In [1]:
```
import numpy as np
import pandas as pd
```

In [2]:
```
dataset=pd.read_csv(r"D:\ML_Course\Works_on_python\Multi_Linear_Regression\Fish.csv")
```

In [3]:
```
dataset.head()
```

Out[3]:

	Species	Weight	Length1	Length2	Length3	Height	Width
0	Bream	242.0	23.2	25.4	30.0	11.5200	4.0200
1	Bream	290.0	24.0	26.3	31.2	12.4800	4.3056
2	Bream	340.0	23.9	26.5	31.1	12.3778	4.6961
3	Bream	363.0	26.3	29.0	33.5	12.7300	4.4555
4	Bream	430.0	26.5	29.0	34.0	12.4440	5.1340

In [4]:
```
dataset.isnull().any()
```

Out[4]:
```
Species    False
Weight     False
Length1    False
Length2    False
Length3    False
Height     False
Width      False
dtype: bool
```

In [5]:
```
dataset["Species"].unique()
```

Out[5]:
```
array(['Bream', 'Roach', 'Whitefish', 'Parkki', 'Perch', 'Pike', 'Smelt'],
      dtype=object)
```

In [6]:
```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
dataset["Species"]=le.fit_transform(dataset["Species"])
```

In [7]:
```
dataset.head()
```

Out[7]:

	Species	Weight	Length1	Length2	Length3	Height	Width
0	0	242.0	23.2	25.4	30.0	11.5200	4.0200
1	0	290.0	24.0	26.3	31.2	12.4800	4.3056
2	0	340.0	23.9	26.5	31.1	12.3778	4.6961
3	0	363.0	26.3	29.0	33.5	12.7300	4.4555
4	0	430.0	26.5	29.0	34.0	12.4440	5.1340

In [8]:
```
x=dataset.iloc[:,[0,2,3,4,5,6]].values
y=dataset.iloc[:,1:2].values
```

In [9]:
```
x.shape
```

Out[9]:
```
(159, 6)
```

In [10]:
```
y.shape
```

Out[10]:
```
(159, 1)
```

In [11]:
```
from sklearn.preprocessing import OneHotEncoder
one=OneHotEncoder()
z=one.fit_transform(x[:,0:1]).toarray()
x=np.delete(x,0,axis=1)
x=np.concatenate((z,x),axis=1)
```

```
C:\Users\anikp\Anaconda3\lib\site-packages\sklearn\preprocessing\_encoders.py:368: FutureWarning: The handling of integer data
will change in version 0.22. Currently, the categories are determined based on the range [0, max(values)], while in the future
they will be determined based on the unique values.
If you want the future behaviour and silence this warning, you can specify "categories='auto'".
In case you used a LabelEncoder before this OneHotEncoder to convert the categories to integers, then you can now use the OneH
otEncoder directly.
  warnings.warn(msg, FutureWarning)
```

In [12]:
```
x.shape
```
Out[12]:
```
(159, 12)
```

In [13]:
```
y.shape
```
Out[13]:
```
(159, 1)
```

In [14]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=0)
```

In [15]:
```
y_test.shape
```
Out[15]:
```
(32, 1)
```

In [16]:
```
from sklearn.linear_model import LinearRegression
mlr=LinearRegression()
mlr.fit(x_train,y_train)
```
Out[16]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
         normalize=False)
```

In [17]:
```
y_predict=mlr.predict(x_test)
```

```
In [18]: x_test
```

```
Out[18]: array([[ 1.    ,  0.    ,  0.    ,  0.    ,  0.    ,  0.    ,  0.    ,
                 27.6   , 30.    , 35.    , 12.67  ,  4.69  ],
               [ 0.    ,  0.    ,  0.    ,  0.    ,  1.    ,  0.    ,  0.    ,
                 19.    , 20.5   , 22.8   ,  6.4752,  3.3516],
               [ 0.    ,  0.    ,  1.    ,  0.    ,  0.    ,  0.    ,  0.    ,
                 21.5   , 23.5   , 25.    ,  6.275 ,  3.725 ],
               [ 0.    ,  0.    ,  0.    ,  0.    ,  1.    ,  0.    ,  0.    ,
                 20.5   , 22.5   , 25.3   ,  7.0334,  3.8203],
               [ 0.    ,  0.    ,  1.    ,  0.    ,  0.    ,  0.    ,  0.    ,
                 32.    , 34.5   , 36.5   , 10.2565,  6.3875],
               [ 0.    ,  0.    ,  1.    ,  0.    ,  0.    ,  0.    ,  0.    ,
                 37.    , 40.    , 42.5   , 11.73  ,  7.225 ],
               [ 0.    ,  0.    ,  0.    ,  0.    ,  0.    ,  0.    ,  1.    ,
                 33.7   , 36.4   , 39.6   , 11.7612,  6.5736],
               [ 0.    ,  0.    ,  0.    ,  1.    ,  0.    ,  0.    ,  0.    ,
                 34.8   , 37.3   , 39.8   ,  6.2884,  4.0198],
               [ 1.    ,  0.    ,  0.    ,  0.    ,  0.    ,  0.    ,  0.    ,
                 37.4   , 41.    , 45.9   , 18.6354,  6.7473],
               [ 0.    ,  0.    ,  1.    ,  0.    ,  0.    ,  0.    ,  0.    ,
                 19.    , 21.    , 22.5   ,  5.9175,  3.3075],
               [ 0.    ,  0.    ,  0.    ,  1.    ,  0.    ,  0.    ,  0.    ,
                 30.    , 32.3   , 34.8   ,  5.568 ,  3.3756],
               [ 0.    ,  0.    ,  0.    ,  1.    ,  0.    ,  0.    ,  0.    ,
                 40.    , 42.5   , 45.5   ,  7.28  ,  4.3225],
               [ 0.    ,  0.    ,  0.    ,  0.    ,  0.    ,  0.    ,  1.    ,
                 37.3   , 40.    , 43.5   , 12.354 ,  6.525 ],
               [ 0.    ,  0.    ,  1.    ,  0.    ,  0.    ,  0.    ,  0.    ,
                 40.2   , 43.5   , 46.    , 12.604 ,  8.142 ],
               [ 0.    ,  1.    ,  0.    ,  0.    ,  0.    ,  0.    ,  0.    ,
                 14.3   , 15.5   , 17.4   ,  6.5772,  2.3142],
               [ 0.    ,  0.    ,  0.    ,  0.    ,  1.    ,  0.    ,  0.    ,
                 17.5   , 18.8   , 21.2   ,  5.5756,  2.9044],
               [ 0.    ,  0.    ,  0.    ,  0.    ,  1.    ,  0.    ,  0.    ,
                 20.5   , 22.    , 24.3   ,  6.6339,  3.5478],
               [ 0.    ,  0.    ,  0.    ,  1.    ,  0.    ,  0.    ,  0.    ,
                 56.    , 60.    , 64.    ,  9.6   ,  6.144 ],
               [ 0.    ,  0.    ,  1.    ,  0.    ,  0.    ,  0.    ,  0.    ,
                 19.3   , 21.3   , 22.8   ,  6.384 ,  3.534 ],
               [ 1.    ,  0.    ,  0.    ,  0.    ,  0.    ,  0.    ,  0.    ,
                 32.    , 35.    , 40.6   , 16.3618,  6.09  ],
               [ 0.    ,  1.    ,  0.    ,  0.    ,  0.    ,  0.    ,  0.    ,
        13.5   , 14.7   , 16.5   ,  6.8475,  2.3265],
       [ 0.    ,  0.    ,  0.    ,  0.    ,  1.    ,  0.    ,  0.    ,
        29.5   , 31.7   , 35.    ,  9.485 ,  5.355 ],
       [ 0.    ,  0.    ,  1.    ,  0.    ,  0.    ,  0.    ,  0.    ,
        20.    , 22.    , 23.5   ,  6.11  ,  3.4075],
       [ 0.    ,  0.    ,  0.    ,  1.    ,  0.    ,  0.    ,  0.    ,
        59.    , 63.4   , 68.    , 10.812 ,  7.48  ],
       [ 0.    ,  1.    ,  0.    ,  0.    ,  0.    ,  0.    ,  0.    ,
        16.3   , 17.7   , 19.8   ,  7.4052,  2.673 ],
       [ 1.    ,  0.    ,  0.    ,  0.    ,  0.    ,  0.    ,  0.    ,
        27.6   , 30.    , 35.1   , 14.0049,  4.8438],
       [ 1.    ,  0.    ,  0.    ,  0.    ,  0.    ,  0.    ,  0.    ,
        31.9   , 35.    , 40.5   , 16.2405,  5.589 ],
       [ 0.    ,  0.    ,  0.    ,  0.    ,  0.    ,  0.    ,  1.    ,
        24.1   , 26.5   , 29.3   ,  8.1454,  4.2485],
       [ 0.    ,  0.    ,  1.    ,  0.    ,  0.    ,  0.    ,  0.    ,
        36.9   , 40.    , 42.3   , 11.9286,  7.1064],
       [ 0.    ,  0.    ,  0.    ,  0.    ,  0.    ,  1.    ,  0.    ,
        10.4   , 11.    , 12.    ,  2.196 ,  1.38  ],
       [ 1.    ,  0.    ,  0.    ,  0.    ,  0.    ,  0.    ,  0.    ,
        31.    , 33.5   , 38.7   , 14.4738,  5.7276],
       [ 0.    ,  0.    ,  1.    ,  0.    ,  0.    ,  0.    ,  0.    ,
        20.    , 22.    , 23.5   ,  5.5225,  3.995 ]])
```

In [19]:
```
y_test
```

Out[19]:
```
array([[ 390. ],
       [   0. ],
       [ 170. ],
       [ 160. ],
       [ 556. ],
       [ 900. ],
       [ 800. ],
       [ 300. ],
       [ 975. ],
       [ 115. ],
       [ 200. ],
       [ 456. ],
       [1000. ],
       [1000. ],
       [  60. ],
       [  78. ],
       [ 145. ],
       [1600. ],
       [ 130. ],
       [ 720. ],
       [  55. ],
       [ 390. ],
       [ 120. ],
       [1650. ],
       [  90. ],
       [ 450. ],
       [ 700. ],
       [ 270. ],
       [ 850. ],
       [   9.7],
       [ 650. ],
       [ 110. ]])
```

In [20]:
```
y_predict
```

Out[20]:
```
array([[ 428.88533577],
       [  98.08363614],
       [ 216.67998922],
       [ 208.66936638],
       [ 657.24094116],
       [ 876.38855413],
       [ 665.97861965],
       [ 407.27203048],
       [ 965.65306863],
       [ 146.62291102],
       [ 255.15532231],
       [ 561.63685124],
       [ 765.67575361],
       [1012.38234027],
       [-118.72798063],
       [  14.47341216],
       [ 137.60789564],
       [1155.53572308],
       [ 170.97092949],
       [ 724.93548455],
       [-128.48675188],
       [ 525.45508599],
       [ 175.3519065 ],
       [1322.74816983],
       [ -23.0873263 ],
       [ 475.44172778],
       [ 719.96841977],
       [ 280.94571114],
       [ 885.12085107],
       [ -16.63412226],
       [ 585.26038657],
       [ 164.22863371]])
```

In [21]:
```
from sklearn.metrics import r2_score
accuracy=r2_score(y_test,y_predict)
```

In [22]:
```
accuracy
```

Out[22]:
```
0.9102350316202583
```

In [23]:
```
y=mlr.predict([[1,0,0,0,0,0,0,28,28,30,15,6]])
```

In [24]:
```
y
```

Out[24]:
```
array([[257.56434014]])
```

In [ ]: