,ticks=np.arange(len(x)))
plt.plot(x,y)
```

```
Out[13]:  [<matplotlib.lines.Line2D at 0x1db59bf43d0>]
```



BAR CHART

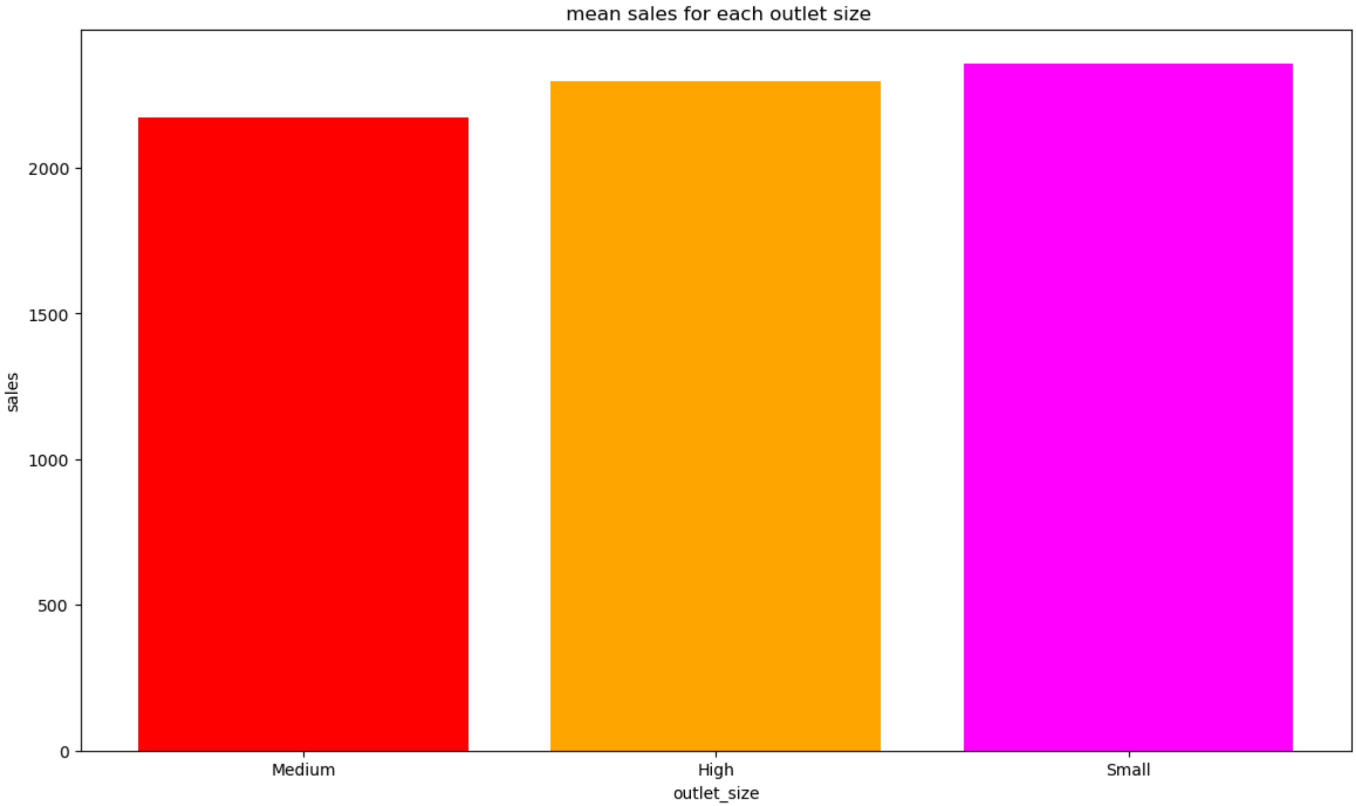
suppose we want to have a look at what is the mean sales for each outlet size ?

a bar chart is another simple type of visulaization that is used fro categorical variable.

you can use plt.bar() instead of plt.plot() to create a bar chart.

In [14]:
```
#mean price based on item type
sales_by_outlet_size=data_bm.groupby("Outlet_Size" ).Item_Outlet_Sales.mean()

#sort by sales
sales_by_outlet_size.sort_values(inplace=True)


x=sales_by_outlet_size.index.tolist()
y=sales_by_outlet_size.values.tolist()

#set figure size
plt.figure(figsize=(14,8))

#title
plt.title(" mean sales for each outlet size" )
#set axis labels
plt.xlabel("outlet_size")
plt.ylabel("sales" )

#set xticks
plt.xticks(labels=x,ticks=np.arange(len(x)))
plt.bar(x,y,color=["red" ,"orange" ,"magenta"])
```

Out[14]:
```
<BarContainer object of 3 artists>
```


mean sales for each outlet size
sales
0
500
1000
1500
2000
Medium
High
Small
outlet_size

HISTOGRAM

distribution of item price histogram are very common type of plots when we are looking at dat like height and weight, stok price,waiting time for a customer, etc which are continuous in nature.

histogram's data is plotted within a eange against its frequency.

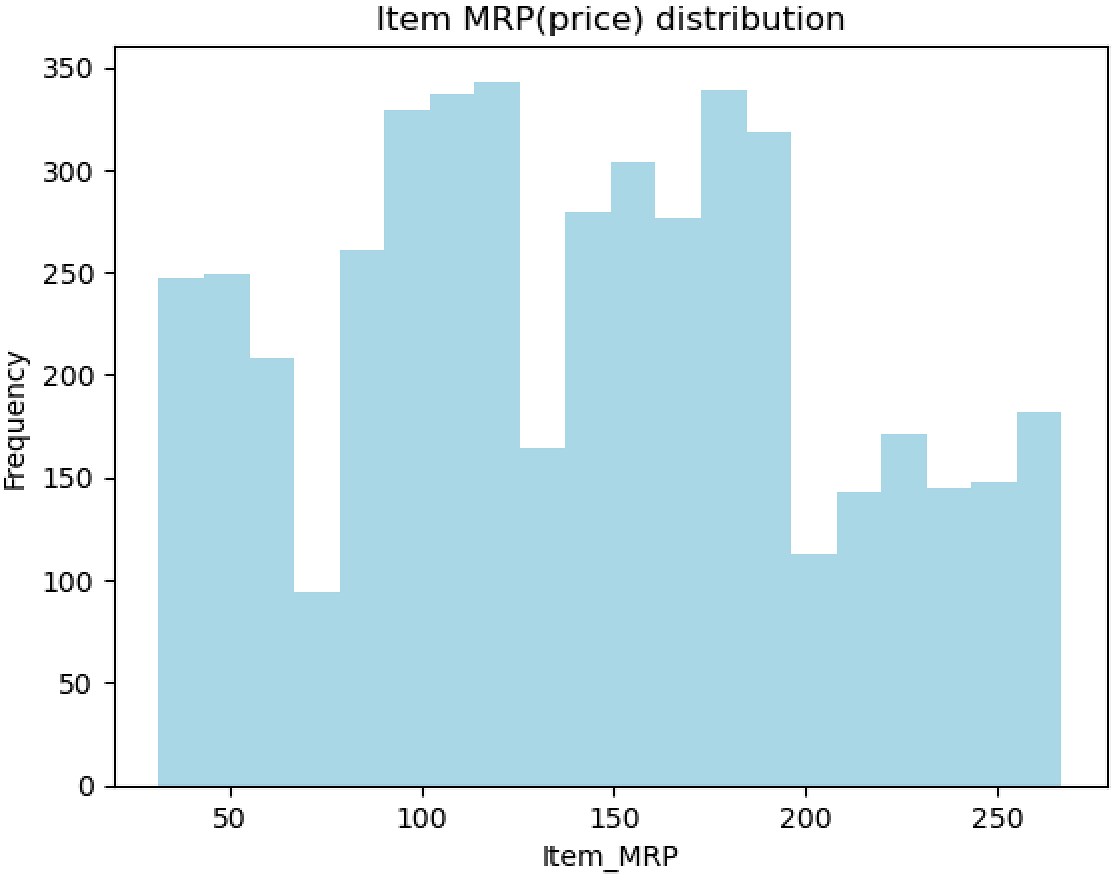
histpograms are very commonly occurring graphs in probablity and statistics and form the basis for various distribution like the normal distribution,t-distribution,etc.

you can use plt.hist() to draw a histogram. it provides many parameters to adjust the plot.

In [15]:

```
#title
plt.title("Item MRP(price) distribution")

#xlabel
plt.xlabel("Item_MRP")

#Y label

plt.ylabel("Frequency")

#plot histogram

plt.hist(data_bm["Item_MRP"],bins =20,color="lightblue" )
```

```
Out[15]:  (array([247., 249., 208.,  94., 261., 329., 337., 343., 164., 279., 304.,
                  276., 339., 318., 113., 143., 171., 145., 148., 182.]),
           array([ 31.49   ,  43.25992,  55.02984,  66.79976,  78.56968,  90.3396 ,
                  102.10952, 113.87944, 125.64936, 137.41928, 149.1892 , 160.95912,
                  172.72904, 184.49896, 196.26888, 208.0388 , 219.80872, 231.57864,
                  243.34856, 255.11848, 266.8884 ]),
           <BarContainer object of 20 artists>)
```


Item MRP(price) distribution
Frequency
350
300
250
200
150
100
50
0
50
100
150
200
250
Item_MRP

BOX PLOT

distribution of sales

box plot shows the threequartile values of the distribution along with extream value.

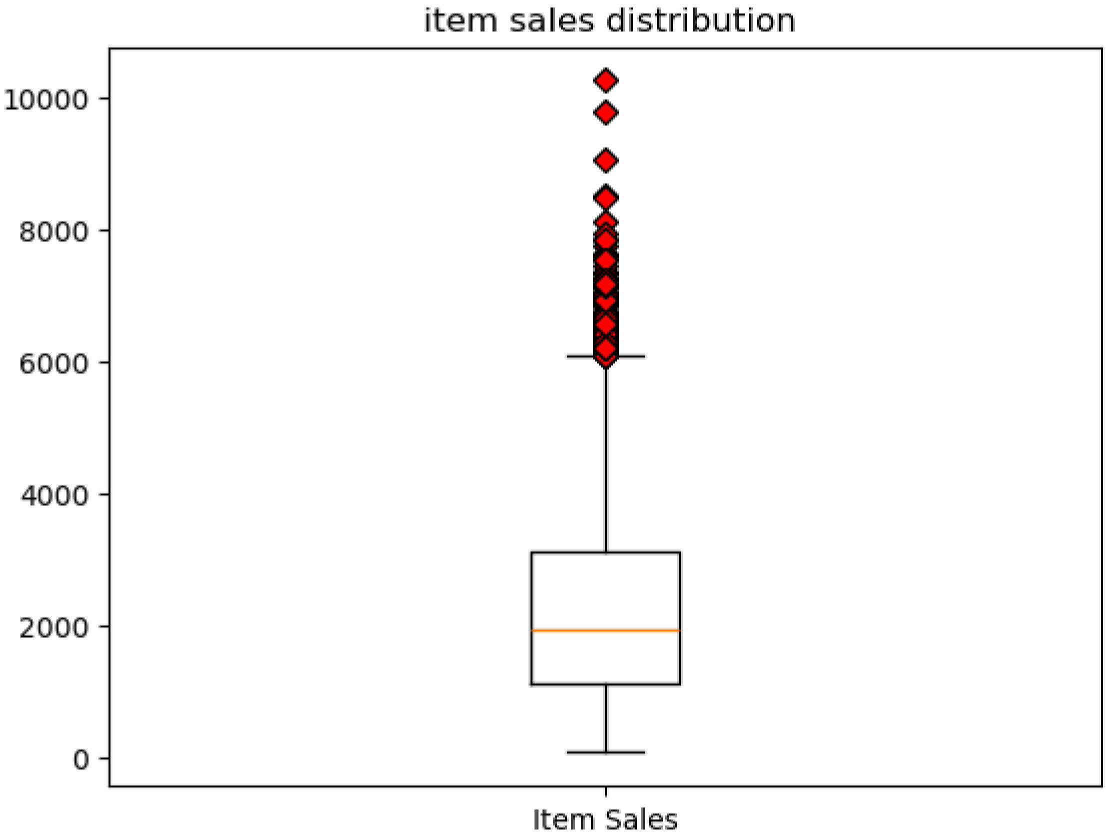
the "whishkers" extend to points thats lie within 1.5 iqrs of the lower aand upper quartile. and then observations that fall outside this range are displayed independently.

this means that each value in the boxplot corresponds to an actuial observation in the data.

let's try to visualize the distribution of item_outlet_sales of items.

In [16]:

