1.Dataset has been downloaded

In [20]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
```

2. Load the dataset into the tool

In [21]:
```
data=pd.read_csv('C:/Users/swapna/Desktop/IBM/abalone.csv')
data.head()
```

Out[21]:

	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

In [22]:
```
data.shape
```

Out[22]:
```
(4177, 9)
```

In [23]:
```
#One additional task is that, we have to add the "Age" column using "Rings" data. We
```

In [24]:
```
Age=1.5+data.Rings
data["Age"]=Age
data=data.rename(columns = {'Whole weight':'Whole_weight','Shucked weight': 'Shucked
                            'Shell weight': 'Shell_weight'})
data=data.drop(columns=["Rings"],axis=1)
data.head()
```

Out[24]:

	Sex	Length	Diameter	Height	Whole_weight	Shucked_weight	Viscera_weight	Shell_weight	A
0	M	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.150	16
1	M	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.070	8
2	F	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.210	10
3	M	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.155	1
4	I	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.055	8

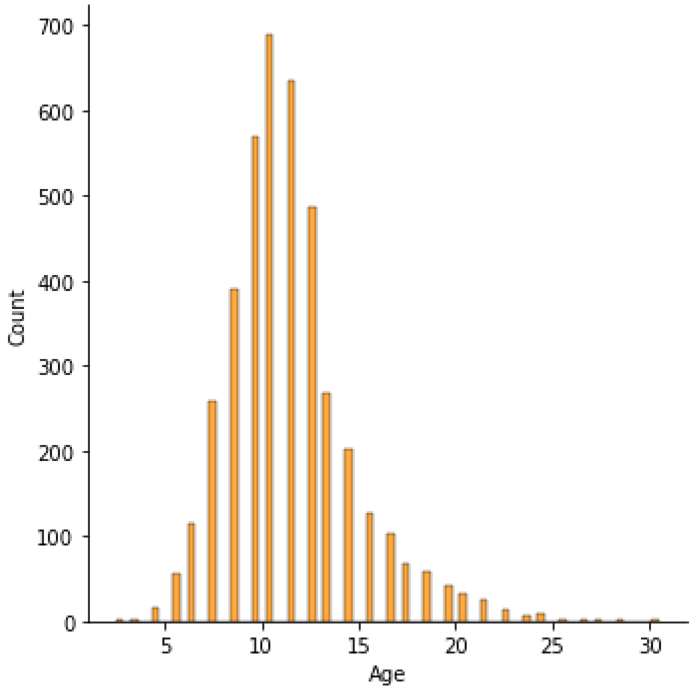
3. Perform Below Visualizations.

(i) Univariate Analysis

In [26]:
```
#The term univariate analysis refers to the analysis of one variable.
#You can remember this because the prefix “uni” means “one.”
#There are three common ways to perform univariate analysis on one variable:
#1. Summary statistics – Measures the center and spread of values.
```

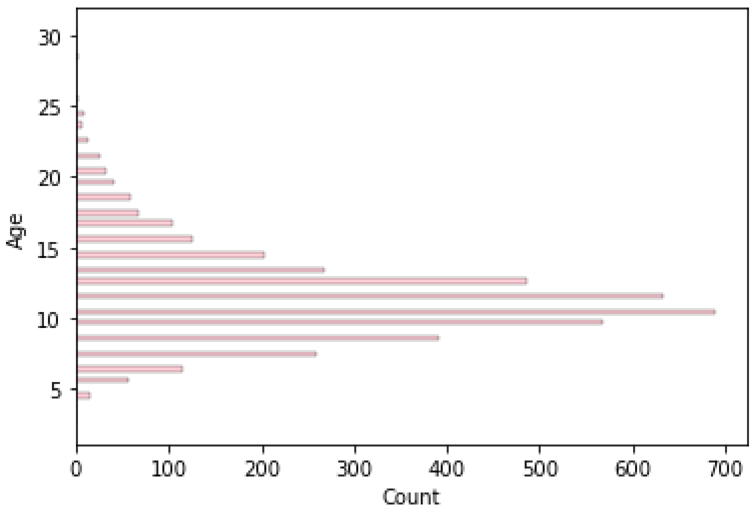
In [27]:
```
sns.displot(data["Age"], color='darkorange')
```

Out[27]: <seaborn.axisgrid.FacetGrid at 0x2b78e098310>



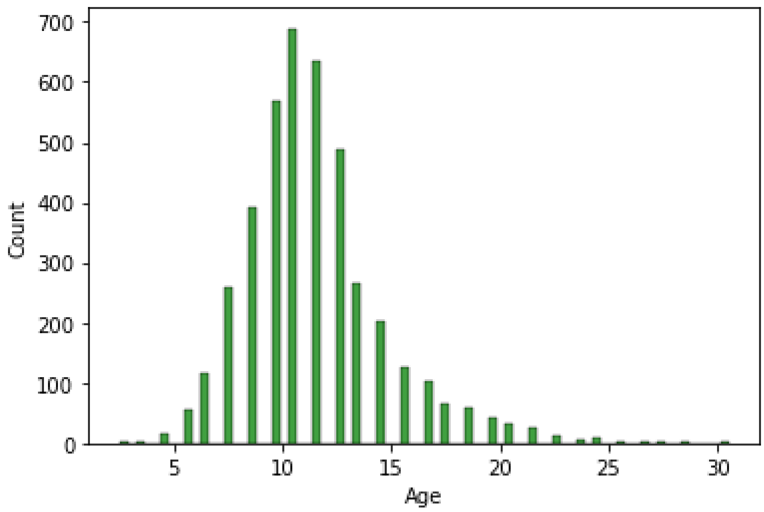
In [28]:
```
sns.histplot(y=data.Age,color='pink')
```

Out[28]: <AxesSubplot:xlabel='Count', ylabel='Age'>



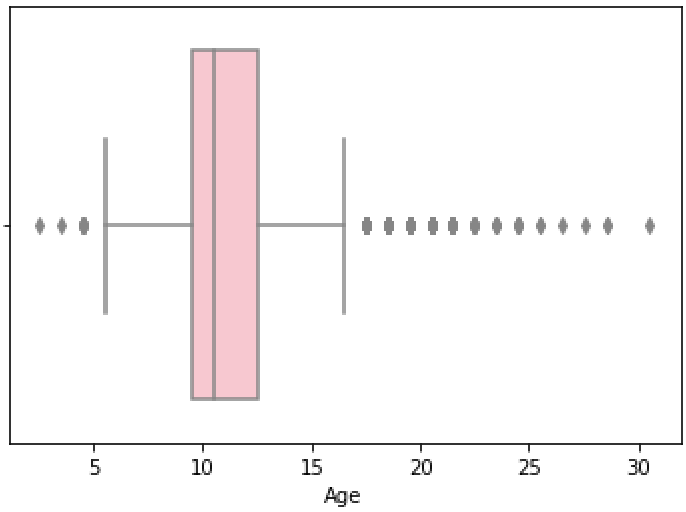
In [29]:
```
sns.histplot(x=data.Age,color='green')
```

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



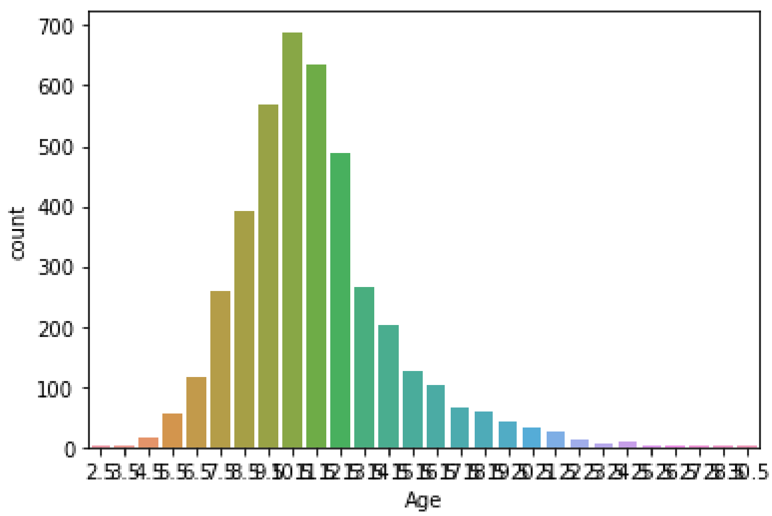
In [30]:
```
sns.boxplot(x=data.Age,color='pink')
```

Out[30]: <AxesSubplot:xlabel='Age'>



In [31]:
```
sns.countplot(x=data.Age)
```

Out[31]: <AxesSubplot:xlabel='Age', ylabel='count'>

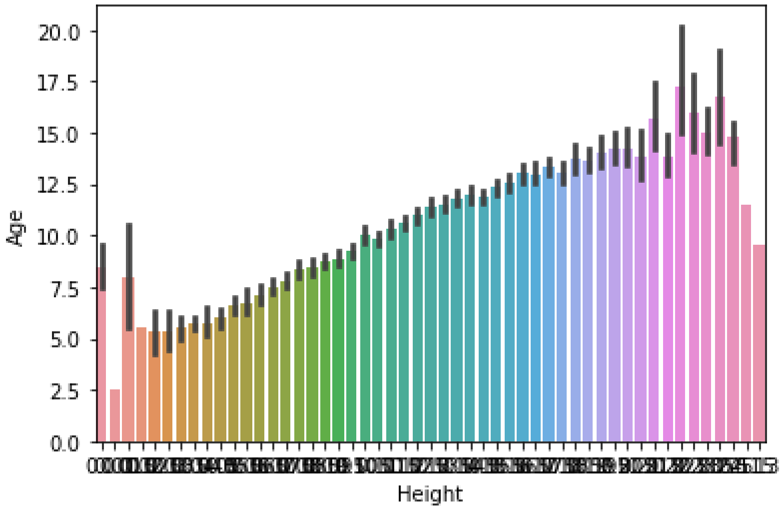


(ii) Bi-Variate Analysis

In [32]:
```
#Image result for bivariate analysis in python It is a methodical statistical techni
#(features/ attributes) of data to determine the empirical relationship between them
#n order words, it is meant to determine any concurrent relations (usually over and
```

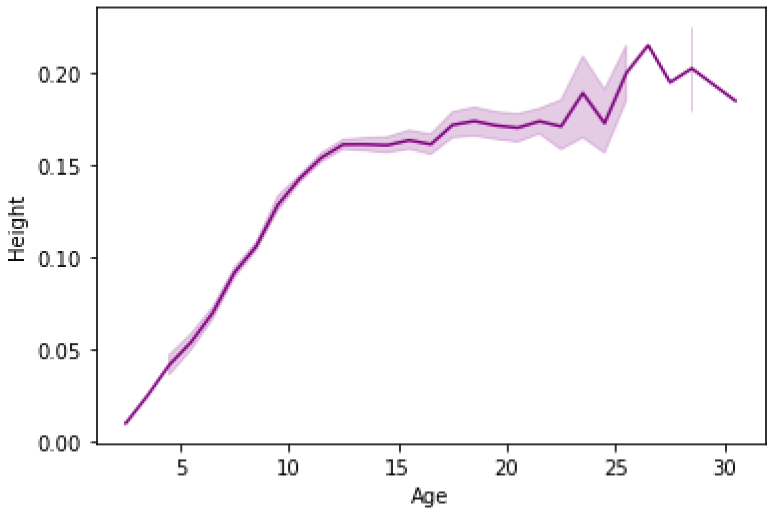
In [33]:
```
sns.barplot(x=data.Height,y=data.Age)
```

Out[33]: <AxesSubplot:xlabel='Height', ylabel='Age'>



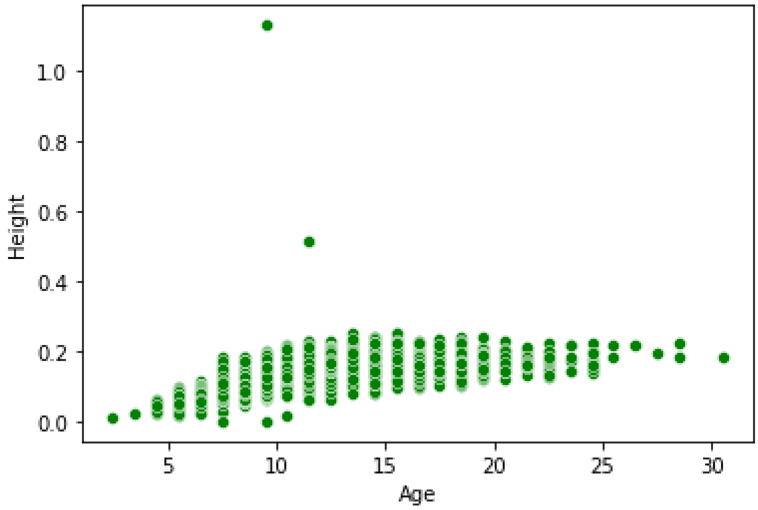
In [34]:
```
sns.lineplot(x=data.Age,y=data.Height, color='purple')
```

Out[34]: <AxesSubplot:xlabel='Age', ylabel='Height'>



In [35]:
```
sns.scatterplot(x=data.Age,y=data.Height,color='green')
```

Out[35]: <AxesSubplot:xlabel='Age', ylabel='Height'>

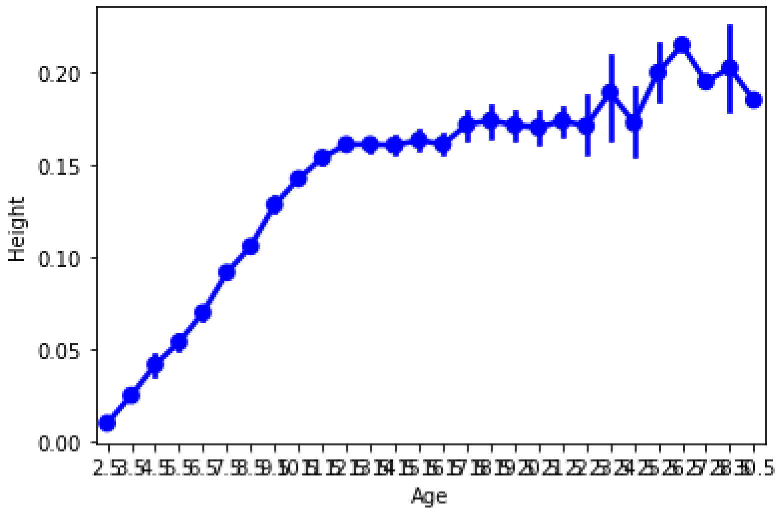


In [36]:

```
sns.pointplot(x=data.Age, y=data.Height, color="blue")
```

Out[36]:

```
<AxesSubplot:xlabel='Age', ylabel='Height'>
```

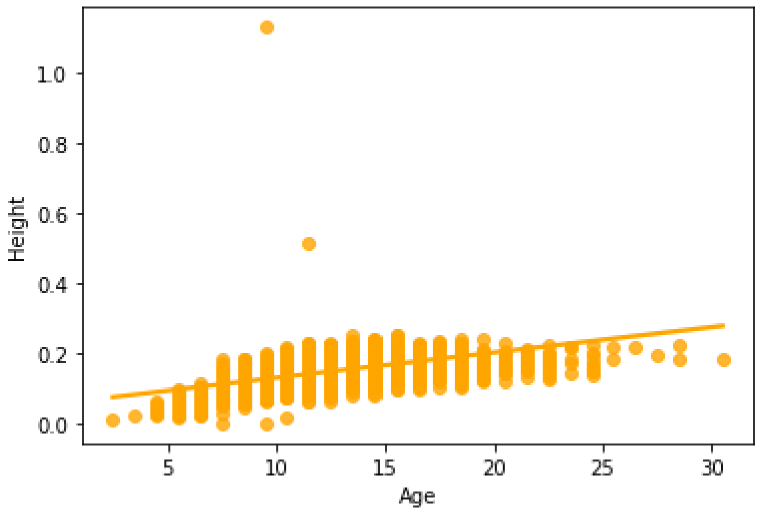


In [37]:

```
sns.regplot(x=data.Age,y=data.Height,color='orange')
```

Out[37]:

```
<AxesSubplot:xlabel='Age', ylabel='Height'>
```

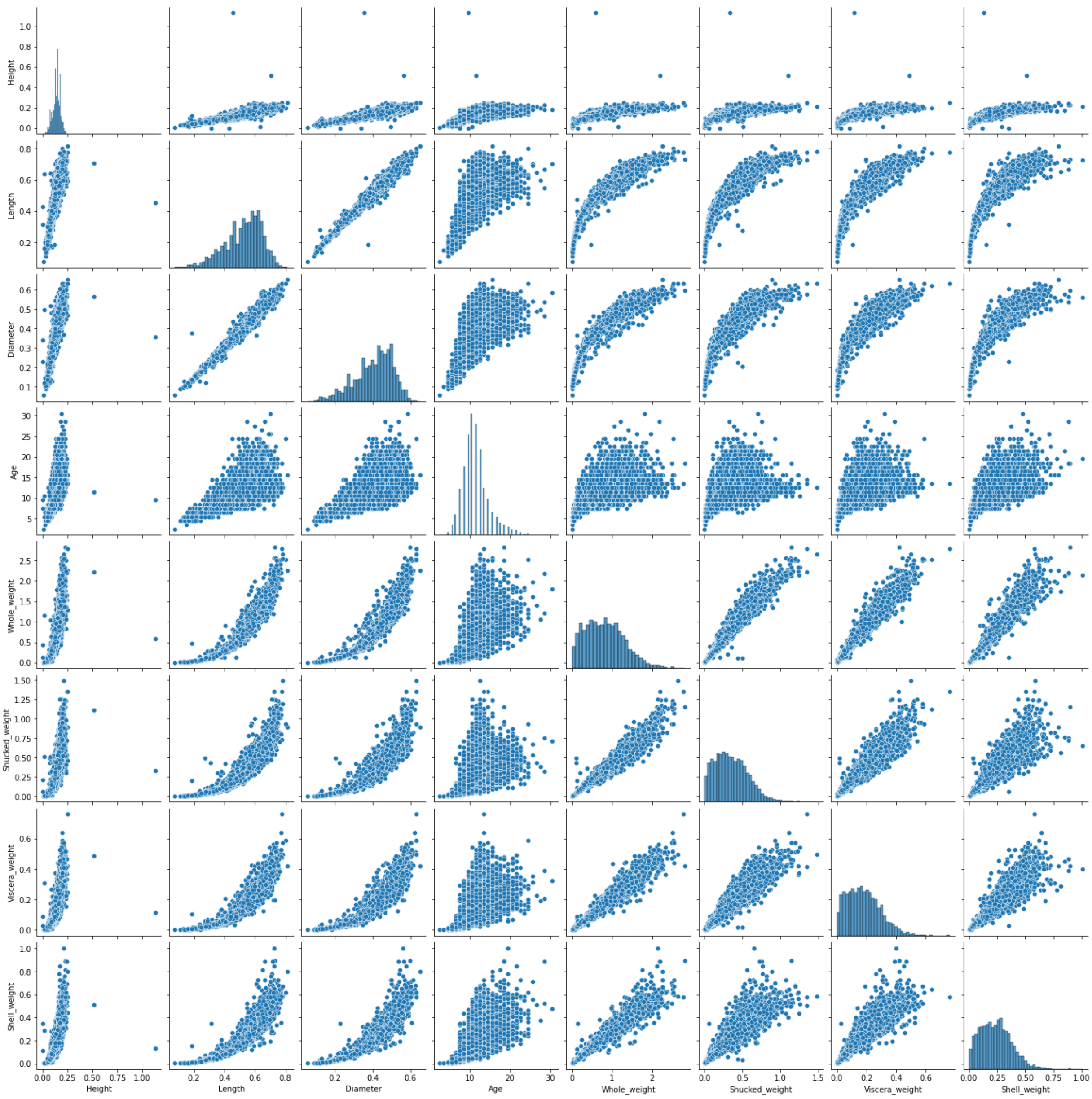


(iii) Multi-Variate Analysis

In [38]:
```
#Multivariate analysis is based in observation and analysis of more than one
#statistical outcome variable at a time. In design and analysis, the technique
#is used to perform trade studies across multiple dimensions while taking into accou
#variables on the responses of interest.
```

In [39]:
```
sns.pairplot(data=data[["Height","Length","Diameter","Age","Whole_weight","Shucked_w
```

Out[39]: <seaborn.axisgrid.PairGrid at 0x2b78f994820>



4. Perform descriptive statistics on the dataset

In [40]:
```
data.describe(include='all')
```

Out[40]:

	Sex	Length	Diameter	Height	Whole_weight	Shucked_weight	Viscera_weigh
count	4177	4177.000000	4177.000000	4177.000000	4177.000000	4177.000000	4177.00000
unique	3	NaN	NaN	NaN	NaN	NaN	Na
top	M	NaN	NaN	NaN	NaN	NaN	Na

	Sex	Length	Diameter	Height	Whole_weight	Shucked_weight	Viscera_weigh
freq	1528	NaN	NaN	NaN	NaN	NaN	NaI
mean	NaN	0.523992	0.407881	0.139516	0.828742	0.359367	0.18059
std	NaN	0.120093	0.099240	0.041827	0.490389	0.221963	0.10961
min	NaN	0.075000	0.055000	0.000000	0.002000	0.001000	0.00050
25%	NaN	0.450000	0.350000	0.115000	0.441500	0.186000	0.09350
50%	NaN	0.545000	0.425000	0.140000	0.799500	0.336000	0.17100
75%	NaN	0.615000	0.480000	0.165000	1.153000	0.502000	0.25300
max	NaN	0.815000	0.650000	1.130000	2.825500	1.488000	0.76000

5. Check for Missing values and deal with them

In [41]:
```
data.isnull().sum()
```

Out[41]:
```
Sex               0
Length            0
Diameter          0
Height            0
Whole_weight      0
Shucked_weight    0
Viscera_weight    0
Shell_weight      0
Age               0
dtype: int64
```

6. Find the outliers and replace them outliers

In [42]:
```
outliers=data.quantile(q=(0.25,0.75))
outliers
```

Out[42]:

	Length	Diameter	Height	Whole_weight	Shucked_weight	Viscera_weight	Shell_weight	Age
0.25	0.450	0.35	0.115	0.4415	0.186	0.0935	0.130	9.5
0.75	0.615	0.48	0.165	1.1530	0.502	0.2530	0.329	12.5

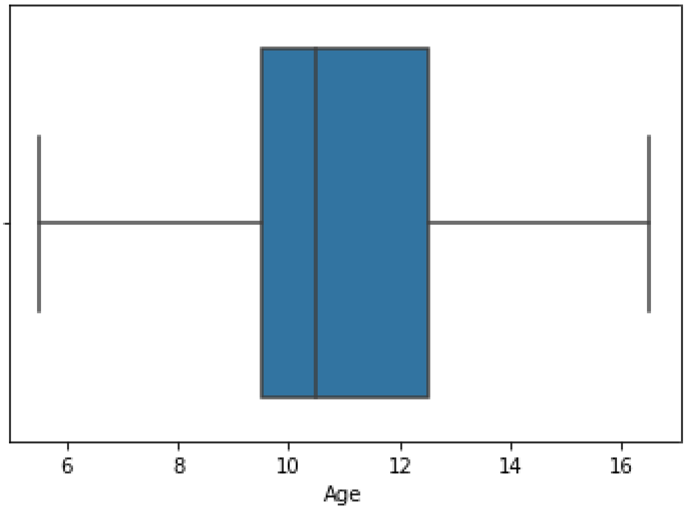
In [43]:
```
a = data.Age.quantile(0.25)
b = data.Age.quantile(0.75)
c = b - a
lower_limit = a - 1.5 * c
data.median(numeric_only=True)
```

Out[43]:
```
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
Age               10.5000
dtype: float64
```

In [44]:
```
data['Age'] = np.where(data['Age'] < lower_limit, 7, data['Age'])
sns.boxplot(x=data.Age,showfliers = False)
```

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



7. Check for Categorical columns and perform encoding

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

Out[45]:

	Sex	Length	Diameter	Height	Whole_weight	Shucked_weight	Viscera_weight	Shell_weight	A
0	M	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.150	16
1	M	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.070	8
2	F	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.210	10
3	M	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.155	11
4	I	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.055	8

In [46]:
```
from sklearn.preprocessing import LabelEncoder

lab = LabelEncoder()
data.Sex = lab.fit_transform(data.Sex)

data.head()
```

Out[46]:

	Sex	Length	Diameter	Height	Whole_weight	Shucked_weight	Viscera_weight	Shell_weight	A
0	2	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.150	16

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

8. Split the data into dependent and independent variables

In [47]:

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

Out[47]:

```
0    2
1    2
2    0
3    2
4    1
Name: Sex, dtype: int32
```

In [48]:

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

Out[48]:

	Length	Diameter	Height	Whole_weight	Shucked_weight	Viscera_weight	Shell_weight	Age
0	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.150	16.5
1	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.070	8.5
2	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.210	10.5
3	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.155	11.5
4	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.055	8.5

9. Scale the independent variables

In [49]:

```
from sklearn.preprocessing import scale
X_Scaled = pd.DataFrame(scale(x), columns=x.columns)
X_Scaled.head()
```

Out[49]:

	Length	Diameter	Height	Whole_weight	Shucked_weight	Viscera_weight	Shell_weight	
0	-0.574558	-0.432149	-1.064424	-0.641898	-0.607685	-0.726212	-0.638217	1.
1	-1.448986	-1.439929	-1.183978	-1.230277	-1.170910	-1.205221	-1.212987	-0.
2	0.050033	0.122130	-0.107991	-0.309469	-0.463500	-0.356690	-0.207139	-0.
3	-0.699476	-0.432149	-0.347099	-0.637819	-0.648238	-0.607600	-0.602294	0.
4	-1.615544	-1.540707	-1.423087	-1.272086	-1.215968	-1.287337	-1.320757	-0.

10. Split the data into training and testing

In [50]:
```
from sklearn.model_selection import train_test_split
X_Train, X_Test, Y_Train, Y_Test = train_test_split(X_Scaled, y, test_size=0.2, rand
```

In [51]:
```
X_Train.shape,X_Test.shape
```

Out[51]:
```
((3341, 8), (836, 8))
```

In [52]:
```
Y_Train.shape,Y_Test.shape
```

Out[52]:
```
((3341,), (836,))
```

In [53]:
```
X_Train.head()
```

Out[53]:

	Length	Diameter	Height	Whole_weight	Shucked_weight	Viscera_weight	Shell_weight
3141	-2.864726	-2.750043	-1.423087	-1.622870	-1.553902	-1.583867	-1.644065
3521	-2.573250	-2.598876	-2.020857	-1.606554	-1.551650	-1.565619	-1.626104
883	1.132658	1.230689	0.728888	1.145672	1.041436	0.286552	1.538726
3627	1.590691	1.180300	1.446213	2.164373	2.661269	2.330326	1.377072
2106	0.591345	0.474853	0.370226	0.432887	0.255175	0.272866	0.906479

In [54]:
```
X_Test.head()
```

Out[54]:

	Length	Diameter	Height	Whole_weight	Shucked_weight	Viscera_weight	Shell_weight
668	0.216591	0.172519	0.370226	0.181016	-0.368878	0.569396	0.690940
1580	-0.199803	-0.079426	-0.466653	-0.433875	-0.443224	-0.343004	-0.325685
3784	0.799543	0.726798	0.370226	0.870348	0.755318	1.764639	0.565209
463	-2.531611	-2.447709	-2.020857	-1.579022	-1.522362	-1.538247	-1.572219
2615	1.007740	0.928354	0.848442	1.390405	1.415417	1.778325	0.996287

In [55]:
```
Y_Train.head()
```

Out[55]:
```
3141    1
3521    1
883     2
3627    2
2106    2
Name: Sex, dtype: int32
```

In [56]:
```
Y_Test.head()
```

Out[56]:
```
668     2
1580    1
3784    2
463     1
2615    2
Name: Sex, dtype: int32
```

11. Build the Model

In [57]:
```
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier(n_estimators=10,criterion='entropy')
```

In [58]:
```
model.fit(X_Train,Y_Train)
```

Out[58]:
```
RandomForestClassifier(criterion='entropy', n_estimators=10)
```

In [59]:
```
y_predict = model.predict(X_Test)
```

In [60]:
```
y_predict_train = model.predict(X_Train)
```

12. Train the Model

In [61]:
```
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
```

In [62]:
```
print('Training accuracy: ',accuracy_score(Y_Train,y_predict_train))
```

```
Training accuracy:  0.9820413049985034
```

13.Test the Model

In [63]:
```
print('Testing accuracy: ',accuracy_score(Y_Test,y_predict))
```

```
Testing accuracy:  0.5239234449760766
```

14. Measure the performance using Metrics

In [64]:
```
pd.crosstab(Y_Test,y_predict)
```

Out[64]:

col_0	0	1	2
Sex			
0	122	31	96
1	40	211	40

col_0	0	1	2
Sex			
2	128	63	105

In [65]:

```
print(classification_report(Y_Test,y_predict))
```

```
              precision    recall  f1-score   support

           0       0.42      0.49      0.45       249
           1       0.69      0.73      0.71       291
           2       0.44      0.35      0.39       296

    accuracy                           0.52       836
   macro avg       0.52      0.52      0.52       836
weighted avg       0.52      0.52      0.52       836
```

In [ ]: