Assignment -3

Assignment Date	29 September 2022
Student Name	Swati S
Student Roll Number	211419104283
Maximum Marks	2 Marks

1. Download the dataset: [Dataset]

2. Load the dataset into the tool.

```
## Importing the libraries

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
import warnings
warnings.filterwarnings('ignore')
```

2. Load the dataset.

```
data= pd.read_csv('F:IBM-project/abalone.csv')
data.head()
```

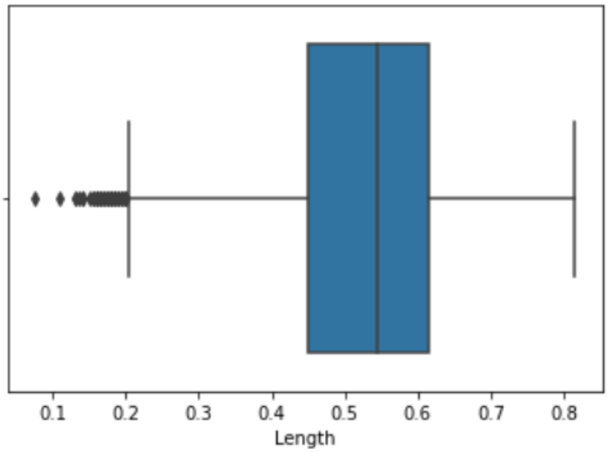
	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
0	M	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.150	15
1	M	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.070	7
2	F	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.210	9
3	M	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.155	10
4	I	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.055	7

3. Perform Below Visualizations.

· Univariate Analysis

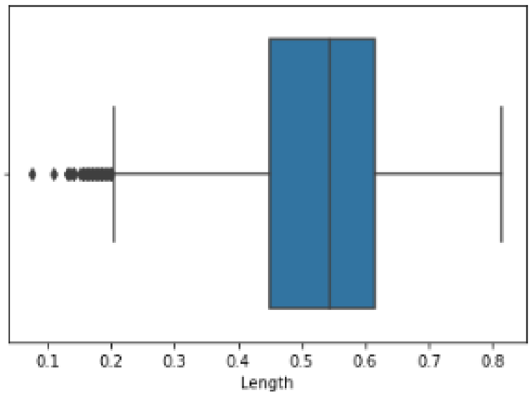
```python
# Boxplot
sns.boxplot(data['Length'])
```

```
<AxesSubplot:xlabel='Length'>
```



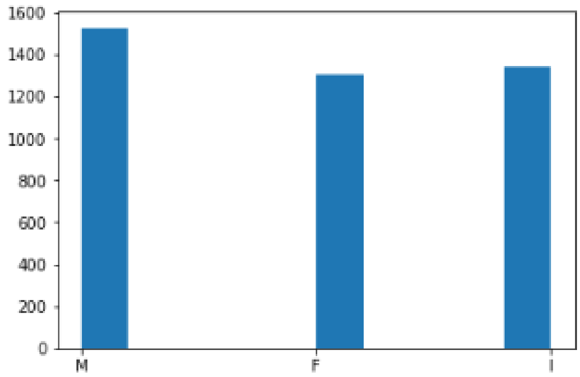

```python
# Boxplot
sns.boxplot(data['Length'])
```

```
<AxesSubplot:xlabel='Length'>
```



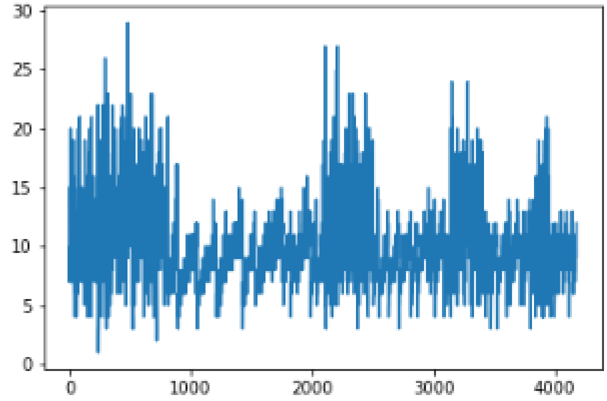

```python
plt.hist(data['Sex'])
```

```
(array([1528.,    0.,    0.,    0.,    0., 1307.,    0.,    0.,    0.,
        1342.]),
 array([0. , 0.2, 0.4, 0.6, 0.8, 1. , 1.2, 1.4, 1.6, 1.8, 2. ]),
 <BarContainer object of 10 artists>)
```



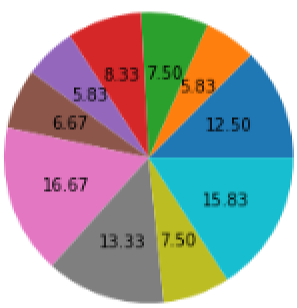
```
plt.plot(data['Rings'])
```

```
[<matplotlib.lines.Line2D at 0x1fca36f8e80>]
```



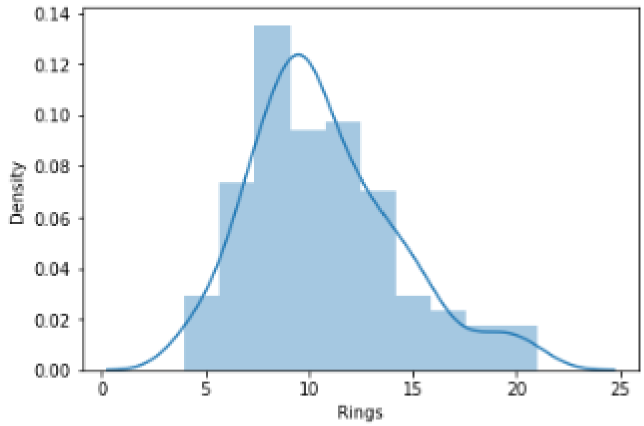

```
: plt.pie(data['Rings'].head(10), autopct="%.2f")
```

```
: ([<matplotlib.patches.Wedge at 0x1fca38708e0>,
   <matplotlib.patches.Wedge at 0x1fca38810a0>,
   <matplotlib.patches.Wedge at 0x1fca38817c0>,
   <matplotlib.patches.Wedge at 0x1fca3881ee0>,
   <matplotlib.patches.Wedge at 0x1fca388f5e0>,
   <matplotlib.patches.Wedge at 0x1fca388fd00>,
   <matplotlib.patches.Wedge at 0x1fca389b460>,
   <matplotlib.patches.Wedge at 0x1fca389bb80>,
   <matplotlib.patches.Wedge at 0x1fca38ab2e0>,
   <matplotlib.patches.Wedge at 0x1fca38aba00>],
  [Text(1.0162674857624154, 0.4209517756015988, ''),
   Text(0.6230468599100139, 0.9065388079703327, ''),
   Text(0.20045906622712806, 1.081580400509989, ''),
   Text(-0.3399187231970732, 1.046162158377023, ''),
   Text(-0.7571900625229377, 0.7979117803469942, ''),
   Text(-1.0049000250498075, 0.4474102587725236, ''),
   Text(-1.0461621424642782, -0.3399187721714579, ''),
   Text(-0.3399185762739153, -1.046162206115244, ''),
   Text(0.3671876940163829, -1.0369055874875646, ''),
   Text(0.9666989009177708, -0.5248744944883245, '')],
  [Text(0.5543277195067721, 0.22961005941905385, '12.50'),
   Text(0.33984374176909843, 0.4944757134383632, '5.83'),
   Text(0.10934130885116075, 0.5899529457327212, '7.50'),
   Text(-0.18541021265294902, 0.5706339045692853, '8.33'),
   Text(-0.4130127613761478, 0.43522460746199676, '5.83'),
   Text(-0.548127286390804, 0.2440419593304674, '6.67'),
   Text(-0.5706338958896062, -0.18541023936624976, '16.67'),
   Text(-0.1854101325130447, -0.5706339306083149, '13.33'),
   Text(0.20028419673620884, -0.5655848659023078, '7.50'),
   Text(0.5272903095915112, -0.2862951788118133, '15.83')])
```



```
sns.distplot(data['Rings'].head(200))
```

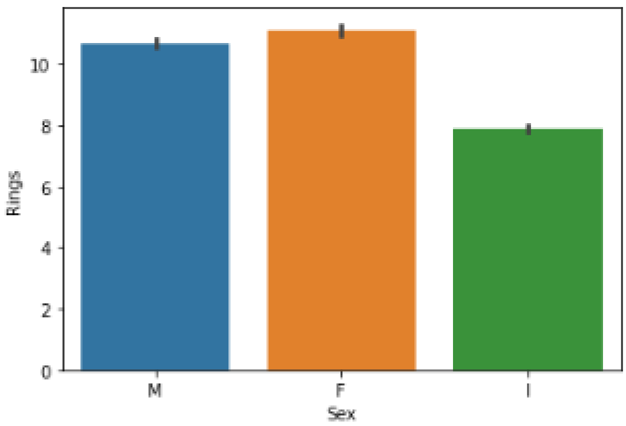
<AxesSubplot:xlabel='Rings', ylabel='Density'>



Bi-Variate Analysis

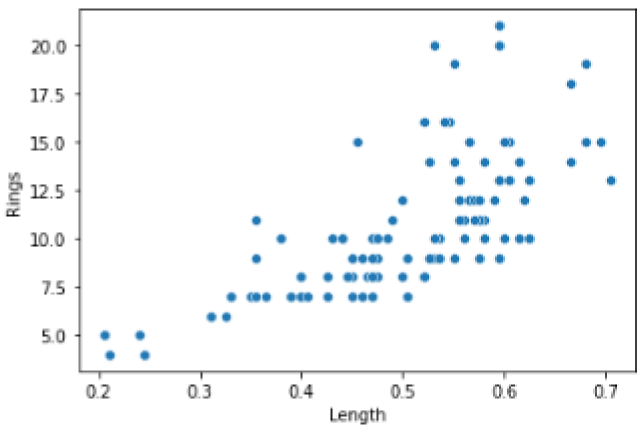
```
sns.barplot(data['Sex'], data['Rings'])
```

<AxesSubplot:xlabel='Sex', ylabel='Rings'>



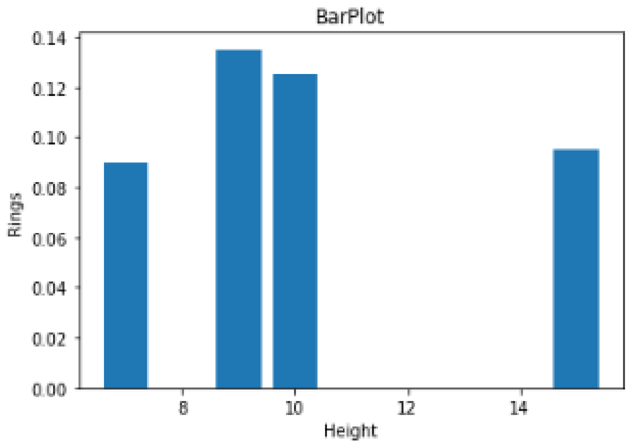

```
sns.scatterplot(data['Length'].head(100),data['Rings'].head(100))
```

<AxesSubplot:xlabel='Length', ylabel='Rings'>



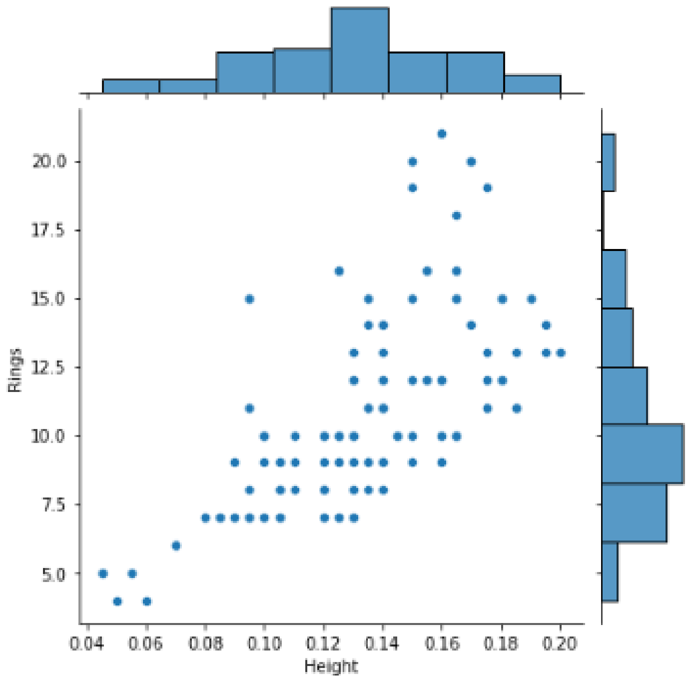
```
plt.bar(data['Rings'].head() ,data['Height'].head(),  )

plt.title('BarPlot')
plt.xlabel('Height')
plt.ylabel('Rings')
```

```
Text(0, 0.5, 'Rings')
```



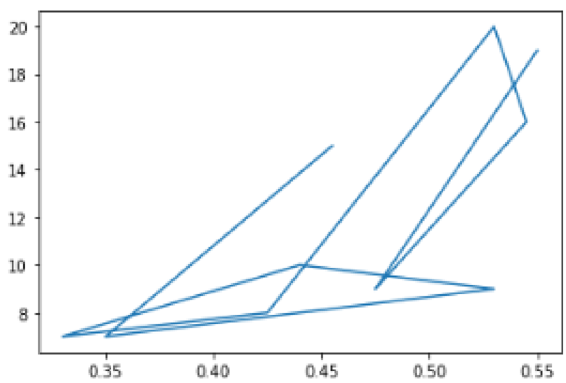

```
sns.jointplot(data['Height'].head(100) ,data['Rings'].head(100),  )
```

```
<seaborn.axisgrid.JointGrid at 0x1fca3abd130>
```




```
plt.plot(data['Length'].head(10), data['Rings'].head(10))
```

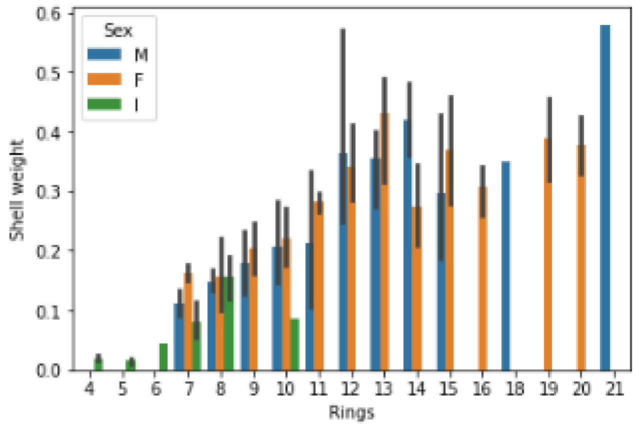
```
[<matplotlib.lines.Line2D at 0x1fca4c1bb50>]
```



· Multi-Variate Analysis

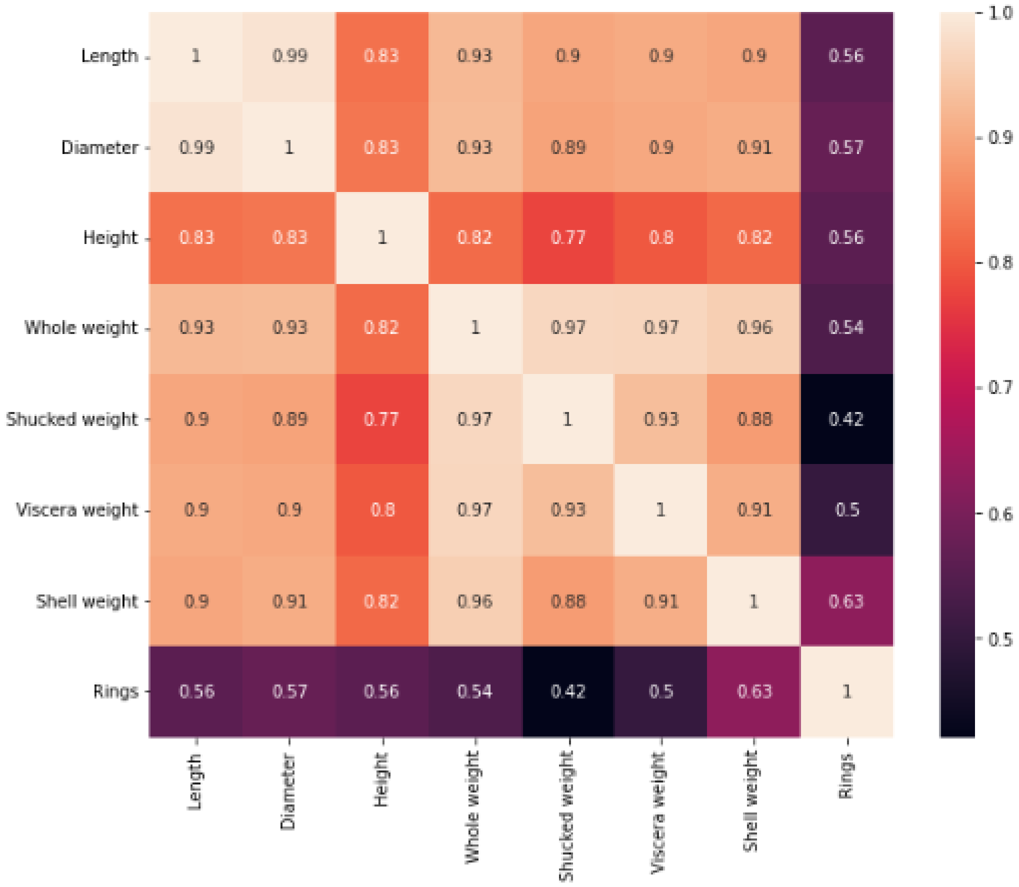
```python
sns.barplot('Rings','Shell weight',hue='Sex', data=data.head(100))
```

<AxesSubplot:xlabel='Rings', ylabel='Shell weight'>




```python
fig= plt.figure(figsize  =(10,8))
sns.heatmap(data.corr(), annot = True)
```

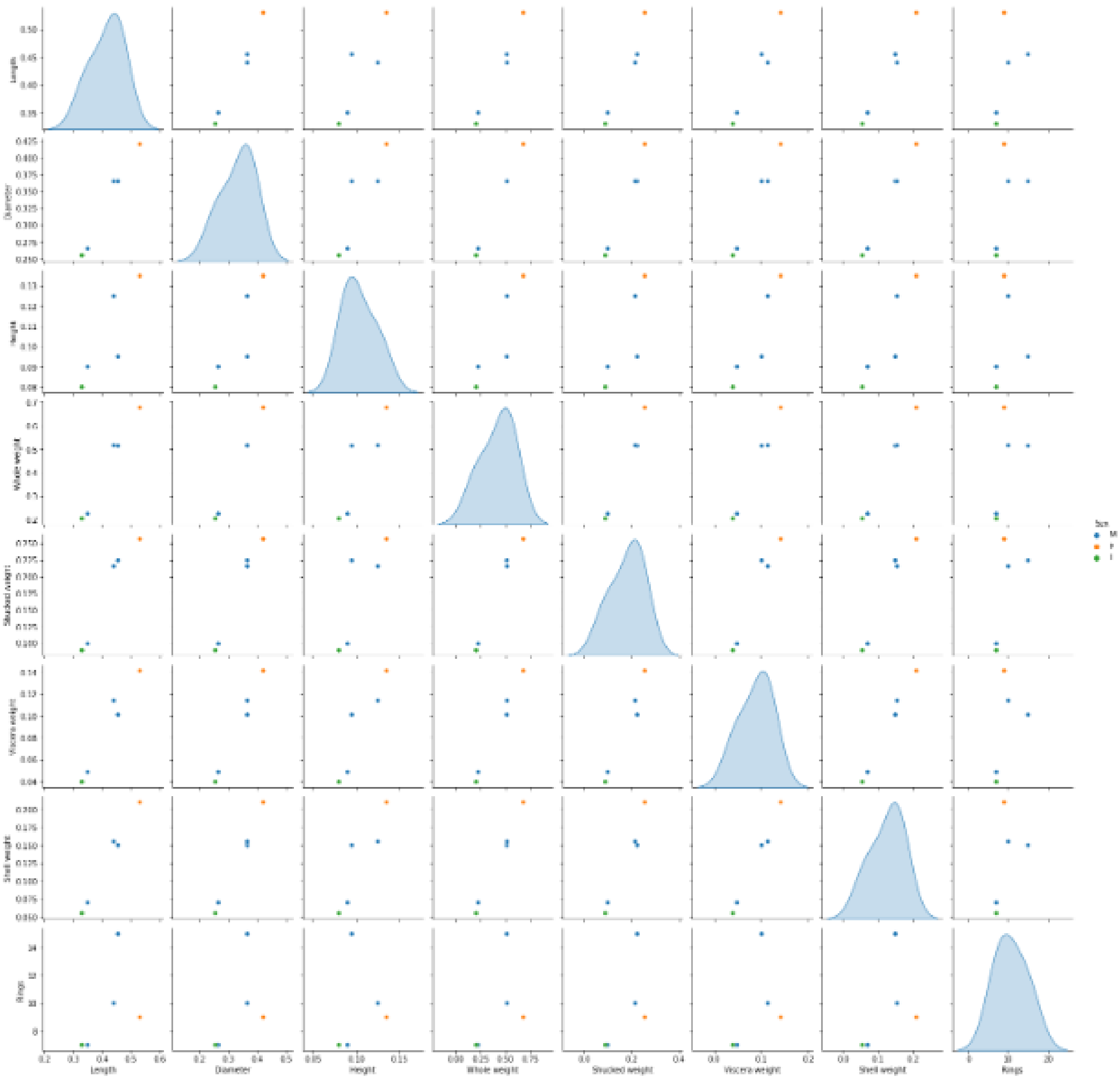
<AxesSubplot:>



```
fig= plt.figure(figsize  =(7,5))
sns.pairplot(data.head(),hue='Sex')
```

<seaborn.axisgrid.PairGrid at 0x1fca4c4a460>

<Figure size 504x360 with 0 Axes>



4. Perform descriptive statistics on the dataset.

```
data.head()
```

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
0	M	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.150	15
1	M	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.070	7
2	F	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.210	9
3	M	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.155	10
4	I	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.055	7

```
data.tail()
```

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
4172	F	0.565	0.450	0.165	0.8870	0.3700	0.2390	0.2490	11
4173	M	0.590	0.440	0.135	0.9660	0.4390	0.2145	0.2605	10
4174	M	0.600	0.475	0.205	1.1760	0.5255	0.2875	0.3080	9
4175	F	0.625	0.485	0.150	1.0945	0.5310	0.2610	0.2960	10
4176	M	0.710	0.555	0.195	1.9485	0.9455	0.3765	0.4950	12

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4177 entries, 0 to 4176
Data columns (total 9 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Sex             4177 non-null   object
 1   Length          4177 non-null   float64
 2   Diameter        4177 non-null   float64
 3   Height          4177 non-null   float64
 4   Whole weight    4177 non-null   float64
 5   Shucked weight  4177 non-null   float64
 6   Viscera weight  4177 non-null   float64
 7   Shell weight    4177 non-null   float64
 8   Rings           4177 non-null   int64
dtypes: float64(7), int64(1), object(1)
memory usage: 293.8+ KB
```

```
data.shape
```

```
(4177, 9)
```

```
data.describe()
```

	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
count	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000
mean	0.523992	0.407881	0.139516	0.828742	0.359367	0.180594	0.238831	9.933684
std	0.120093	0.099240	0.041827	0.490389	0.221963	0.109614	0.139203	3.224169
min	0.075000	0.055000	0.000000	0.002000	0.001000	0.000500	0.001500	1.000000
25%	0.450000	0.350000	0.115000	0.441500	0.186000	0.093500	0.130000	8.000000
50%	0.545000	0.425000	0.140000	0.799500	0.336000	0.171000	0.234000	9.000000
75%	0.615000	0.480000	0.165000	1.153000	0.502000	0.253000	0.329000	11.000000
max	0.815000	0.650000	1.130000	2.825500	1.488000	0.760000	1.005000	29.000000

```
data.median()
```

```
Length            0.5450
Diameter          0.4250
Height            0.1400
Whole weight      0.7995
Shucked weight    0.3360
Viscera weight    0.1710
Shell weight      0.2340
Rings             9.0000
dtype: float64
```

```
data.mode()
```

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
0	M	0.550	0.45	0.15	0.2225	0.175	0.1715	0.275	9.0
1	NaN	0.625	NaN	NaN	NaN	NaN	NaN	NaN	NaN

```
data['Length'].mode()
```

```
0    0.550
1    0.625
dtype: float64
```

```
data.var()
```

```
Length             0.014422
Diameter           0.009849
Height             0.001750
Whole weight       0.240481
Shucked weight     0.049268
Viscera weight     0.012015
Shell weight       0.019377
Rings             10.395266
dtype: float64
```

```
data.skew()
```

```
Length           -0.639873
Diameter         -0.609198
Height            3.128817
Whole weight      0.530959
Shucked weight    0.719098
Viscera weight    0.591852
Shell weight      0.620927
Rings             1.114102
dtype: float64
```

```
data.kurt()
```

```
Length             0.064621
Diameter          -0.045476
Height            76.025509
Whole weight      -0.023644
Shucked weight     0.595124
Viscera weight     0.084012
Shell weight       0.531926
Rings              2.330687
dtype: float64
```

```
data.nunique()
```

```
Sex                  3
Length             134
Diameter           111
Height              51
Whole weight      2429
Shucked weight    1515
Viscera weight     880
Shell weight       926
Rings               28
dtype: int64
```

```
data.isna().any()
```

```
Sex               False
Length            False
Diameter          False
Height            False
Whole weight      False
Shucked weight    False
Viscera weight    False
Shell weight      False
Rings             False
dtype: bool
```

5. Check for Missing values and deal with them.

```
data.isna().any().sum()
```

```
0
```

No missing values

6. Find the outliers and replace them outliers

```
qut= data.quantile(q=[0.25,0.75])
qut
```

	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
0.25	0.450	0.35	0.115	0.4415	0.186	0.0935	0.130	8.0
0.75	0.615	0.48	0.165	1.1530	0.502	0.2530	0.329	11.0

```
irq=qut.loc[0.75]- qut.loc[0.25] # q3 and q1
irq
```

```
Length            0.1650
Diameter          0.1300
Height            0.0500
Whole weight      0.7115
Shucked weight    0.3160
Viscera weight    0.1595
Shell weight      0.1990
Rings             3.0000
dtype: float64
```

```
# upper
upper= qut.loc[0.75]+(1.5*irq)
upper
```

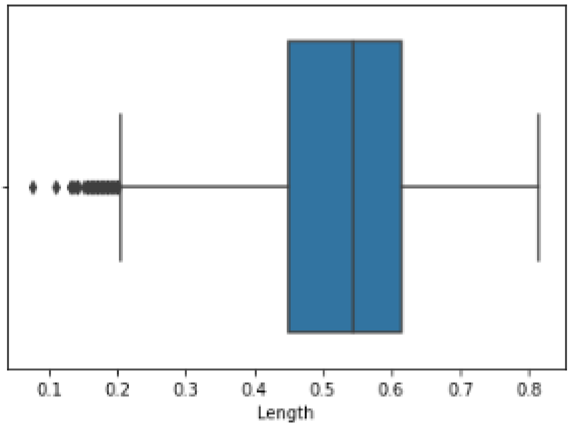
```
Length             0.86250
Diameter           0.67500
Height             0.24000
Whole weight       2.22025
Shucked weight     0.97600
Viscera weight     0.49225
Shell weight       0.62750
Rings             15.50000
dtype: float64
```

```
# lower
lower= qut.loc[0.25]+(1.5*irq)
lower
```

```
Length             0.69750
Diameter           0.54500
Height             0.19000
Whole weight       1.50875
Shucked weight     0.66000
Viscera weight     0.33275
Shell weight       0.42850
Rings             12.50000
dtype: float64
```

```python
## Length

sns.boxplot(data['Length'])
```

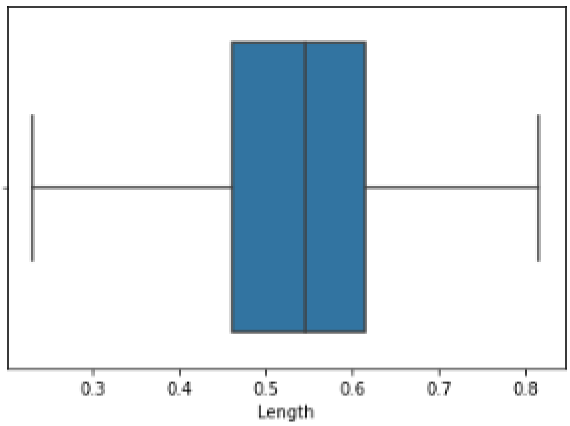
```
<AxesSubplot:xlabel='Length'>
```




```python
data['Length']=np.where(data['Length']<0.23,0.52, data['Length'])
```

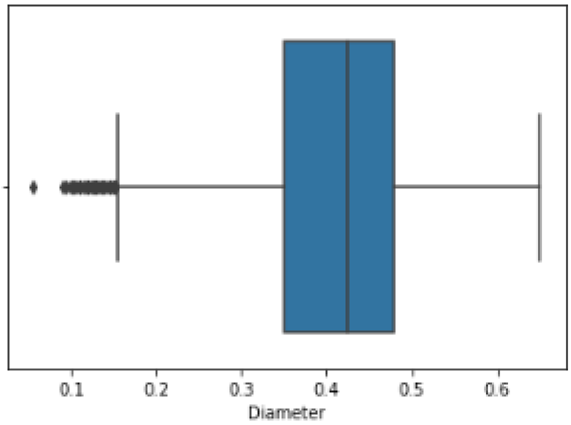
```python
sns.boxplot(data['Length'])
```

```
<AxesSubplot:xlabel='Length'>
```



```python
## Diameter

sns.boxplot(data['Diameter'])
```

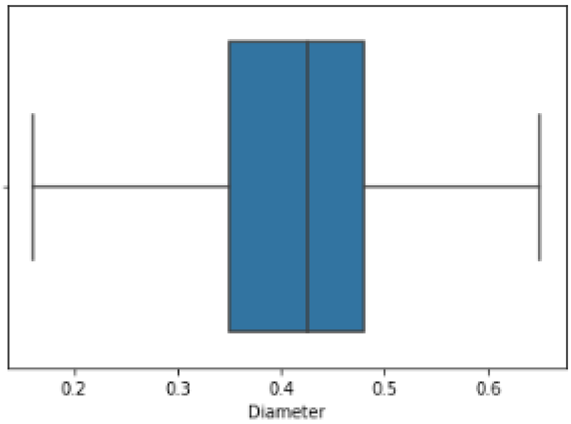
<AxesSubplot:xlabel='Diameter'>




```python
data['Diameter']=np.where(data['Diameter']<0.16,0.40, data['Diameter'])
```

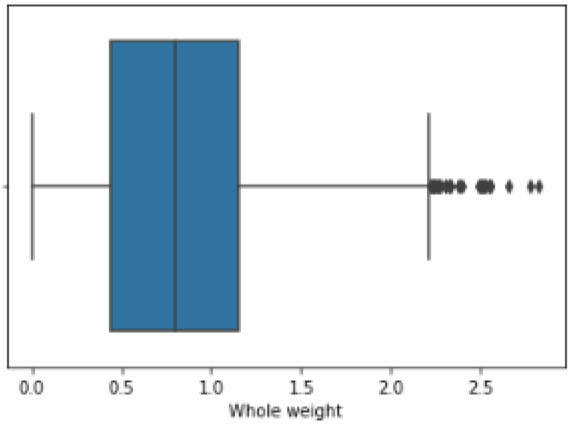
```python
sns.boxplot(data['Diameter'])
```

<AxesSubplot:xlabel='Diameter'>



```python
## Whole weight

sns.boxplot(data['Whole weight'])
```

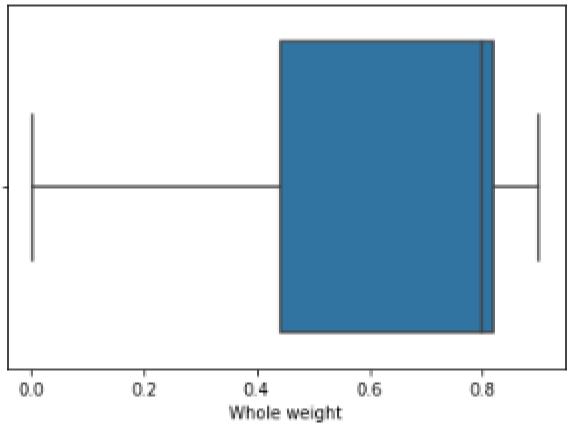
<AxesSubplot:xlabel='Whole weight'>




```python
data['Whole weight']=np.where(data['Whole weight']>0.9,0.82, data['Whole weight'])
```

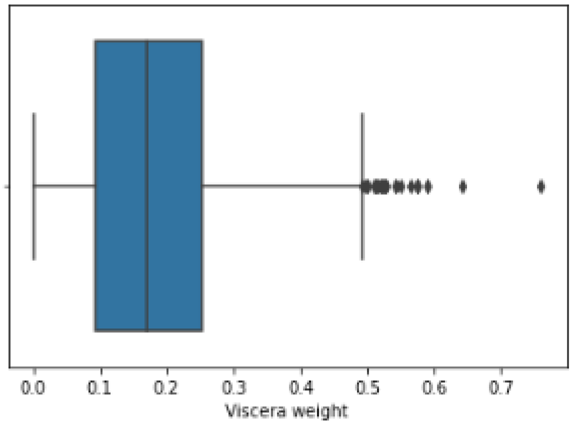
```python
sns.boxplot(data['Whole weight'])
```

<AxesSubplot:xlabel='Whole weight'>



```
## Viscera weight

sns.boxplot(data['Viscera weight'])
```

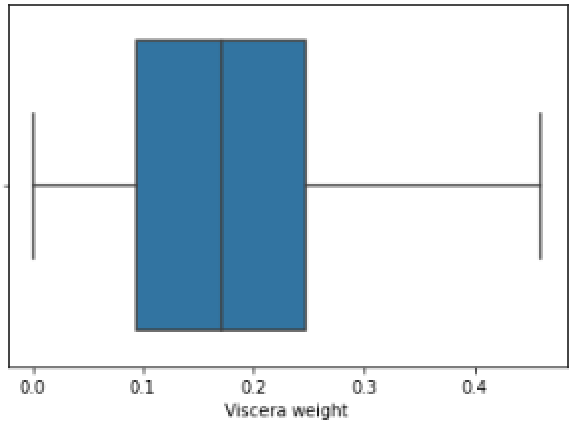
<AxesSubplot:xlabel='Viscera weight'>




```
data['Viscera weight']=np.where(data['Viscera weight']>0.46,0.18, data['Viscera weight'])
```

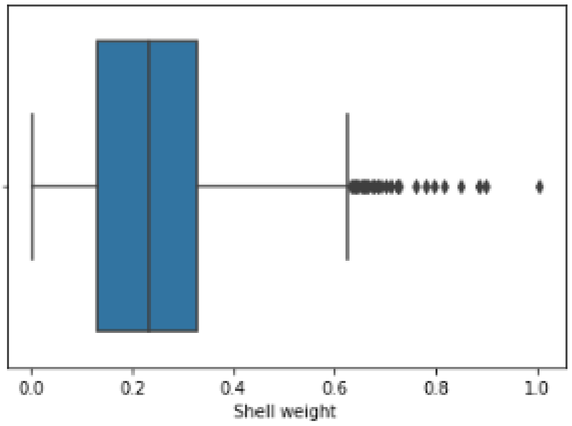
```
sns.boxplot(data['Viscera weight'])
```

<AxesSubplot:xlabel='Viscera weight'>



```
## Shell weight

sns.boxplot(data['Shell weight'])
```

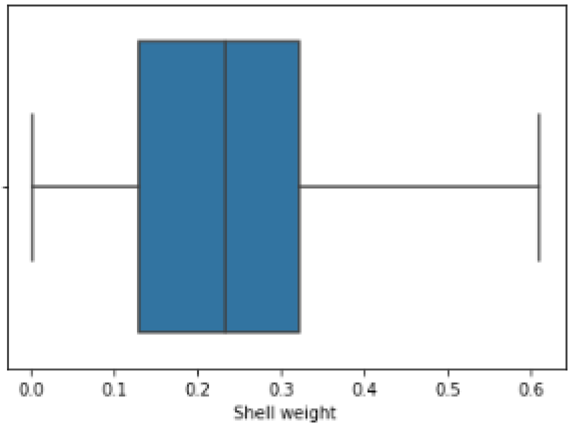
<AxesSubplot:xlabel='Shell weight'>




```
data['Shell weight']=np.where(data['Shell weight']>0.61,0.2388, data['Shell weight'])
```

```
sns.boxplot(data['Shell weight'])
```

<AxesSubplot:xlabel='Shell weight'>



7. Check for Categorical columns and perform encoding.

```
data.Sex.unique()
```

array(['M', 'F', 'I'], dtype=object)

```
## one hot encoding

data['Sex'].replace({'F':0, 'I':1, 'M': 2 }, inplace=True)
data.head()
```

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
0	2	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.150	15
1	2	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.070	7
2	0	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.210	9
3	2	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.155	10
4	1	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.055	7

8. Split the data into dependent and independent variables.

```
x=data.iloc[:,:8]
y= data.iloc[:,-1]

x
```

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight
0	2	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.1500
1	2	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.0700
2	0	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.2100
3	2	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.1550
4	1	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.0550
...	...	...	...	...	...	...	...	...
4172	0	0.565	0.450	0.165	0.8870	0.3700	0.2390	0.2490
4173	2	0.590	0.440	0.135	0.8200	0.4390	0.2145	0.2605
4174	2	0.600	0.475	0.205	0.8200	0.5255	0.2875	0.3080
4175	0	0.625	0.485	0.150	0.8200	0.5310	0.2610	0.2960
4176	2	0.710	0.555	0.195	0.8200	0.3500	0.3765	0.4950

4177 rows × 8 columns

```
y
```

```
0       15
1        7
2        9
3       10
4        7
        ..
4172    11
4173    10
4174     9
4175    10
4176    12
Name: Rings, Length: 4177, dtype: int64
```

9. Scale the independent variables

```
from sklearn.preprocessing import scale
```

```
x = scale(x)
x
```

```
array([[ 1.15198011, -0.67088921, -0.5049092 , ..., -0.61037964,
        -0.7328165 , -0.64358742],
       [ 1.15198011, -1.61376082, -1.57994526, ..., -1.22513334,
        -1.24343929, -1.25742181],
       [-1.28068972,  0.00259051,  0.08636063, ..., -0.45300269,
        -0.33890749, -0.18321163],
       ...,
       [ 1.15198011,  0.63117159,  0.67763046, ...,  0.86994729,
         1.08111018,  0.56873549],
       [-1.28068972,  0.85566483,  0.78513407, ...,  0.89699645,
         0.82336724,  0.47666033],
       [ 1.15198011,  1.61894185,  1.53765931, ...,  0.00683308,
         1.94673739,  2.00357336]])
```

10. Split the data into training and testing

```
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)
```

```
print(x_train.shape, x_test.shape)
```

```
(3341, 8) (836, 8)
```

11. Build the Model

```
from  sklearn.linear_model import LinearRegression
```

```
MLR = LinearRegression()
```

12. Train the Model

```
## Learn from training dataset

MLR.fit(x_train, y_train)
```

```
LinearRegression()
```

13. Test the Model

```
# predict on test data

y_predict = MLR.predict(x_test)
y_predict
```

```
array([10.8456925 , 12.05103635, 11.67402082,  9.6822331 ,  8.39864913,
       13.19691767, 11.65007272, 12.38988766,  7.24665349, 11.70391722,
        9.16675747,  9.24890829, 13.49751745, 10.08005211, 12.27729059,
       10.72907018, 15.31814036,  9.32451452, 10.07315385,  6.71719202,
       10.43105308, 10.05139585,  6.29862255,  6.52890621, 13.07980009,
       10.16184185,  9.50750974, 13.03162396, 11.2670134 ,  6.2923276 ,
        7.67918961, 10.89478025,  8.6057803 ,  9.4008726 , 12.39130464,
       10.68565172, 11.08200061,  8.43458669, 10.8256542 , 10.71115037,
        8.39012218, 10.96076672, 14.73647861,  8.86557419,  9.82543938,
        6.19475622,  8.50778533,  8.8799639 ,  6.85927894, 11.62467972,
        8.11713029, 10.05248336, 12.04300159,  7.51441504, 10.87899973,
       10.06581893,  9.20283032, 11.08359089, 12.31794251, 10.88833487,
        9.2342281 , 10.44300559, 11.72520635, 14.88030393,  6.56620956,
       10.71450985, 13.20942301,  9.70573658, 11.81963869, 12.60421606,
        8.74480293,  7.96722619,  9.01024075, 11.90141438,  7.0442236 ,
       10.48160999,  6.7048433 , 14.02109375, 10.38121758, 10.96753519,
       10.63784306,  9.44118633, 10.29440671,  5.96959205, 10.04724621,
        8.81766434,  6.88733868, 10.52472912,  9.43797478, 10.70424969,
        8.38742965, 13.29026493,  8.86873015, 11.21931919, 14.41904655,
```

14. Measure the performance using Metrics.

```
## Indicating the accuracy

from sklearn.metrics import r2_score
r2_score(y_test, y_predict)
```

```
0.43557418439159945
```

```
## Error

from sklearn.metrics import mean_squared_error

np.sqrt(mean_squared_error(y_test, y_predict))
```

```
2.3248014273565674
```

```
## Predicting new value

MLR.predict([[2, 0.455, 0.365, 0.095, 0.5140, 0.2240, 0.1010, 0.1500]])
```

```
array([10.18066208])
```

Lasso

```
from sklearn.linear_model import Lasso, Ridge
```

```
## Initialization

lso=Lasso(alpha=0.01, normalize= True)
```

```
## fit the model

lso.fit(x_train,y_train)
```

```
Lasso(alpha=0.01, normalize=True)
```

```
lso_pred=lso.predict(x_test)
```

```
# coeff
lso.coef_
```

```
array([-0.        ,  0.        ,  0.        ,  0.46411794,  0.16277028,
        0.        ,  0.        ,  0.84336817])
```

```
lso.alpha
```

```
0.01
```

```
## Accuracy

from sklearn import metrics
from sklearn.metrics import mean_squared_error
metrics.r2_score(y_test, lso_pred)
```

```
0.3493769048184049
```

```
## error

np.sqrt(mean_squared_error(y_test, lso_pred))
```

```
2.4960148605220445
```

Ridge

```
rdg= Ridge(alpha= 0.01, normalize =True)
rdg.fit(x_train, y_train)
```

```
Ridge(alpha=0.01, normalize=True)
```

**In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.**

```
rdg_pred= rdg.predict(x_test)
rdg_pred
```

```
array([10.8456442 , 12.05124816, 11.73542849,  9.67206237,  8.35896287,
       13.1028181 , 11.61968312, 12.43199278,  7.2757619 , 11.61417486,
        9.17106534,  9.26235646, 13.45859471, 10.08743447, 12.20229321,
       10.70579557, 15.06729377,  9.37335345, 10.05315596,  6.73862063,
       10.424592  , 10.08914379,  6.46100492,  6.52524655, 13.05138915,
       10.20822031,  9.55195673, 13.0827306 , 11.31232121,  6.29915901,
        7.71450843, 10.94134994,  8.57781362,  9.4164158 , 12.26158907,
       10.74992334, 11.04851607,  8.52831679, 10.85281158, 10.67970627,
        8.39966223, 10.97666718, 14.58843546,  8.82614522,  9.78442468,
        6.17004339,  8.56815203,  8.92810223,  6.88076953, 11.57563954,
        8.11904036, 10.06812424, 12.00185367,  7.48819841, 10.83016296,
       10.02467959,  9.09303557, 11.09907357, 12.24293595, 10.91812745,
        9.2012187 , 10.44994145, 11.7666963 , 14.73076453,  6.59964069,
       10.71320327, 13.07013444,  9.85144525, 11.78675966, 12.51738789,
        8.8654463 ,  7.95415684,  9.19252721, 11.91561211,  7.01187554,
       10.42054959,  6.72506681, 13.82942432, 10.37543094, 11.03200473,
       10.69867459,  9.4374555 , 10.28132648,  5.89542635,  9.99932419,
        8.99201657,  6.89064412, 10.56647153,  9.4761134 , 10.74819551,
        8.3872948 , 13.2215344 ,  8.84338415, 11.27226853, 14.25718321,
```

```
rdg.coef_
```

```
array([-0.02388191, -0.90121399,  0.36785195,  0.99472218,  1.06720705,
       -1.4420271 ,  0.02392643,  1.84609037])
```

```
rdg.alpha
```

```
0.01
```

```
## acc
metrics.r2_score(y_test, rdg_pred)
```

```
0.43662461242674633
```

```
## error

np.sqrt(mean_squared_error(y_test, rdg_pred))
```

```
2.3226371271260766
```