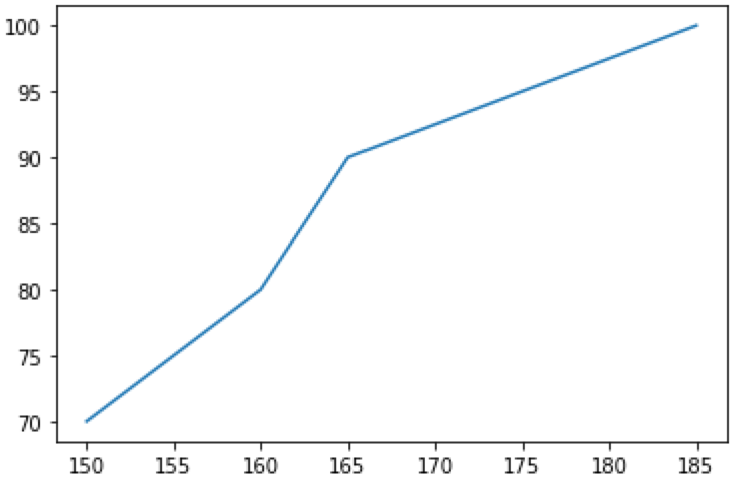
In [1]:
```python
# importing required libraries
import numpy as np
import pandas as pd

# importing matplotlib
import matplotlib.pyplot as plt

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

2. Matplotlib basics

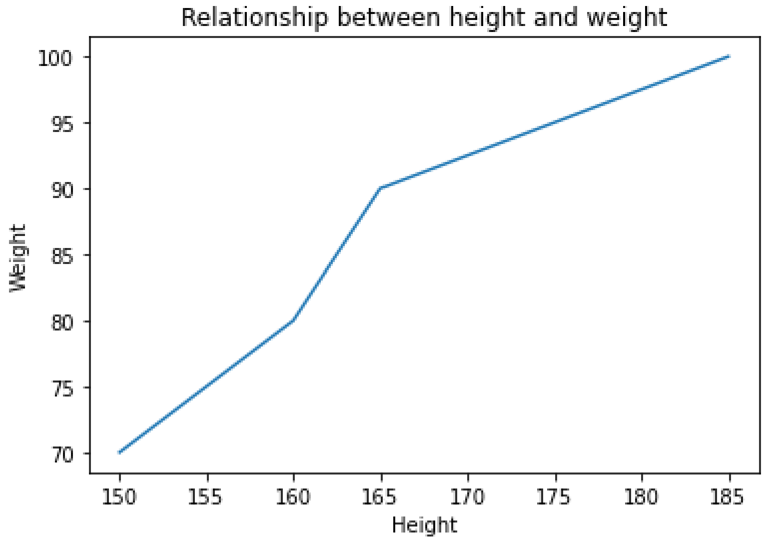
In [2]:
```python
height = [150,160,165,185]
weight = [70, 80, 90, 100]

# draw the plot
plt.plot(height, weight)
```

Out[2]:
```
[<matplotlib.lines.Line2D at 0x286fccdc760>]
```



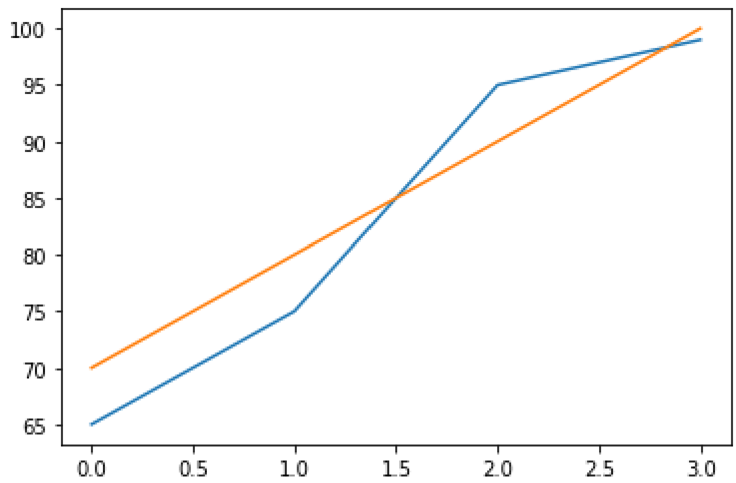
In [3]:
```python
# draw the plot
plt.plot(height,weight)
# add title
plt.title("Relationship between height and weight")
# label x axis
plt.xlabel("Height")
# label y axis
plt.ylabel("Weight")
```

Out[3]:
```
Text(0, 0.5, 'Weight')
```



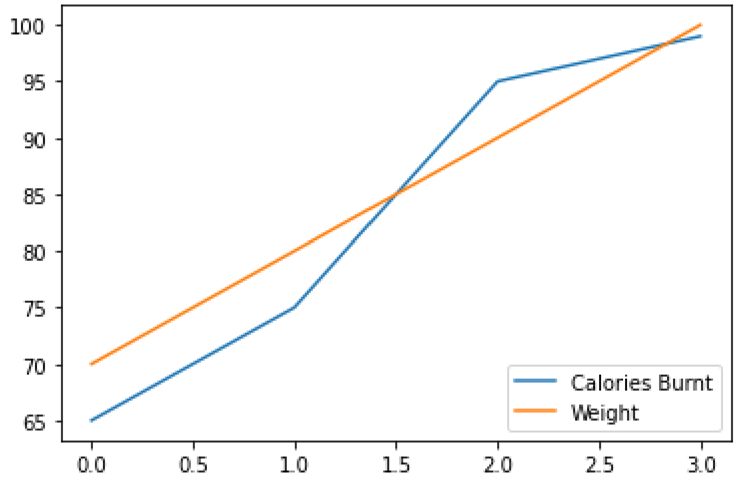
In [4]:
```
calories_burnt = [65, 75, 95, 99]

# draw the plot for calories burnt
plt.plot(calories_burnt)
# draw the plot for weight
plt.plot(weight)
```

Out[4]: [<matplotlib.lines.Line2D at 0x286fee7cd00>]



In [5]:
```
# draw the plot for calories burnt
plt.plot(calories_burnt)
# draw the plot for weight
plt.plot(weight)

# add legend in the lower right part of the figure
plt.legend(labels=['Calories Burnt', 'Weight'], loc='lower right')
```

Out[5]: <matplotlib.legend.Legend at 0x286fef02be0>

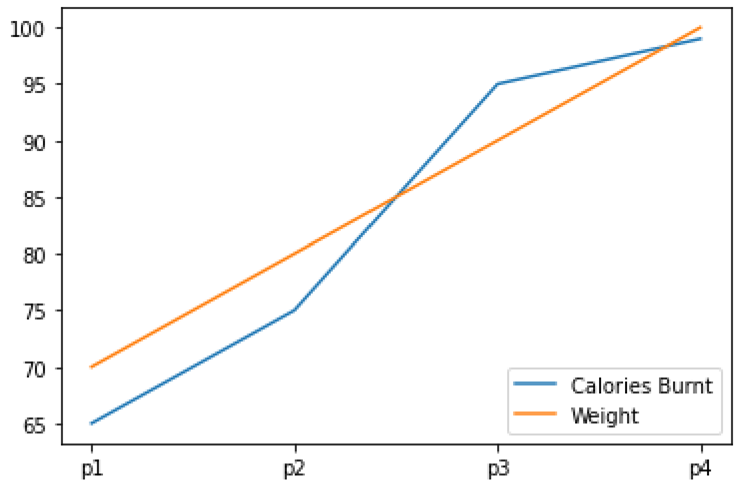


In [6]:

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

# add legend in the lower right part of the figure
plt.legend(labels=['Calories Burnt', 'Weight'], loc='lower right')

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

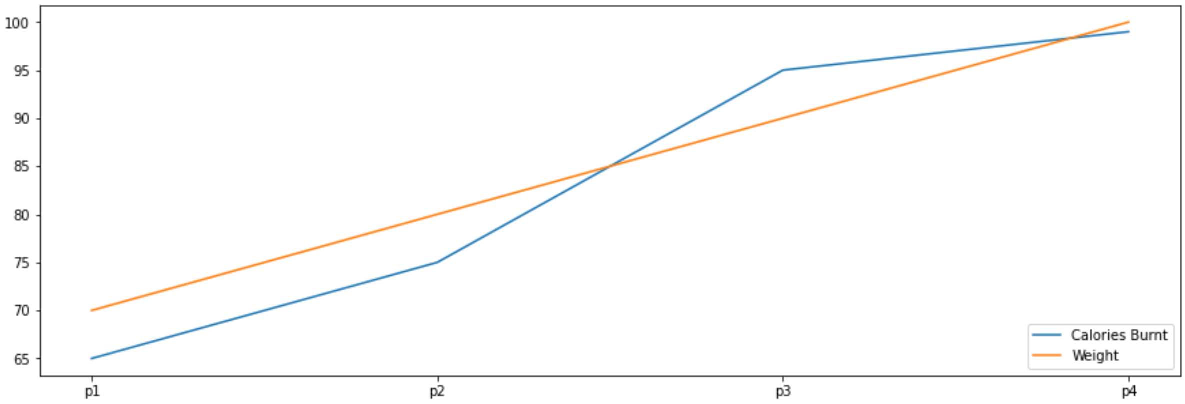
Out[6]:

```
([<matplotlib.axis.XTick at 0x286fff4a910>,
  <matplotlib.axis.XTick at 0x286fff4a8e0>,
  <matplotlib.axis.XTick at 0x286fff4a370>,
  <matplotlib.axis.XTick at 0x286fff83e50>],
 [Text(0, 0, 'p1'), Text(1, 0, 'p2'), Text(2, 0, 'p3'), Text(3, 0, 'p4')])
```

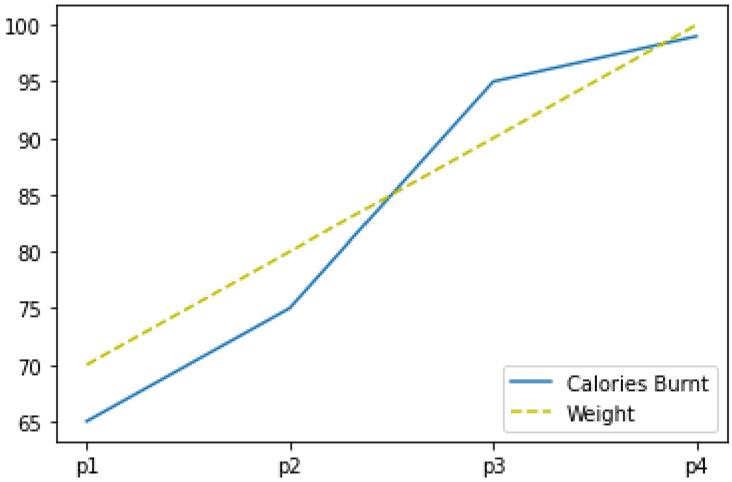


In [7]:

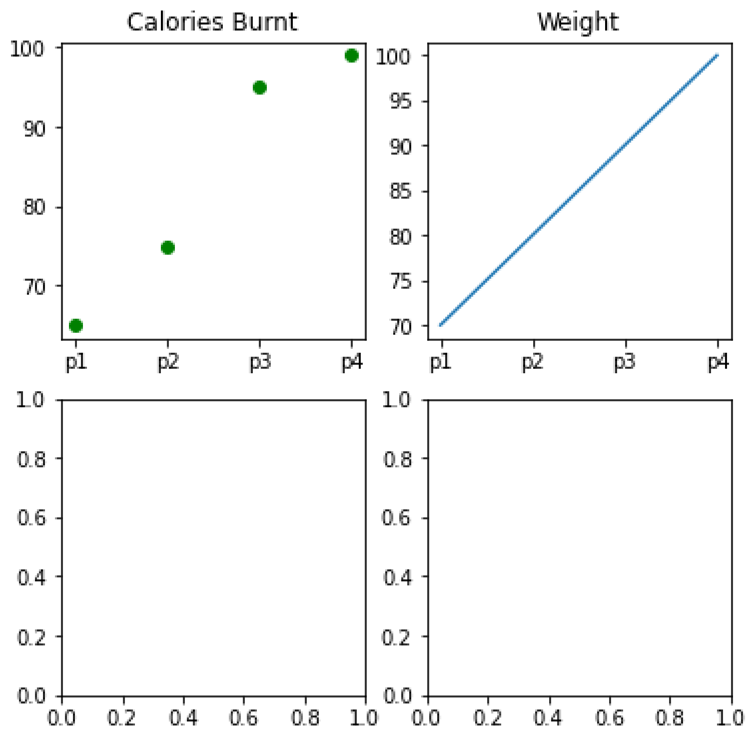
```
# figure size in inches
plt.figure(figsize=(15,5))