```
data=data_bm[["Item_Outlet_Sales"]]
#create outlier point_shape

red_diamond=dict(markerfacecolor="r",marker="D" )

#set title

plt.title(" item sales distribution" )

# make the box plot
plt.boxplot(data.values,labels=["Item Sales"],flierprops=red_diamond);
```

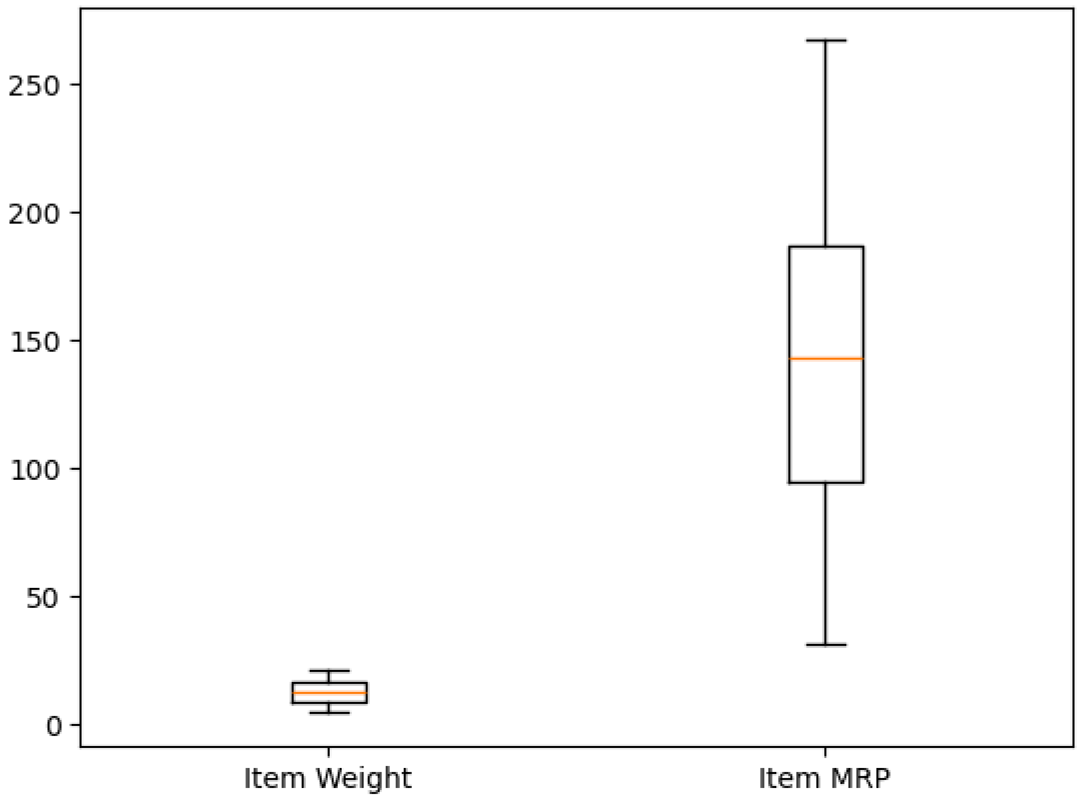


you can also create multiple boxplots for different columns of your dataset.

in order to plot multiple boxplots,you can use the same subplots()that we saw earlier. lets see iterm_weight,item_mrp distribution together.

In [17]:

```
data=data_bm[["Item_Weight" ,"Item_MRP"]]
#create outlier point shape
red_diamond=dict(markerfacecolor="r",marker="D" )

#generate subplots
fig, ax=plt.subplots()

#make the boxplot

plt.boxplot(data.values,labels=["Item Weight","Item MRP"]);
```

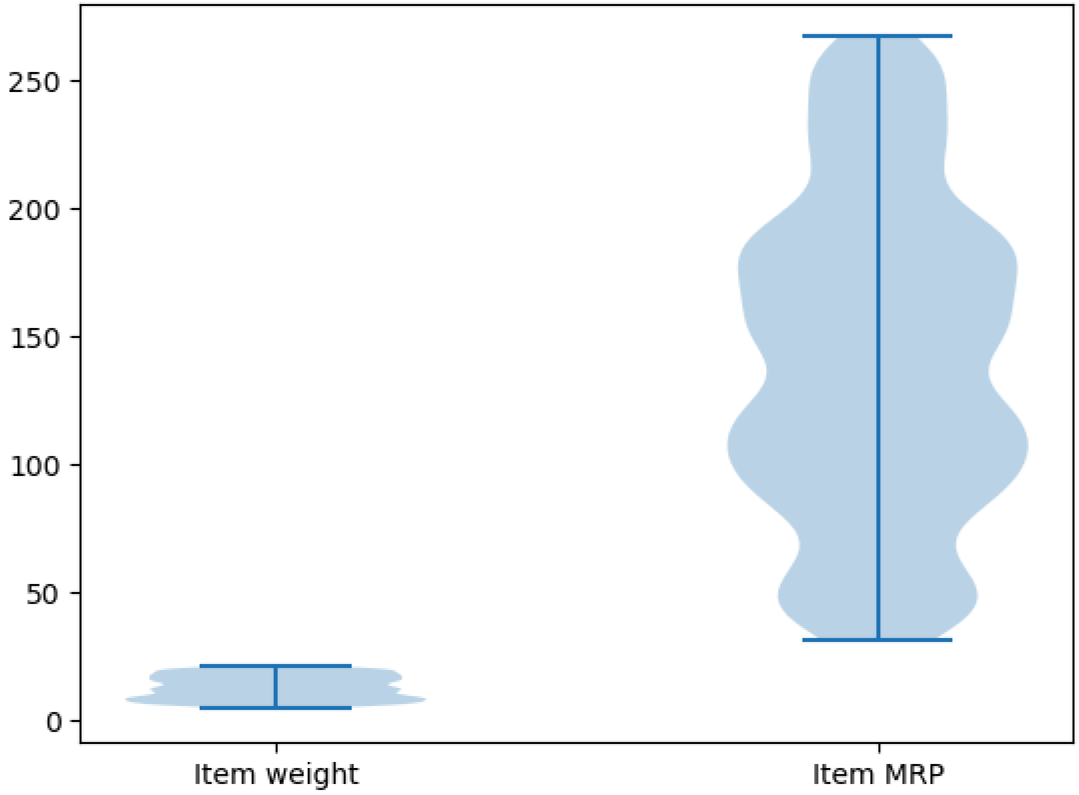


VIOlIN PLOTS

DENSITY DISTRIBUTION OF ITEM WEIGHTS AND ITEM PRICE

In [18]:

```
data=data_bm[["Item_Weight","Item_MRP"]]
#generate saubplots
fig,ax=plt.subplots()

