Part 1: The purpose of the dataset is to predict the age of an abalone through physical characteristics, determined by cutting through the cone, staining it and counting the no. of rings through a microscope. The dataset description from the UCI machine learning repository is as follows:

Name	Data Type	Measure	Description

Sex	nominal		M, F, and I (infant)
Length	continuous	mm	Longest shell measurement
Diameter	continuous	mm	perpendicular to length
Height	continuous	mm	with meat in shell
Whole weight	continuous	grams	whole abalone
Shucked weight	continuous	grams	weight of meat
Viscera weight	continuous	grams	gut weight (after bleeding)
Shell weight	continuous	grams	after being dried
Rings	integer		+1.5 gives the age in years

Run the cells below to examine the dataset.

In [1]:

```
# Load abalone dataset
import csv
path = r"D:\TRL\OneDrive\CourseraPython\MiniProject_Course1\BDPV - Mini Project\abalone
f = open(path,'rt',encoding = 'utf8')
all_lines = csv.reader(f,delimiter = ",")
```

In [2]:

```
# we define the header ourselves as the dataset contains only raw nos.
dataset = []
header = ['Sex', 'Length','Diameter','Height','Whole Weight','Shucked Weight',
          'Viscera Weight', 'Shell Weight','Rings']

for line in all_lines:
    d = dict(zip(header,line))
    d['Length'] = float(d['Length'])
    d['Diameter'] = float(d['Diameter'])
    d['Height'] = float(d['Height'])
    d['Whole Weight'] = float(d['Whole Weight'])
    d['Shucked Weight'] = float(d['Shucked Weight'])
    d['Viscera Weight'] = float(d['Viscera Weight'])
    d['Shell Weight'] = float(d['Shell Weight'])
    d['Rings'] = int(d['Rings'])
    dataset.append(d)

dataset[0]
#type(dataset[0]['Length']), type(d['Diameter'])

```

Out[2]:

```
{'Sex': 'M',
 'Length': 0.455,
 'Diameter': 0.365,
 'Height': 0.095,
 'Whole Weight': 0.514,
 'Shucked Weight': 0.2245,
 'Viscera Weight': 0.101,
 'Shell Weight': 0.15,
 'Rings': 15}
```

In [3]:

```
d
```

Out[3]:

```
{'Sex': 'M',
 'Length': 0.71,
 'Diameter': 0.555,
 'Height': 0.195,
 'Whole Weight': 1.9485,
 'Shucked Weight': 0.9455,
 'Viscera Weight': 0.3765,
 'Shell Weight': 0.495,
 'Rings': 12}
```

Part 2: Simple Statistics

This dataset is already cleaned for us and relatively straightforward, without strings or time data. In your final project you will have to take care of missing or tricky values yourself.

Fill in the following cells with the requested information about the dataset. The answers are given so you can check the output of your own code. For floating numbers, don't worry too much about the exact numbers as long as they are quite close - different systems may have different rounding protocols.

'import numpy' if you want more practise with it, or just use Python's native structures to play around with the numbers.

Question: What is the total number of entries in the dataset?

In [4]:

```
1 len(dataset)
```

Out[4]:

```
4177
```

Question: What is the average length of an abalone?

In [5]:

```
1  lengths = [d['Length'] for d in dataset]
2  type(lengths)
3  type(lengths[0])
4
5  import statistics
6  s = statistics.fmean(lengths)
7
8  avg = sum(lengths)/len(lengths)
9  s,avg
10
11 import numpy
12 numpy.mean(lengths), type(lengths) # this creates a list from a dictionary object
```

Out[5]:

```
(0.5239920995930094, list)
```

Question: What is the widest abalone in the dataset (diameter)?

In [6]:

```
1 a = []
2 for d in dataset:
3     a.append(d['Diameter'])
4 max(a)
```

Out[6]:

```
0.65
```

Question: What is the average number of rings of smaller abalones compared to that of larger ones?

That is, do smaller abalones tend to be younger or older than larger abalones? We will count small abalones as abalones with lengths less than or equal to the average length of an abalone. THe average length of an abalone is 0.524

In [7]:

```
small = [d['Rings'] for d in dataset if d['Length'] < 0.524]
len(small), numpy.mean(small), statistics.fmean(small)
```

Out[7]:

```
(1828, 8.315645514223196, 8.315645514223196)
```

In [8]:

```
large = [d['Rings'] for d in dataset if d['Length'] > 0.524]
len(large), numpy.mean(large)
```

Out[8]:

```
(2349, 11.192848020434228)
```

Part 3: Data Visualizations

In this course we learned about [Matplotlib], a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. There are a variety of plots and figures we can make with Matplotlib, and in conjunction with NumPy, becomes a powerful and versatile tool in your skillset.

We covered the basics of line plots, histograms, scatter plots, bar plots and box plots. Let's try a few below:

In [9]:

```
import matplotlib.pyplot as plt
from matplotlib import colors
import numpy
from collections import defaultdict
```

Line Plots

Line plots show the change in data over time. The example Line Plot below plots the change in density as abalones age (i.e. the distribution of rings). Note that a line plot is not necessarily the best way to show this data since it doesn't deal with a trend! Use histogram in the next step to better showcase this data.

In [10]:

```
# Parse out Rings column from dataset
rings = [d['Rings'] for d in dataset]
rings.sort()
rings[-10], rings[0], rings[10], rings[4000],len(rings),rings[4176]
```

Out[10]:

```
(23, 1, 3, 17, 4177, 29)
```

Question: Count the number of abalones with each number of rings with

defaultdict

In [11]:

```
abalone_rings = defaultdict(int)
for r in rings:
    abalone_rings[r] += 1

X = list(abalone_rings.keys())
Y = list(abalone_rings.values())
len(X),len(Y)
```

Out[11]:

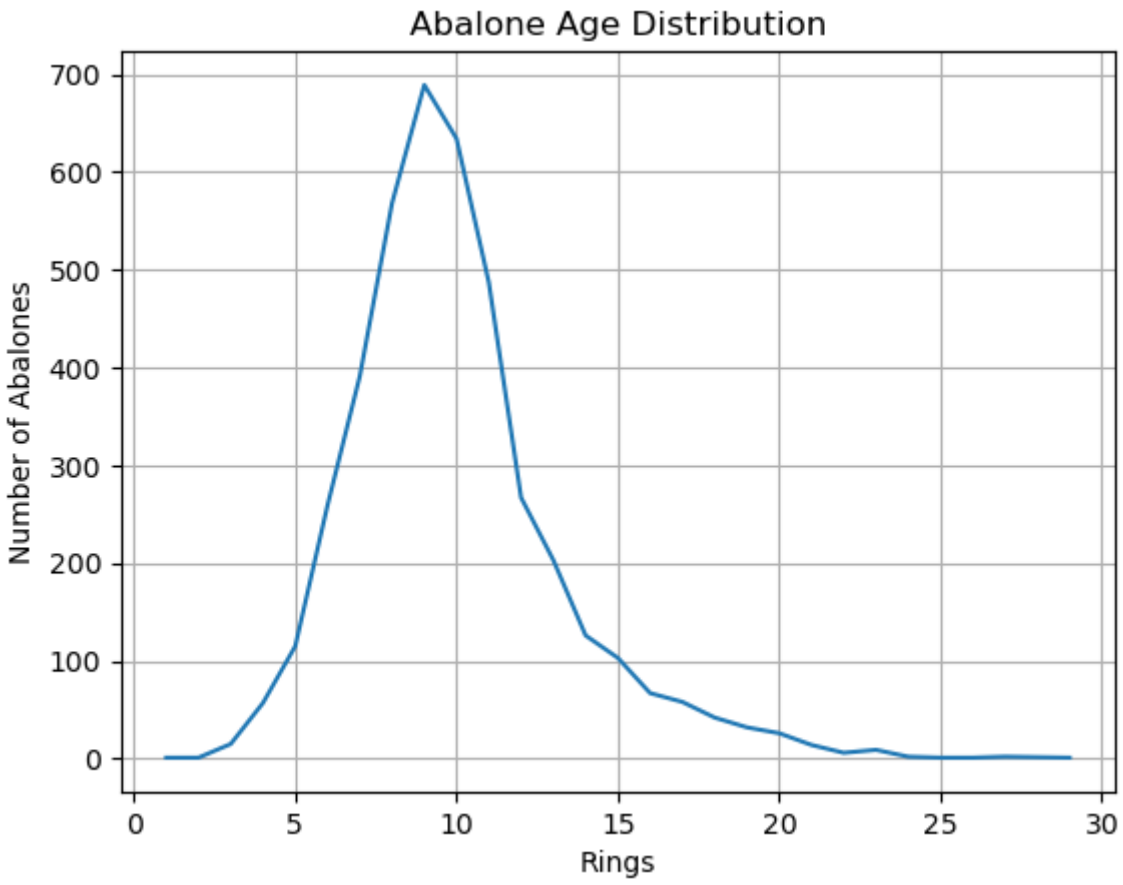
```
(28, 28)
```

In [12]:

```
### Customize Plot:
plt.gca().set(xlabel = 'Rings', ylabel = 'Number of Abalones', title = 'Abalone Age Dis
plt.grid()

# Show the plot of Rings vs Number of Abalones
plt.plot(X,Y)
```

Out[12]:

```
[<matplotlib.lines.Line2D at 0x1d50d42b970>]
```



Histograms:

Histograms show the distribution of numeric continuous variables with central tendency and skewness. Using the line plot from above, plot a histogram showing the distribution of abalone age. Explore matplotlib to customize your histogram and the following visualizations.

In [13]:

```
# Histogram of abalone age distribution
# Flatten distribution list into frequency distribution.
age_freq = []
for key in abalone_rings.keys():
    for i in range(0,abalone_rings.get(key)):
        age_freq.append(key)
print(age_freq[:10])
# dictionary has the key,value pairs of the 28 keys,
# each key has the corresponding value as the frequency,
# here we add each value to a list, freq number of times, so
# that we can easily plot a histogram
```

```
[1, 2, 3, 3, 3, 3, 3, 3, 3, 3]
```

In [14]:

```
abalone_rings
```

Out[14]:

```
defaultdict(int,
            {1: 1,
             2: 1,
             3: 15,
             4: 57,
             5: 115,
             6: 259,
             7: 391,
             8: 568,
             9: 689,
             10: 634,
             11: 487,
             12: 267,
             13: 203,
             14: 126,
             15: 103,
             16: 67,
             17: 58,
             18: 42,
             19: 32,
             20: 26,
             21: 14,
             22: 6,
             23: 9,
             24: 2,
             25: 1,
             26: 1,
             27: 2,
             29: 1})
```

In [15]:

```
1 abalone_rings.keys()
```

Out[15]:

```
dict_keys([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 1
9, 20, 21, 22, 23, 24, 25, 26, 27, 29])
```

In [16]:

```
1 len(abalone_rings.keys()), len(abalone_rings), len(age_freq)
```

Out[16]:

```
(28, 28, 4177)
```

In [17]:

```
1 abalone_rings.get(9)
```

Out[17]:

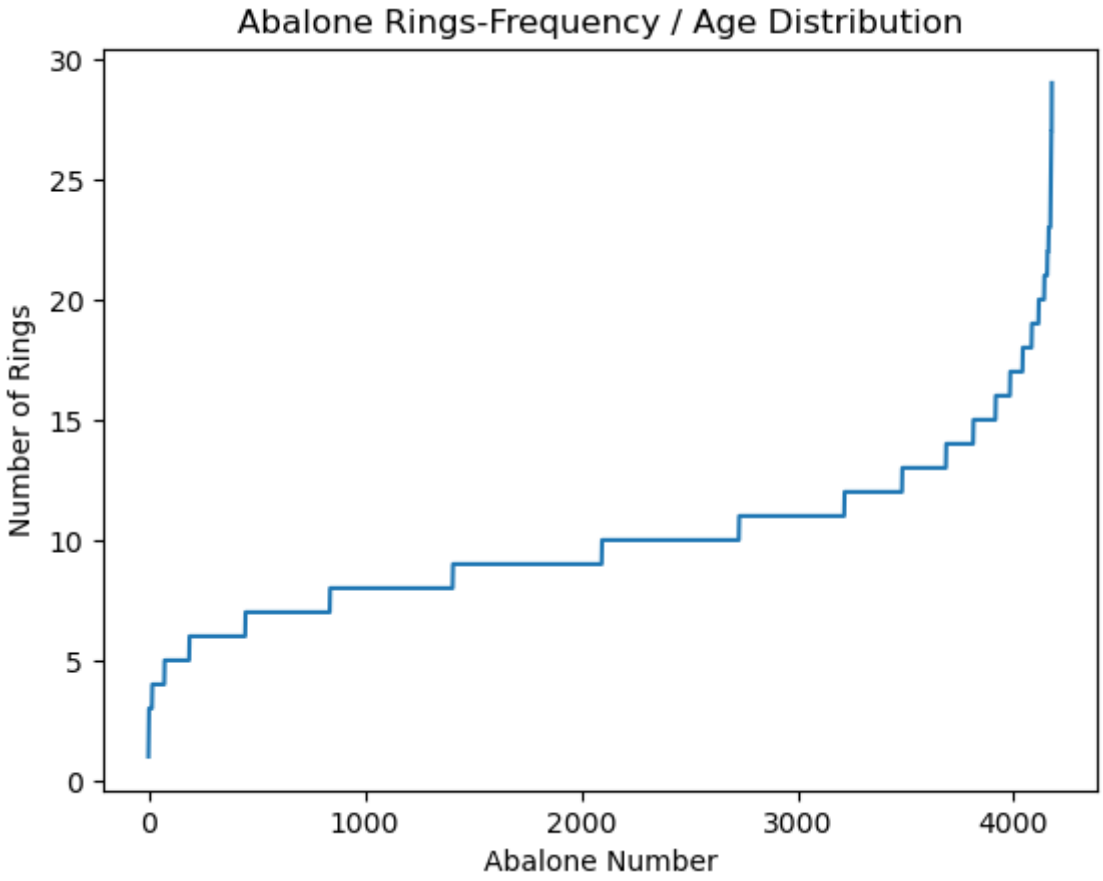
```
689
```

In [18]:

```
# Plot your histogram here
X = list(abalone_rings.keys())
Y = age_freq
age_freq[-1], type(X), type(Y)
plt.gca().set(xlabel = 'Abalone Number', ylabel = 'Number of Rings', title = 'Abalone F
plt.plot(Y)
```

Out[18]:

```
[<matplotlib.lines.Line2D at 0x1d50db1dd30>]
```

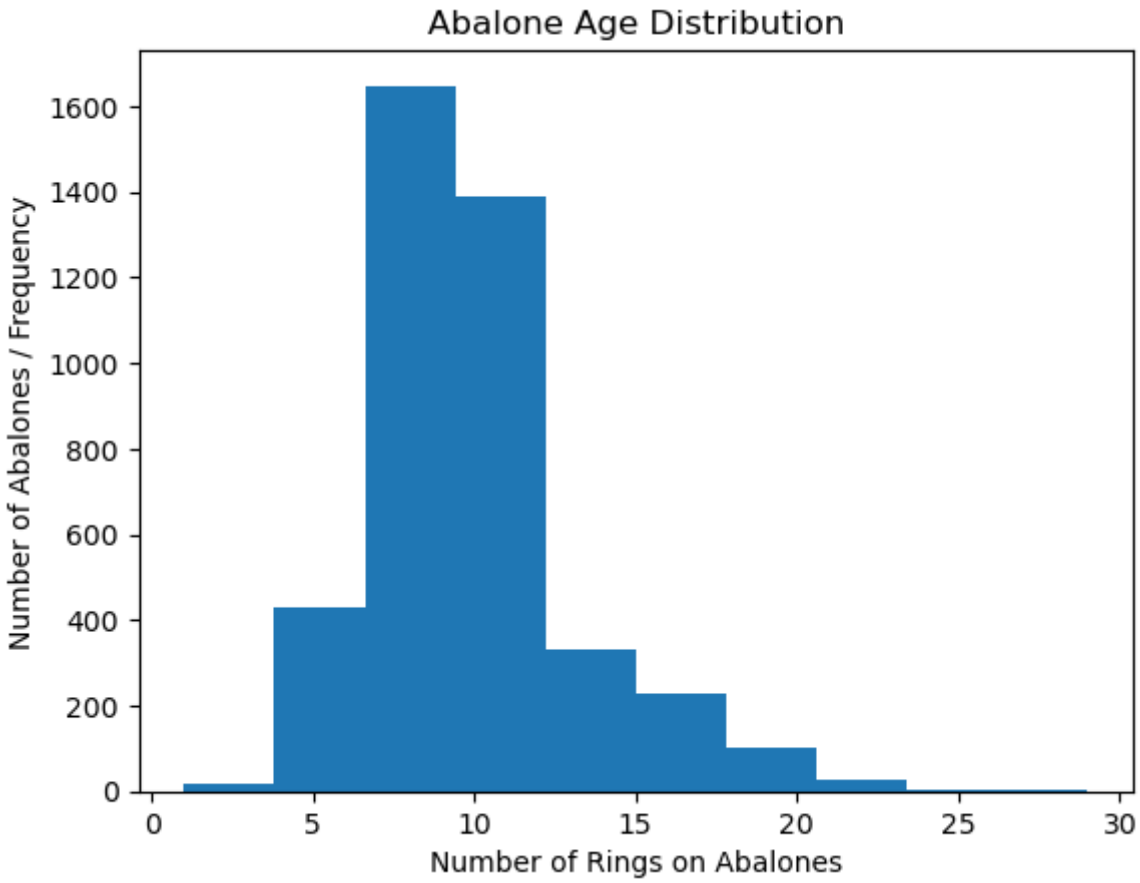


In [19]:

```
plt.gca().set(xlabel = 'Number of Rings on Abalones', ylabel = 'Number of Abalones / Fr
plt.hist(Y)
# From the histogram, we can conclude that the age
# of abalones in the sample is relatively less.
# The mode of (number of rings) the sample is about 10, indicating
# a relatively young population
```

Out[19]:

```
(array([  17.,  431., 1648., 1388.,  329.,  228.,  100.,   29.,    4.,
          3.]),
 array([ 1. ,  3.8,  6.6,  9.4, 12.2, 15. , 17.8, 20.6, 23.4, 26.2, 29. ]),
 <BarContainer object of 10 artists>)
```



Scatter Plots

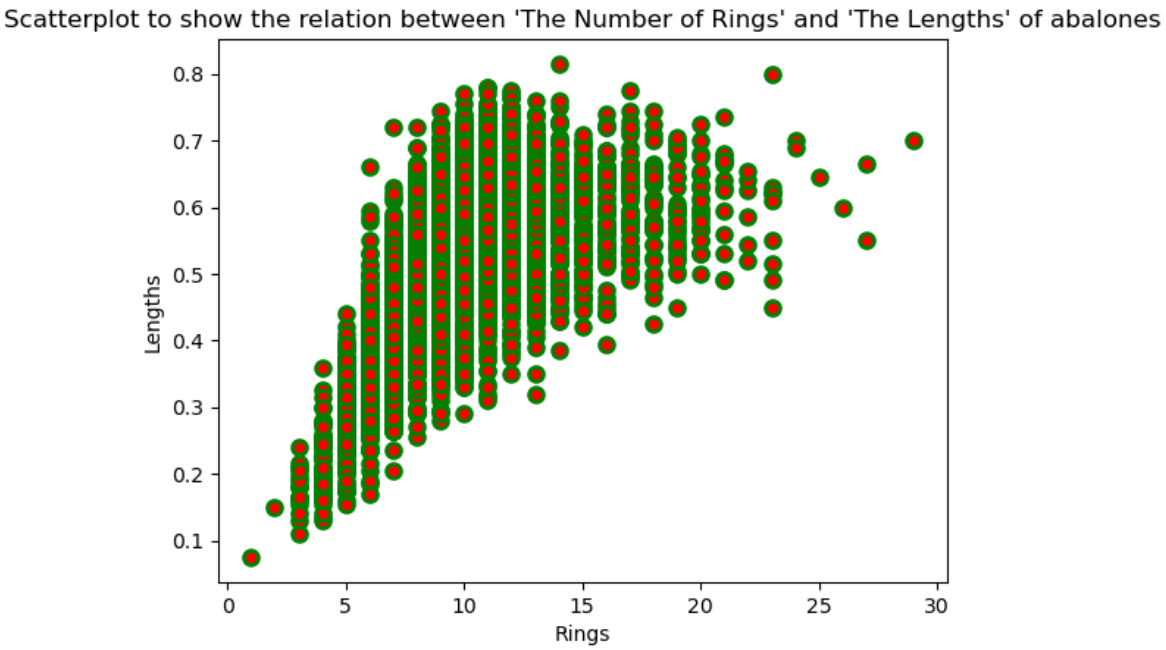
Scatter Plots show the strength of a relationship between two variables (also known as correlation). From Part:2 Simple Statistics, we see that larger abalones tend to be larger, at least from a numbers perspective. Let's see if this is actually true by creating a scatter plot showing the relationship between 'Rings' and 'Length'.

In [20]:

```
# Scatter plot of age vs length
rings = [d['Rings'] for d in dataset]
length = [d['Length'] for d in dataset]
plt.scatter(rings,length, c = 'red', linewidths =2, edgecolor = 'green', s = 50 )
# linewidths = width of marker border, edgecolor = marker color value.
# s = marker size.
plt.xlabel("Rings")
plt.ylabel("Lengths")
# From the graph note that: abalones with lesser rings have smaller lengths.
plt.title("Scatterplot to show the relation between 'The Number of Rings' and 'The Leng
```

Out[20]:

```
Text(0.5, 1.0, "Scatterplot to show the relation between 'The Number of Ring
s' and 'The Lengths' of abalones")
```



In [22]:

```
# To generate proper input to use the seaborn package in python
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd

df1 = pd.DataFrame(rings, columns = ['Rings'])
df2 = pd.DataFrame(length, columns = ['Lengths'])
#concatenated = pd.concat([rings.assign(dataset='rings'), lengths.assign(dataset = 'ler
df2
df = pd.concat([df1,df2], axis = 1) # gives a propoer dataframe, we need two dfs to cor
df
```

Out[22]:

	Rings	Lengths
0	15	0.455
1	7	0.350
2	9	0.530
3	10	0.440
4	7	0.330
...	...	...
4172	11	0.565
4173	10	0.590
4174	9	0.600
4175	10	0.625
4176	12	0.710

4177 rows × 2 columns

In [23]:

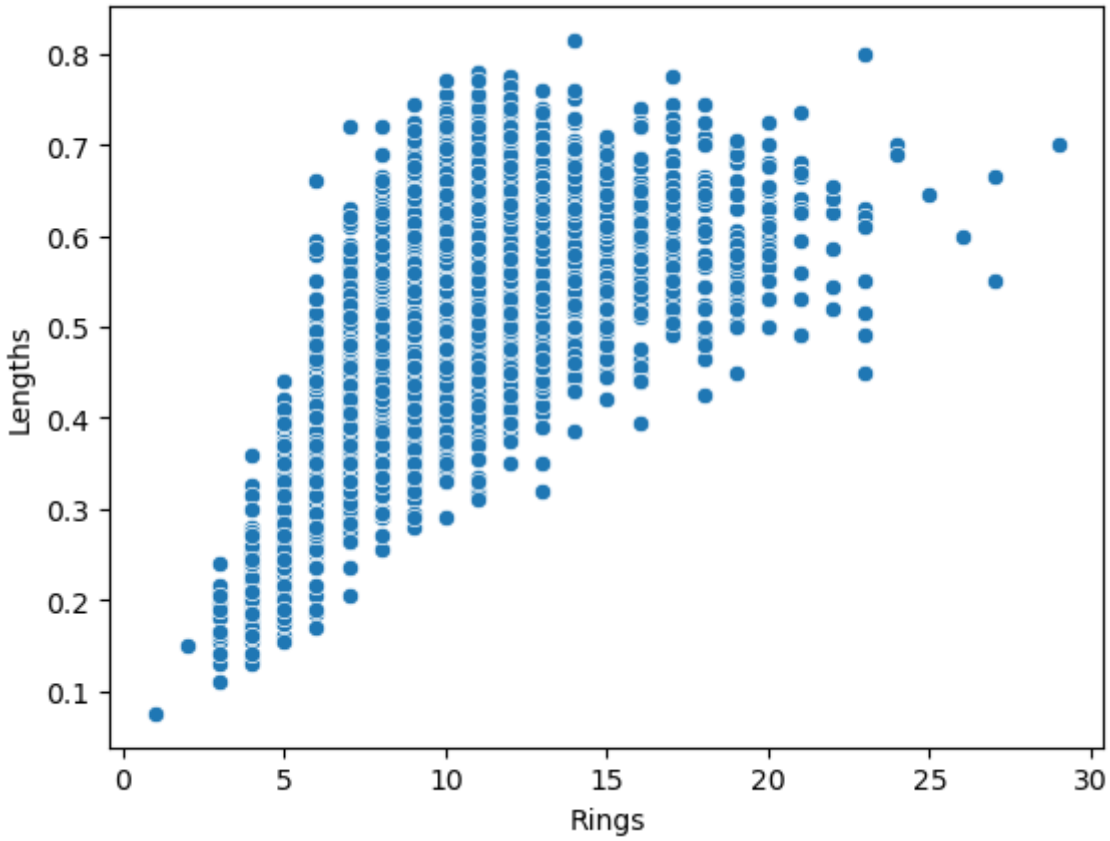
```
data1 = df
```

In [24]:

```
# how do we concatenate two lists to form a single list with two columns ?
import seaborn as sns
sns.scatterplot(x= "Rings", y = "Lengths", data = data1 )

```

Out[24]:

```
<AxesSubplot:xlabel='Rings', ylabel='Lengths'>
```

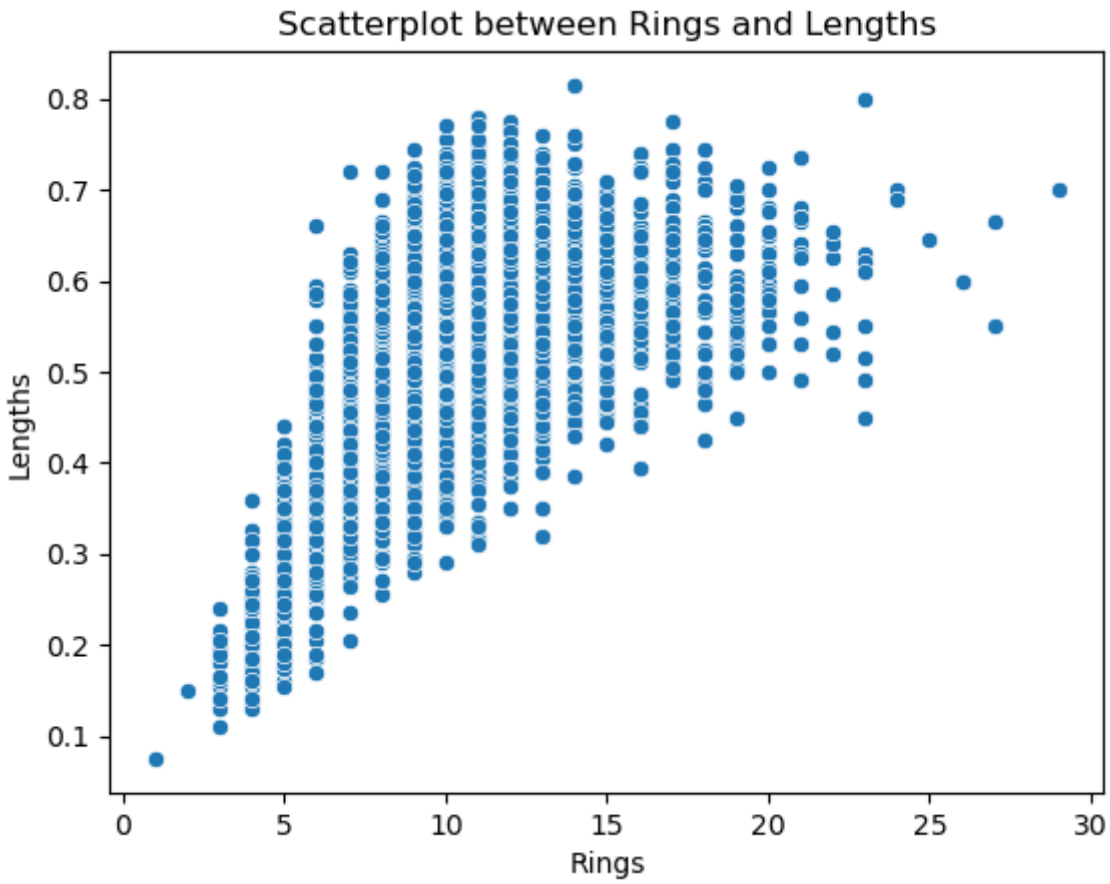


In [25]:

```
# now add the labels:
p = sns.scatterplot(x= "Rings", y = "Lengths", data = data1 )
p.set_title("Scatterplot between Rings and Lengths")
#p.set_xlabel("Rings") # this is correct.
```

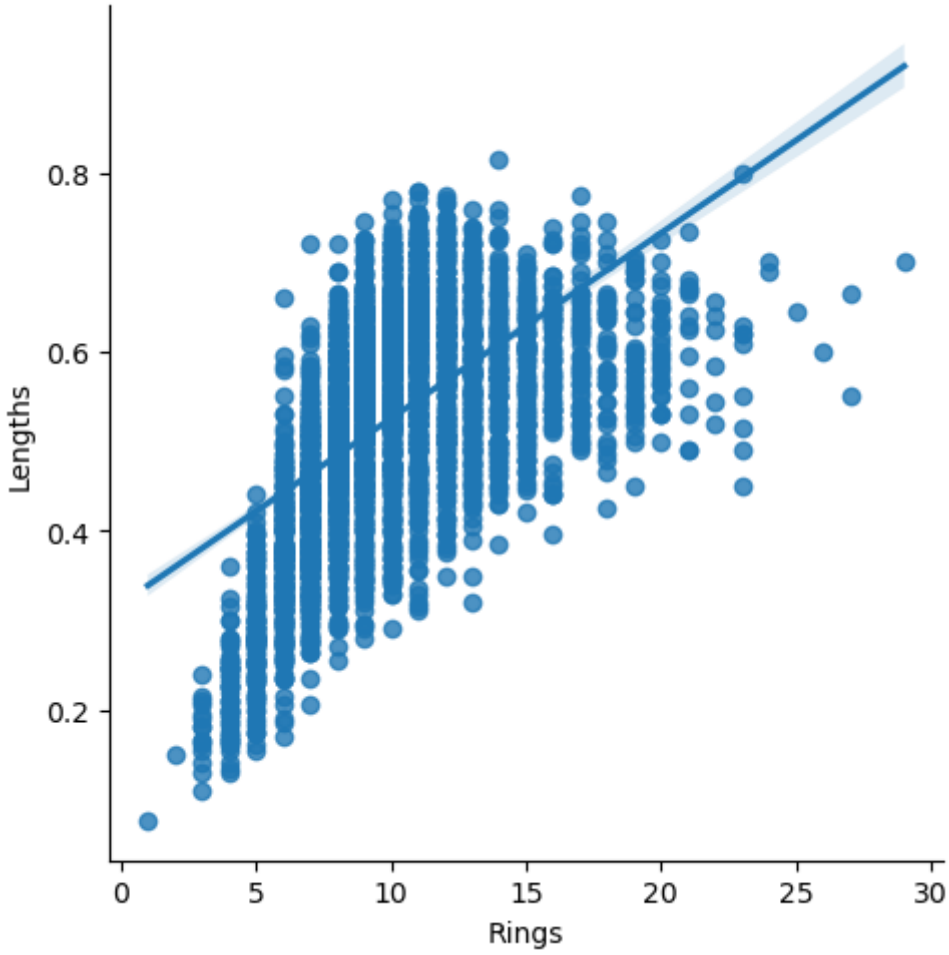
Out[25]:

```
Text(0.5, 1.0, 'Scatterplot between Rings and Lengths')
```



In [26]:

```
# Add the line of best fit:
sns.lmplot(x = "Rings", y = "Lengths", data = data1);
#sns.lmplot(x = "Rings", y = "Lengths", hue ="AirEntrain", data = data1)
# hue is color in seaborn package to add a third dimension to a
# two - dimensional scatterplot.

```



In [27]:

```
# To see how we can divide this dataset into small and large rings
# and plot two regression lines, maybe this can better estimate the length?

```

In [28]:

```
category = []
avg = numpy.mean(rings)
for i in range(len(rings)):
    if rings[i] > avg:
        category.append(1) # category[i] =1 does not work here
    else :
        category.append(0)

```

In [29]:

```
type(category), len(category), avg, len(rings)
```

Out[29]:

```
(list, 4177, 9.933684462532918, 4177)
```

In [30]:

```
len(category)
```

Out[30]:

```
4177
```

In [31]:

```
category[1:10]
```

Out[31]:

```
[0, 0, 1, 0, 0, 1, 1, 0, 1]
```

In [32]:

```
df3 = pd.DataFrame(category, columns = ['Category'])
# the category is 1 for small and 0 for large. Use this to generate two reg lines on a
dfnew = pd.concat([df,df3], axis = 1) # gives a propoer dataframe, we need two dfs to c
dfnew
```

Out[32]:

	Rings	Lengths	Category
0	15	0.455	1
1	7	0.350	0
2	9	0.530	0
3	10	0.440	1
4	7	0.330	0
...	...	...	...
4172	11	0.565	1
4173	10	0.590	1
4174	9	0.600	0
4175	10	0.625	1
4176	12	0.710	1

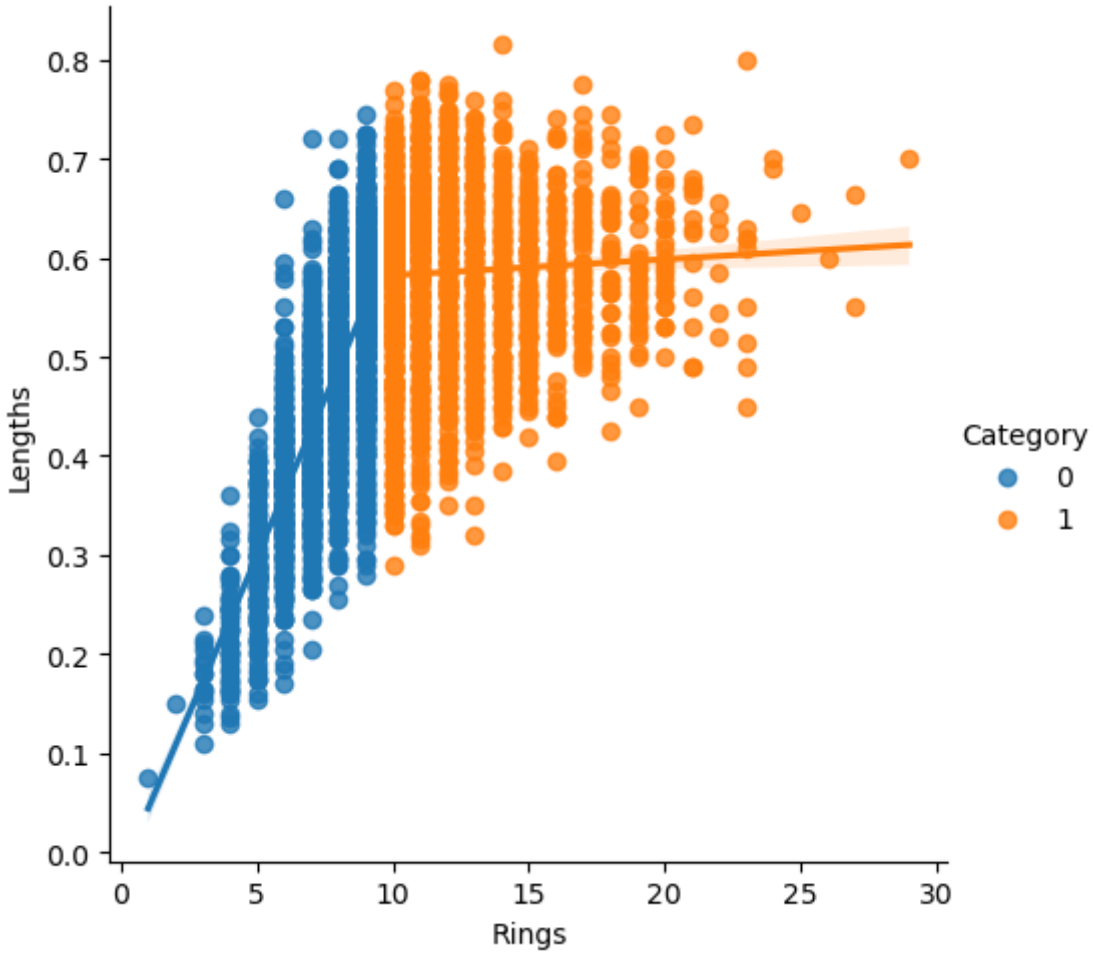
4177 rows × 3 columns

In [33]:

```
# Add the line of best fit:
sns.lmplot(x = "Rings", y = "Lengths", hue ="Category", data = dfnew)
```

Out[33]:

```
<seaborn.axisgrid.FacetGrid at 0x1d51132abb0>
```



In [34]:

```
# what if we categorize according to shucked weight, i.e.
# What is the expected weight of the Abalone, given its length and the
# number of rings?
dataset[0]
swgt = [d['Shucked Weight'] for d in dataset]
```

In [35]:

```
category1 = []
avg1 = numpy.mean(swgt)
for i in range(len(swgt)):
    if swgt[i] > avg1:
        category1.append(1) # category[i] =1 does not work here
    else :
        category1.append(0)

df4 = pd.DataFrame(category1, columns = ['CategoryWeight'])
# the category is 1 for small and 0 for large. Use this to generate two reg lines on a
df = pd.concat([dfnew,df4], axis = 1) # gives a proper dataframe, we need two dfs to co
df
sum(category1)
```

Out[35]:

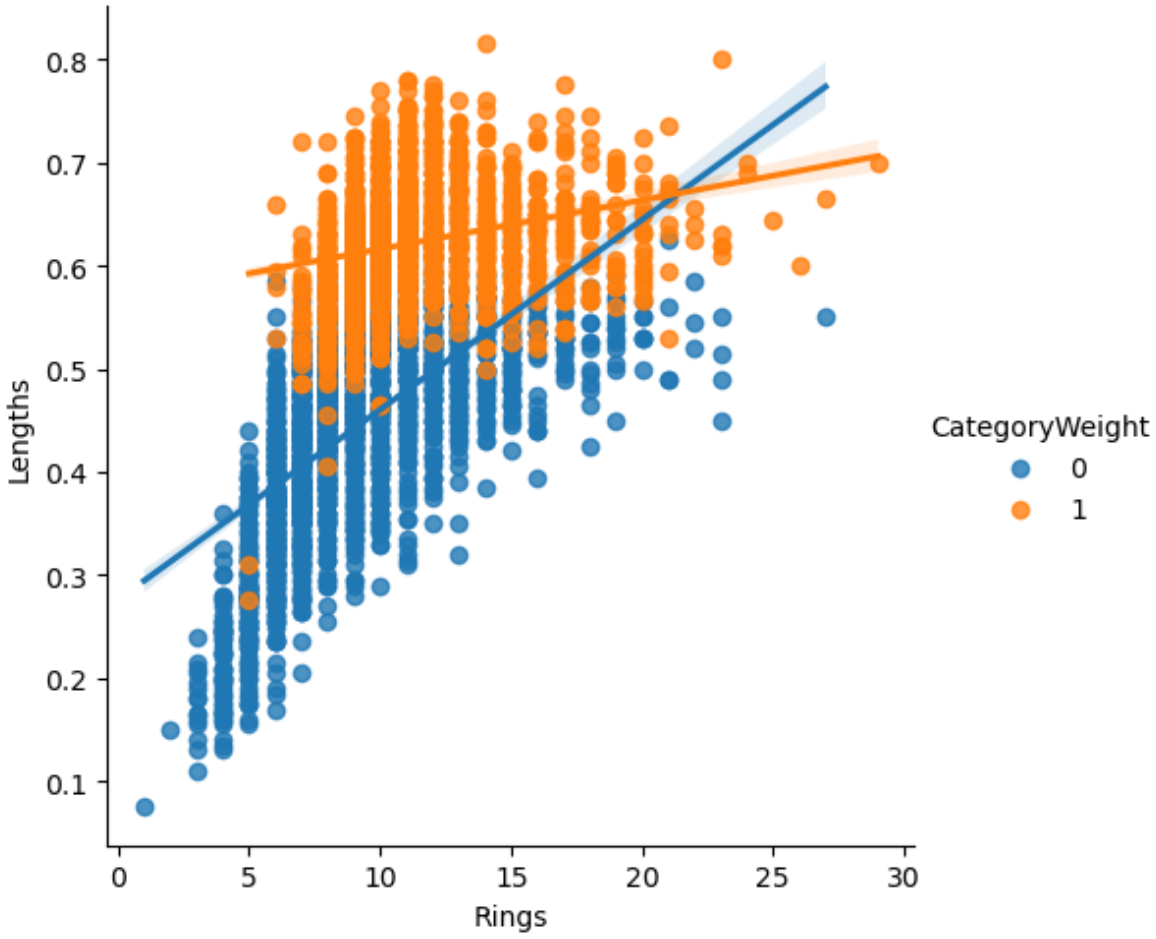
```
1933
```

In [36]:

```
sns.lmplot(x = "Rings", y = "Lengths", hue ="CategoryWeight", data = df)
# this plot gives the estimate of lengths of the abalones using the rings
# as a predictor. It also plots two different regression lines for the two
# categories of shucked weight - smaller than average or larger than average.
```

Out[36]:

```
<seaborn.axisgrid.FacetGrid at 0x1d5114f9760>
```



How to calculate and graph correlation using 'sciPy'

In [37]:

```
from scipy import stats
stats.pearsonr(data1['Rings'],data1['Lengths'])
```

Out[37]:

```
PearsonRResult(statistic=0.556719576929618, pvalue=0.0)
```

Note that p-value is the probability that the given observations are likely if the null hypothesis is true.

Ho: r = 0 or there is no correlation between Rings and Lengths.

p-value = 0 < 0.05, so the result is statistically significant.

This means that there is strong evidence against the null hypothesis and that we reject the null hypothesis.

In [38]:

```
# We can calculate the correlation matrix for all the variables on
# abalones in the given data. First we transform the abalone data into a
# dataframe and then we calculate the correlation matrix :

rings = [d['Rings'] for d in dataset]
length = [d['Length'] for d in dataset]
diameter= [d['Diameter'] for d in dataset]
height = [d['Height'] for d in dataset]
WholeWeight = [d['Whole Weight'] for d in dataset]
ShuckedWeight = [d['Shucked Weight'] for d in dataset]
VisceraWeight = [d['Viscera Weight'] for d in dataset]
ShellWeight = [d['Shell Weight'] for d in dataset]

df1 = pd.DataFrame(rings, columns = ['Rings'])
df2 = pd.DataFrame(length, columns = ['Lengths'])
df3 = pd.DataFrame(diameter, columns = ['Diameter'])
df4 = pd.DataFrame(height, columns = ['Height'])
df5 = pd.DataFrame(WholeWeight, columns = ['Whole Weight'])
df6 = pd.DataFrame(ShuckedWeight, columns = ['Shucked Weight'])
df7 = pd.DataFrame(VisceraWeight, columns = ['Viscera Weight'])
df8 = pd.DataFrame(ShellWeight, columns = ['Shell Weight'])

df = pd.concat([df1,df2,df3,df4,df5,df6,df7,df8], axis = 1) # gives a propoer dataframe
df
```

Out[38]:

	Rings	Lengths	Diameter	Height	Whole Weight	Shucked Weight	Viscera Weight	Shell Weight
0	15	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.1500
1	7	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.0700
2	9	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.2100
3	10	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.1550
4	7	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.0550
...	...	...	...	...	...	...	...	...
4172	11	0.565	0.450	0.165	0.8870	0.3700	0.2390	0.2490
4173	10	0.590	0.440	0.135	0.9660	0.4390	0.2145	0.2605
4174	9	0.600	0.475	0.205	1.1760	0.5255	0.2875	0.3080
4175	10	0.625	0.485	0.150	1.0945	0.5310	0.2610	0.2960
4176	12	0.710	0.555	0.195	1.9485	0.9455	0.3765	0.4950

4177 rows × 8 columns

In [39]:

```
correlation_mat = df.corr()
round(correlation_mat,2)
```

Out[39]:

	Rings	Lengths	Diameter	Height	Whole Weight	Shucked Weight	Viscera Weight	Shell Weight
Rings	1.00	0.56	0.57	0.56	0.54	0.42	0.50	0.63
Lengths	0.56	1.00	0.99	0.83	0.93	0.90	0.90	0.90
Diameter	0.57	0.99	1.00	0.83	0.93	0.89	0.90	0.91
Height	0.56	0.83	0.83	1.00	0.82	0.77	0.80	0.82
Whole Weight	0.54	0.93	0.93	0.82	1.00	0.97	0.97	0.96
Shucked Weight	0.42	0.90	0.89	0.77	0.97	1.00	0.93	0.88
Viscera Weight	0.50	0.90	0.90	0.80	0.97	0.93	1.00	0.91
Shell Weight	0.63	0.90	0.91	0.82	0.96	0.88	0.91	1.00

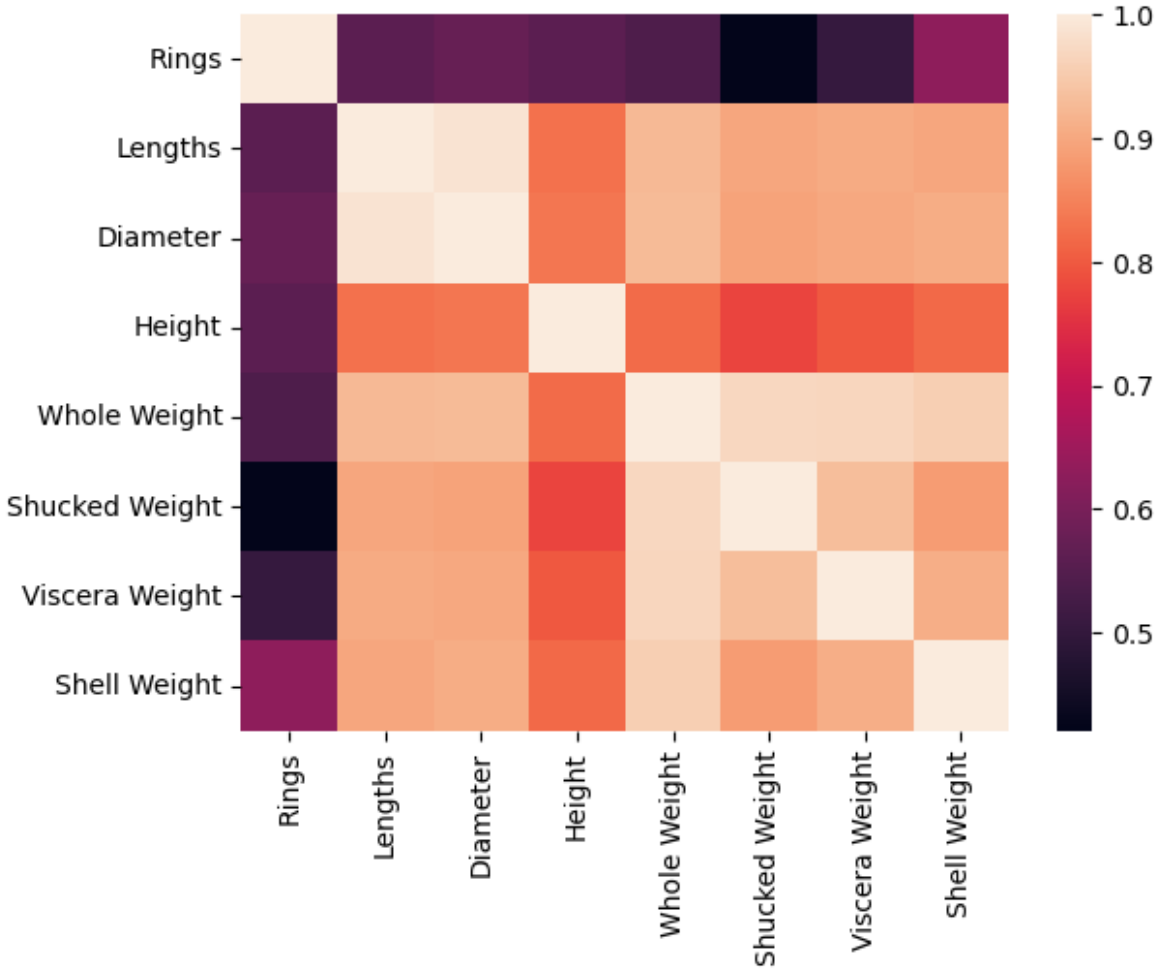
From the above correlation matrix, we see that the maximum correlation between two variables is 0.99 - between diameter and lengths and the minimum correlation is 0.42 - between Shucked weight and rings. This gives a general idea as to what variables are better correlated and which variables are not.

In [40]:

```
1 sns.heatmap(correlation_mat)
```

Out[40]:

<AxesSubplot:>



In [41]:

```
# From the above heatmap, at a glance, we can see that the correlation
# of the number of rings with all the parameters is lower than that of
# all other variables (<0.6 for all except shell weight according to the
# heatmap and <0.7 for shell weight)


```

Bar Plots

Bar Plots are great for comparing categorical variables. There are a few subtypes of bar plots, such as the grouped bar chart or stacked bar chart. Since we have the 'Sex' field to play with, we can compare data across 'M' and 'F' abalones. Below is a simple stacked bar chart comparing the 'Sex' category with the 'Shucked Weight' data.

1. Create a bar chart of your choice of data.
2. Refer to the cell below to parse out fields by sex.

A bar plot is a graph that represents the category of data with rectangular bars - with heights proportional to values which they represent. Can be plotted horizontally or vertically. A bar chart describes the comparisons between the discrete categories. One of the axis of the plot represents the specific categories being compared, while the other represents the measured value corresponding to those categories.

In [42]:

```
# 2. Example Stacked Bar chart- Comparison between Sexes.
Mweight = sum([d['Shucked Weight'] for d in dataset if d['Sex'] == 'M'])
Fweight = sum([d['Shucked Weight'] for d in dataset if d['Sex'] == 'F'])
index = [1] # x-axis is index+- 0.4
```

In [43]:

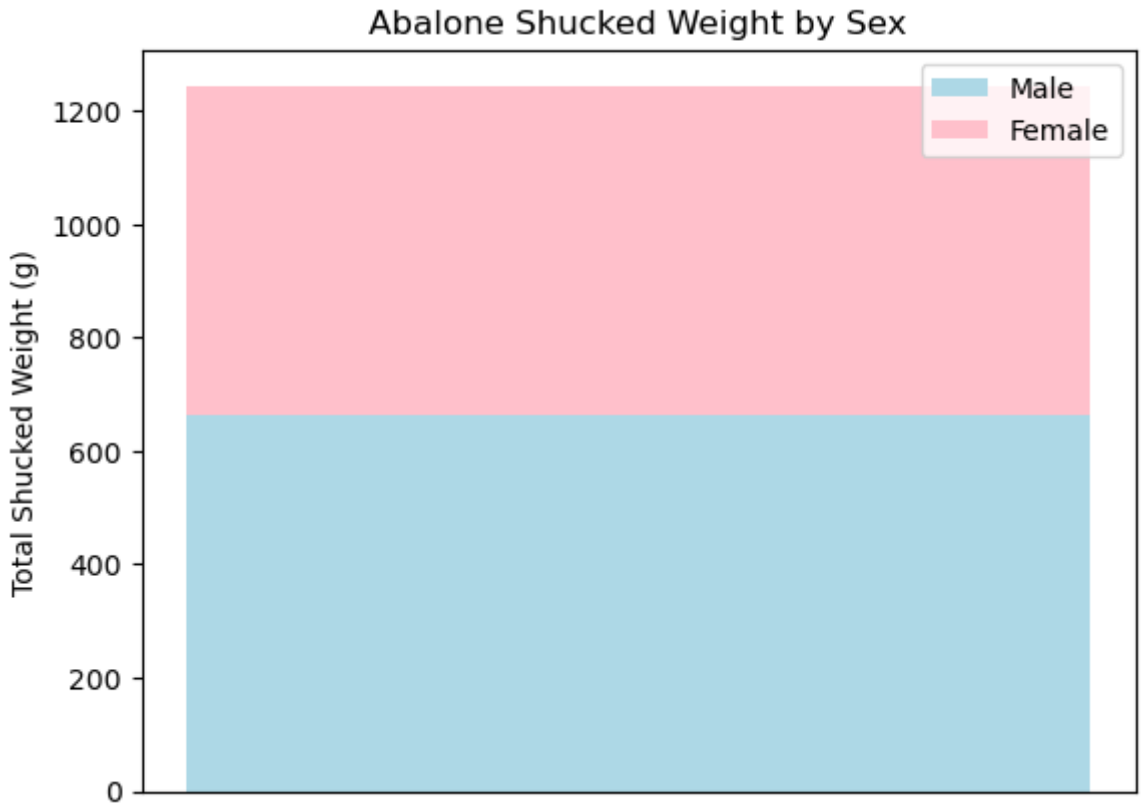
```
Mweight, Fweight
```

Out[43]:

```
(661.5415000000003, 583.1675)
```

In [44]:

```
p1 = plt.bar(index, Mweight, color = 'lightblue')
p2 = plt.bar(index, Fweight, bottom = Mweight, color = 'pink')
plt.gca().set(title = 'Abalone Shucked Weight by Sex', ylabel = 'Total Shucked Weight (
plt.xticks([])
plt.legend((p1[0],p2[0]),('Male','Female'))
plt.show()
```



In [45]:

```
arr_m = [d['Shucked Weight'] for d in dataset if d['Sex'] == 'M']
```

In [46]:

```
max(arr_m), min(arr_m)
```

Out[46]:

```
(1.351, 0.0065)
```

In [47]:

```
# Alternative method:
import numpy as np
import matplotlib.pyplot as plt

dat = {'M': Mweight, 'F': Fweight}
dat
```

Out[47]:

```
{'M': 661.5415000000003, 'F': 583.1675}
```

In [48]:

```
Gender = list(dat.keys())
Values = list(dat.values())
Gender, Values
```

Out[48]:

```
(['M', 'F'], [661.5415000000003, 583.1675])
```

In [49]:

```
fig = plt.figure(figsize = (10,5))
```

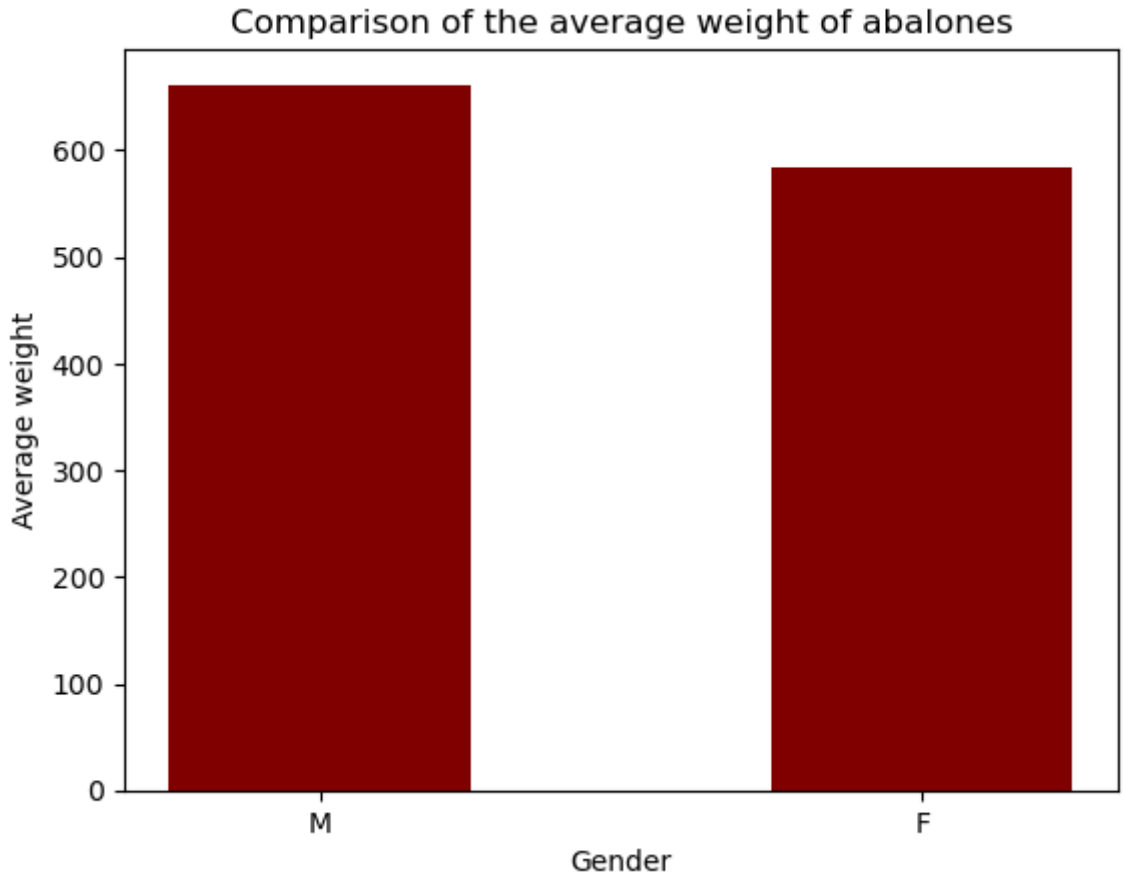
```
<Figure size 1000x500 with 0 Axes>
```

In [50]:

```
# let us put some labels
plt.xlabel("Gender")
plt.ylabel("Average weight")
plt.title("Comparison of the average weight of abalones")
# Creating the bar plot
plt.bar(Gender, Values, color = 'maroon', width = 0.5)
# plt.bar(x, height, width, bottom, align)
```

Out[50]:

```
<BarContainer object of 2 artists>
```



Box Plots:

Box Plots are useful in comparing distributions of data and are commonly found in research papers. The box portion of a box plot represents 50% of the data, and there are versions where you can mark outliers and other extremes. We have the distribution of rings already from the line plot example under the variable name 'age_freq'.

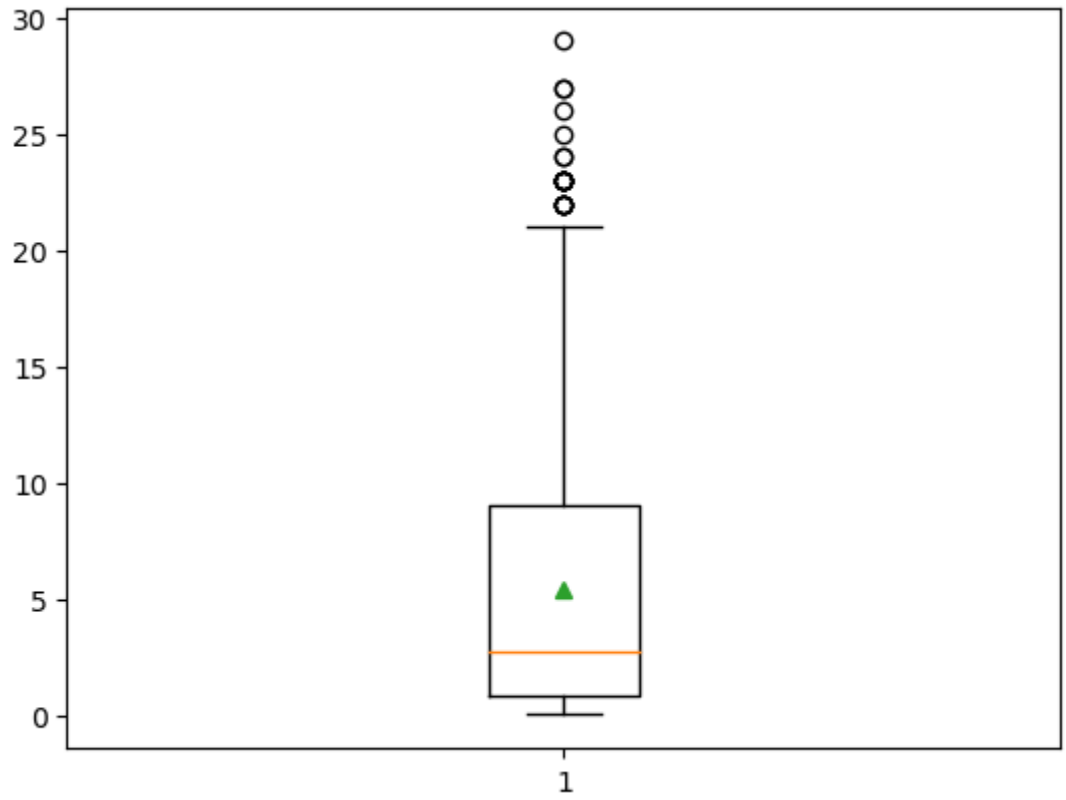
1. Find the distribution of another field of your choice and create one or more box plots with both of these fields.
2. Plot multiple box plots using the command plt.boxplot([plot1,plot2,..,plotn]) or use subplots() to draw multiple separate plots at the same time. See the example at matplotlib website.

In [51]:

```
# Multiple box plots
dataset
# let us compare the boxplots between - the number of rings and whole weight
# as we are interested in whether more rings means more weight.
rings = [d['Rings'] for d in dataset]
WholeWeight = [d['Whole Weight'] for d in dataset]
NewData = np.concatenate((rings,WholeWeight))

ax = plt.boxplot(NewData, showmeans=True)
[item.get_ydata() for item in ax['whiskers']]
```

Out[51]:

```
[array([0.799625, 0.002   ]), array([ 9., 21.])]
```



In [52]:

```
type(ax)
ax.keys() # these keys represent the main elements of a boxplot.
ax
ax['means']
# now to get the values of the whiskers, medians etc:
```

Out[52]:

```
[<matplotlib.lines.Line2D at 0x1d5127fdeb0>]
```

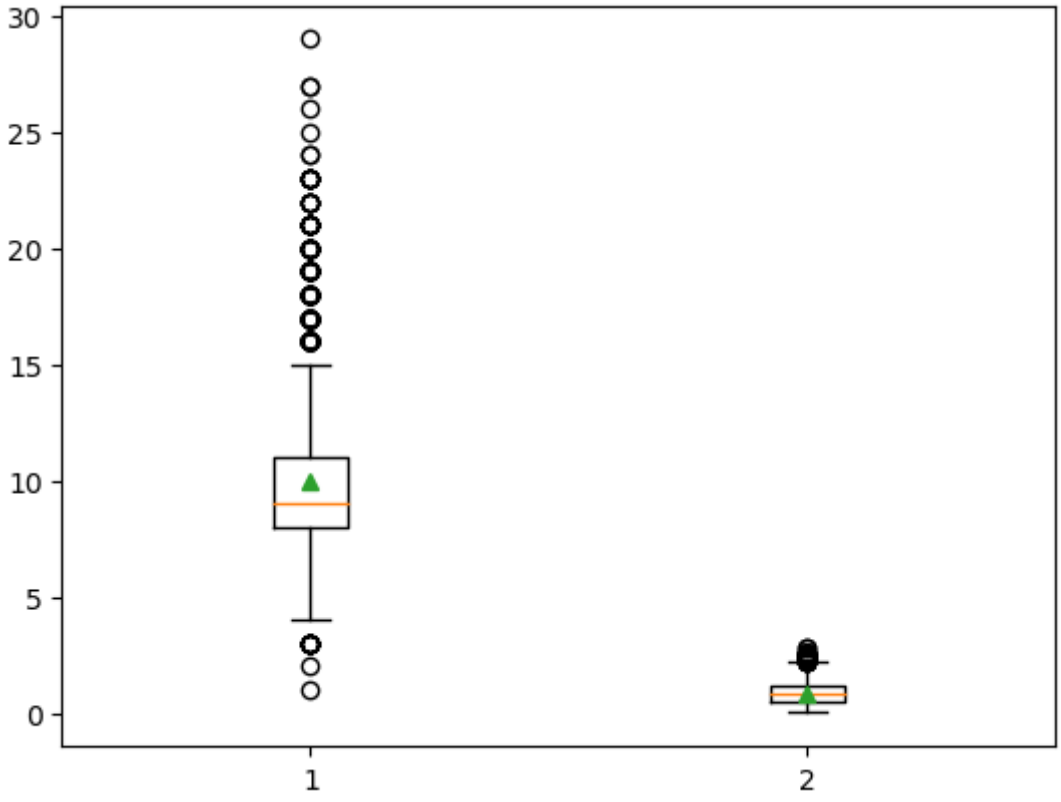
In [53]:

```
for key in ax:
    print(f'{key}:{[item.get_ydata() for item in ax[key]]}\n')

# the first box plot is for the data in two columns rings and wholeWeight
# written as one column to compute the different statistics.
# maximum and minimum are called caps - here the maximum is 21 and the
# minimum is 0.002

# whisker is the distance between third quartile and maximum/cap
# the whiskers are from 0.79 - the first quartile to 0.002 and
# from 9 - the third quartile to 21 - the maximum.

# outliers - are called fliers- data on outliers is also given.
# any number 1.5 *IQR above the third quartile or below the first quartile
# is called an outlier.
#numpy.mean(rings), numpy.mean(WholeWeight)
#numpy.median(rings), numpy.median(WholeWeight)
#numpy.var(rings), numpy.var(WholeWeight)
```

```
whiskers:[array([0.799625, 0.002   ]), array([ 9., 21.])]

caps:[array([0.002, 0.002]), array([21., 21.])]

boxes:[array([0.799625, 0.799625, 9.      , 9.      , 0.799625])]

medians:[array([2.71825, 2.71825])]

fliers:[array([22., 22., 22., 26., 23., 23., 22., 22., 29., 23., 23., 22., 2
3.,
       27., 25., 27., 23., 23., 23., 23., 24., 24.])]

means:[array([5.38121331])]
```

In [54]:

```
# let us now use subplots and plot multiple boxplots together:
#import matplotlib.pyplot as axs
import numpy as np
import matplotlib.pyplot as axs
### We can use the following three statements to draw multiple boxplots,
### But, we won't be able to derive the values of the keys easily
###fig,axs = plt.subplots(1,2) # gives 1 X 2 subplots.
###axs[0].boxplot(NewData)
###axs[1].boxplot(rings)

type(np.array(rings))
type(np.array(WholeWeight))
rings_new = np.array(rings)
WholeWeight_new = np.array(WholeWeight)
data_values = [rings_new,WholeWeight_new]
type(data_values)
bp = axs.boxplot(data_values, showmeans = True)
```



In [55]:

```
maximums = [item.get_ydata()[0] for item in bp['caps']][1::2]
minimums = [item.get_ydata()[0] for item in bp['caps']][::2]
# Note ::2 gives every second item of a sequence.
# a[start:end:step] - and any of start, end and step can be skipped.
# [1::2] indicates that we start at 1 and then jumps at every 2 steps.
# so, we get all the odd elements of the sequence.
#[0] or [1] give the same values
maximums, minimums
```

Out[55]:

```
([15, 2.21], [4, 0.002])
```

In [56]:

```
quartiles = [item.get_ydata() for item in bp['boxes']]
quartiles
# Q1 is the min of these values and Q3 is the maximum of these values.
```

Out[56]:

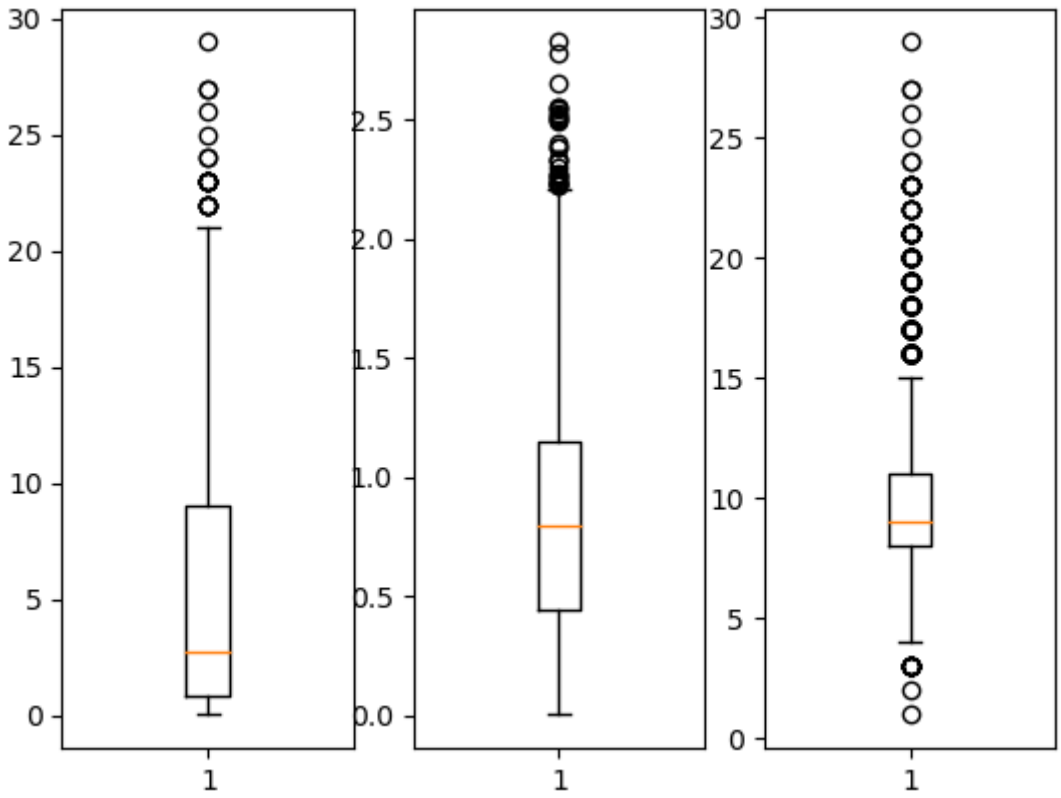
```
[array([ 8.,  8., 11., 11.,  8.]),
 array([0.4415, 0.4415, 1.153 , 1.153 , 0.4415])]
```

In [57]:

```
# Now let us see how rings and WholeWeight compare with the
# let us now use subplots and plot multiple boxplots together:
fig,axs = plt.subplots(1,3) # gives 1 X 3 subplots.
axs[0].boxplot(NewData)
axs[1].boxplot(WholeWeight)
axs[2].boxplot(rings)
```

Out[57]:

```
{'whiskers': [<matplotlib.lines.Line2D at 0x1d512cca070>,
  <matplotlib.lines.Line2D at 0x1d512cca340>],
 'caps': [<matplotlib.lines.Line2D at 0x1d512cca610>,
  <matplotlib.lines.Line2D at 0x1d512cca8e0>],
 'boxes': [<matplotlib.lines.Line2D at 0x1d512cd4d60>],
 'medians': [<matplotlib.lines.Line2D at 0x1d512ccabb0>],
 'fliers': [<matplotlib.lines.Line2D at 0x1d512ccae80>],
 'means': []}
```



In [ ]:

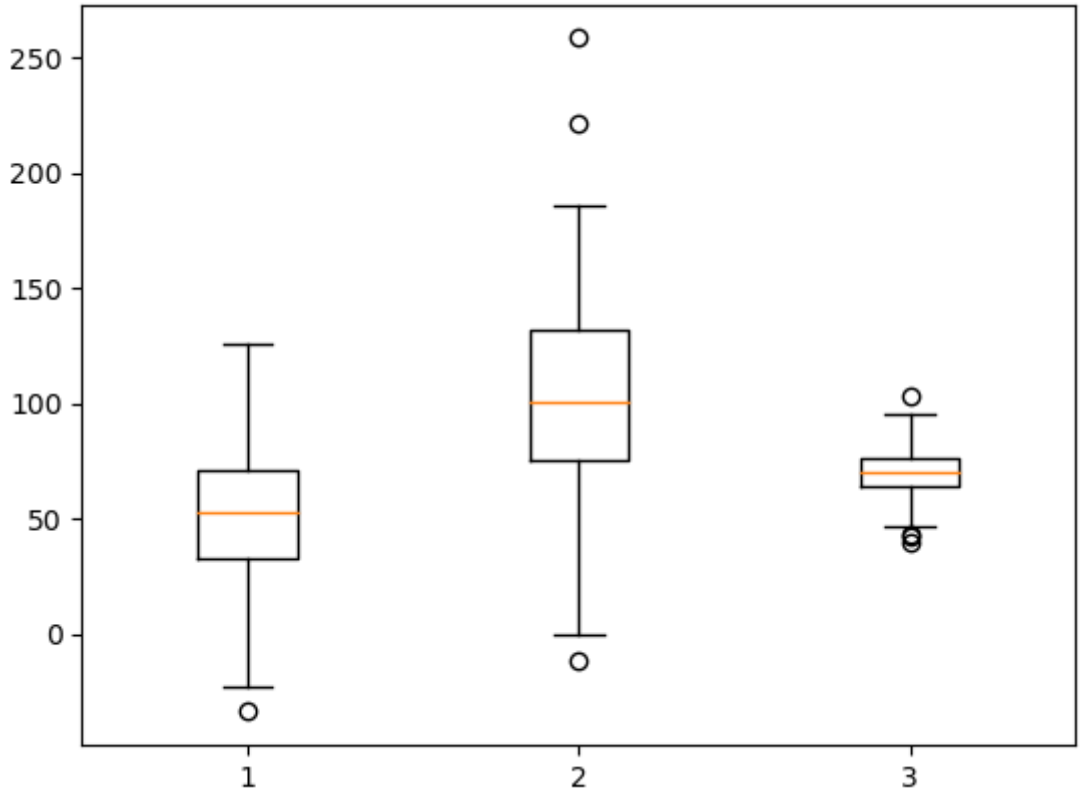
```
1
```

In [58]:

```
# Additional example to plot:
import matplotlib.pyplot as plt
import numpy as np
np.random.seed(1)
data_1 = np.random.normal(50, 30, 300)
data_2 = np.random.normal(100, 40, 300)
data_3 = np.random.normal(70, 10, 300)
data = [data_1, data_2, data_3]
bp = plt.boxplot(data)
type(data_1) # this is an nparray
type(rings) # this is a list
type(bp), bp.keys()
```

Out[58]:

```
(dict, dict_keys(['whiskers', 'caps', 'boxes', 'medians', 'fliers', 'mean
s']))
```



In [59]:

```
for key in bp:
    print(f'{key}: {[item.get_ydata() for item in bp[key]]}\n')
```

```
whiskers: [array([ 32.27380667, -23.04513292]), array([ 70.59264691, 125.849
7712 ]), array([75.68154245, -0.65215444]), array([131.88978143, 185.5131227
]), array([63.74451462, 46.95092062]), array([76.33158616, 95.05980285])]

caps: [array([-23.04513292, -23.04513292]), array([125.8497712, 125.849771
2]), array([-0.65215444, -0.65215444]), array([185.5131227, 185.5131227]), a
rray([46.95092062, 46.95092062]), array([95.05980285, 95.05980285])]

boxes: [array([32.27380667, 32.27380667, 70.59264691, 70.59264691, 32.273806
67]), array([ 75.68154245,  75.68154245, 131.88978143, 131.88978143,
        75.68154245]), array([63.74451462, 63.74451462, 76.33158616, 76.3315
8616, 63.74451462])]

medians: [array([52.64528282, 52.64528282]), array([100.43803244, 100.438032
44]), array([70.10978367, 70.10978367])]

fliers: [array([-33.79255]), array([-11.30137871, 221.23428449, 258.3441081
6]), array([ 42.09003593,  43.01638258,  39.4623562 , 103.21078756])]

means: []
```

In [60]:

```
for key in bp:
    print(f'{key}: {[item.get_ydata() for item in bp[key]]}\n')
```

```
whiskers: [array([ 32.27380667, -23.04513292]), array([ 70.59264691, 125.849
7712 ]), array([75.68154245, -0.65215444]), array([131.88978143, 185.5131227
]), array([63.74451462, 46.95092062]), array([76.33158616, 95.05980285])]

caps: [array([-23.04513292, -23.04513292]), array([125.8497712, 125.849771
2]), array([-0.65215444, -0.65215444]), array([185.5131227, 185.5131227]), a
rray([46.95092062, 46.95092062]), array([95.05980285, 95.05980285])]

boxes: [array([32.27380667, 32.27380667, 70.59264691, 70.59264691, 32.273806
67]), array([ 75.68154245,  75.68154245, 131.88978143, 131.88978143,
        75.68154245]), array([63.74451462, 63.74451462, 76.33158616, 76.3315
8616, 63.74451462])]

medians: [array([52.64528282, 52.64528282]), array([100.43803244, 100.438032
44]), array([70.10978367, 70.10978367])]

fliers: [array([-33.79255]), array([-11.30137871, 221.23428449, 258.3441081
6]), array([ 42.09003593,  43.01638258,  39.4623562 , 103.21078756])]

means: []
```

In [61]:

```
rings = [d['Rings'] for d in dataset]
length = [d['Length'] for d in dataset]
diameter= [d['Diameter'] for d in dataset]
height = [d['Height'] for d in dataset]
WholeWeight = [d['Whole Weight'] for d in dataset]
ShuckedWeight = [d['Shucked Weight'] for d in dataset]
VisceraWeight = [d['Viscera Weight'] for d in dataset]
ShellWeight = [d['Shell Weight'] for d in dataset]

df1 = pd.DataFrame(rings, columns = ['Rings'])
df2 = pd.DataFrame(length, columns = ['Lengths'])
df3 = pd.DataFrame(diameter, columns = ['Diameter'])
df4 = pd.DataFrame(height, columns = ['Height'])
df5 = pd.DataFrame(WholeWeight, columns = ['Whole Weight'])
df6 = pd.DataFrame(ShuckedWeight, columns = ['Shucked Weight'])
df7 = pd.DataFrame(VisceraWeight, columns = ['Viscera Weight'])
df8 = pd.DataFrame(ShellWeight, columns = ['Shell Weight'])

df = pd.concat([df2,df3,df4,df5,df6,df7,df8], axis = 1) # gives a propoer dataframe, we
df
```

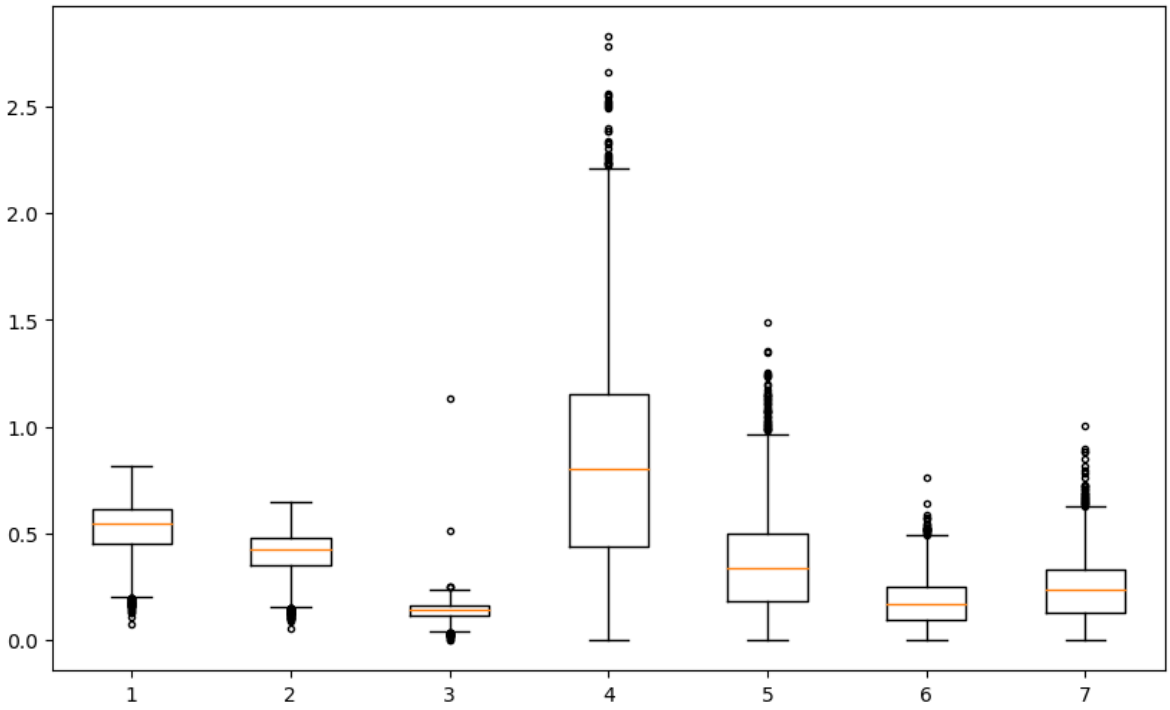
Out[61]:

	Lengths	Diameter	Height	Whole Weight	Shucked Weight	Viscera Weight	Shell Weight
0	0.455	0.365	0.095	0.5140	0.2245	0.1010	0.1500
1	0.350	0.265	0.090	0.2255	0.0995	0.0485	0.0700
2	0.530	0.420	0.135	0.6770	0.2565	0.1415	0.2100
3	0.440	0.365	0.125	0.5160	0.2155	0.1140	0.1550
4	0.330	0.255	0.080	0.2050	0.0895	0.0395	0.0550
...	...	...	...	...	...	...	...
4172	0.565	0.450	0.165	0.8870	0.3700	0.2390	0.2490
4173	0.590	0.440	0.135	0.9660	0.4390	0.2145	0.2605
4174	0.600	0.475	0.205	1.1760	0.5255	0.2875	0.3080
4175	0.625	0.485	0.150	1.0945	0.5310	0.2610	0.2960
4176	0.710	0.555	0.195	1.9485	0.9455	0.3765	0.4950

4177 rows × 7 columns

In [62]:

```
# Data Visualizations
# The purpose of this chart is to see the variation in the different
# attributes of abalones (other than rings) to note how the median and
# quartiles of different attributes vary for the same data.
df
type(df)
fig, ax1= plt.subplots(figsize=(10,6))
bp = ax1.boxplot(df, notch=False, sym='.', vert=True, whis=1.5)

```

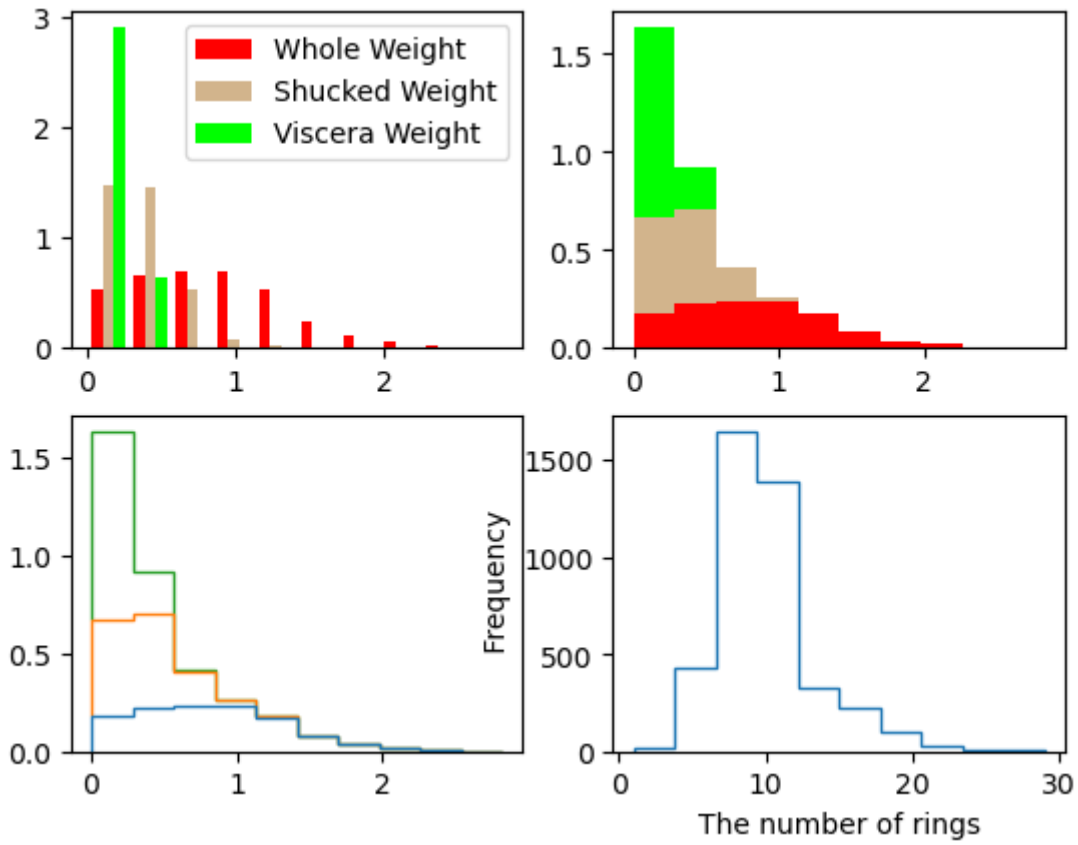


In [63]:

```
# let us also look at a histogram for three attributes: Whole Weight,
# Shucked Weight and Viscera Weight  - df5, df6, df7
data_frame = pd.concat([df5,df6,df7], axis =1)
data_frame

n_bins = 10

fig, ((ax0, ax1), (ax2, ax3)) = plt.subplots(nrows=2, ncols=2)
colors = ['red', 'tan', 'lime']
ax0label = ['Whole Weight' ,'Shucked Weight','Viscera Weight']
ax0.hist(data_frame, n_bins, density = True, histtype = 'bar', color = colors, label =
ax0.legend(prop={'size': 10})

ax1.hist(data_frame, n_bins, density = True, histtype = 'bar', stacked = True, color =

ax2.hist(data_frame, n_bins, density = True, histtype = 'step', stacked = True, fill =

counts, bins = np.histogram(rings)
plt.stairs(counts, bins)
plt.xlabel('The number of rings')
plt.ylabel(' Frequency')
#plt.title('Histogram for the number of rings')
```

Out[63]:

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



In [ ]:

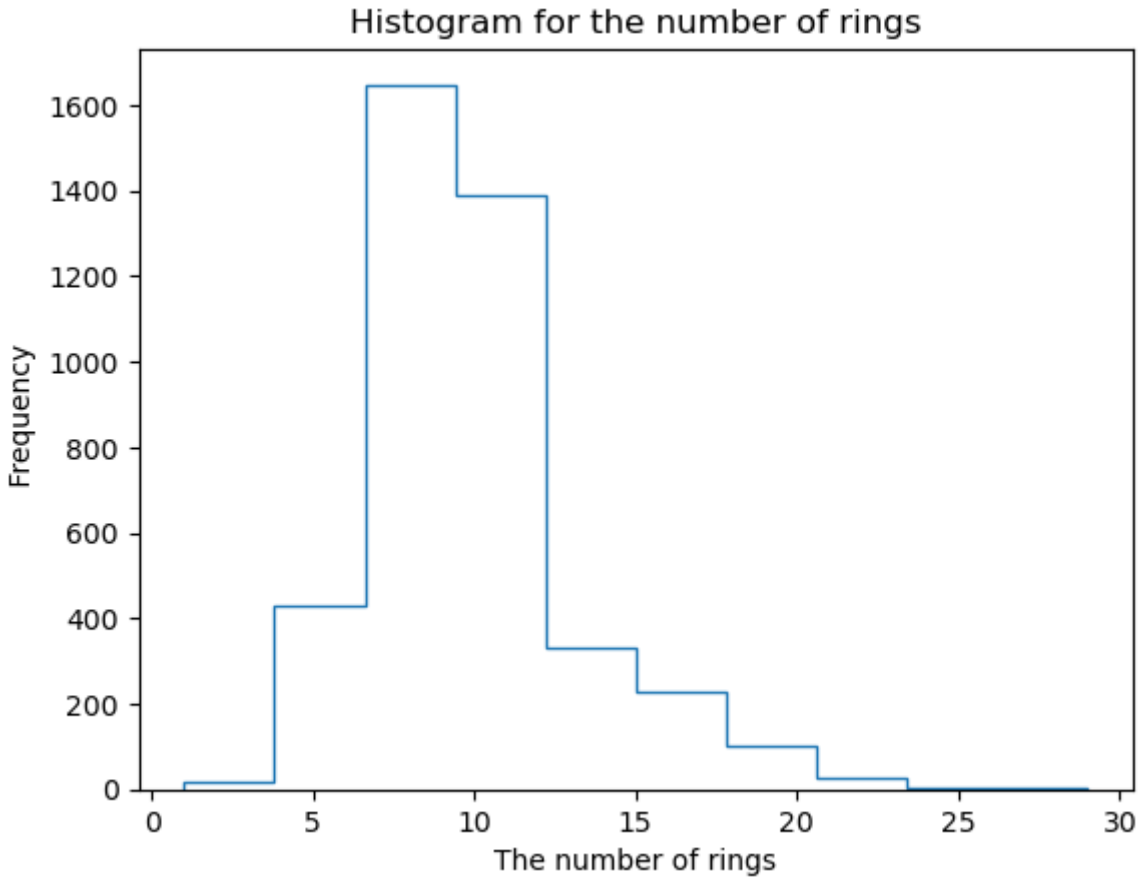
```
1
```

In [64]:

```
1 counts, bins = np.histogram(rings)
2 plt.stairs(counts, bins)
3 plt.xlabel('The number of rings')
4 plt.ylabel(' Frequency')
5 plt.title('Histogram for the number of rings')
```

Out[64]:

```
Text(0.5, 1.0, 'Histogram for the number of rings')
```



In [65]:

```
1 counts, bins
```

Out[65]:

```
(array([  17,  431, 1648, 1388,  329,  228,  100,   29,    4,    3],
       dtype=int64),
 array([ 1. ,  3.8,  6.6,  9.4, 12.2, 15. , 17.8, 20.6, 23.4, 26.2, 29. ]))
```

Part 4: Web Scrapping

This part contains some valuable tips for web scrapping. This is another way to parse a webpage with BeautifulSoup. We will be using a short story from Project Gutenberg "Little Boy" by Harry Neal, 1954

In [66]:

```
# Introduction to Beautoful Soup
# Below are a few useful commands we will be using throughout the next
# section as we parse a webpage/ read the html code of a webpage.
from urllib.request import urlopen
from bs4 import BeautifulSoup
```

In [67]:

```
# Open and extract html from the webpage
f = urlopen("http://www.gutenberg.org/files/58743/58743-h/58743-h.htm")
html = str(f.read())
# The first 100 characters of the HTML are:
html[:100]
```

Out[67]:

```
'b\'<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"\\r\\n    "htt
p://www.w3.org/TR/xhtml1/DTD/x'
```

In [68]:

```
# Convert our HTML object to a BeautifulSoup object and make it readable.
soup = BeautifulSoup(html,'html.parser')
print(soup.prettify())
```

```
b'
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"\r\n    "http://ww
w.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
\r\n
<html lang="en" xml:lang="en" xmlns="http://www.w3.org/1999/xhtml">
 \r\n
 <head>
  \r\n
  <meta content="text/html;charset=utf-8" http-equiv="Content-Type"/>
  \r\n
  <meta content="text/css" http-equiv="Content-Style-Type"/>
  \r\n
  <title>
   \r\n      The Project Gutenberg eBook of Little Boy, by Harry Neal.\r\n
  </title>
  \r\n
  <link href="images/cover.jpg" rel="coverpage"/>
  \r\n\r\n
  <style type="text/css">
```

With a BeautifulSoup object, we can easily search through HTML and create lists and other structures.

In [69]:

```
# Number of paragraph tags
len(soup.find_all('p'))
```

Out[69]:

```
165
```

In [70]:

```
# Create list of all paragraphs
paragraph_list = soup.find_all('p')
paragraph_list[0]
```

Out[70]:

```
<p class="ph1"><i>There are times when the animal in Mankind<br/>\r\nsavagel
y asserts itself. Even children become<br/>\r\nsnarling little beasts. Fortu
nately, however,<br/>\r\nin childhood laughter is not buried deep.</i></p>
```

In [71]:

```
paragraph_list[100]
```

Out[71]:

```
<p>Slowly he felt his own lips curl back into an expression he could\r\nhard
ly remember. He felt the way he felt sometimes late at night when,\r\nsafe a
nd alone in his room, he would play a little with his toys. He\r\ndidn\'t fe
el like killing her any more. He felt like ... like <i>friends</i>.</p>
```

We an also extract all the text from a page and use it to create a bag of words or other measures.

In [72]:

```
# Extract all text from the page
text = soup.get_text()
text[:100]
```

Out[72]:

```
"b'\\r\\n\\r\\n  \\r\\n    \\r\\n    \\r\\n    \\r\\n      The Project Guten
berg eBook of Little Boy, by Harry Ne"
```

In [73]:

```
import string
from collections import defaultdict
letters = defaultdict(int)
punctuation = set(string.punctuation)

for char in text:
   # print(char)
    if char not in punctuation:
         letters[char] += 1
letters.items()

```

Out[73]:

```
dict_items([('b', 594), ('r', 3484), ('n', 3772), (' ', 7958), ('T', 160),
('h', 2175), ('e', 4876), ('P', 119), ('o', 2967), ('j', 115), ('c', 968),
('t', 3615), ('G', 119), ('u', 1046), ('g', 825), ('B', 61), ('k', 451),
('f', 790), ('L', 74), ('i', 2328), ('l', 1346), ('y', 755), ('H', 138),
('a', 2707), ('N', 74), ('E', 130), ('s', 1987), ('w', 841), ('U', 53),
('d', 1728), ('S', 181), ('m', 818), ('p', 647), ('v', 313), ('Y', 40),
('I', 134), ('A', 110), ('R', 71), ('D', 49), ('J', 7), ('2', 11), ('1', 5
9), ('0', 19), ('9', 11), ('5', 18), ('8', 15), ('7', 14), ('4', 17), ('3',
17), ('C', 48), ('O', 71), ('F', 75), ('K', 11), ('W', 34), ('M', 33), ('x',
44), ('—', 54), ('6', 9), ('z', 11), ('q', 42), ('V', 8), ('X', 2), ('Q',
1)])
```

In [74]:

```
char
type(letters)
```

Out[74]:

```
collections.defaultdict
```

In [75]:

```
#Letters
```

In [76]:

```
letters.keys()
```

Out[76]:

```
dict_keys(['b', 'r', 'n', ' ', 'T', 'h', 'e', 'P', 'o', 'j', 'c', 't', 'G',
'u', 'g', 'B', 'k', 'f', 'L', 'i', 'l', 'y', 'H', 'a', 'N', 'E', 's', 'w',
'U', 'd', 'S', 'm', 'p', 'v', 'Y', 'I', 'A', 'R', 'D', 'J', '2', '1', '0',
'9', '5', '8', '7', '4', '3', 'C', 'O', 'F', 'K', 'W', 'M', 'x', '—', '6',
'z', 'q', 'V', 'X', 'Q'])
```

In [77]:

```
1 letters.values()
```

Out[77]:

```
dict_values([594, 3484, 3772, 7958, 160, 2175, 4876, 119, 2967, 115, 968, 36
15, 119, 1046, 825, 61, 451, 790, 74, 2328, 1346, 755, 138, 2707, 74, 130, 1
987, 841, 53, 1728, 181, 818, 647, 313, 40, 134, 110, 71, 49, 7, 11, 59, 19,
11, 18, 15, 14, 17, 17, 48, 71, 75, 11, 34, 33, 44, 54, 9, 11, 42, 8, 2, 1])
```

Creating our own dataset.

In previous notebooks/lectures, we wrote our own parse method to extract parts of the text. Here is a trivial example of how to do the same with BeautifulSoup using a list of [Top 10 Chefs by Gazetter Review] website: (https://gazettereview.com/2017/04/top-10-chefs/ (https://gazettereview.com/2017/04/top-10-chefs/)).

Introduction to Beautiful Soup

Below are a few useful commands we will be using throughout the next section as we parse a webpage

In [82]:

```
1 from urllib.request import Request, urlopen
2 req = Request(
3         url = "https://gazettereview.com/top-10-best-chefs-world/"
4 )
```

In [92]:

```
# Open and extract HTML from the webpage.
#f = urllib.request.urlopen("https://gazettereview.com/top-10-best-chefs-world/")
#html = str(f.read())
from requests import get
from bs4 import BeautifulSoup
url = "https://gazettereview.com/top-10-best-chefs-world/"
req = get(url)
html = req.text
html
```

Out[92]:

```
'<!doctype html >\n<!--[if IE 8]><html class="ie8" lang="en"> <![endif]-->
\n<!--[if IE 9]><html class="ie9" lang="en"> <![endif]-->\n<!--[if gt IE
8]><!--><html lang="en-US" prefix="og: https://ogp.me/ns#"> (https://ogp.m
e/ns#">) <!--<![endif]--><head><script data-no-optimize="1">var litespeed_
docref=sessionStorage.getItem("litespeed_docref");litespeed_docref&&(Objec
t.defineProperty(document,"referrer",{get:function(){return litespeed_docr
ef}}),sessionStorage.removeItem("litespeed_docref"));</script> <title>Top
 10 Chefs In The World - The Best in 2023 - Gazette Review</title><meta ch
arset="UTF-8" /><style id="litespeed-ccss">h1,h3,h4{overflow-wrap:break-wo
rd}ul{overflow-wrap:break-word}p{overflow-wrap:break-word}:root{--wp--pres
et--font-size--normal:16px;--wp--preset--font-size--huge:42px}body{--wp--p
reset--color--black:#000;--wp--preset--color--cyan-bluish-gray:#abb8c3;--w
p--preset--color--white:#fff;--wp--preset--color--pale-pink:#f78da7;--wp--
preset--color--vivid-red:#cf2e2e;--wp--preset--color--luminous-vivid-orang
e:#ff6900;--wp--preset--color--luminous-vivid-amber:#fcb900;--wp--preset--
color--light-green-cyan:#7bdcb5;--wp--preset--color--vivid-green-cyan:#00d
084;--wp--preset--color--pale-cyan-blue:#8ed1fc;--wp--preset--color--vivid
-cyan-blue:#0693e3;--wp--preset--color--vivid-purple:#9b51e0;--wp--preset-
```

In [84]:

```
import sys
sys.version

```

Out[84]:

```
'3.9.13 (main, Aug 25 2022, 23:51:50) [MSC v.1916 64 bit (AMD64)]'
```

In [94]:

```
# Open and extract HTML from the webpage:
f = urlopen("http://www.gutenberg.org/files/58743/58743-h/58743-h.htm")
html = str(f.read())
# First 100 characters of the HTML
html[:100]
```

Out[94]:

```
'b\'<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"\\r\\n    "htt
p://www.w3.org/TR/xhtml1/DTD/x'
```

In [96]:

```
# COnvert out HTML object to a BeautifulSoup object and make it readable.
soup = BeautifulSoup(html,'html.parser')
print(soup.prettify())
```

```
b'
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"\r\n    "http://ww
w.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
\r\n
<html lang="en" xml:lang="en" xmlns="http://www.w3.org/1999/xhtml">
 \r\n
 <head>
  \r\n
  <meta content="text/html;charset=utf-8" http-equiv="Content-Type"/>
  \r\n
  <meta content="text/css" http-equiv="Content-Style-Type"/>
  \r\n
  <title>
   \r\n      The Project Gutenberg eBook of Little Boy, by Harry Neal.\r\n
  </title>
  \r\n
  <link href="images/cover.jpg" rel="coverpage"/>
  \r\n\r\n
  <style type="text/css">
```

With a BeautifulSoup object, we can easily search through HTML and create lists and other structures.

In [97]:

```
# Number of paragraph tags
len(soup.find_all('p'))
```

Out[97]:

```
165
```

We can also extract all the text froma page and use it to create a bag of words or other measures.

In [98]:

```
# Extract all the text from a page:
text = soup.get_text()
text[:100]
```

Out[98]:

```
"b'\\r\\n\\r\\n  \\r\\n    \\r\\n    \\r\\n    \\r\\n      The Project Guten
berg eBook of Little Boy, by Harry Ne"
```

In [99]:

```
import string
from collections import defaultdict
letters = defaultdict(int)
punctuation = set(string.punctuation)
for char in text:
    if char not in punctuation:
        letters[char] += 1
letters.items()
```

Out[99]:

```
dict_items([('b', 594), ('r', 3484), ('n', 3772), (' ', 7958), ('T', 160),
('h', 2175), ('e', 4876), ('P', 119), ('o', 2967), ('j', 115), ('c', 968),
('t', 3615), ('G', 119), ('u', 1046), ('g', 825), ('B', 61), ('k', 451),
('f', 790), ('L', 74), ('i', 2328), ('l', 1346), ('y', 755), ('H', 138),
('a', 2707), ('N', 74), ('E', 130), ('s', 1987), ('w', 841), ('U', 53),
('d', 1728), ('S', 181), ('m', 818), ('p', 647), ('v', 313), ('Y', 40),
('I', 134), ('A', 110), ('R', 71), ('D', 49), ('J', 7), ('2', 11), ('1', 5
9), ('0', 19), ('9', 11), ('5', 18), ('8', 15), ('7', 14), ('4', 17), ('3',
17), ('C', 48), ('O', 71), ('F', 75), ('K', 11), ('W', 34), ('M', 33), ('x',
44), ('—', 54), ('6', 9), ('z', 11), ('q', 42), ('V', 8), ('X', 2), ('Q',
1)])
```

In [109]:

```
# Sorry, the above part got repeated.
# Creating Our Own Dataset with BeautifulSoup
# Open and extract HTML from the webpage:
# f = urlopen("https://gazettereview.com/top-10-best-chefs-world/")

from urllib.request import Request, urlopen
req = Request(
        url = "https://gazettereview.com/top-10-best-chefs-world/"
)
```

In [110]:

```
from requests import get
from bs4 import BeautifulSoup
url = "https://gazettereview.com/top-10-best-chefs-world/"
req = get(url)
html = req.text
html
soup = BeautifulSoup(html,'html.parser')
print(soup.prettify())
```

```
<!DOCTYPE html >
<!--[if IE 8]><html class="ie8" lang="en"> <![endif]-->
<!--[if IE 9]><html class="ie9" lang="en"> <![endif]-->
<!--[if gt IE 8]><!-->
<html lang="en-US" prefix="og: https://ogp.me/ns#"> (https://ogp.me/ns#">)
 <!--<![endif]-->
 <head>
  <script data-no-optimize="1">
   var litespeed_docref=sessionStorage.getItem("litespeed_docref");litespe
ed_docref&&(Object.defineProperty(document,"referrer",{get:function(){retu
rn litespeed_docref}}),sessionStorage.removeItem("litespeed_docref"));
  </script>
  <title>
   Top 10 Chefs In The World - The Best in 2023 - Gazette Review
  </title>
  <meta charset="utf-8"/>
  <style id="litespeed-ccss">
   h1,h3,h4{overflow-wrap:break-word}ul{overflow-wrap:break-word}p{overflo
w-wrap:break-word}:root{--wp--preset--font-size--normal:16px;--wp--preset-
```

Note that all the names of the chefs are in between `<h3>` and `</h3>` tags and the descriptions are between `<p>` and `</p>` tags. We can get the names of the chefs quite easily, as seen below.

In [124]:

```
# List of chef names
# Note that find_all() function returns a bs4 object rather than a
# Python list
# The HTML tags are also a part of the object.
chefs = soup.find_all('h3')
chefs
chefs = chefs[0:10]
chefs
```

Out[124]:

```
[<h3><span class="ez-toc-section" id="10_Anthony_Bourdain"></span>10. Anthon
y Bourdain<span class="ez-toc-section-end"></span></h3>,
 <h3><span class="ez-toc-section" id="9_Paul_Bocuse"></span>9. Paul Bocuse<s
pan class="ez-toc-section-end"></span></h3>,
 <h3><span class="ez-toc-section" id="8_Alain_Ducasse"></span>8. Alain Ducas
se<span class="ez-toc-section-end"></span></h3>,
 <h3><span class="ez-toc-section" id="7_Emeril_Lagasse"></span>7. Emeril Lag
asse<span class="ez-toc-section-end"></span></h3>,
 <h3><span class="ez-toc-section" id="6_Vikas_Khanna"></span>6. Vikas Khanna
<span class="ez-toc-section-end"></span></h3>,
 <h3><span class="ez-toc-section" id="5_Marco_Pierre_White"></span>5. Marco
Pierre White<span class="ez-toc-section-end"></span></h3>,
 <h3><span class="ez-toc-section" id="4_Heston_Blumenthal"></span>4. Heston
Blumenthal<span class="ez-toc-section-end"></span></h3>,
 <h3><span class="ez-toc-section" id="3_Wolfgang_Puck"></span>3. Wolfgang Pu
ck<span class="ez-toc-section-end"></span></h3>,
 <h3><span class="ez-toc-section" id="2_Jamie_Oliver"></span>2. Jamie Oliver
<span class="ez-toc-section-end"></span></h3>,
 <h3><span class="ez-toc-section" id="1_Gordon_Ramsay"></span>1. Gordon Rams
ay<span class="ez-toc-section-end"></span></h3>]
```

In [125]:

```
print(type(chefs))
print(chefs[0])
```

```
<class 'list'>
<h3><span class="ez-toc-section" id="10_Anthony_Bourdain"></span>10. Anthony
Bourdain<span class="ez-toc-section-end"></span></h3>
```

In [158]:

```
# Clean and strip spaces and numbers from the bs4 element and turn it into
# a Python list
import string
letters = set(string.ascii_letters)
chef_name = []
# Grab relevant letters/spaces and remove extra HTML tags and spaces.
for chef in chefs:
    chef = [letter for letter in str(chef) if letter in letters or letter == ' ']
    chef = ''.join(chef[26:len(chef)-25])
    chef_name.append(chef)
```

In [159]:

```
chef_name
```

Out[159]:

```
['AnthonyBourdainspan Anthony Bourdainspan ',
 'PaulBocusespan Paul Bocusespan ',
 'AlainDucassespan Alain Ducassespan ',
 'EmerilLagassespan Emeril Lagassespan ',
 'VikasKhannaspan Vikas Khannaspan ',
 'MarcoPierreWhitespan Marco Pierre Whitespan ',
 'HestonBlumenthalspan Heston Blumenthalspan ',
 'WolfgangPuckspan Wolfgang Puckspan ',
 'JamieOliverspan Jamie Oliverspan ',
 'GordonRamsayspan Gordon Ramsayspan ']
```

In [149]:

```
chef[0].find("span")
```

Out[149]:

```
-1
```

In [157]:

```
chef_name[0][-24]
```

Out[157]:

```
'c'
```

In [156]:

```
len( "classeztocsectionendspan")
```

Out[156]:

```
24
```

In [164]:

```
test_s = "Iam Batman"
sub = "am"
idx = test_s.index(sub)
idx
```

Out[164]:

```
1
```

In [166]:

```
sub = "span"
idx = str(chef_name[0]).index(sub)
idx
```

Out[166]:

```
15
```

In [174]:

```
# this means that span starts at 15, so we can take the string from 19 onwards
#so we can just take the letters in the string upto 15 for chef[0]
sub = "span"
chef_new = []
for i in chef_name:
    indx = str(i).index(sub) + 4
    i = i[indx:-5]
    print(i)
    chef_new.append(i)
```

```
Anthony Bourdain
Paul Bocuse
Alain Ducasse
Emeril Lagasse
Vikas Khanna
Marco Pierre White
Heston Blumenthal
Wolfgang Puck
Jamie Oliver
Gordon Ramsay
```

In [175]:

```
chef_new
```

Out[175]:

```
[' Anthony Bourdain',
 ' Paul Bocuse',
 ' Alain Ducasse',
 ' Emeril Lagasse',
 ' Vikas Khanna',
 ' Marco Pierre White',
 ' Heston Blumenthal',
 ' Wolfgang Puck',
 ' Jamie Oliver',
 ' Gordon Ramsay']
```

Getting the list of chef names is trivial with the 'find_all' function (and a little Python cleaning), but what abou the descriptions? This is a little trickier since there may be overlapping uses for the `<p>` and `</p>` tags, so let's try navigating the BeautifulSoup tree (https://www.crummy.com/software/BeautifulSoup/bs4/doc/#navigating-the-tree).

This website is simple in that every chef has a two-paragraph description in the same format. We can use this to our advantage once we know what to look for. We want to extract the text from these two paragraphs. How can we do so? With the `contents` attribute, we can access the cheildren of each tag

In [213]:

```
from requests import get
from bs4 import BeautifulSoup
url1 = "https://www.crummy.com/software/BeautifulSoup/bs4/doc/#navigating-the-tree"
req1 = get(url1)
html1 = req.text
html1
soup1 = BeautifulSoup(html,'html.parser')
print(soup.prettify())
```

```
<!DOCTYPE html >
<!--[if IE 8]><html class="ie8" lang="en"> <![endif]-->
<!--[if IE 9]><html class="ie9" lang="en"> <![endif]-->
<!--[if gt IE 8]><!-->
<html lang="en-US" prefix="og: https://ogp.me/ns#"> (https://ogp.me/ns#">)
 <!--<![endif]-->
 <head>
  <script data-no-optimize="1">
   var litespeed_docref=sessionStorage.getItem("litespeed_docref");litespe
ed_docref&&(Object.defineProperty(document,"referrer",{get:function(){retu
rn litespeed_docref}}),sessionStorage.removeItem("litespeed_docref"));
  </script>
  <title>
   Top 10 Chefs In The World - The Best in 2023 - Gazette Review
  </title>
  <meta charset="utf-8"/>
  <style id="litespeed-ccss">
   h1,h3,h4{overflow-wrap:break-word}ul{overflow-wrap:break-word}p{overflo
w-wrap:break-word}:root{--wp--preset--font-size--normal:16px;--wp--preset-
```

In [214]:

```
descriptions = soup1.find_all('p')
len(descriptions)
```

Out[214]:

```
41
```

In [215]:

```
del descriptions[-12:]
del descriptions[0]
print("The number of paragraphs is :", len(descriptions))
descriptions[5]
```

```
The number of paragraphs is : 28
```

Out[215]:

```
<p class="p1">Tragically, his career came to an abrupt end in 2018 when he w
as found dead in an apparent suicide while filming <i>Parts Unknown </i>in S
trasbourg. He is survived by his daughter, Ariane, and his girlfriend at the
time, Asia Argento.</p>
```

In [246]:

```
# Set up the loop
i = 0
chef_description = [''] * 20
chef_image = []
# Grab description text from paragraphs
for d in descriptions:
    curr_desc = []
    if i % 2 == 0:
        curr_desc = d.contents[0]
        chef_image.append(d.contents[0])#['src']) # get images as well
        # Append relevant parts to corresponding index
       # chef_description[int(i/2)] = 1 #= chef_description[int(i/2)] + ' ' + curr_desc
    else:
        curr_desc = d.contents[0]
    i += 1

chef_image
```

Out[246]:

```
['Table of Contents',
 'Anthony Bourdain rose to fame in 2000 with his book, ‘Kitchen Confidentia
l: Adventures in the Culinary Underbelly. He was later given his own TV show
on the Travel Channel, ',
 'After graduating from The Culinary Institute of America, he went on to run
various kitchens in New York City, including the Supper Club, Sullivan’s, an
d One Fifth Avenue. By 1998, he was the executive chef at Brasserie Les Hall
es, a French-brasserie-style restaurant in Manhattan. From there, he got inv
olved with culinary writing and television presentations.',
 <img alt="" class="size-medium wp-image-65284 alignright" data-lazyloaded
="1" data-sizes="(max-width: 300px) 100vw, 300px" data-src="https://gazetter
eview.com/wp-content/uploads/2017/04/Paul-Bocuse-300x180.jpg" data-srcset="h
ttps://gazettereview.com/wp-content/uploads/2017/04/Paul-Bocuse-300x180.jpg
300w, https://gazettereview.com/wp-content/uploads/2017/04/Paul-Bocuse.jpg
 (https://gazettereview.com/wp-content/uploads/2017/04/Paul-Bocuse.jpg) 468
w" height="180" loading="lazy" src="
0iaHR0cDovL3d3dy53My5vcmcvMjAwMC9zdmciIHdpZHRoPSIzMDAiIGhlaWdodD0iMTgwIiB2aW
V3Qm94PSIwIDAgMzAwIDE4MCI+PHJlY3Qgd2lkdGg9IjEwMCUiIGhlaWdodD0iMTAwJSIgZmlsbD
0iI2NmZDRkYiIvPjwvc3ZnPg==" width="300"/>,
 'Sadly, it was in the room above this very restaurant that he passed away f
rom Parkinson’s disease on January 20, 2018. He was 91 years old.',
 'Ducasse has opened many successful and innovative restaurants in his lifet
ime. He has earned 21 total Michelin stars. He has also been featured as a g
uest judge on the cooking competition series, ',
 'His cooking style is a fusion of Creole and Cajun which he calls “New New
 Orleans”. His television appearances, restaurants, and endorsements earn ov
er $150 million in revenue per year.',
 'Vikas Khanna is known for creating the most expensive cookbook in the worl
d. The book was created over a span of 12 years and details the rich history
of Indian cuisine. The book costs $13,000, and only 12 have ever been made.
 Khanna has personally gifted it to famous people including Queen Elizabet
h.',
 'Marco Pierre White has been featured on popular shows like Masterchef and
 Hell’s Kitchen. White returned his stars later in his career, citing negati
ve impacts on his personal life. He is nonetheless highly respected.',
 'The British chef’s advocation for a scientific understanding of food has e
arned him a spot as fellow in the Royal Society of Chemistry. Blumenthal has
also pioneered many unique recipes. He is known for pairing food with molecu
lar similarities together, even if such pairings seem bizarre.',
```

'Wolfgang Puck has released multiple cookbooks and appeared on many popular cooking shows. He has even acted in famous TV shows. In 2017, he was given a star on the Hollywood Walk of Fame.',
'In 2002, Oliver released the five-part series Jamie's Kitchen. He is also the creator of The Naked Chef. Jamie Oliver has also pushed for healthy eating in schools. Oliver was the face of the British supermarket chain Sainsbury's until 2005. He has been featured in hundreds of commercials.',
,
'Wanting to advance his career, he later moved to Paris, where he studied French Cuisine. Upon returning to London in 1993, he was offered the position of head chef at La Tante Claire, a three-Michelin-star restaurant in Chelsea. After working there for several years, he opened up his own restaurant, Restaurant Gordon Ramsay in 1998.']

In [222]:

```
1 d.contents[0]
2
```

Out[222]:

'Table of Contents'

In [223]:

```
1 d
```

Out[223]:

Table of Contents

In [224]:

```
1 descriptions
```

Out[224]:

```
[<p class="ez-toc-title">Table of Contents</p>,
 <p><img alt="" class="size-medium wp-image-65278 alignleft" data-lazyload
ed="1" data-sizes="(max-width: 200px) 100vw, 200px" data-src="https://gaze
ttereview.com/wp-content/uploads/2017/04/Anthony-Bourdain-200x300.jpg" dat
a-srcset="https://gazettereview.com/wp-content/uploads/2017/04/Anthony-Bou
rdain-200x300.jpg 200w, https://gazettereview.com/wp-content/uploads/2017/
04/Anthony-Bourdain-280x420.jpg (https://gazettereview.com/wp-content/uplo
ads/2017/04/Anthony-Bourdain-280x420.jpg) 280w, https://gazettereview.com/
wp-content/uploads/2017/04/Anthony-Bourdain.jpg (https://gazettereview.co
m/wp-content/uploads/2017/04/Anthony-Bourdain.jpg) 320w" height="300" src
="
9zdmciIHdpZHRoPSIyMDAiIGhlaWdodD0iMzAwIiB2aWV3Qm94PSIwIDAgMjAwIDMwMCI+PHJl
Y3Qgd2lkdGg9IjEwMCUiIGhlaWdodD0iMTAwJSIgZmlsbD0iI2NmZDRkYiIvPjwvc3ZnPg=="
 width="200"/></p>,
 <p class="p1">Anthony Bourdain rose to fame in 2000 with his book, ‘Kitch
en Confidential: Adventures in the Culinary Underbelly. He was later given
his own TV show on the Travel Channel, <i>Anthony Bourdain: No Reservation
s</i>, which ran from 2005 to 2012. He also hosted <i>A Cook’s Tour. The L
```

In [228]:

```
1 descriptions[2].contents[0] # and viola :)
```

Out[228]:

```
'Anthony Bourdain rose to fame in 2000 with his book, ‘Kitchen Confidential:
Adventures in the Culinary Underbelly. He was later given his own TV show on
the Travel Channel, '
```

In [231]:

```
1 descriptions[1].contents[0]
```

Out[231]:

```
<img alt="" class="size-medium wp-image-65278 alignleft" data-lazyloaded="1"
data-sizes="(max-width: 200px) 100vw, 200px" data-src="https://gazetterevie
w.com/wp-content/uploads/2017/04/Anthony-Bourdain-200x300.jpg" data-srcset
="https://gazettereview.com/wp-content/uploads/2017/04/Anthony-Bourdain-200x
300.jpg 200w, https://gazettereview.com/wp-content/uploads/2017/04/Anthony-B
ourdain-280x420.jpg (https://gazettereview.com/wp-content/uploads/2017/04/An
thony-Bourdain-280x420.jpg) 280w, https://gazettereview.com/wp-content/uploa
ds/2017/04/Anthony-Bourdain.jpg (https://gazettereview.com/wp-content/upload
s/2017/04/Anthony-Bourdain.jpg) 320w" height="300" src="data:image/svg+xml;b
ase64,PHN2ZyB4bWxucz0iaHR0cDovL3d3dy53My5vcmcvMjAwMC9zdmciIHdpZHRoPSIyMDAiIG
hlaWdodD0iMzAwIiB2aWV3Qm94PSIwIDAgMjAwIDMwMCI+PHJlY3Qgd2lkdGg9IjEwMCUiIGhlaW
dodD0iMTAwJSIgZmlsbD0iI2NmZDRkYiIvPjwvc3ZnPg==" width="200"/>
```

In [238]:

```
1 4 % 2
2 i
```

Out[238]:

```
20
```

In [243]:

```
1 chef_image[3]
```

Out[243]:

```
<img alt="" class="size-medium wp-image-65284 alignright" data-lazyloaded
="1" data-sizes="(max-width: 300px) 100vw, 300px" data-src="https://gazetter
eview.com/wp-content/uploads/2017/04/Paul-Bocuse-300x180.jpg" data-srcset="h
ttps://gazettereview.com/wp-content/uploads/2017/04/Paul-Bocuse-300x180.jpg
300w, https://gazettereview.com/wp-content/uploads/2017/04/Paul-Bocuse.jpg
 (https://gazettereview.com/wp-content/uploads/2017/04/Paul-Bocuse.jpg) 468
w" height="180" loading="lazy" src="
0iaHR0cDovL3d3dy53My5vcmcvMjAwMC9zdmciIHdpZHRoPSIzMDAiIGhlaWdodD0iMTgwIiB2aW
V3Qm94PSIwIDAgMzAwIDE4MCI+PHJlY3Qgd2lkdGg9IjEwMCUiIGhlaWdodD0iMTAwJSIgZmlsbD
0iI2NmZDRkYiIvPjwvc3ZnPg==" width="300"/>
```

In [281]:

```
1 d = descriptions[2]
2 d
3
```

Out[281]:

```
<p class="p1">Anthony Bourdain rose to fame in 2000 with his book, ‘Kitchen
Confidential: Adventures in the Culinary Underbelly. He was later given his
own TV show on the Travel Channel, <i>Anthony Bourdain: No Reservations</i>,
which ran from 2005 to 2012. He also hosted <i>A Cook’s Tour, The Layover, T
he Mind of a Chef, Parts Unknown,</i> <i>No Reservations,</i> and was a gues
t judge in the cooking competition series, <i>Top Chef.</i></p>
```

In [253]:

```
1 d.contents[0]
```

Out[253]:

```
'Anthony Bourdain rose to fame in 2000 with his book, ‘Kitchen Confidential:
Adventures in the Culinary Underbelly. He was later given his own TV show on
the Travel Channel, '
```

In [ ]:

```
1  from lxml import etree
2  from io import StringIO
3
4  parser = etree.HTMLParser()
5  tree = descriptions[0]
6
7  for p in tree.contents[0]("//p"):
8      if len(p):
9          continue
10     t = p.text
11     if not (t and t.strip()):
12         p.getparent().remove(p)
```

In [282]:

```
# Now let us see if we can clean the html of tags, or basically
# everything inside <>
import re
# as per recommendation by @freylis, compile only once.
CLEANR = re.compile('<.*?>')

```

In [283]:

```
def cleanhtml(raw_html):
    cleantext = re.sub(CLEANR, "", raw_html)
    return cleantext
```

In [284]:

```
cleanhtml(html)
```

Out[284]:

```
'\n \n \n var litespeed_docref=sessionStorage.getItem("litespeed_docref");
litespeed_docref&&(Object.defineProperty(document,"referrer",{get:function
(){return litespeed_docref}}),sessionStorage.removeItem("litespeed_docre
f")); Top 10 Chefs In The World - The Best in 2023 - Gazette Reviewh1,h3,h
4{overflow-wrap:break-word}ul{overflow-wrap:break-word}p{overflow-wrap:bre
ak-word}:root{--wp--preset--font-size--normal:16px;--wp--preset--font-size
--huge:42px}body{--wp--preset--color--black:#000;--wp--preset--color--cyan
-bluish-gray:#abb8c3;--wp--preset--color--white:#fff;--wp--preset--color--
pale-pink:#f78da7;--wp--preset--color--vivid-red:#cf2e2e;--wp--preset--col
or--luminous-vivid-orange:#ff6900;--wp--preset--color--luminous-vivid-ambe
r:#fcb900;--wp--preset--color--light-green-cyan:#7bdcb5;--wp--preset--colo
r--vivid-green-cyan:#00d084;--wp--preset--color--pale-cyan-blue:#8ed1fc;--
wp--preset--color--vivid-cyan-blue:#0693e3;--wp--preset--color--vivid-purp
le:#9b51e0;--wp--preset--gradient--vivid-cyan-blue-to-vivid-purple:linear-
gradient(135deg,rgba(6,147,227,1) 0%,#9b51e0 100%);--wp--preset--gradient-
-light-green-cyan-to-vivid-green-cyan:linear-gradient(135deg,#7adcb4 0%,#0
0d082 100%);--wp--preset--gradient--luminous-vivid-amber-to-luminous-vivid
-orange:linear-gradient(135deg,rgba(252,185,0,1) 0%,rgba(255,105,0,1) 10
```

In [ ]:

```

```