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

# add legend in the lower right part of the figure
plt.legend(labels=['Calories Burnt', 'Weight'], loc='lower right')

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



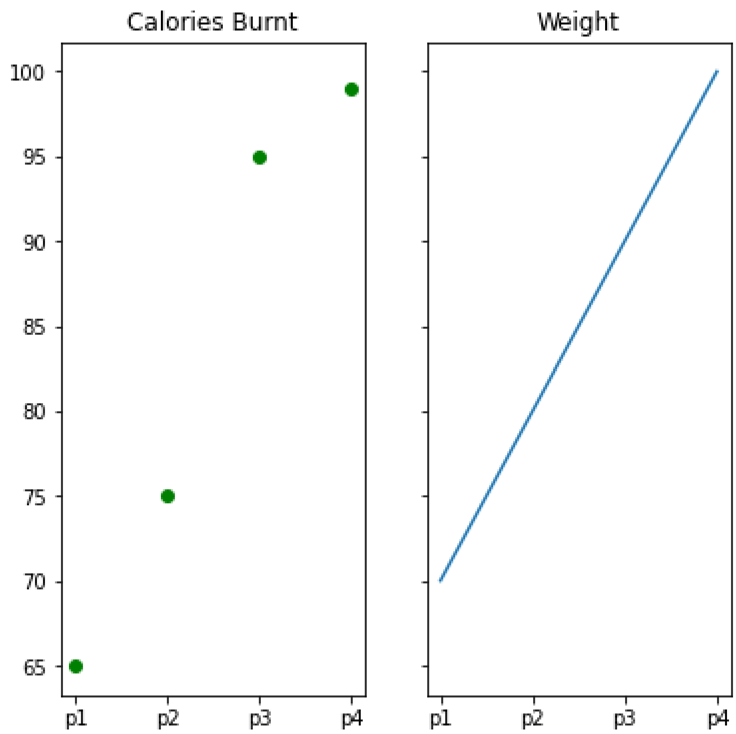
In [8]:
```
# draw the plot
plt.plot(calories_burnt)
plt.plot(weight,  'y--')

# add legend in the lower right part of the figure
plt.legend(labels=['Calories Burnt', 'Weight'], loc='lower right')

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



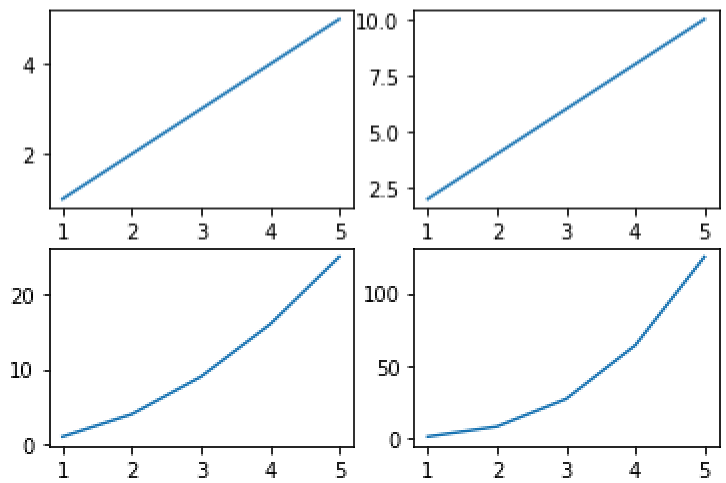
In [9]:
```
# create 2 plots
fig, ax = plt.subplots(nrows=2, ncols=2, figsize=(6,6))

# plot on 0 row and 0 column
ax[0,0].plot(calories_burnt, 'go')

# plot on 0 row and 1 column
ax[0,1].plot(weight)

# set titles for subplots
ax[0,0].set_title("Calories Burnt")
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']);
```



In [10]:
```
# create 2 plots
fig, ax = plt.subplots(nrows=1, ncols=2, figsize=(6,6), sharex=True, sharey=True)

# plot on 0 row and 0 column
ax[0].plot(calories_burnt,'go')

# plot on 0 row and 1 column
ax[1].plot(weight)

# set titles for subplots
ax[0].set_title("Calories Burnt")
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']);
```

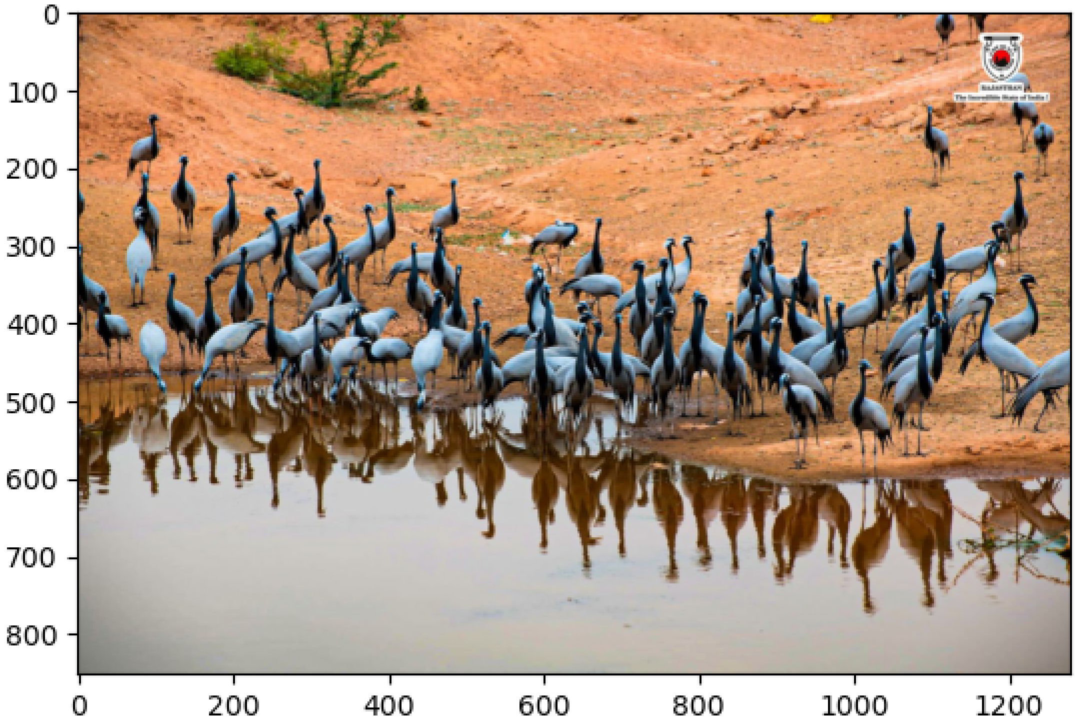


In [32]:
```
x=np.array([1, 2, 3, 4, 5])

# making subplots
fig, ax = plt.subplots(2, 2)

# set data with subplots and plot
ax[0, 0].plot(x, x)
ax[0, 1].plot(x, x*2)
ax[1, 0].plot(x, x*x)
ax[1, 1].plot(x, x*x*x)
plt.show()
```



In [1]:
```
import matplotlib.pyplot as plt
import matplotlib.image as img
# reading the image
testImage = img.imread('Bharatpur.jpg')
# displaying the image
plt.imshow(testImage)
```

Out[1]: <matplotlib.image.AxesImage at 0x21cab864640>



In [2]:
```
# displaying the image as an array
print(testImage)
```

```
[[[  0   4   5]
  [  0   3   5]
  [  0   3   5]
  ...
  [ 21   7   0]
  [ 15  12   7]
  [ 14  16  15]]

 [[  0   6   5]
  [  0   6   5]
  [  0   3   3]
  ...
  [ 23   6   0]
  [ 14   9   5]
  [ 10  12   9]]

 [[  0   2   0]
  [  4   6   3]
  [  6   5   3]
  ...
  [ 31  13   0]
  [ 19  12   4]
  [ 13  12   8]]

 ...

 [[121 123 109]
  [121 123 109]
  [122 124 111]
  ...
  [131 121 111]
  [131 121 111]
  [132 122 112]]

 [[121 123 109]
  [121 123 109]
  [122 124 111]
  ...
  [131 121 111]
  [131 121 111]
  [131 121 111]]

 [[121 123 109]
  [121 123 109]
  [122 124 111]
  ...
  [130 120 110]
  [130 120 110]
  [130 120 110]]]
```

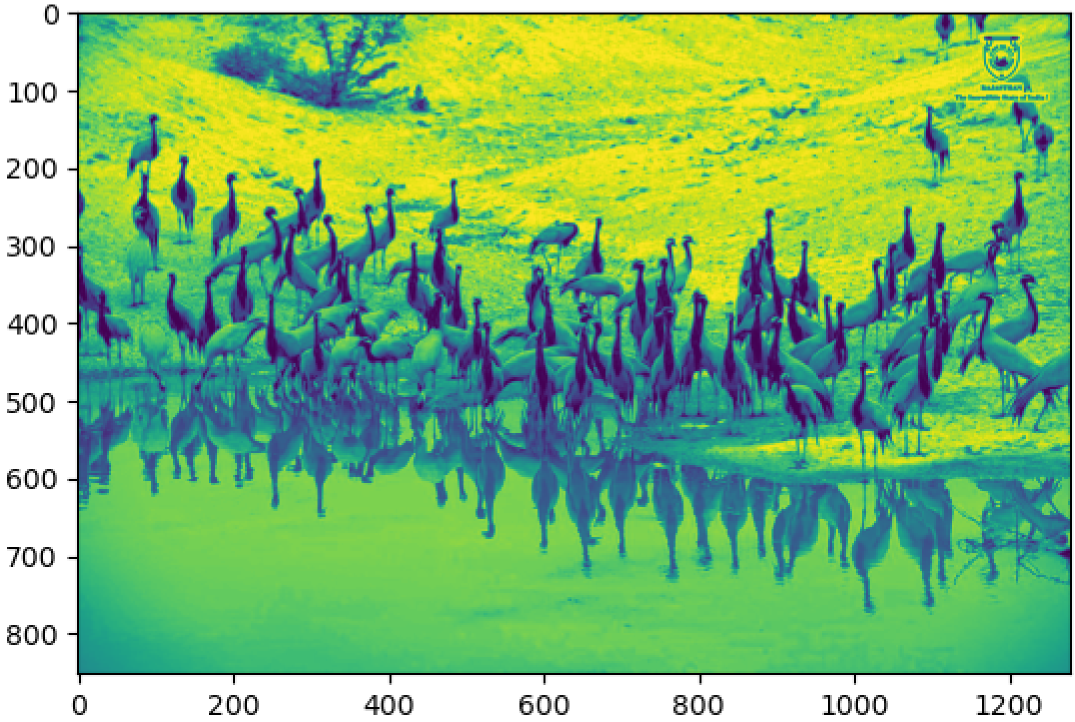
In [3]:

```
# displaying the shape of the image
print(testImage.shape)

# modifying the shape of the image
modifiedImage = testImage[:, :,  0]

# displaying the modified image
plt.imshow(modifiedImage)
```

```
(852, 1280, 3)
```

Out[3]:

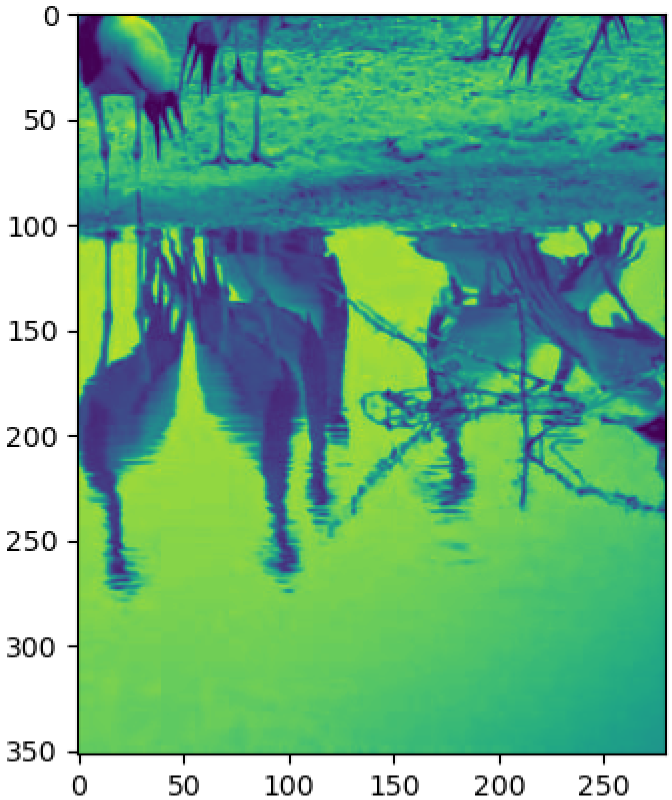
```
<matplotlib.image.AxesImage at 0x21cabcacaf0>
```



In [4]:
```
print(testImage.shape)

# modifying the shape of the image
modifiedImage = testImage[500:2000, 1000:2000, 1]

# displaying the modified image
plt.imshow(modifiedImage)
```

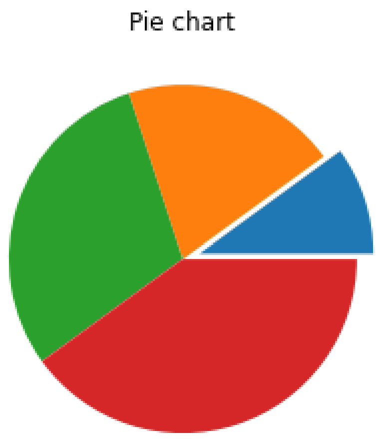
```
(852, 1280, 3)
```

Out[4]:
```
<matplotlib.image.AxesImage at 0x21cac293d90>
```



In [15]:
```
x = [1, 2, 3, 4]
```

```
# this will explode the 1st wedge
# i.e. will separate the 1st wedge
# from the chart
e  =(0.1, 0, 0, 0)

# This will plot a simple pie chart
plt.pie(x, explode = e)

# Title to the plot
plt.title("Pie chart")
plt.show()
```

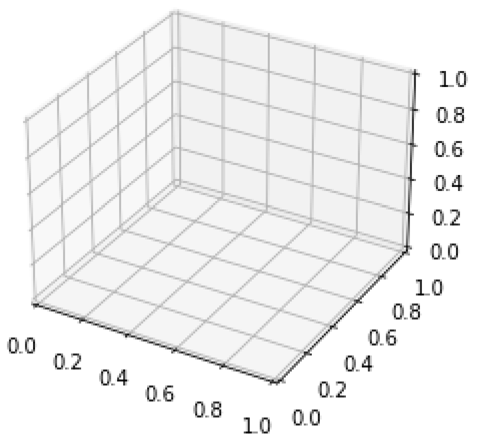


In [ ]:

Working on 3D picture

In [16]:

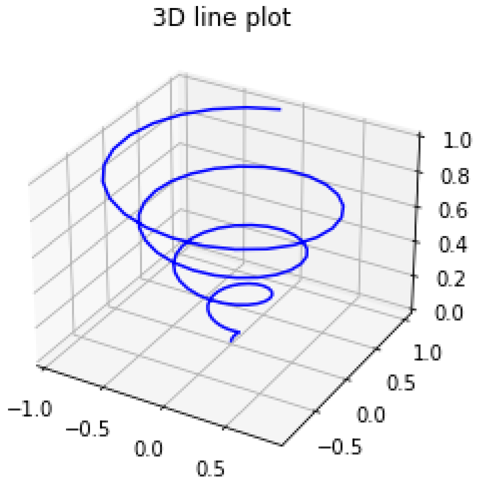
```
fig = plt.figure()
axis = plt.axes(projection='3d')
```



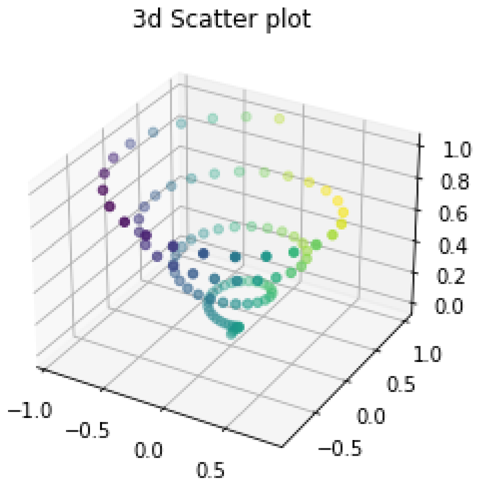
In [17]:

```
# syntax for 3-D projection
ax = plt.axes(projection ='3d')

# defining all 3 axis
z = np.linspace(0, 1, 100)
x = z * np.sin(25 * z)
y = z * np.cos(25 * z)

# plotting
ax.plot3D(x, y, z, 'blue')
```

```
ax.set_title('3D line plot')
plt.show()
```

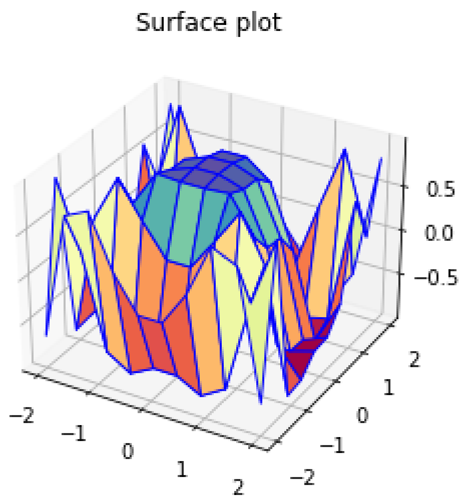


In [18]:
```
# syntax for 3-D projection
ax = plt.axes(projection ='3d')

# defining axes
z = np.linspace(0, 1, 100)
x = z * np.sin(25 * z)
y = z * np.cos(25 * z)
c = x + y
ax.scatter(x, y, z, c = c)

# syntax for plotting
ax.set_title('3d Scatter plot')
plt.show()
```



In [19]:
```
# defining surface and axes
x = np.outer(np.linspace(-2, 2, 10), np.ones(10))
y = x.copy().T
z = np.cos(x ** 2 + y ** 3)

fig = plt.figure()

# syntax for 3-D plotting
ax = plt.axes(projection='3d')

# syntax for plotting
ax.plot_surface(x, y, z, cmap='Spectral',\
                edgecolor='blue')
```

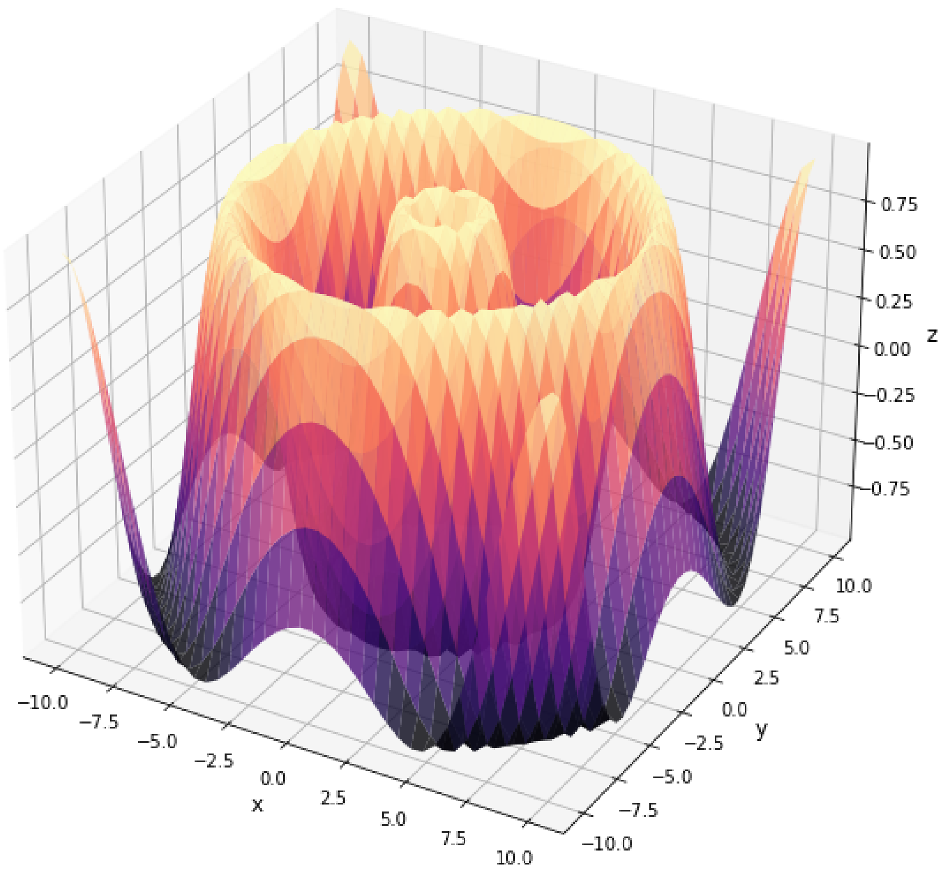
```python
ax.set_title('Surface plot')
plt.show()
```



In [20]:
```python
def function(x, y):
    return np.sin(np.sqrt(x ** 2 + y ** 2))
```

In [21]:
```python
x = np.linspace(-10, 10, 40)
y = np.linspace(-10, 10, 40)

X, Y = np.meshgrid(x, y)
Z = function(X, Y)

fig = plt.figure(figsize=(15, 10))
ax = plt.axes(projection='3d')

ax.plot_surface(X, Y, Z, cmap='magma', alpha=0.8)

ax.set_title('3D Contour Plot of function(x, y) =\
                sin(sqrt(x^2 + y^2))', fontsize=20)
ax.set_xlabel('x', fontsize=12)
ax.set_ylabel('y', fontsize=12)
ax.set_zlabel('z', fontsize=12)

plt.show()
```

3D Contour Plot of function(x, y) = sin(sqrt(x^2 + y^2))



Load dataset

In [22]:
```
# 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[22]:

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

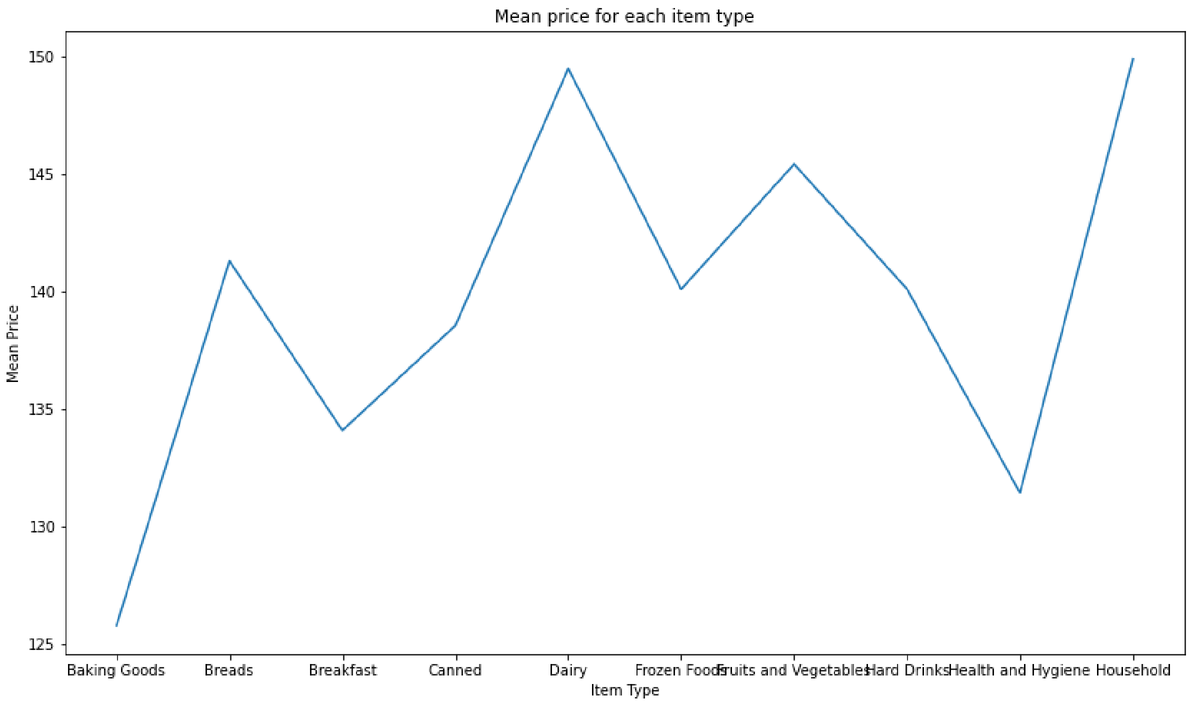
3. Line Chart.

In [23]:
```
price_by_item = data_BM.groupby('Item_Type').Item_MRP.mean()[:10]
price_by_item
```

Out[23]:
```
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 [24]:
```python
# 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))

# set 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[24]:
```
[<matplotlib.lines.Line2D at 0x28681366550>]
```

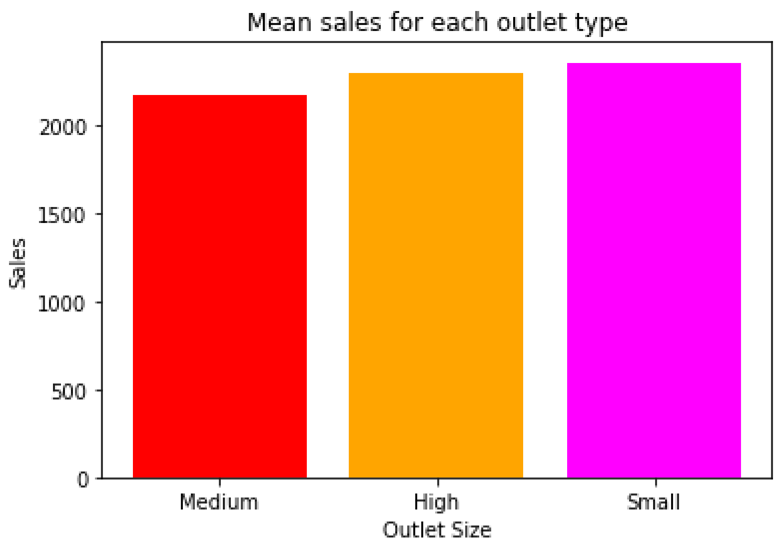


4. Bar Chart

In [25]:
```python
# sales by outlet size
sales_by_outlet_size = data_BM.groupby('Outlet_Size').Item_Outlet_Sales.mean()

# sort by sales
```

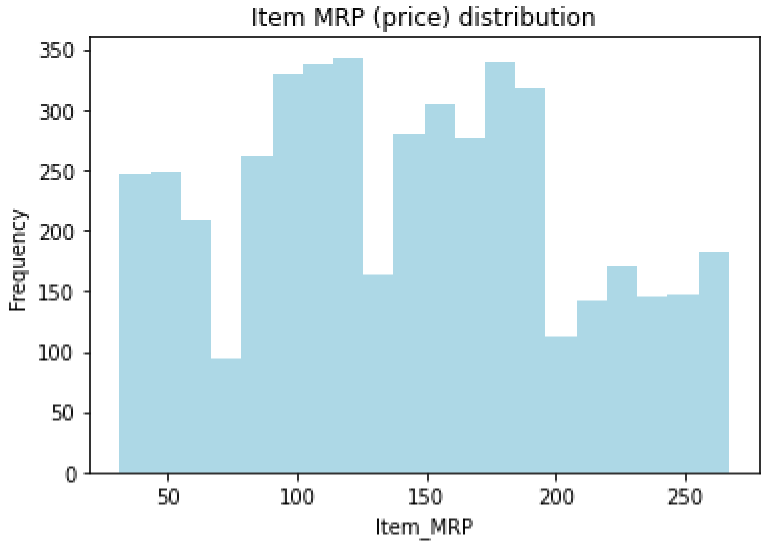
```
sales_by_outlet_size.sort_values(inplace=True)

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

# set axis labels
plt.xlabel('Outlet Size')
plt.ylabel('Sales')

# set title
plt.title('Mean sales for each outlet type')

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

plt.bar(x, y, color=['red', 'orange', 'magenta'])
```

Out[25]: <BarContainer object of 3 artists>



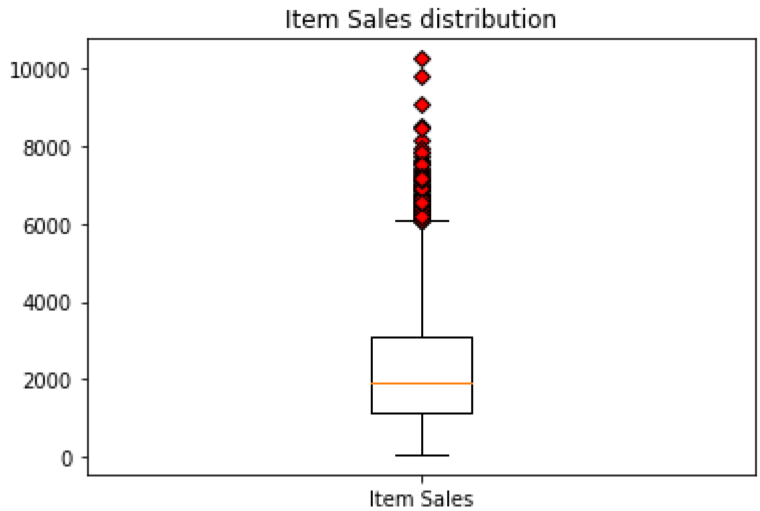
5. Histogram

In [26]:
```
# title
plt.title('Item MRP (price) distribution')

# xlabel
plt.xlabel('Item_MRP')

# ylabel
plt.ylabel('Frequency')

# plot histogram
plt.hist(data_BM['Item_MRP'], bins=20, color='lightblue');
```

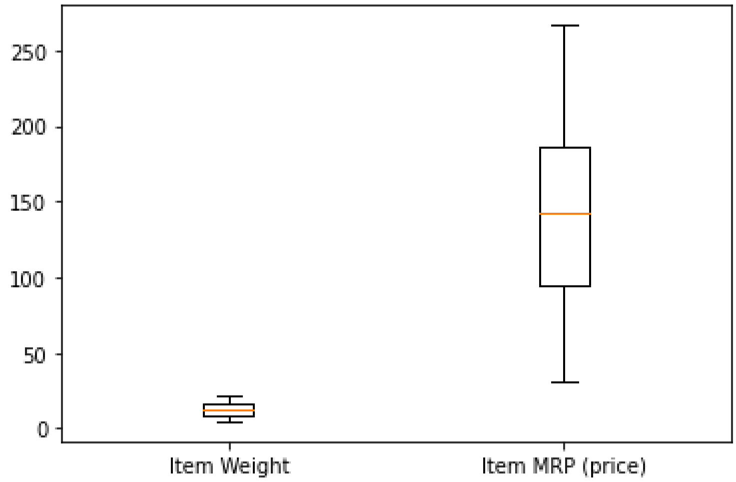


6. Box Plots Distribution of sales

In [27]:
```
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 boxplot
plt.boxplot(data.values, labels=['Item Sales'], flierprops=red_diamond);
```

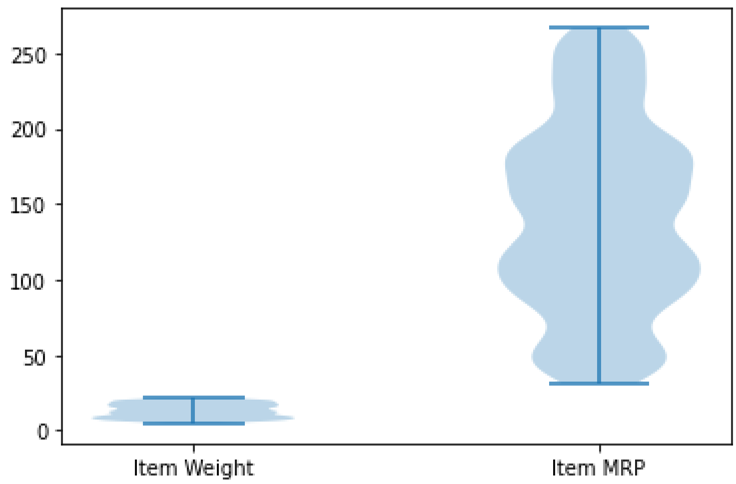


In [28]:
```
data = data_BM[['Item_Weight', 'Item_MRP']]

# 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 (price)'], flierprops=rec
```



7. Violin Plots

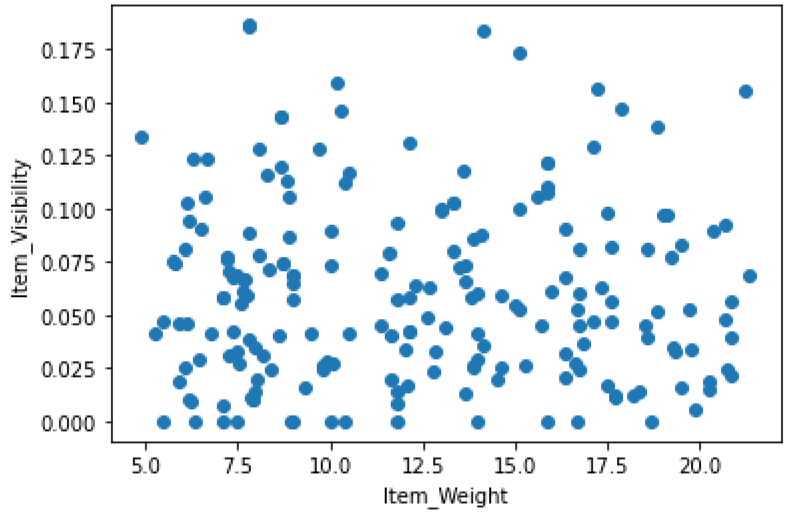
In [29]:
```
data = data_BM[['Item_Weight', 'Item_MRP']]

# generate subplots
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);
```



8. Scatter Plots

In [30]:
```
# set label of axes
plt.xlabel('Item_Weight')
plt.ylabel('Item_Visibility')

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

Out[30]:
```
<matplotlib.collections.PathCollection at 0x2868156f6a0>
```



9. Bubble Plots

In [31]:
```
# set label of axes
plt.xlabel('Item_MRP')
plt.ylabel('Item_Outlet_Sales')

# set title
plt.title('Item Outlet Sales vs Item MRP (price)')

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

Out[31]: <matplotlib.collections.PathCollection at 0x28681401df0>