In [5]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [6]:
```python
from google.colab import files
uploaded = files.upload()
```

Choose Files No file chosen

Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving abalone.csv to abalone.csv
```

In [7]:
```python
df = pd.read_csv('abalone.csv')
```

In [8]:
```python
df.head()
```

Out[8]:

	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 [9]:
```python
#univariate statistical analysis
df['Length'].mean()
```

Out[9]:
```
0.5239920995930094
```

In [10]:
```python
df['Length'].median()
```

Out[10]:
```
0.545
```

In [11]:
```python
df['Length'].std()
```

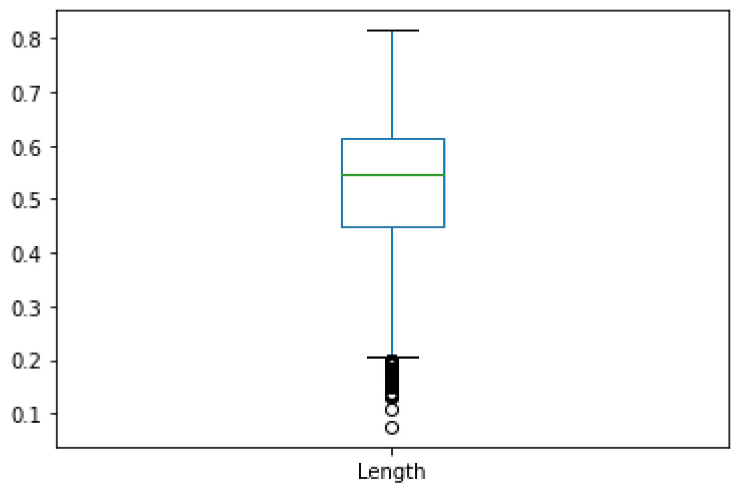
Out[11]:
```
0.12009291256479956
```

In [12]:
```
df['Length'].value_counts()
```

Out[12]:
```
0.625    94
0.550    94
0.575    93
0.580    92
0.600    87
         ..
0.075     1
0.815     1
0.110     1
0.150     1
0.800     1
Name: Length, Length: 134, dtype: int64
```

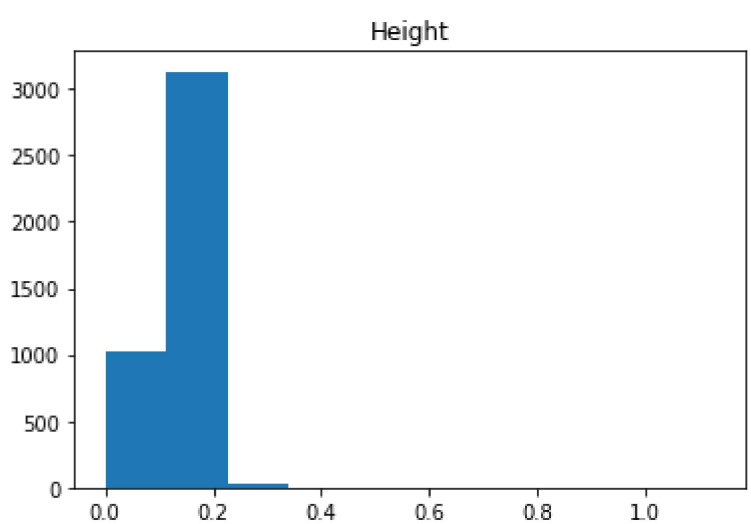
In [13]:
```
import matplotlib.pyplot as plt

df.boxplot(column=['Length'],grid=False)
```

Out[13]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc4711a9f50>
```



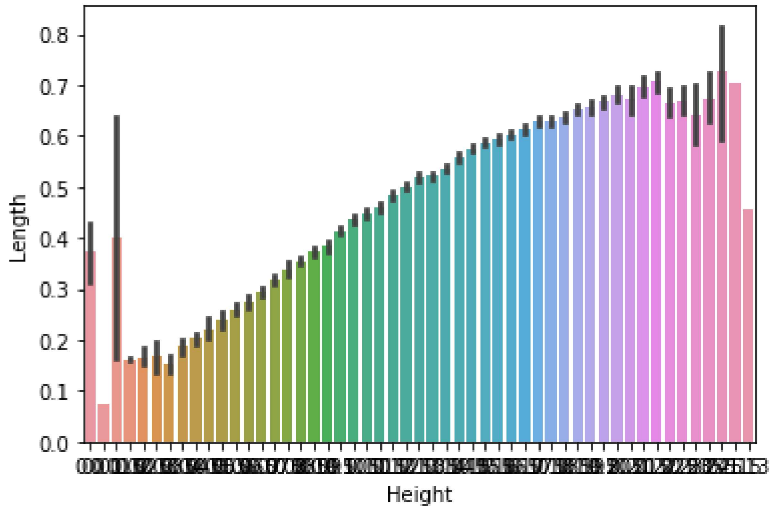
In [14]:
```
df.hist(column=['Height'],grid=False)
```

Out[14]:
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7fc473595950>]],
      dtype=object)
```



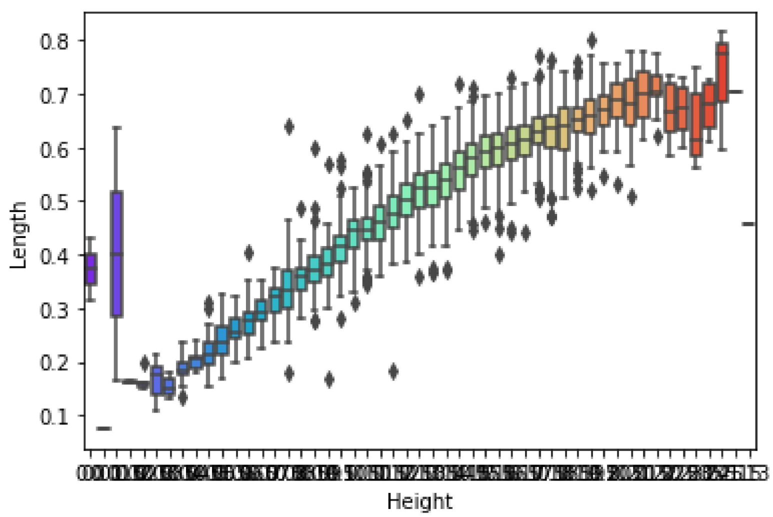
In [15]:
```
#Bivariate analysis
sns.barplot(x='Height',y='Length',data = df)
```

Out[15]: <matplotlib.axes._subplots.AxesSubplot at 0x7fc470ba6b90>



In [16]:
```
sns.boxplot(x="Height", y="Length", data=df,palette='rainbow')
```

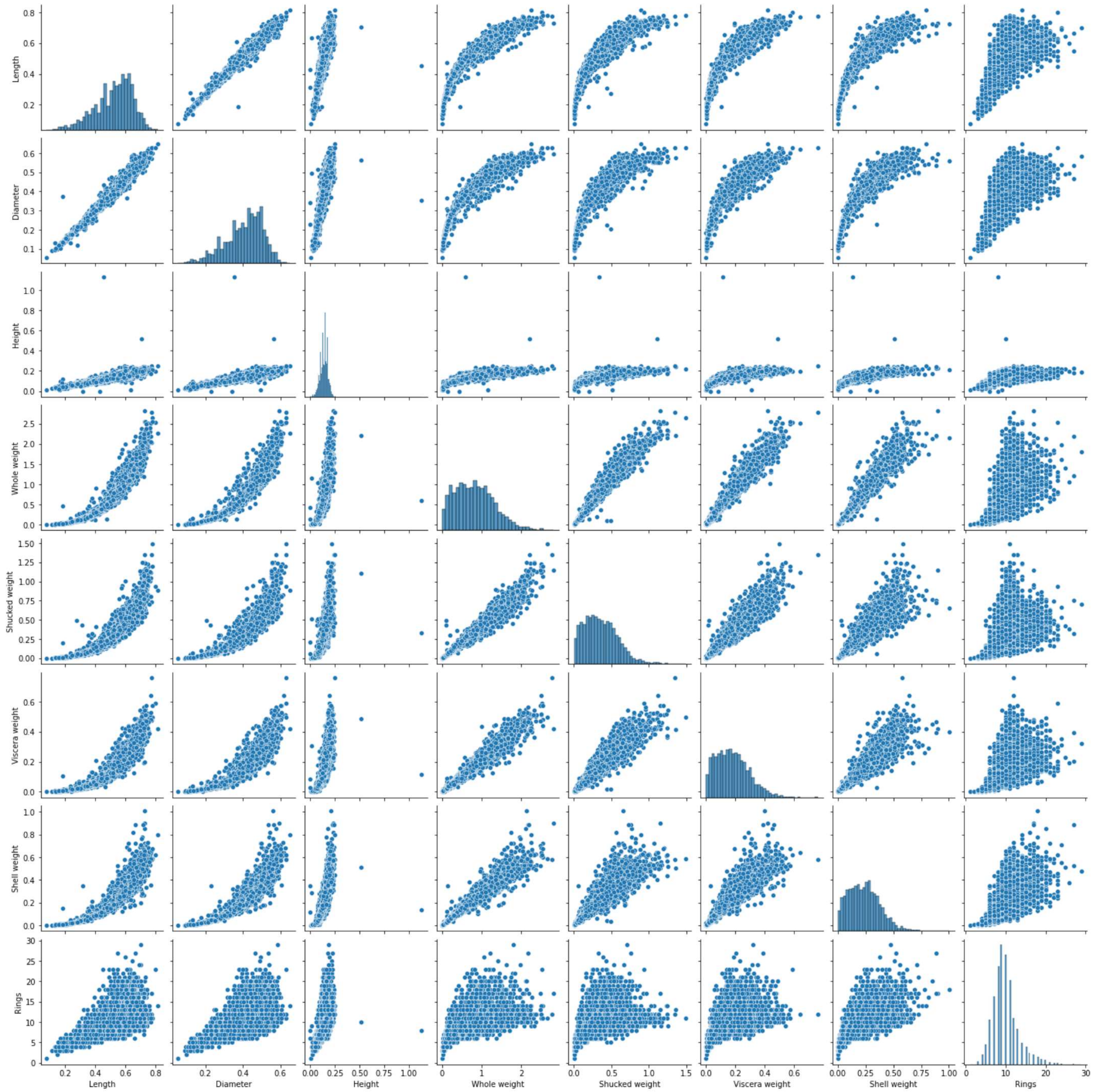
Out[16]: <matplotlib.axes._subplots.AxesSubplot at 0x7fc470912d90>



In [17]:
```
#Multivariate analysis
import seaborn as sns
```

In [18]:
```
sns.pairplot(df)
```

Out[18]:
```
<seaborn.axisgrid.PairGrid at 0x7fc47042e2d0>
```



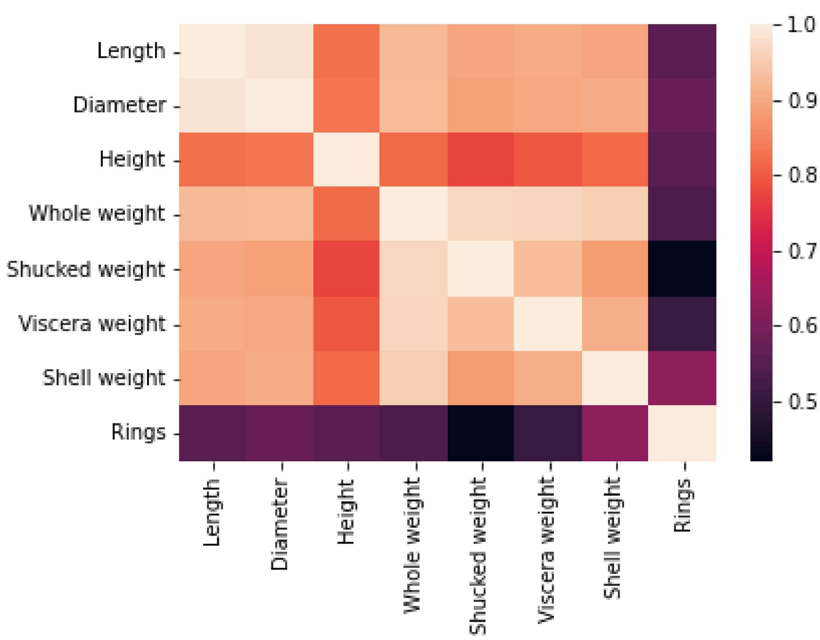
In [19]:
```
df.corr()
```

Out[19]:

	Length	Diameter	Height	Whole weight	Shucked weight	Viscera weight	Shell weight	Rings
Length	1.000000	0.986812	0.827554	0.925261	0.897914	0.903018	0.897706	0.556720
Diameter	0.986812	1.000000	0.833684	0.925452	0.893162	0.899724	0.905330	0.574660
Height	0.827554	0.833684	1.000000	0.819221	0.774972	0.798319	0.817338	0.557467
Whole weight	0.925261	0.925452	0.819221	1.000000	0.969405	0.966375	0.955355	0.540390
Shucked weight	0.897914	0.893162	0.774972	0.969405	1.000000	0.931961	0.882617	0.420884
Viscera weight	0.903018	0.899724	0.798319	0.966375	0.931961	1.000000	0.907656	0.503819
Shell weight	0.897706	0.905330	0.817338	0.955355	0.882617	0.907656	1.000000	0.627574
Rings	0.556720	0.574660	0.557467	0.540390	0.420884	0.503819	0.627574	1.000000

In [20]:
```
sns.heatmap(df.corr())
```

Out[20]: <matplotlib.axes._subplots.AxesSubplot at 0x7fc46c94dc50>

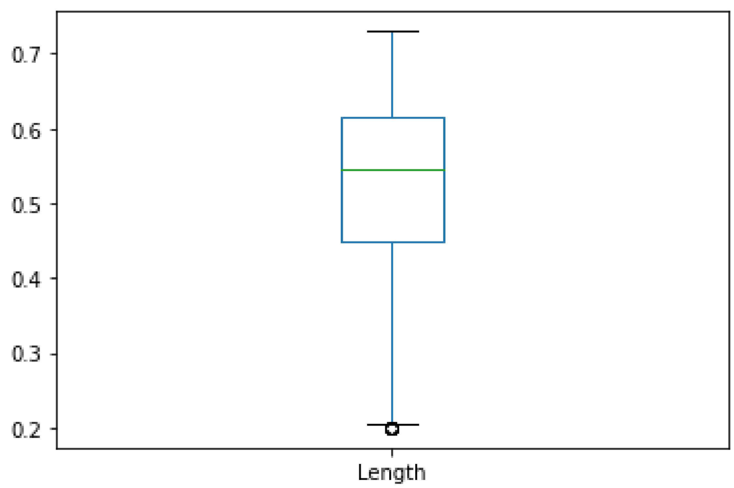


In [21]:
```
#Missing values
df.isna().sum()
```

Out[21]:
```
Sex               0
Length            0
Diameter          0
Height            0
Whole weight      0
Shucked weight    0
Viscera weight    0
Shell weight      0
Rings             0
dtype: int64
```

In [22]:
```
#Find outliers and reject
q_low = df["Length"].quantile(0.01)
q_hi  = df["Length"].quantile(0.99)

df_filtered = df[(df["Length"] < q_hi) & (df["Length"] > q_low)]
df_filtered.boxplot(column=['Length'],grid=False)
```

Out[22]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc46bc25290>
```



In [23]:
```
#Categorical values - encoding
df['Sex']
```

Out[23]:
```
0       M
1       M
2       F
3       M
4       I
       ..
4172    F
4173    M
4174    M
4175    F
4176    M
Name: Sex, Length: 4177, dtype: object
```

In [24]:
```
df['Sex'].replace({'M':0, 'F':1, 'I':2}, inplace=True)
df['Sex']
```

Out[24]:
```
0       0
1       0
2       1
3       0
4       2
       ..
4172    1
4173    0
4174    0
4175    1
4176    0
Name: Sex, Length: 4177, dtype: int64
```

In [25]:
```python
#independent and dependent variable
df["Rings"].value_counts()
```

Out[25]:
```
9     689
10    634
8     568
11    487
7     391
12    267
6     259
13    203
14    126
5     115
15    103
16     67
17     58
4      57
18     42
19     32
20     26
3      15
21     14
23      9
22      6
27      2
24      2
1       1
26      1
29      1
2       1
25      1
Name: Rings, dtype: int64
```

In [26]:
```python
#independent variables
X = df.iloc[:, :-1].values
X
```

Out[26]:
```
array([[0.    , 0.455 , 0.365 , ..., 0.2245, 0.101 , 0.15  ],
       [0.    , 0.35  , 0.265 , ..., 0.0995, 0.0485, 0.07  ],
       [1.    , 0.53  , 0.42  , ..., 0.2565, 0.1415, 0.21  ],
       ...,
       [0.    , 0.6   , 0.475 , ..., 0.5255, 0.2875, 0.308 ],
       [1.    , 0.625 , 0.485 , ..., 0.531 , 0.261 , 0.296 ],
       [0.    , 0.71  , 0.555 , ..., 0.9455, 0.3765, 0.495 ]])
```

In [27]:
```python
#dependent variables
y = df.iloc[:, -1].values
print(y)
```

```
[15  7  9 ...  9 10 12]
```

In [28]:
```python
#train test split
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4)
print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
```

```
(2506, 8) (1671, 8) (2506,) (1671,)
```

In [29]:
```
# Build the model
from sklearn.linear_model import LinearRegression
from sklearn import  metrics
import numpy as np
linear = LinearRegression()
```

In [30]:
```
# Train the model
linear.fit(X_train, y_train)
```

Out[30]:
```
LinearRegression()
```

In [31]:
```
# Test the model
y_pred = linear.predict(X_test)
```

In [32]:
```
# Measure the metrics
print('Mean Absolute Error :',metrics.mean_absolute_error(y_test, y_pred))
print('Mean Squared Error :',metrics.mean_squared_error(y_test, y_pred))
print('RMSE :',np.sqrt(metrics.mean_absolute_error(y_test, y_pred)))
print('R2 Score :',metrics.r2_score(y_test, y_pred))
```

```
Mean Absolute Error : 1.6002436298763874
Mean Squared Error : 4.914524375073977
RMSE : 1.2650073635660732
R2 Score : 0.5248938529449565
```