#add labels to x axis
plt.xticks(ticks=[1,2],labels=["Item weight" ,"Item MRP"])
#make the violinplot
plt.violinplot(data.values);
```



SCATTER PLOTS

RELATIVES DISTRIBUTION OF ITEM WEIGHT AND IT'S VISIBILITY it depicts the distribution of two variables using a cloud of point, where each point represents an observation in the dataset.

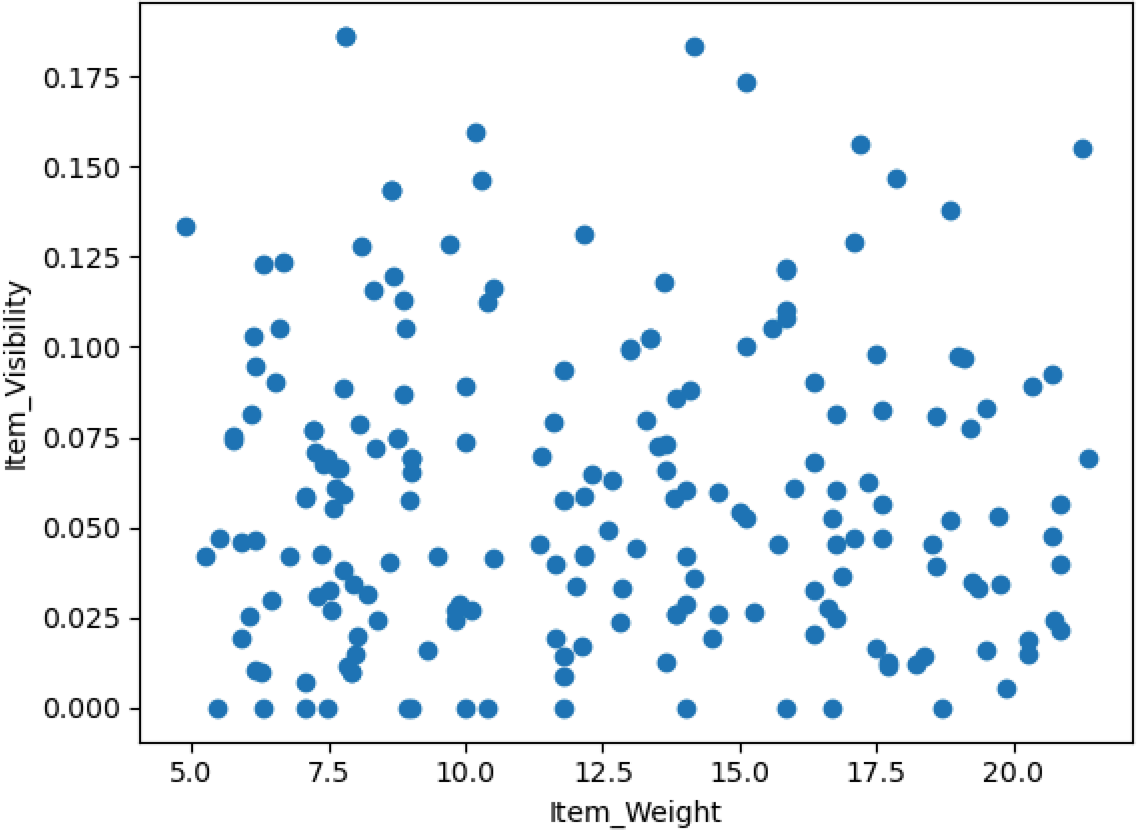
this depiction allows the eye to infer a substantial amount of information about whether there is any meaningful relationships between them.

In [19]:

```
plt.xlabel("Item_Weight")
plt.ylabel("Item_Visibility")
#plot
plt.scatter(data_bm["Item_Weight"][:200],data_bm["Item_Visibility"][:200])
```

Out[19]:

```
<matplotlib.collections.PathCollection at 0x1db5b302dd0>
```



Bubble plots

relatives distribution of sales,item price and item visiblity

lets make a scatter plot of item_outlet_sales and item_MRP and make the size of bubbles by the column item_visiblity.

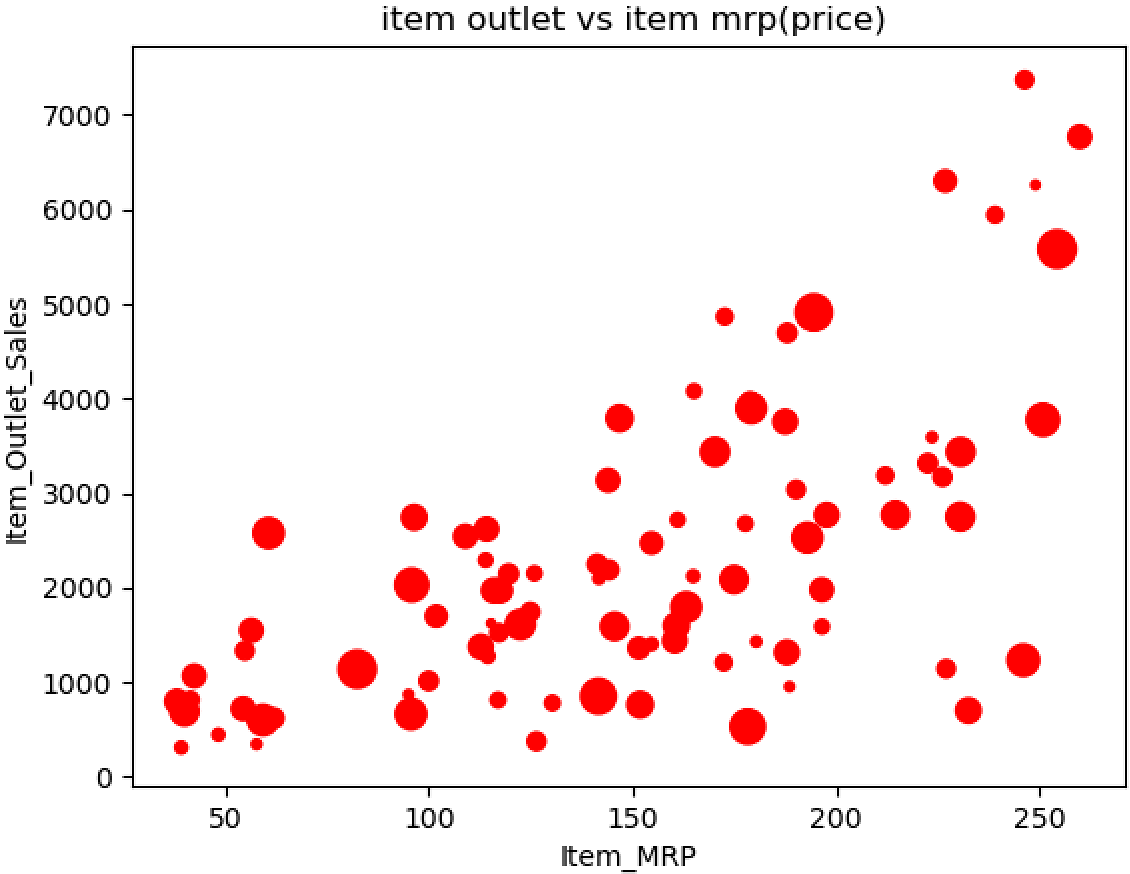
bubble plots let you understand the interdependent relations among 3 variables.

In [20]:

```
#set label of axes
plt.xlabel("Item_MRP")
plt.ylabel("Item_Outlet_Sales")
#set title
plt.title(" item outlet vs item mrp(price)")

#plot
plt.scatter(data_bm["Item_MRP"][:100],data_bm["Item_Outlet_Sales"][:100],s=data_bm["Item_Visi
```

Out[20]:

```
<matplotlib.collections.PathCollection at 0x1db5b27b310>
```


item outlet vs item mrp(price)
Item_Outlet_Sales
0
1000
2000
3000
4000
5000
6000
7000
50
100
150
200
250
Item_MRP

In [ ]:

In [ ]: