In [*]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

Load the dataset

In [*]:
```
df=pd.read_csv('abalone.csv')
```

In [3]:
```
df
```

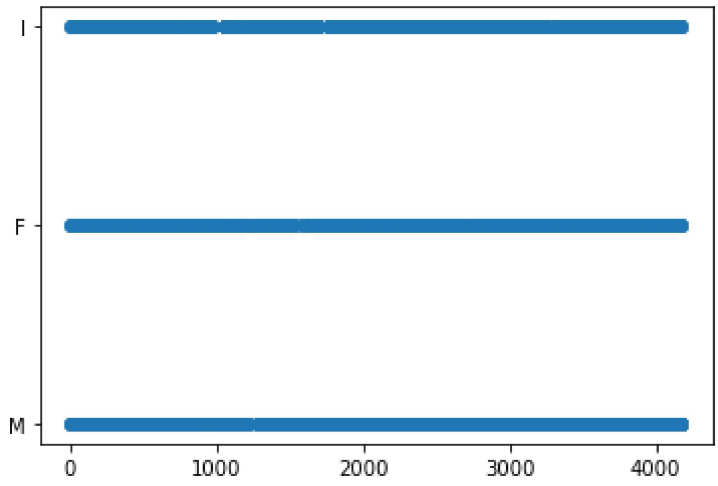
Out[3]:

	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	0.1500	15
1	M	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.0700	7
2	F	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.2100	9
3	M	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.1550	10
4	I	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.0550	7
...	...	...	...	...	...	...	...	...	...
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

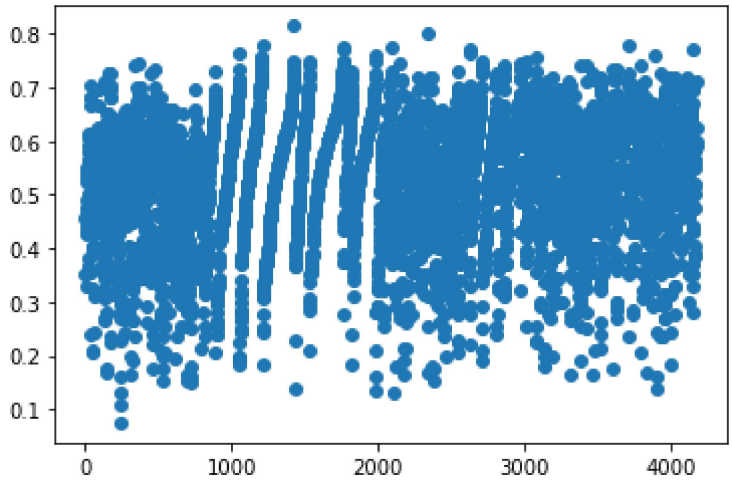
4177 rows × 9 columns

Univariate analysis

In [4]:
```
import seaborn as sns
plt.scatter(df.index,df['Sex'])
plt.show()
```

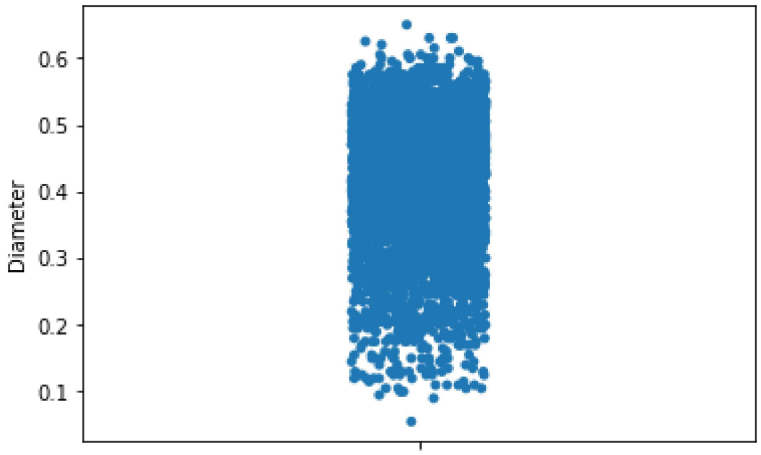


In [5]:
```
plt.scatter(df.index,df['Length'])
plt.show()
```



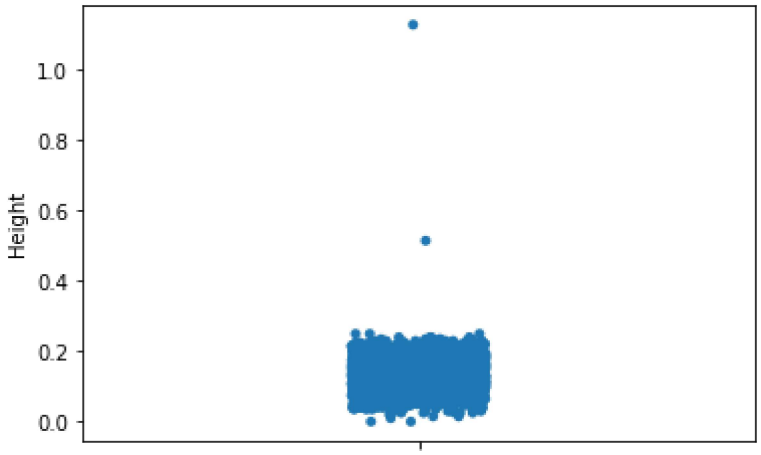
In [6]:
```
sns.stripplot(y=df['Diameter'])
```

Out[6]: <AxesSubplot:ylabel='Diameter'>



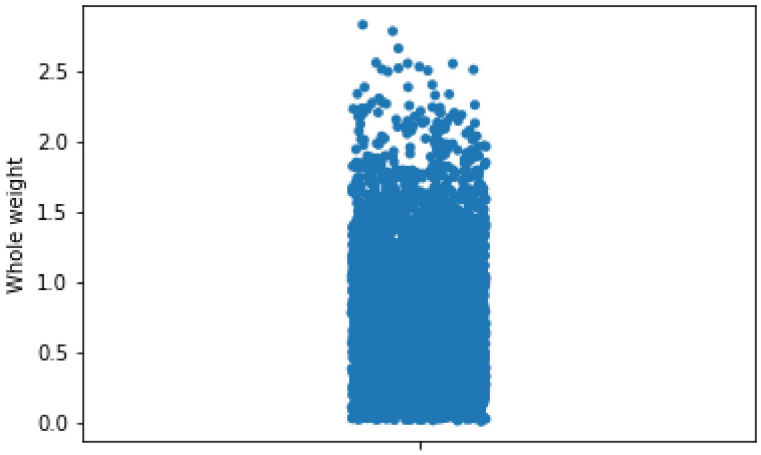
In [7]:
```
sns.stripplot(y=df['Height'])
```

