1.Download the dataset

2. Load the Dataset

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import scale
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score,confusion_matrix,classification_
```

In [2]:
```
data = pd.read_csv("abalone.csv")
data
```

Out[2]:

	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 [3]:
```
data.head()
```

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	0.150	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

One additional task is that, we have to add the "Age" column using "Rings" data. We just have to add '1.5' to the ring data

In [4]:
```
Age=1.5+data.Rings
data["Age"]=Age
data=data.rename(columns = {'Whole weight':'Whole_weight','Shucked weight':
                            'Shell weight': 'Shell_weight'})
data=data.drop(columns=["Rings"],axis=1)
data.head()
```

Out[4]:

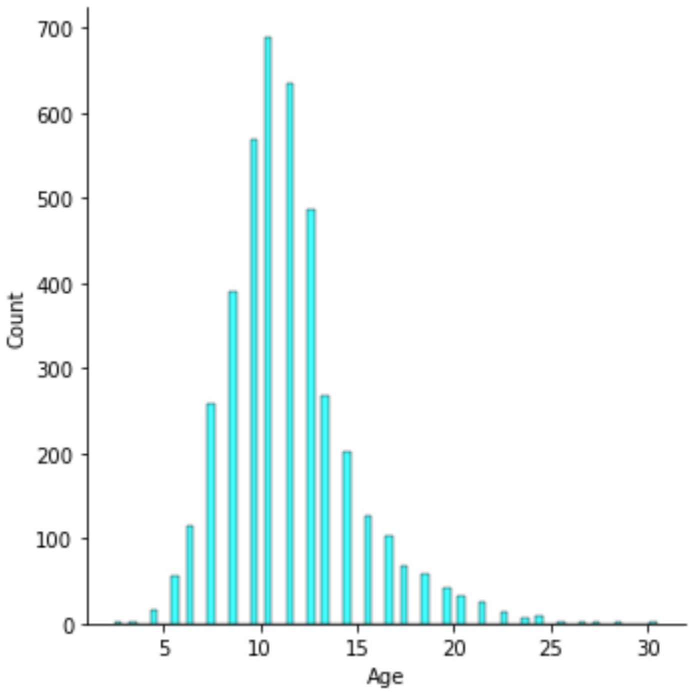
	Sex	Length	Diameter	Height	Whole_weight	Shucked_weight	Viscera_weight	Shell_wei
0	M	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.1
1	M	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.0
2	F	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.2
3	M	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.1
4	I	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.0

3.Performing Analysis

1. Univariate Analysis

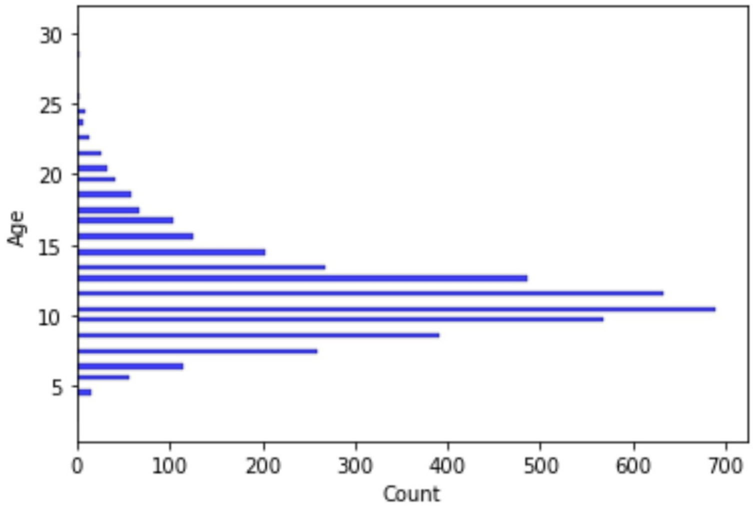
In [5]:
```
#Histogram
sns.displot(data["Age"], color='Cyan')
```

Out[5]: `<seaborn.axisgrid.FacetGrid at 0x2734f29e9a0>`



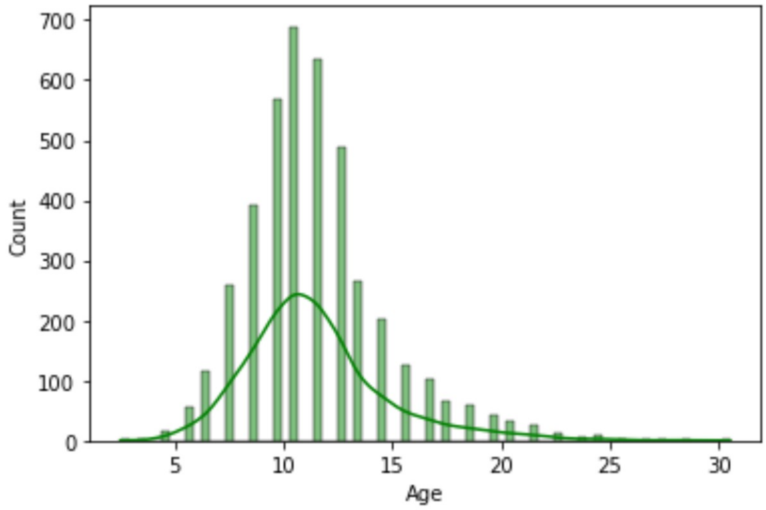
In [6]:
```
sns.histplot(y=data.Age,color='blue')
```

Out[6]: <AxesSubplot:xlabel='Count', ylabel='Age'>



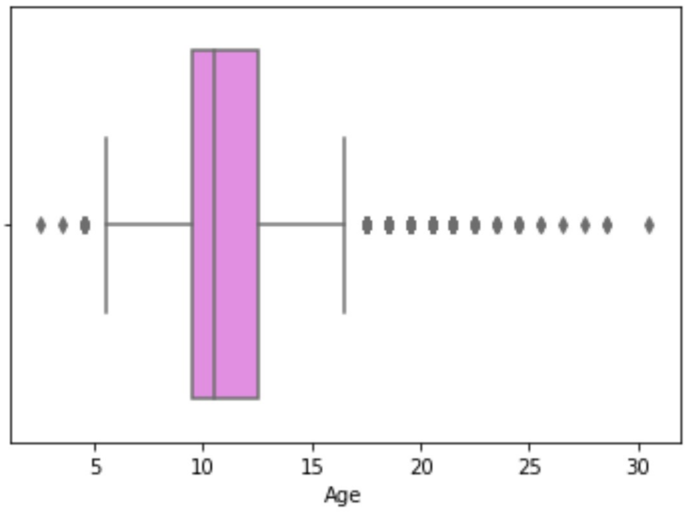
In [7]:
```
sns.histplot(x=data.Age,kde=True,color='green')
```

Out[7]: <AxesSubplot:xlabel='Age', ylabel='Count'>



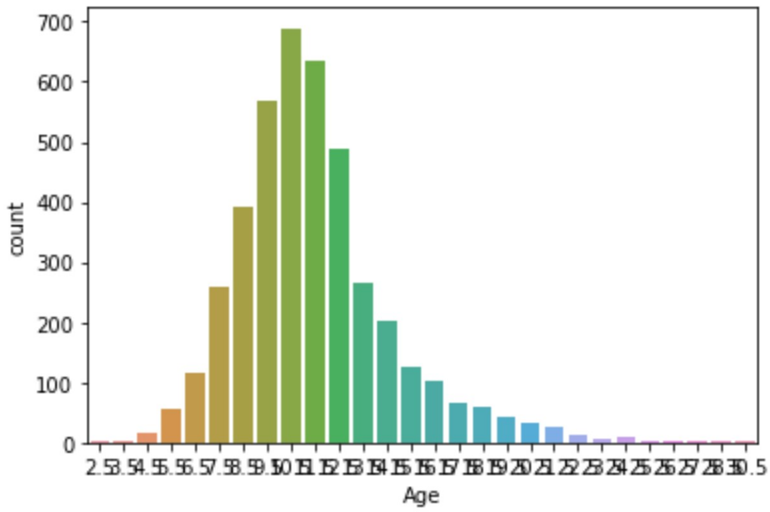
In [8]:
```
sns.boxplot(x=data.Age,color='violet')
```

Out[8]: <AxesSubplot:xlabel='Age'>



In [9]:
```
# Count-plot
sns.countplot(x=data.Age)
```

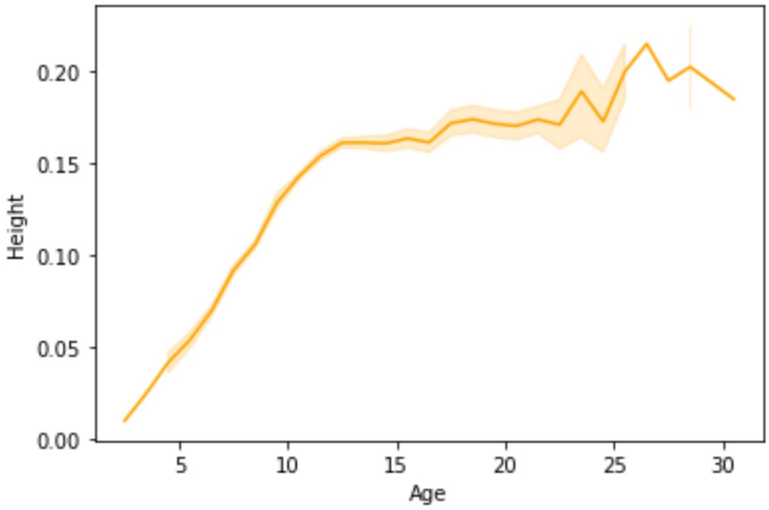
Out[9]: <AxesSubplot:xlabel='Age', ylabel='count'>



2. Bi - variate Analysis

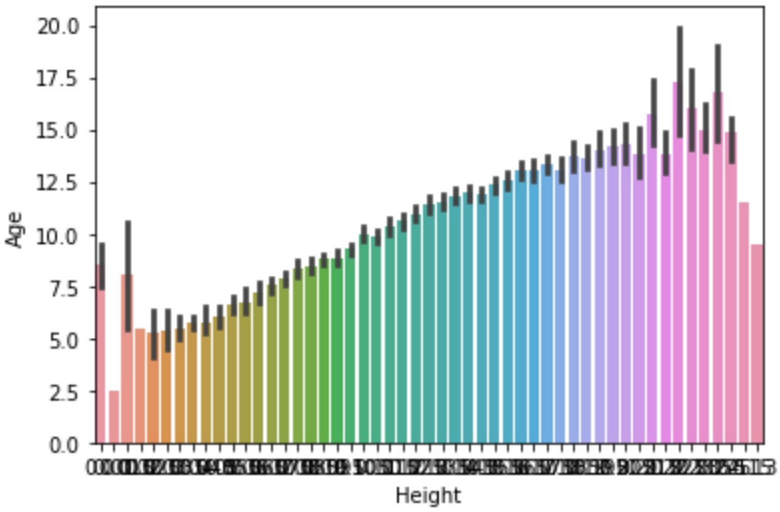
In [10]:
```
#Linearplot
sns.lineplot(x=data.Age,y=data.Height, color='orange')
```

Out[10]: <AxesSubplot:xlabel='Age', ylabel='Height'>



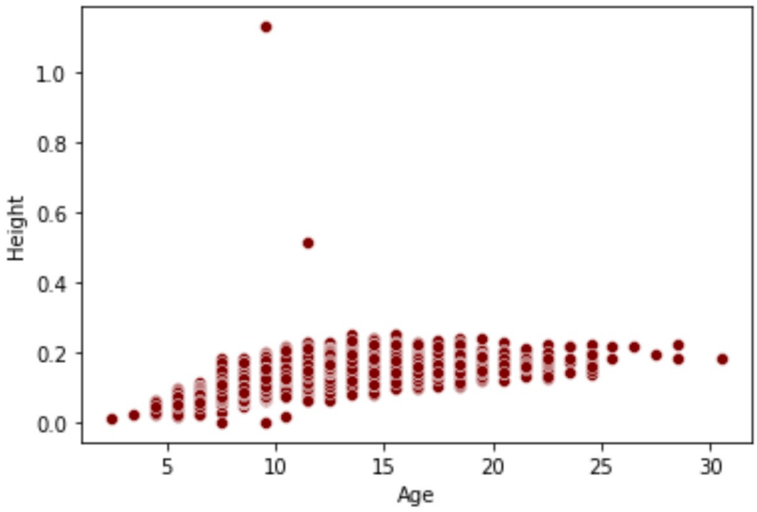
In [11]:
```
#Bar-plot
sns.barplot(x=data.Height,y=data.Age)
```

Out[11]: <AxesSubplot:xlabel='Height', ylabel='Age'>



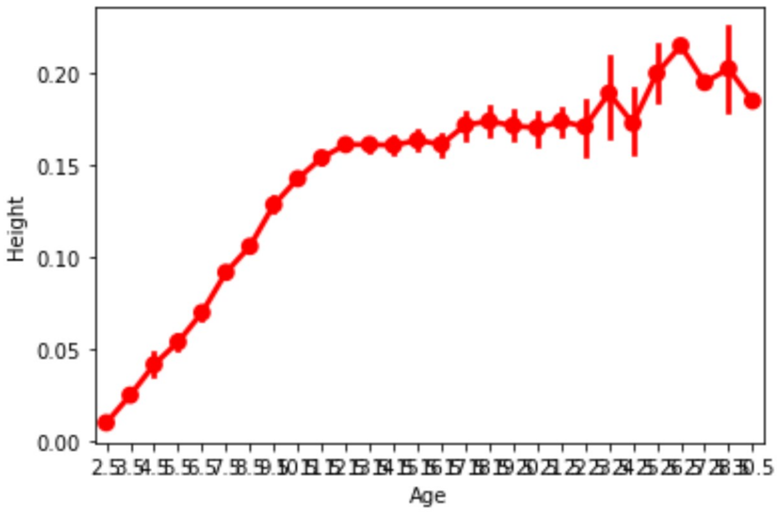
In [12]:
```
#Scatterplot
sns.scatterplot(x=data.Age,y=data.Height,color='maroon')
```

Out[12]: <AxesSubplot:xlabel='Age', ylabel='Height'>



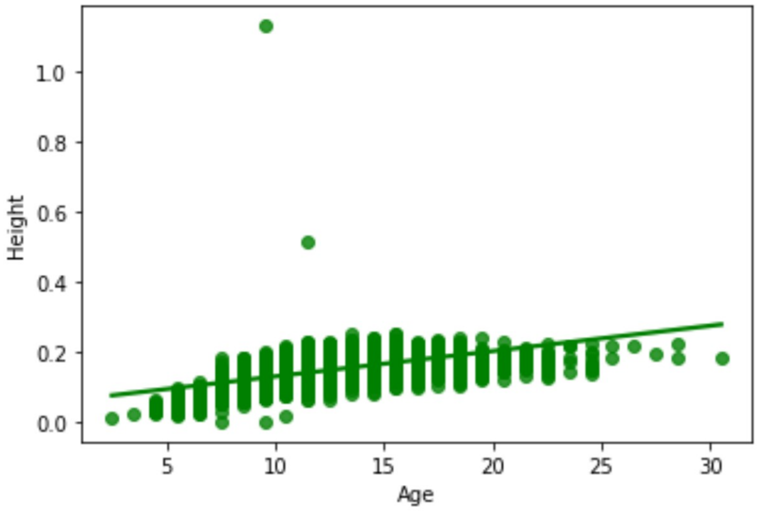
In [13]:
```
#point-plot
sns.pointplot(x=data.Age, y=data.Height, color="red")
```

Out[13]: <AxesSubplot:xlabel='Age', ylabel='Height'>



In [14]:
```
sns.regplot(x=data.Age,y=data.Height,color='green')
```

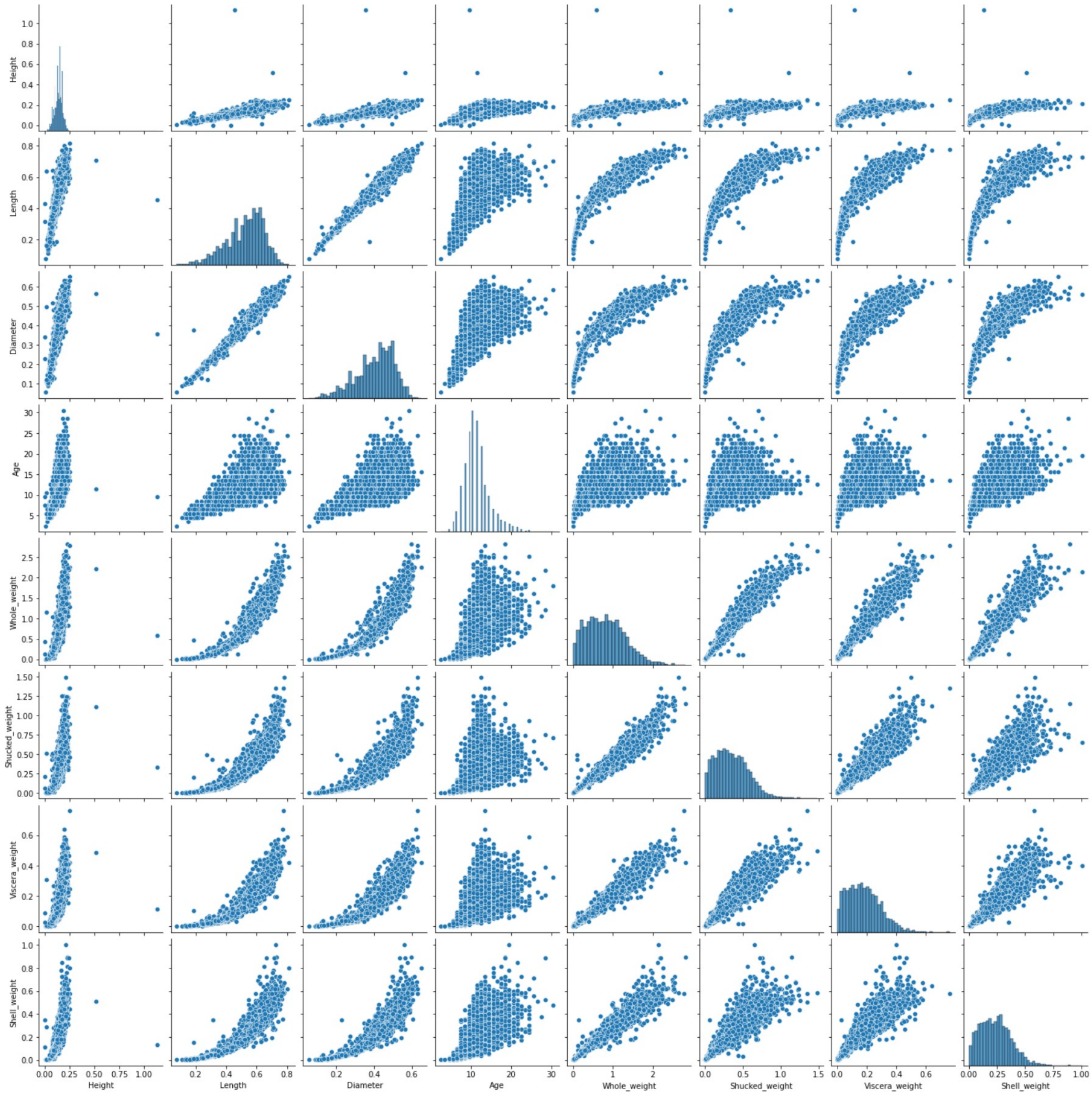
Out[14]:
```
<AxesSubplot:xlabel='Age', ylabel='Height'>
```



3. Multi-Variate Analysis

In [15]:
```
#pair-plot
sns.pairplot(data=data[["Height","Length","Diameter","Age","Whole_weight","S
```

Out[15]: <seaborn.axisgrid.PairGrid at 0x27351115520>



4. Perform descriptive statistics on the dataset

In [16]:
```
#mode
data[['Length','Diameter','Whole_weight','Shucked_weight','Viscera_weight',
```

Out[16]:
```
Length            0.523992
Diameter          0.407881
Whole_weight      0.828742
Shucked_weight    0.359367
Viscera_weight    0.180594
Shell_weight      0.238831
dtype: float64
```

In [17]:
```
#median
data[['Length','Diameter','Whole_weight','Shucked_weight','Viscera_weight',
```

Out[17]:
```
Length            0.5450
Diameter          0.4250
Whole_weight      0.7995
Shucked_weight    0.3360
Viscera_weight    0.1710
Shell_weight      0.2340
dtype: float64
```

In [18]:
```
data[['Length','Diameter','Whole_weight','Shucked_weight','Viscera_weight',
```

Out[18]:

	Length	Diameter	Whole_weight	Shucked_weight	Viscera_weight	Shell_weight
0	0.550	0.45	0.2225	0.175	0.1715	0.275
1	0.625	NaN	NaN	NaN	NaN	NaN

In [19]:
```
#qunatile
data[['Length','Diameter','Whole_weight','Shucked_weight','Viscera_weight',
```

Out[19]:
```
Length            0.5450
Diameter          0.4250
Whole_weight      0.7995
Shucked_weight    0.3360
Viscera_weight    0.1710
Shell_weight      0.2340
Name: 0.5, dtype: float64
```

In [20]:
```
#Standard - deviatiion
data[['Length','Diameter','Whole_weight','Shucked_weight','Viscera_weight',
```

Out[20]:
```
Length            0.120093
Diameter          0.099240
Whole_weight      0.490389
Shucked_weight    0.221963
Viscera_weight    0.109614
Shell_weight      0.139203
dtype: float64
```

In [21]:
```
#min
data[['Length','Diameter','Whole_weight','Shucked_weight','Viscera_weight',
```

Out[21]:
```
Length            0.0750
Diameter          0.0550
Whole_weight      0.0020
Shucked_weight    0.0010
Viscera_weight    0.0005
Shell_weight      0.0015
dtype: float64
```

In [22]:
```
#max
data[['Length','Diameter','Whole_weight','Shucked_weight','Viscera_weight',
```

Out[22]:
```
Length            0.8150
Diameter          0.6500
Whole_weight      2.8255
Shucked_weight    1.4880
Viscera_weight    0.7600
Shell_weight      1.0050
dtype: float64
```

In [23]:
```
#Skew
data[['Length','Diameter','Whole_weight','Shucked_weight','Viscera_weight',
```

Out[23]:
```
Length           -0.639873
Diameter         -0.609198
Whole_weight      0.530959
Shucked_weight    0.719098
Viscera_weight    0.591852
Shell_weight      0.620927
dtype: float64
```

In [24]:
```
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   Age             4177 non-null   float64
dtypes: float64(8), object(1)
memory usage: 293.8+ KB
```

In [25]:
```
data.shape
```

Out[25]:
```
(4177, 9)
```

In [26]:
```
data.describe(include='all')
```

Out[26]:

	Sex	Length	Diameter	Height	Whole_weight	Shucked_weight	Viscer
count	4177	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000	417
unique	3	NaN	NaN	NaN	NaN	NaN	
top	M	NaN	NaN	NaN	NaN	NaN	
freq	1528	NaN	NaN	NaN	NaN	NaN	
mean	NaN	0.523992	0.407881	0.139516	0.828742	0.359367	
std	NaN	0.120093	0.099240	0.041827	0.490389	0.221963	
min	NaN	0.075000	0.055000	0.000000	0.002000	0.001000	
25%	NaN	0.450000	0.350000	0.115000	0.441500	0.186000	
50%	NaN	0.545000	0.425000	0.140000	0.799500	0.336000	
75%	NaN	0.615000	0.480000	0.165000	1.153000	0.502000	
max	NaN	0.815000	0.650000	1.130000	2.825500	1.488000	

5. Check for Missing values and deal with them.

In [27]:
```
data.isnull().sum()
```

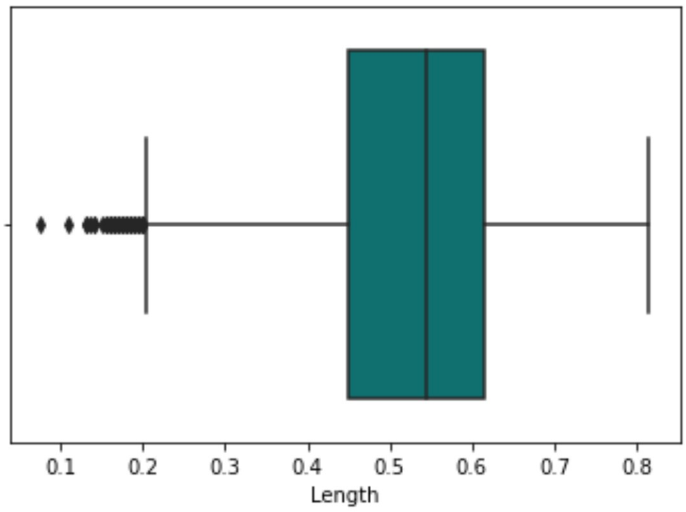
Out[27]:
```
Sex               0
Length            0
Diameter          0
Height            0
Whole_weight      0
Shucked_weight    0
Viscera_weight    0
Shell_weight      0
Age               0
dtype: int64
```

Hence ,There is no missing values in the dataset

6. Find the outliers and replace them outliers

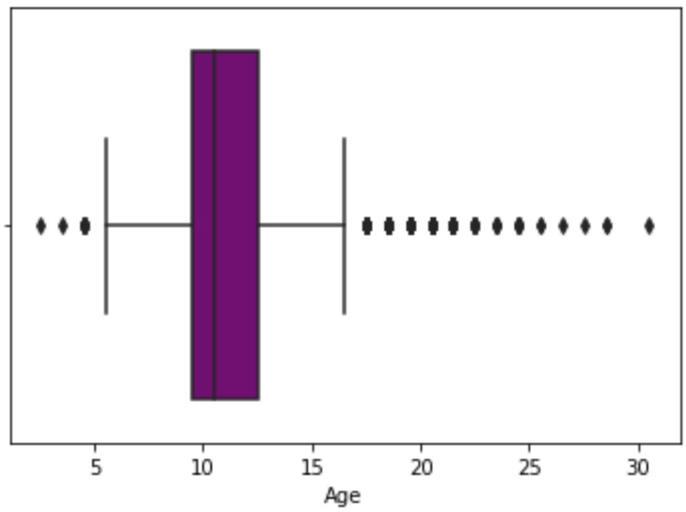
In [28]:
```
sns.boxplot(x=data["Length"],color="Teal")
```

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



In [29]:
```
sns.boxplot(x=data["Age"],color="purple")
```

Out[29]: <AxesSubplot:xlabel='Age'>



In [30]:
```
outliers=data.quantile(q=(0.25,0.75))
outliers
```

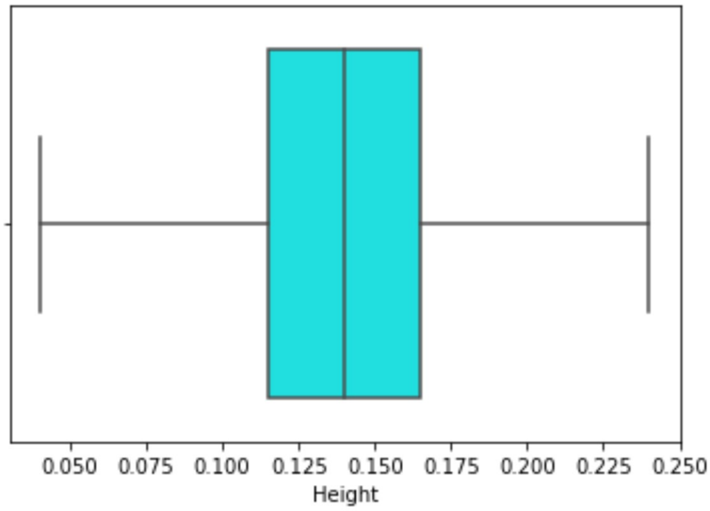
Out[30]:

	Length	Diameter	Height	Whole_weight	Shucked_weight	Viscera_weight	Shell_weight
0.25	0.450	0.35	0.115	0.4415	0.186	0.0935	0.130
0.75	0.615	0.48	0.165	1.1530	0.502	0.2530	0.329

In [31]:
```
for i in data:
    if data[i].dtype=='int64' or data[i].dtypes=='float64':
        q1=data[i].quantile(0.25)
        q3=data[i].quantile(0.75)
        iqr=q3-q1
        upper=q3+1.5*iqr
        lower=q1-1.5*iqr
        data[i]=np.where(data[i] >upper, upper, data[i])
        data[i]=np.where(data[i] <lower, lower, data[i])
```

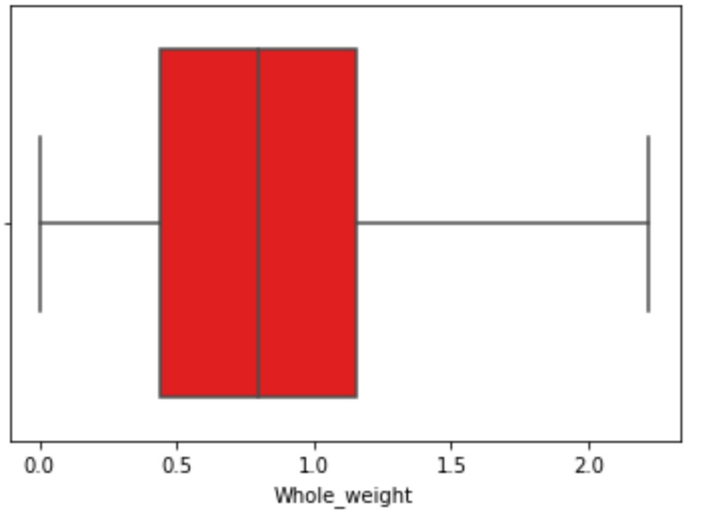
In [32]:
```
print(sns.boxplot(x=data["Height"],color="cyan"))
```

```
AxesSubplot(0.125,0.125;0.775x0.755)
```



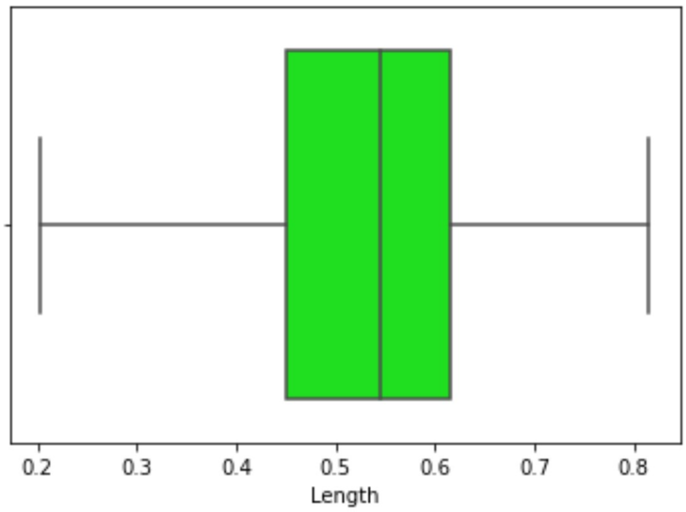
In [33]:
```
sns.boxplot(x=data["Whole_weight"],color="red")
```

Out[33]:
```
<AxesSubplot:xlabel='Whole_weight'>
```



In [34]:
```
sns.boxplot(x=data["Length"],color="lime")
```

Out[34]: <AxesSubplot:xlabel='Length'>



7. Checking for Categorical columns and performing encoding

In [35]:
```
label = LabelEncoder()
data.Sex = label.fit_transform(data.Sex)
data.head()
```

Out[35]:

	Sex	Length	Diameter	Height	Whole_weight	Shucked_weight	Viscera_weight	Shell_weig
0	2	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.1
1	2	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.0
2	0	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.2
3	2	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.1
4	1	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.0

8. Splitting the data into dependent and independent variables

In [36]:
```
y = data["Sex"]
y.head()
```

Out[36]:
```
0    2
1    2
2    0
3    2
4    1
Name: Sex, dtype: int32
```

In [37]:
```
x=data.drop(columns=["Sex"],axis=1)
x.head()
```

Out[37]:

	Length	Diameter	Height	Whole_weight	Shucked_weight	Viscera_weight	Shell_weight	A
0	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.150	1
1	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.070	
2	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.210	1
3	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.155	1
4	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.055	

9. Scale the independent variables

In [38]:
```
X_Scaled = pd.DataFrame(scale(x), columns=x.columns)
X_Scaled.head()
```

Out[38]:

	Length	Diameter	Height	Whole_weight	Shucked_weight	Viscera_weight	Shell_weig
0	-0.583117	-0.440884	-1.158093	-0.644740	-0.614985	-0.730304	-0.6451
1	-1.465694	-1.459762	-1.288751	-1.238208	-1.191637	-1.213890	-1.231
2	0.047295	0.119499	-0.112828	-0.309436	-0.467362	-0.357253	-0.205
3	-0.709200	-0.440884	-0.374145	-0.640626	-0.656504	-0.610559	-0.608
4	-1.633804	-1.561650	-1.550067	-1.280378	-1.237770	-1.296790	-1.341

10. Splitting the data into training and testing

In [39]:
```
X_Train, X_Test, Y_Train, Y_Test = train_test_split(X_Scaled, y, test_size=
```

In [40]:
```
X_Train.shape,X_Test.shape
```

Out[40]: ((3341, 8), (836, 8))

In [41]:
```
Y_Train.shape,Y_Test.shape
```

Out[41]: ((3341,), (836,))

In [42]:
```
X_Train.head()
```

Out[42]:

	Length	Diameter	Height	Whole_weight	Shucked_weight	Viscera_weight	Shell_\
3141	-2.705504	-2.580528	-1.550067	-1.634195	-1.583761	-1.596153	-1.6
3521	-2.600436	-2.580528	-2.203358	-1.617739	-1.581454	-1.577730	-1.6
883	1.140009	1.240265	0.801778	1.158289	1.073452	0.292134	1.5
3627	1.602311	1.189321	1.585727	2.185800	2.731903	2.355432	1.4
2106	0.593652	0.476107	0.409804	0.439340	0.268446	0.278317	0.9

In [43]:
```
X_Test.head()
```

Out[43]:

	Length	Diameter	Height	Whole_weight	Shucked_weight	Viscera_weight	Shell_\
668	0.215405	0.170443	0.409804	0.185291	-0.370485	0.577680	0.7
1580	-0.204870	-0.084276	-0.504803	-0.434918	-0.446603	-0.343436	-0.3
3784	0.803789	0.730826	0.409804	0.880584	0.780513	1.784341	0.5
463	-2.558408	-2.478640	-2.203358	-1.589968	-1.551468	-1.550097	-1.5
2615	1.013926	0.934602	0.932437	1.405139	1.456349	1.798157	1.0

In [44]:
```
Y_Train.head()
```

Out[44]:
```
3141    1
3521    1
883     2
3627    2
2106    2
Name: Sex, dtype: int32
```

In [45]:
```
Y_Test.head()
```

Out[45]:
```
668     2
1580    1
3784    2
463     1
2615    2
Name: Sex, dtype: int32
```

11. Building the Model

In [46]:
```
model = RandomForestClassifier(n_estimators=10,criterion='entropy')

model.fit(X_Train,Y_Train)
```

Out[46]:
```
RandomForestClassifier(criterion='entropy', n_estimators=10)
```

In [47]:
```
y_predict = model.predict(X_Test)

y_predict_train = model.predict(X_Train)
```

12. Train the Model

In [48]:
```
print('Training accuracy: ',accuracy_score(Y_Train,y_predict_train))
```

```
Training accuracy:  0.9856330439988028
```

13.Test the Model

In [49]:
```
print('Testing accuracy: ',accuracy_score(Y_Test,y_predict))
```

```
Testing accuracy:  0.5466507177033493
```

14. Measuring the performance using Metrics

In [50]:
```
pd.crosstab(Y_Test,y_predict)
```

Out[50]:

col_0	0	1	2
Sex			
0	118	38	93
1	39	216	36
2	125	48	123

In [51]:
```
print(classification_report(Y_Test,y_predict))
```

```
              precision    recall  f1-score   support

           0       0.42      0.47      0.44       249
           1       0.72      0.74      0.73       291
           2       0.49      0.42      0.45       296

    accuracy                           0.55       836
   macro avg       0.54      0.54      0.54       836
weighted avg       0.55      0.55      0.54       836
```