Out[7]: <AxesSubplot:ylabel='Height'>

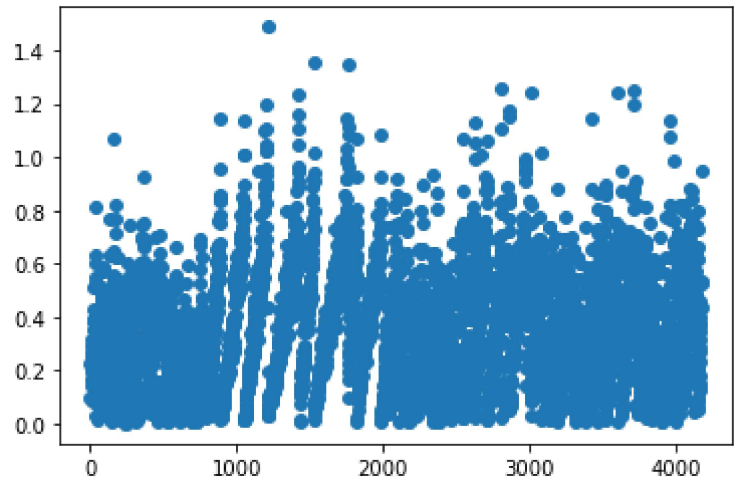


In [8]:
```
sns.stripplot(y=df['Whole weight'])
```

Out[8]: <AxesSubplot:ylabel='Whole weight'>

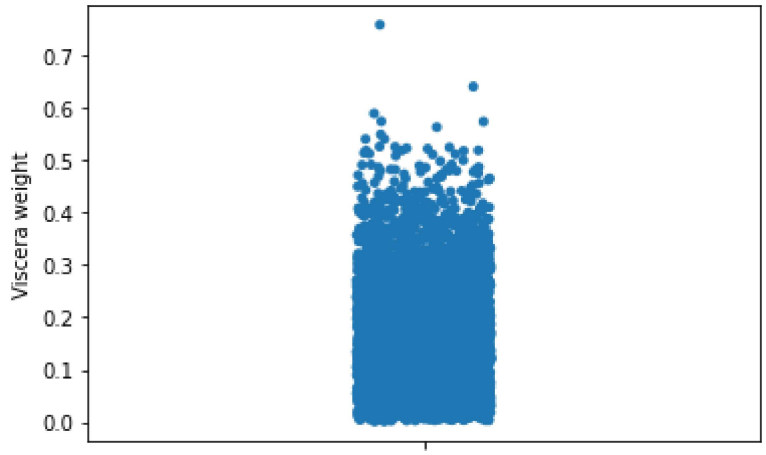


In [9]:
```
plt.scatter(df.index,df['Shucked weight'])
plt.show()
```



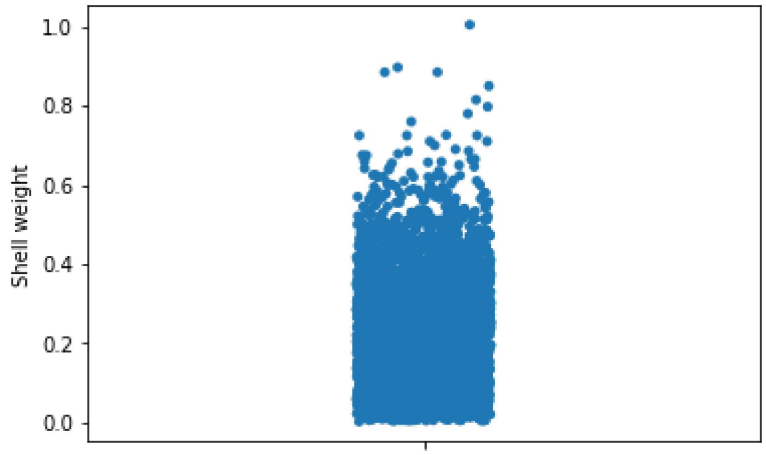
In [10]:
```
sns.stripplot(y=df['Viscera weight'])
```

Out[10]: <AxesSubplot:ylabel='Viscera weight'>

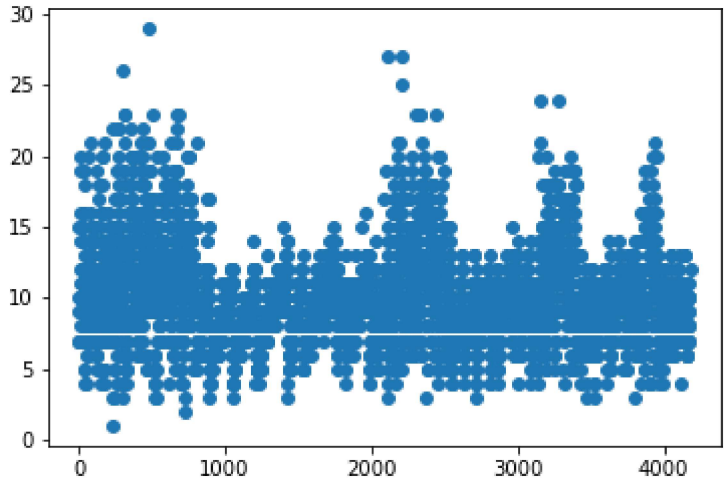


In [11]:
```
sns.stripplot(y=df['Shell weight'])
```

Out[11]: <AxesSubplot:ylabel='Shell weight'>

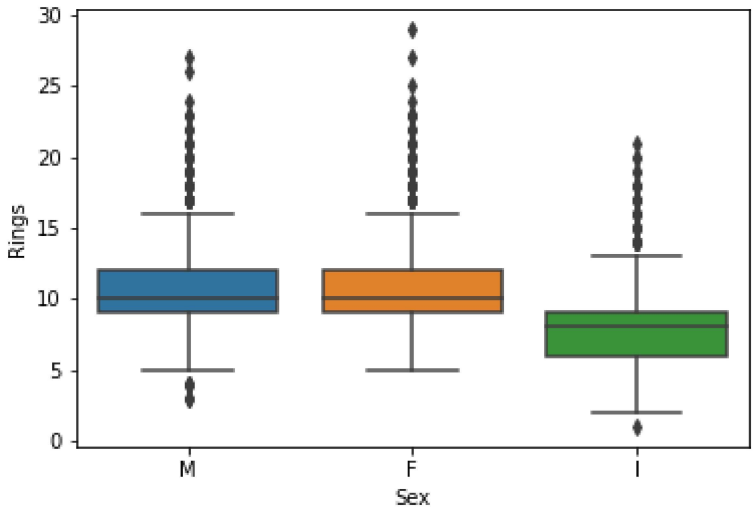


In [12]:
```
plt.scatter(df.index,df['Rings'])
plt.show()
```

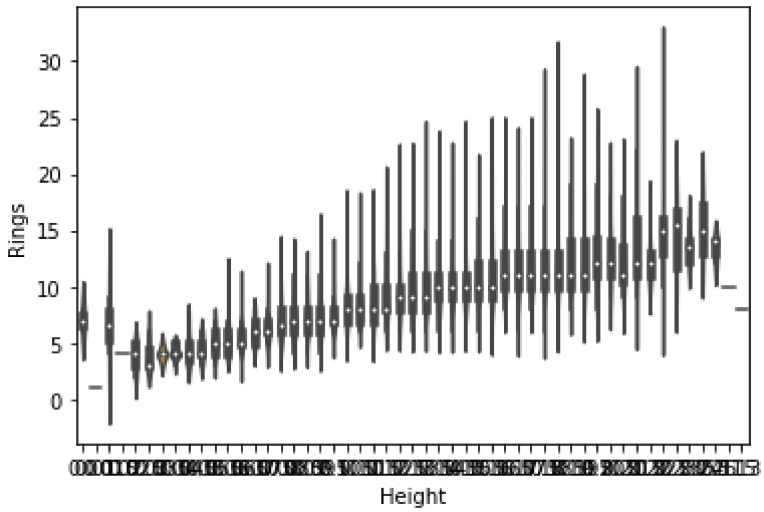


Bi-variate analysis

In [13]:
```
sns.boxplot(x='Sex',y='Rings',data=df)
plt.show()
```

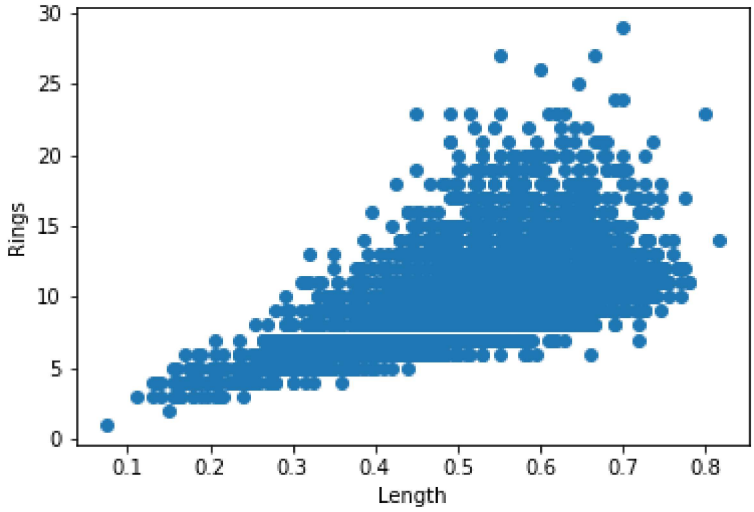


In [14]:
```
sns.violinplot(x="Height",y="Rings",size=8,data=df)
plt.show()
```



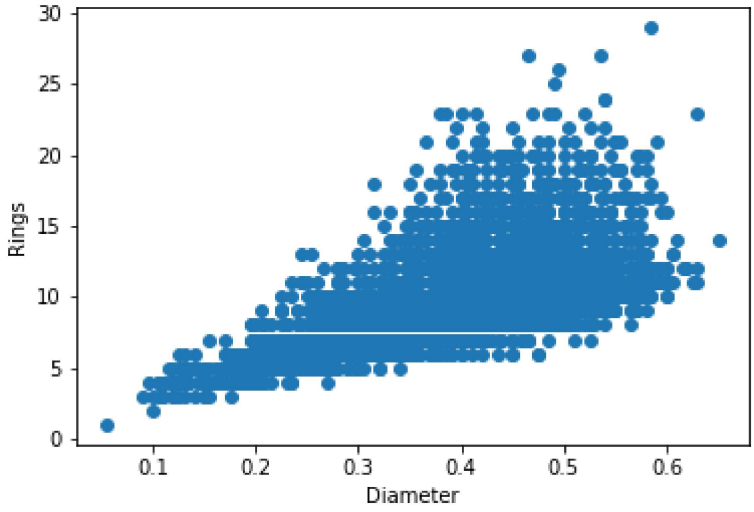
In [15]:
```
plt.plot(df["Length"],df["Rings"],'o')
plt.ylabel("Rings")
plt.xlabel("Length")
```

Out[15]: Text(0.5, 0, 'Length')



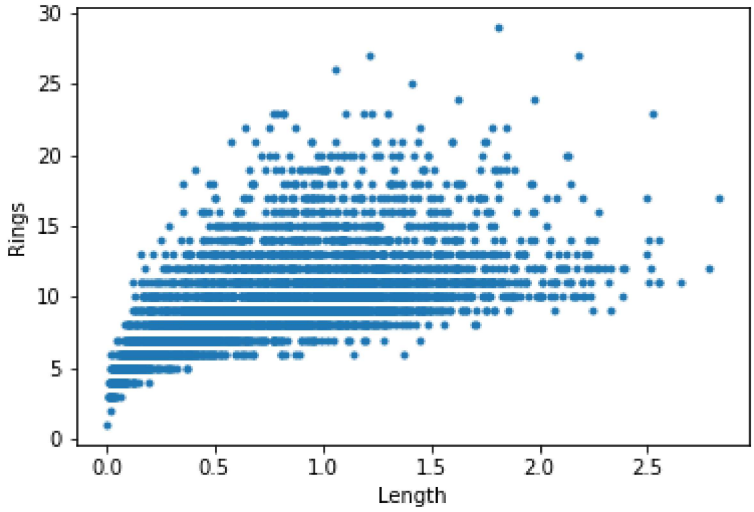
In [16]:
```
plt.plot(df["Diameter"],df["Rings"],'o')
plt.ylabel("Rings")
plt.xlabel("Diameter")
```

Out[16]: Text(0.5, 0, 'Diameter')



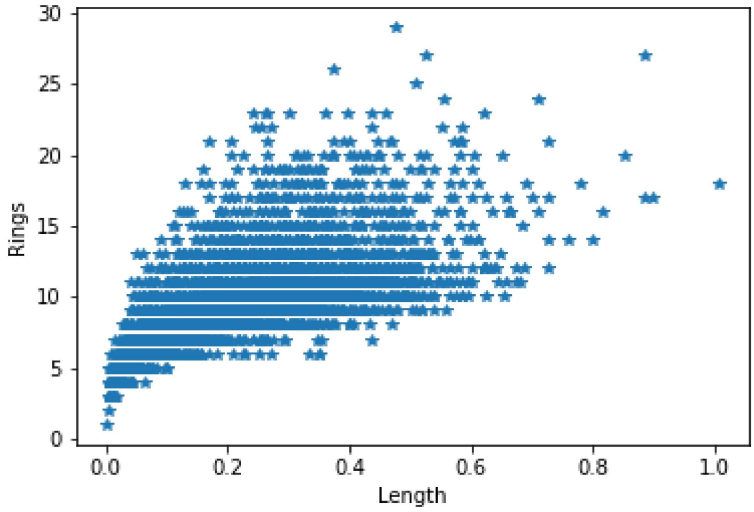
In [17]:
```
plt.plot(df["Whole weight"],df["Rings"],'.')
plt.ylabel("Rings")
plt.xlabel("Length")
```

Out[17]: Text(0.5, 0, 'Length')



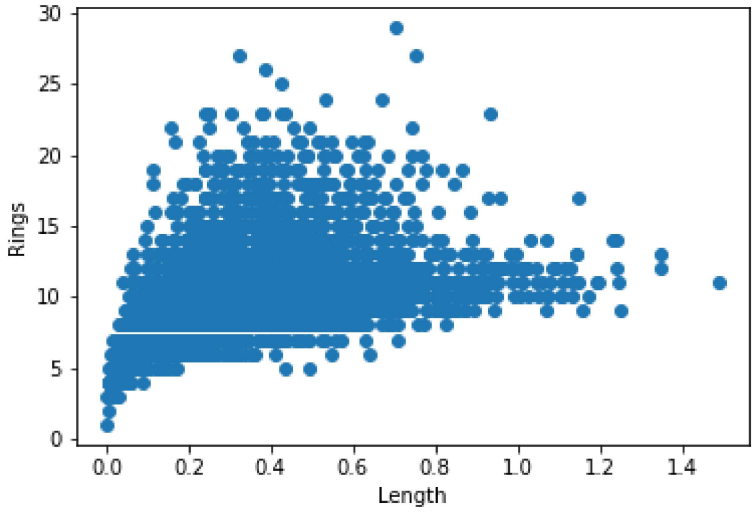
In [18]:
```
plt.plot(df["Shell weight"],df["Rings"],'*')
plt.ylabel("Rings")
plt.xlabel("Length")
```

Out[18]: Text(0.5, 0, 'Length')



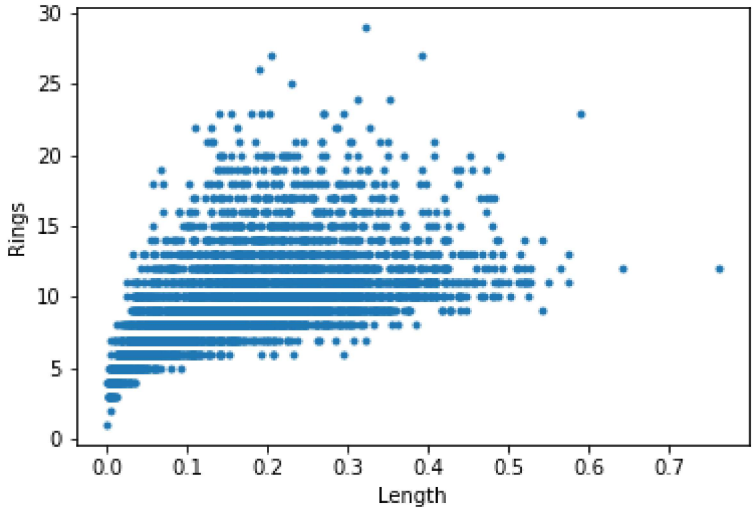
In [19]:
```
plt.plot(df["Shucked weight"],df["Rings"],'o')
plt.ylabel("Rings")
plt.xlabel("Length")
```

Out[19]: Text(0.5, 0, 'Length')



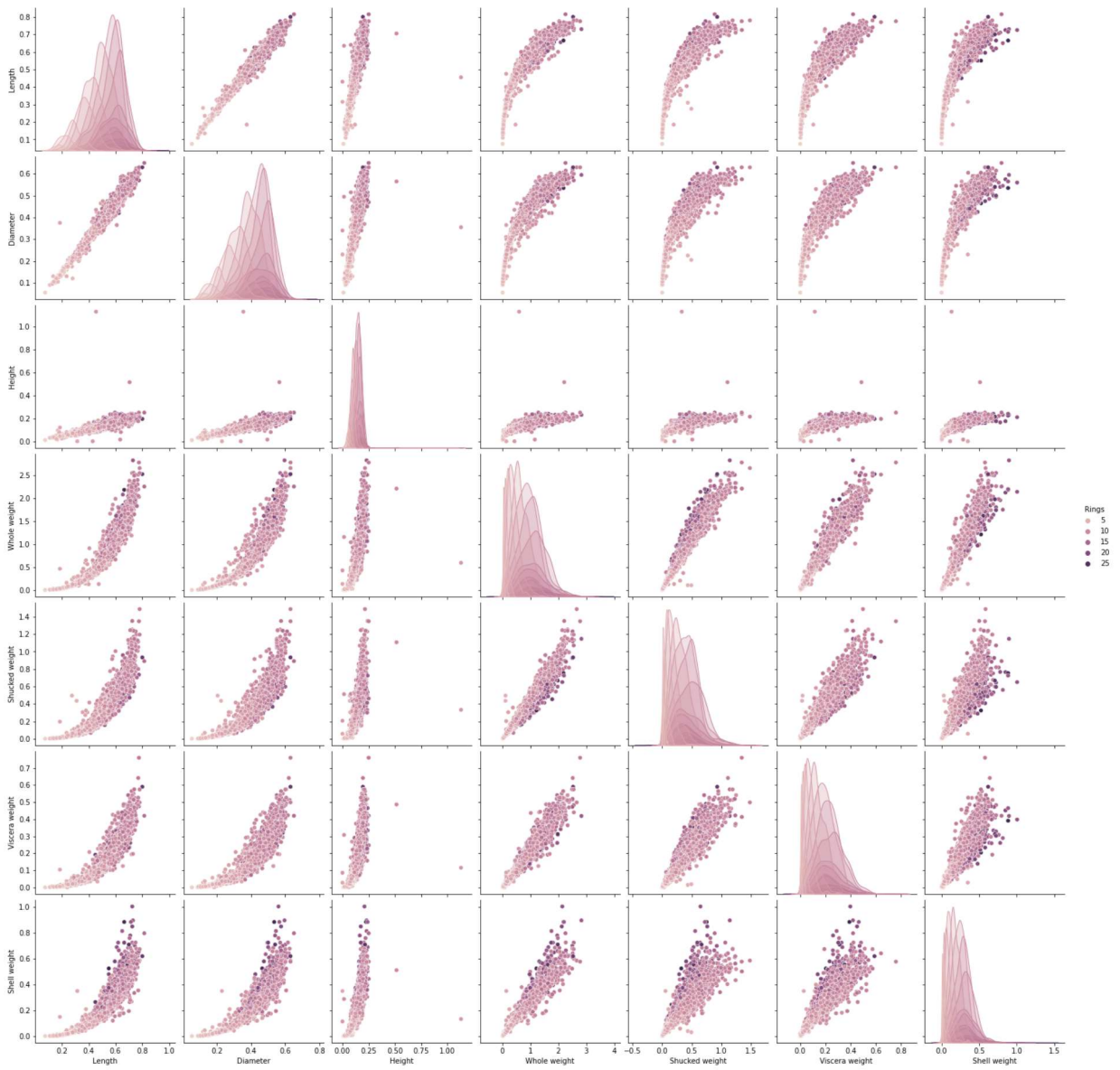
In [20]:
```
plt.plot(df["Viscera weight"],df["Rings"],'.')
plt.ylabel("Rings")
plt.xlabel("Length")
```

Out[20]: Text(0.5, 0, 'Length')



Multivariate Analysis

In [21]:
```
sns.pairplot(df,hue="Rings",height=3)
plt.show()
```



Descriptive analysis

In [22]:
```
df.describe().T
```

Out[22]:

	count	mean	std	min	25%	50%	75%	max
Length	4177.0	0.523992	0.120093	0.0750	0.4500	0.5450	0.615	0.8150
Diameter	4177.0	0.407881	0.099240	0.0550	0.3500	0.4250	0.480	0.6500
Height	4177.0	0.139516	0.041827	0.0000	0.1150	0.1400	0.165	1.1300
Whole weight	4177.0	0.828742	0.490389	0.0020	0.4415	0.7995	1.153	2.8255
Shucked weight	4177.0	0.359367	0.221963	0.0010	0.1860	0.3360	0.502	1.4880
Viscera weight	4177.0	0.180594	0.109614	0.0005	0.0935	0.1710	0.253	0.7600
Shell weight	4177.0	0.238831	0.139203	0.0015	0.1300	0.2340	0.329	1.0050
Rings	4177.0	9.933684	3.224169	1.0000	8.0000	9.0000	11.000	29.0000

Handling with missing data

In [23]:
```
df = pd.DataFrame(df)
df.isnull()
```

Out[23]:

	Sex	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
0	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...
4172	False	False	False	False	False	False	False	False	False
4173	False	False	False	False	False	False	False	False	False
4174	False	False	False	False	False	False	False	False	False
4175	False	False	False	False	False	False	False	False	False
4176	False	False	False	False	False	False	False	False	False

4177 rows × 9 columns

In [24]:
```
df.fillna(0)
```

Out[24]:

	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	0.1500	15
1	M	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.0700	7
2	F	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.2100	9
3	M	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.1550	10
4	I	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.0550	7
...	...	...	...	...	...	...	...	...	...
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

4177 rows × 9 columns

In [25]:
```
df.isnull().sum()
```

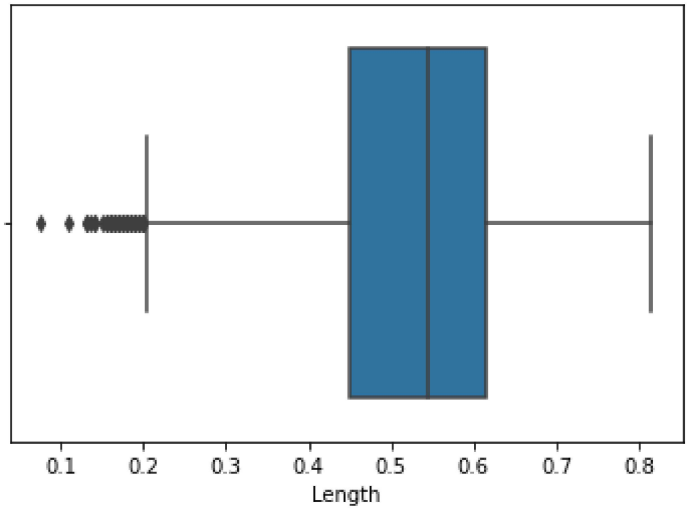
Out[25]:
```
Sex               0
Length            0
Diameter          0
Height            0
Whole weight      0
Shucked weight    0
Viscera weight    0
Shell weight      0
Rings             0
dtype: int64
```

Outliers in each attribute

In [26]:
```
sns.boxplot(df['Length'],data=df)
```

```
C:\Users\ABI\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarni
ng: Pass the following variable as a keyword arg: x. From version 0.12, the onl
y valid positional argument will be `data`, and passing other arguments without
an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

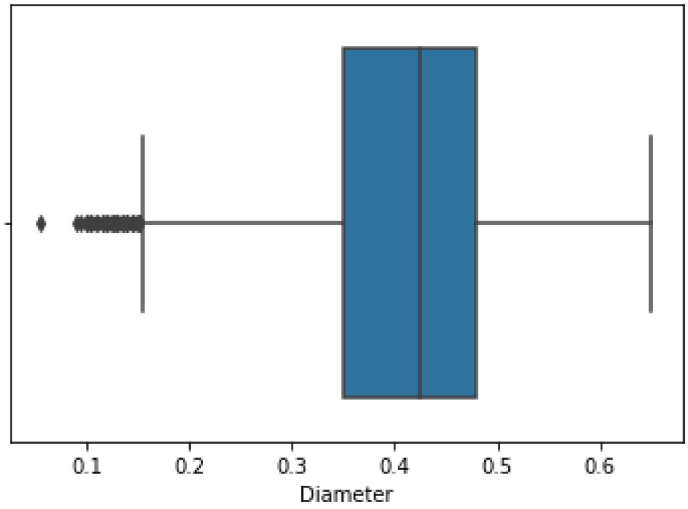
Out[26]: <AxesSubplot:xlabel='Length'>



In [27]:
```
sns.boxplot(df['Diameter'],data=df)
```

```
C:\Users\ABI\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarni
ng: Pass the following variable as a keyword arg: x. From version 0.12, the onl
y valid positional argument will be `data`, and passing other arguments without
an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

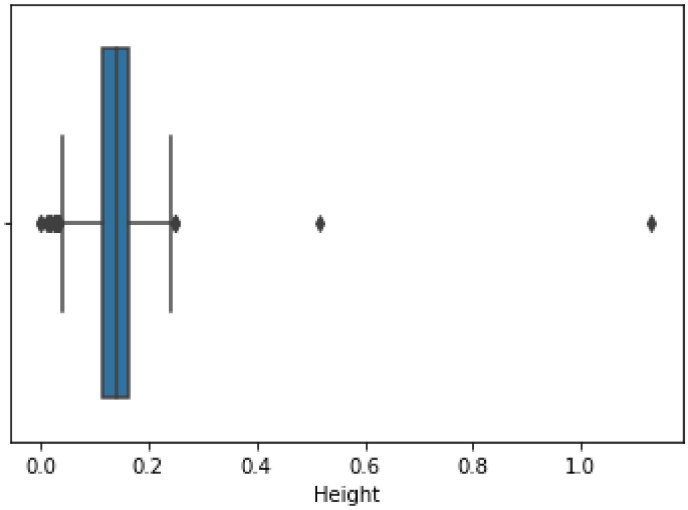
Out[27]: <AxesSubplot:xlabel='Diameter'>



In [28]:
```
sns.boxplot(df['Height'],data=df)
```

```
C:\Users\ABI\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarni
ng: Pass the following variable as a keyword arg: x. From version 0.12, the onl
y valid positional argument will be `data`, and passing other arguments without
an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

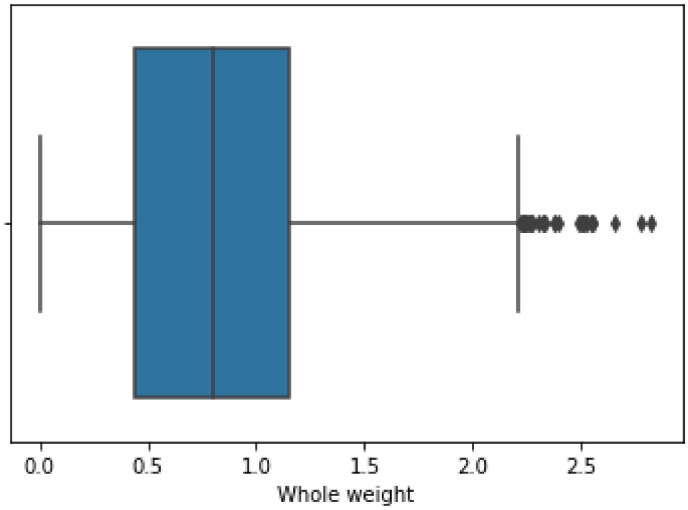
Out[28]: <AxesSubplot:xlabel='Height'>



In [29]:
```
sns.boxplot(df['Whole weight'],data=df)
```

```
C:\Users\ABI\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarni
ng: Pass the following variable as a keyword arg: x. From version 0.12, the onl
y valid positional argument will be `data`, and passing other arguments without
an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

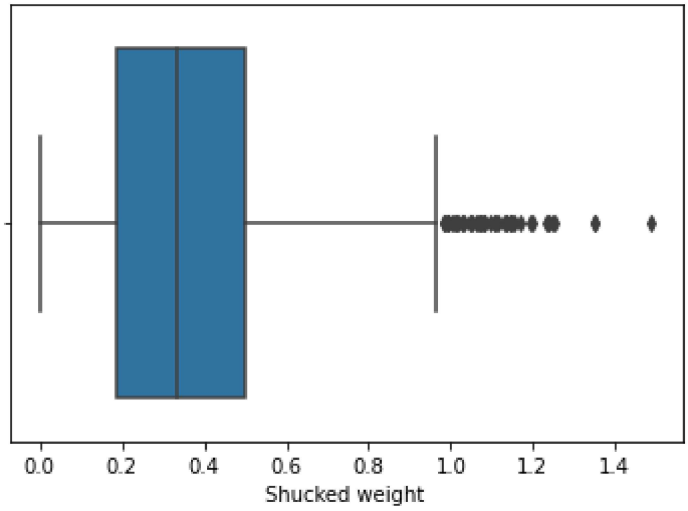
Out[29]: <AxesSubplot:xlabel='Whole weight'>



In [30]:
```
sns.boxplot(df['Shucked weight'],data=df)
```

```
C:\Users\ABI\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarni
ng: Pass the following variable as a keyword arg: x. From version 0.12, the onl
y valid positional argument will be `data`, and passing other arguments without
an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

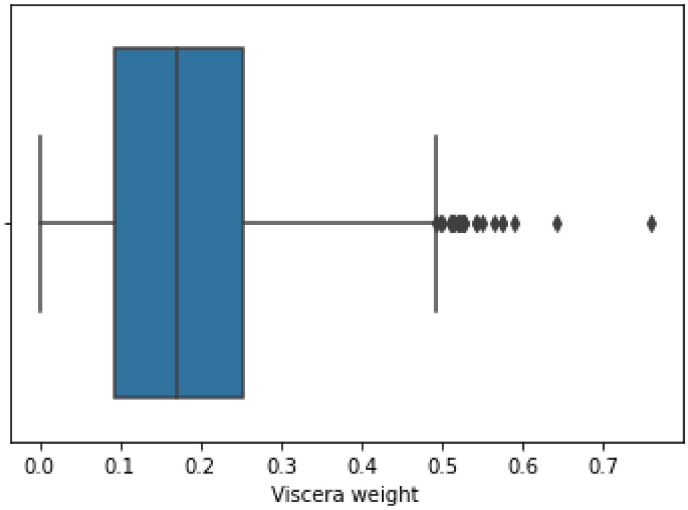
Out[30]: <AxesSubplot:xlabel='Shucked weight'>



In [31]:
```
sns.boxplot(df['Viscera weight'],data=df)
```

```
C:\Users\ABI\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarni
ng: Pass the following variable as a keyword arg: x. From version 0.12, the onl
y valid positional argument will be `data`, and passing other arguments without
an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

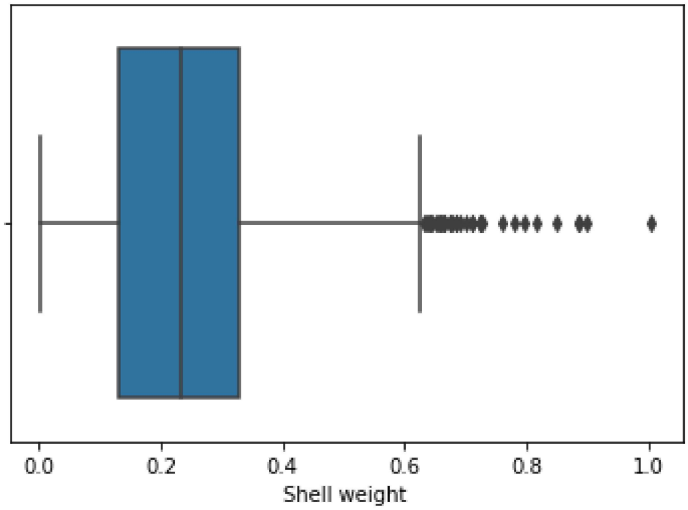
Out[31]: <AxesSubplot:xlabel='Viscera weight'>



In [32]:
```
sns.boxplot(df['Shell weight'],data=df)
```

```
C:\Users\ABI\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarni
ng: Pass the following variable as a keyword arg: x. From version 0.12, the onl
y valid positional argument will be `data`, and passing other arguments without
an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

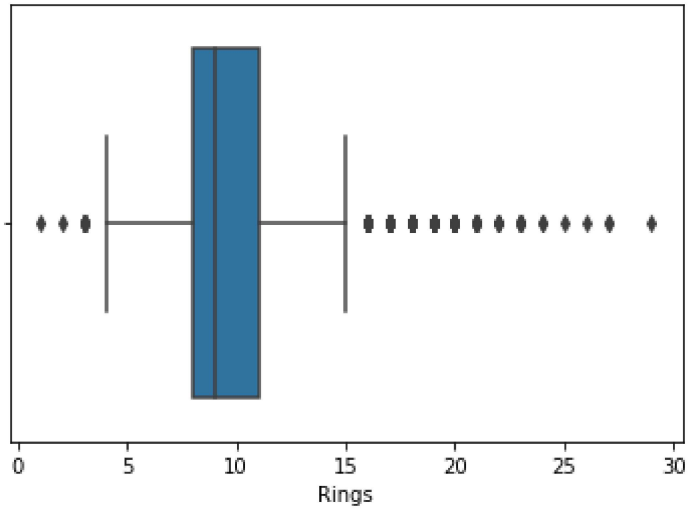
Out[32]: <AxesSubplot:xlabel='Shell weight'>



In [33]:
```
sns.boxplot(df['Rings'],data=df)
```

```
C:\Users\ABI\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarni
ng: Pass the following variable as a keyword arg: x. From version 0.12, the onl
y valid positional argument will be `data`, and passing other arguments without
an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[33]: <AxesSubplot:xlabel='Rings'>



In [38]:
```
Q1 = df.quantile(0.25)
Q3 = df.quantile(0.75)
IQR = Q3-Q1
print(IQR)
```

```
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
```

In [39]:
```
df.shape
```

Out[39]: (4177, 9)

In [43]:
```
upper = np.where(df >= (Q3+1.5*IQR))
lower = np.where(df <= (Q1-1.5*IQR))
df.drop(upper[0], inplace = True)
df.drop(lower[0], inplace = True)
df.shape
```

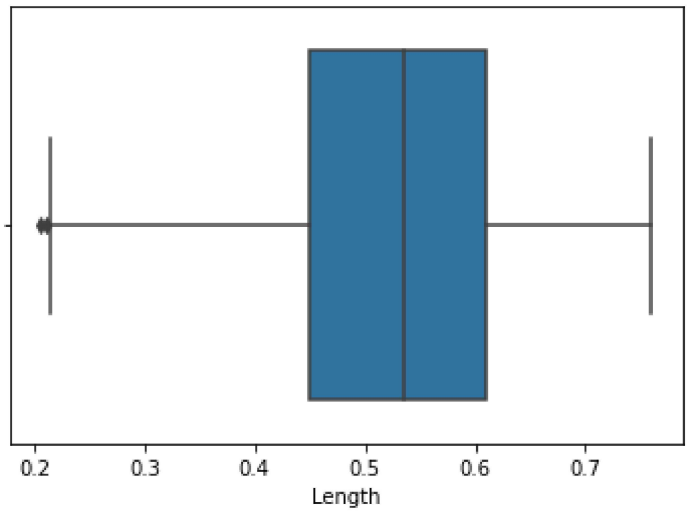
```
C:\Users\ABI\AppData\Local\Temp/ipykernel_15784/1776641308.py:1: FutureWarning:
Automatic reindexing on DataFrame vs Series comparisons is deprecated and will
raise ValueError in a future version.  Do `left, right = left.align(right, axis
=1, copy=False)` before e.g. `left == right`
  upper = np.where(df >= (Q3+1.5*IQR))
C:\Users\ABI\AppData\Local\Temp/ipykernel_15784/1776641308.py:2: FutureWarning:
Automatic reindexing on DataFrame vs Series comparisons is deprecated and will
raise ValueError in a future version.  Do `left, right = left.align(right, axis
=1, copy=False)` before e.g. `left == right`
  lower = np.where(df <= (Q1-1.5*IQR))
```

Out[43]:
```
(3781, 9)
```

In [44]:
```
sns.boxplot(df['Length'],data=df)
```

```
C:\Users\ABI\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarni
ng: Pass the following variable as a keyword arg: x. From version 0.12, the onl
y valid positional argument will be `data`, and passing other arguments without
an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

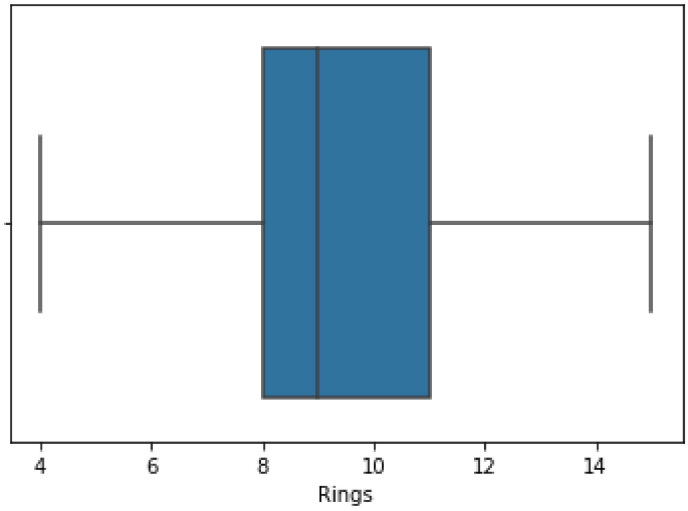
Out[44]:
```
<AxesSubplot:xlabel='Length'>
```



In [45]:
```
sns.boxplot(df['Rings'],data=df)
```

```
C:\Users\ABI\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarni
ng: Pass the following variable as a keyword arg: x. From version 0.12, the onl
y valid positional argument will be `data`, and passing other arguments without
an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

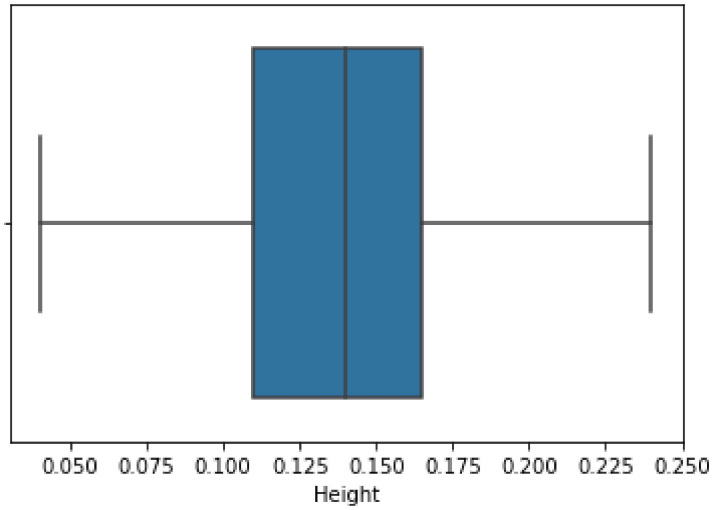
Out[45]: <AxesSubplot:xlabel='Rings'>



In [46]:
```
sns.boxplot(df['Height'],data=df)
```

```
C:\Users\ABI\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarni
ng: Pass the following variable as a keyword arg: x. From version 0.12, the onl
y valid positional argument will be `data`, and passing other arguments without
an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

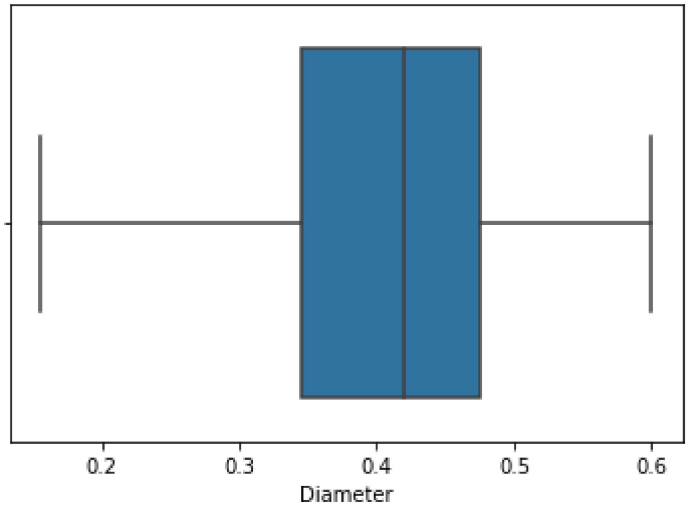
Out[46]: <AxesSubplot:xlabel='Height'>



In [47]:
```
sns.boxplot(df['Diameter'],data=df)
```

```
C:\Users\ABI\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarni
ng: Pass the following variable as a keyword arg: x. From version 0.12, the onl
y valid positional argument will be `data`, and passing other arguments without
an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

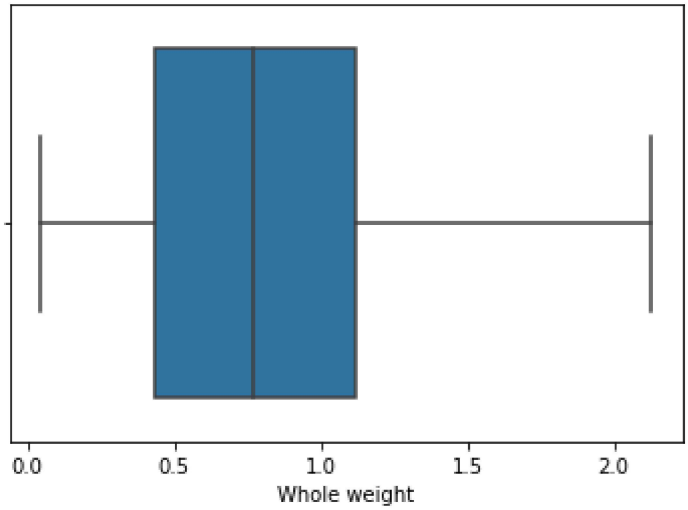
Out[47]: <AxesSubplot:xlabel='Diameter'>



In [48]:
```
sns.boxplot(df['Whole weight'],data=df)
```

```
C:\Users\ABI\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarni
ng: Pass the following variable as a keyword arg: x. From version 0.12, the onl
y valid positional argument will be `data`, and passing other arguments without
an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

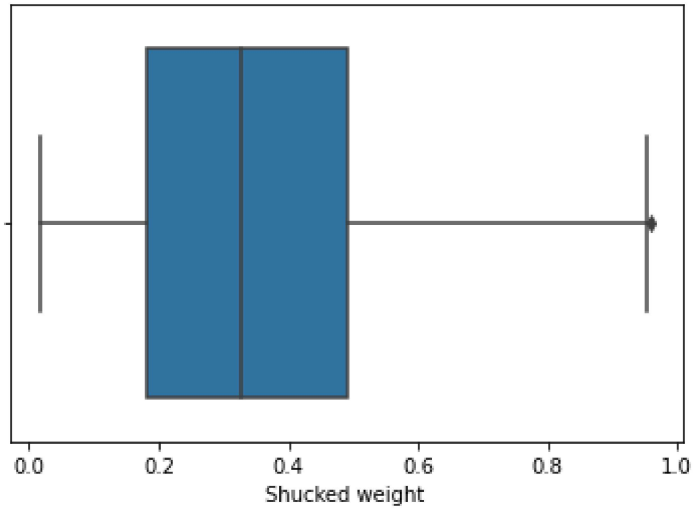
Out[48]: <AxesSubplot:xlabel='Whole weight'>



In [49]:
```
sns.boxplot(df['Shucked weight'],data=df)
```

```
C:\Users\ABI\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarni
ng: Pass the following variable as a keyword arg: x. From version 0.12, the onl
y valid positional argument will be `data`, and passing other arguments without
an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

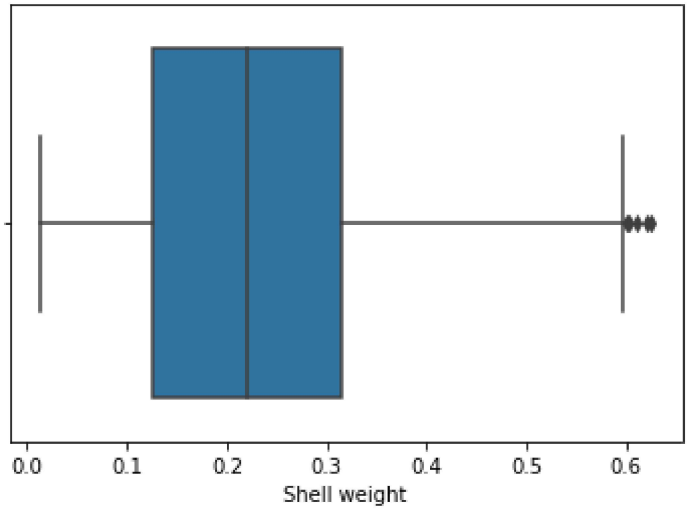
Out[49]: <AxesSubplot:xlabel='Shucked weight'>



In [50]:
```
sns.boxplot(df['Shell weight'],data=df)
```

```
C:\Users\ABI\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarni
ng: Pass the following variable as a keyword arg: x. From version 0.12, the onl
y valid positional argument will be `data`, and passing other arguments without
an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

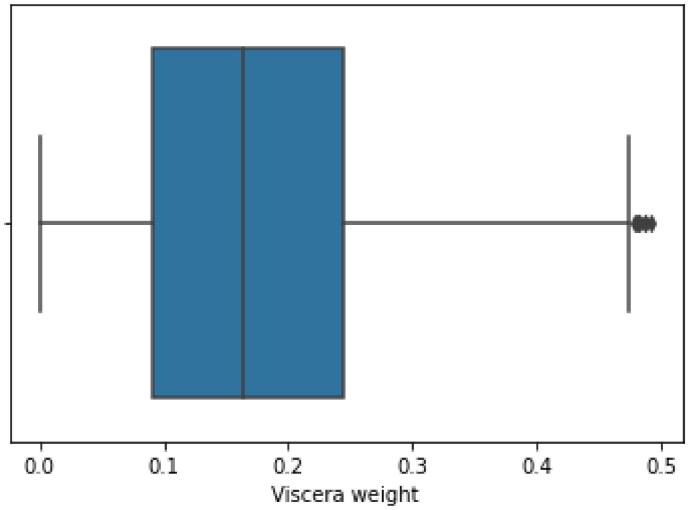
Out[50]: <AxesSubplot:xlabel='Shell weight'>



In [51]:
```
sns.boxplot(df['Viscera weight'],data=df)
```

```
C:\Users\ABI\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarni
ng: Pass the following variable as a keyword arg: x. From version 0.12, the onl
y valid positional argument will be `data`, and passing other arguments without
an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

Out[51]:
```
<AxesSubplot:xlabel='Viscera weight'>
```



Label encoding for categorical data

In [53]:
```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler

from sklearn.linear_model import LinearRegression
from sklearn.svm import SVR
from sklearn.tree import DecisionTreeRegressor
from sklearn import metrics
```

In [55]:
```
le=LabelEncoder()
df['Sex']=le.fit_transform(df['Sex'])
```

In [56]:
```
df
```

Out[56]:

	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	0.1500	15
1	2	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.0700	7
2	0	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.2100	9
3	2	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.1550	10
4	1	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.0550	7
...	...	...	...	...	...	...	...	...	...
4172	0	0.565	0.450	0.165	0.8870	0.3700	0.2390	0.2490	11
4173	2	0.590	0.440	0.135	0.9660	0.4390	0.2145	0.2605	10
4174	2	0.600	0.475	0.205	1.1760	0.5255	0.2875	0.3080	9
4175	0	0.625	0.485	0.150	1.0945	0.5310	0.2610	0.2960	10
4176	2	0.710	0.555	0.195	1.9485	0.9455	0.3765	0.4950	12

3781 rows × 9 columns

Spliting the Data into dependent and Independent Variables

In [57]:
```
X = df.iloc[:, :-1].values
y = df.iloc[:, -1].values
```

Scaling independent variables

In [58]:
```
scaler = StandardScaler()
scaler.fit(df)
```

Out[58]:
```
StandardScaler()
```

Spliting training and test data

In [59]:
```
train_X,val_X,train_y,val_y = train_test_split(X, y, test_size = 0.2, random_stat
```

In [60]:
```
print("Shape of Training X :",train_X.shape)
print("Shape of Validation X :",val_X.shape)
```

```
Shape of Training X : (3024, 8)
Shape of Validation X : (757, 8)
```

In [61]:
```
print("Shape of Training y :",train_y.shape)
print("Shape of Validation y :",val_y.shape)
```

```
Shape of Training y : (3024,)
Shape of Validation y : (757,)
```

Linear regression

In [62]:
```
lr = LinearRegression()
lr.fit(train_X,train_y)
```

Out[62]:
```
LinearRegression()
```

In [63]:
```
%%time
y_pred_val_lr = lr.predict(val_X)
print('MAE on Validation set :',metrics.mean_absolute_error(val_y, y_pred_val_lr)
print("\n")
print('MSE on Validation set :',metrics.mean_squared_error(val_y, y_pred_val_lr))
print("\n")
print('RMSE on Validation set :',np.sqrt(metrics.mean_absolute_error(val_y, y_pre
print("\n")
print('R2 Score on Validation set :',metrics.r2_score(val_y, y_pred_val_lr))
print("\n")
```

```
MAE on Validation set : 1.2719689486359296


MSE on Validation set : 2.7606215450501024


RMSE on Validation set : 1.1278160083257949


R2 Score on Validation set : 0.5119499107890585


Wall time: 4.82 ms
```

Support vector machine

In [64]:
```
svm = SVR()
svm.fit(train_X,train_y)
```

Out[64]:
```
SVR()
```

In [65]:
```
%%time
y_pred_val_svm = svm.predict(val_X)
print('MAE on Validation set :',metrics.mean_absolute_error(val_y, y_pred_val_svm
print("\n")
print('MSE on Validation set :',metrics.mean_squared_error(val_y, y_pred_val_svm)
print("\n")
print('RMSE on Validation set :',np.sqrt(metrics.mean_absolute_error(val_y, y_pre
print("\n")
print('R2 Score on Validation set :',metrics.r2_score(val_y, y_pred_val_svm))
print("\n")
```

```
MAE on Validation set : 1.220895278727089


MSE on Validation set : 2.7012620714060267


RMSE on Validation set : 1.104941301032362


R2 Score on Validation set : 0.5224440679687887


Wall time: 496 ms
```

Decision tree regressor

In [66]:
```
dc = DecisionTreeRegressor(random_state = 0)
dc.fit(train_X,train_y)
```

Out[66]:
```
DecisionTreeRegressor(random_state=0)
```

In [67]:
```
%%time
y_pred_val_dc = dc.predict(val_X)
print('MAE on Validation set :',metrics.mean_absolute_error(val_y, y_pred_val_dc)
print("\n")
print('MSE on Validation set :',metrics.mean_squared_error(val_y, y_pred_val_dc))
print("\n")
print('RMSE on Validation set :',np.sqrt(metrics.mean_absolute_error(val_y, y_pre
print("\n")
print('R2 Score on Validation set :',metrics.r2_score(val_y, y_pred_val_dc))
print("\n")
```

```
MAE on Validation set : 1.6393659180977542


MSE on Validation set : 4.88110964332893


RMSE on Validation set : 1.2803772561623212


R2 Score on Validation set : 0.13706896870869845


Wall time: 6.78 ms
```

In [ ]:
```
Overview of R2 scores of all models
```

In [68]:
```
print('Logistic Regression R2 Score on Validation set :',metrics.r2_score(val_y,
print('SVR R2 Score on Validation set :',metrics.r2_score(val_y, y_pred_val_svm))
print('Decision Tree Regressor R2 Score on Validation set :',metrics.r2_score(val
```

```
Logistic Regression R2 Score on Validation set : 0.5119499107890585
SVR R2 Score on Validation set : 0.5224440679687887
Decision Tree Regressor R2 Score on Validation set : 0.13706896870869845
```

In [ ]:
```

```