PROBLEM SET 2, MRIDUL HARISH, CED18I034

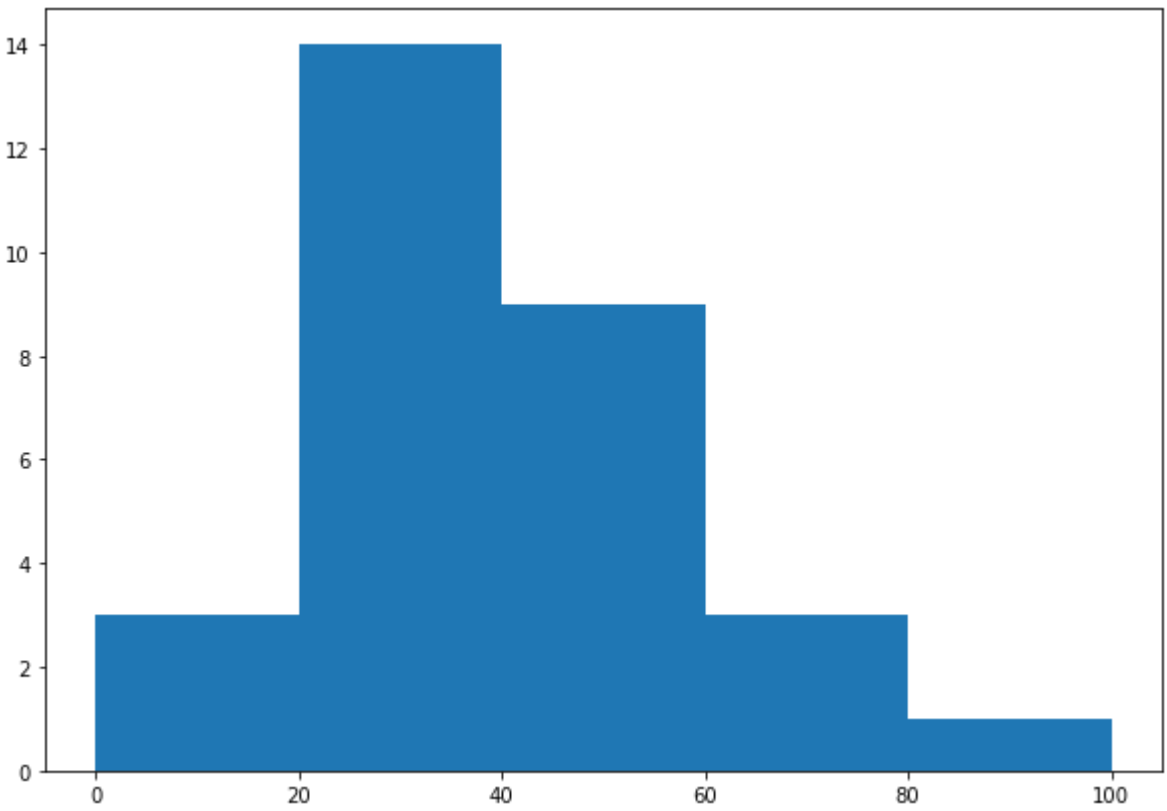
Question 1 - On New Year's Eve, Tina walked into a random shop and surprised to see a huge crowd there. She is interested to find what kind of products they sell the most, for which she needs the age distribution of customers. Help her to find out the same using histogram. The age details of the customers are given below 7, 9, 27, 28, 55, 45, 34, 65, 54, 67, 34, 23, 24, 66, 53, 45, 44, 88, 22, 33, 55, 35, 33, 37, 47, 41,31, 30, 29, 12. Identify the type of histogram (eg. Bimodal, Multimodal, Skewed..etc). Use different bin sizes.

In [ ]:
```
from matplotlib import pyplot as plt
import numpy as np
```

In [ ]:
```
age = np.array([7, 9, 27, 28, 55, 45, 31, 65, 54, 67, 34, 23, 24, 66, 53, 45,
```

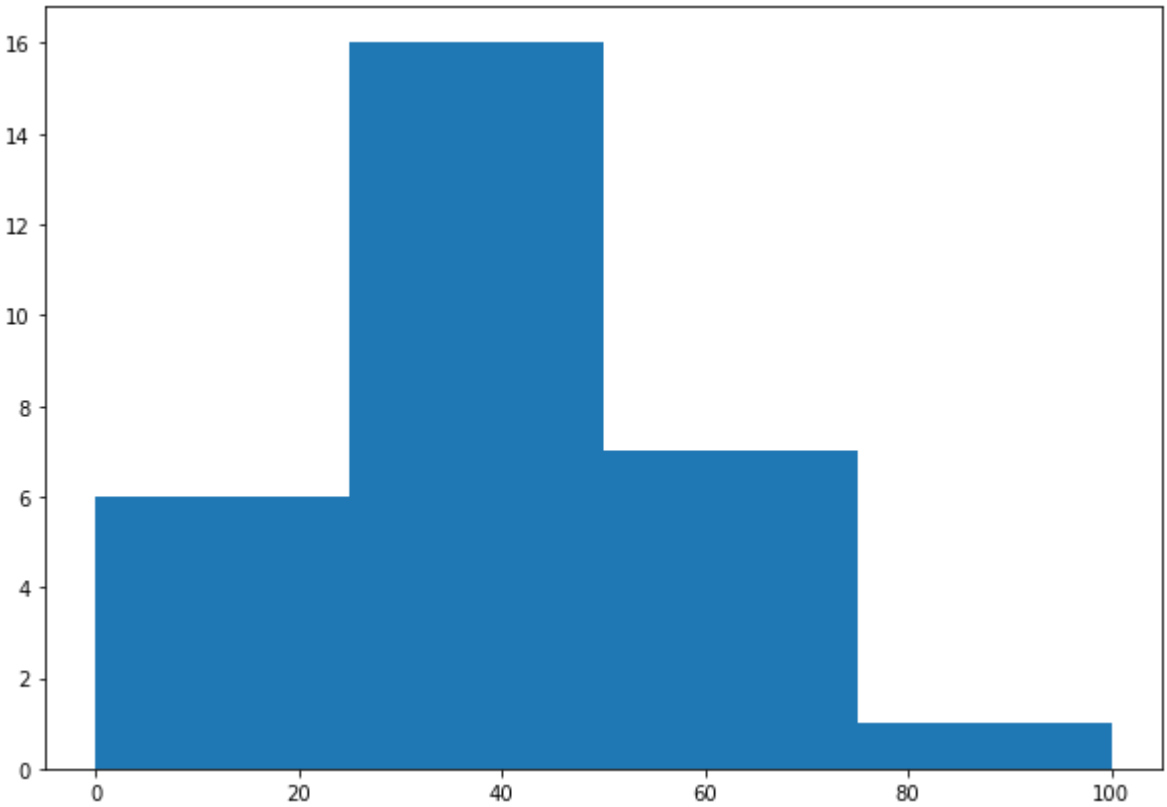
In [ ]:
```
fig, ax = plt.subplots(figsize =(10, 7))
ax.hist(age, bins = [0, 20, 40, 60, 80, 100])
```

Out[ ]:
```
(array([ 3., 14.,  9.,  3.,  1.]),
 array([  0,  20,  40,  60,  80, 100]),
 <BarContainer object of 5 artists>)
```



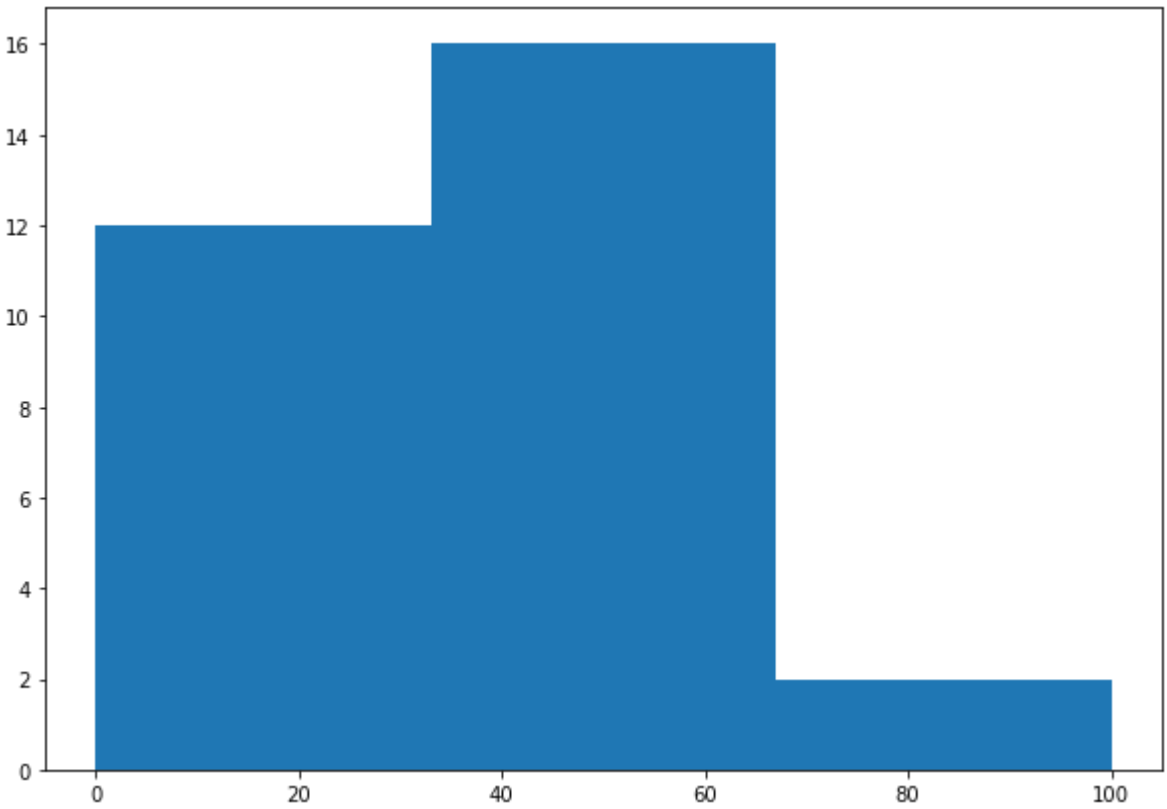
In [ ]:
```
fig, ax = plt.subplots(figsize =(10, 7))
ax.hist(age, bins = [0, 25, 50, 75, 100])
```

Out[ ]:
```
(array([ 6., 16.,  7.,  1.]),
 array([  0,  25,  50,  75, 100]),
```



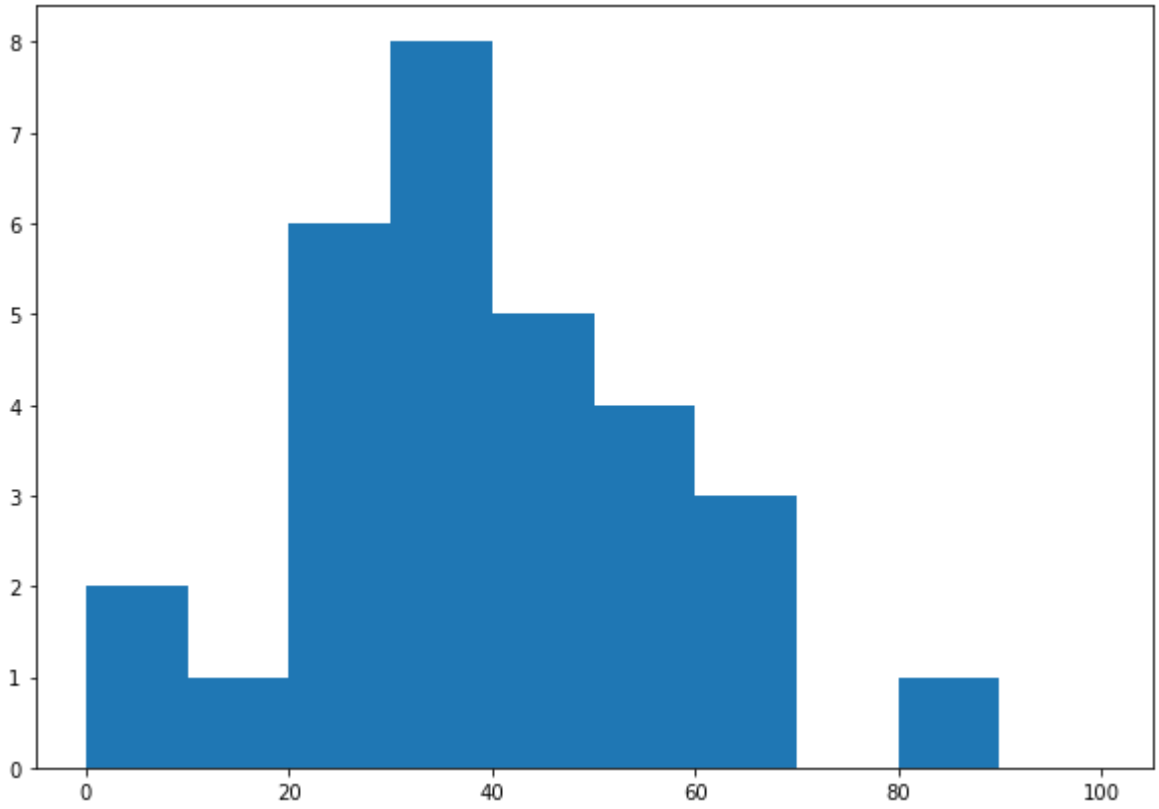
In [ ]:
```
fig, ax = plt.subplots(figsize =(10, 7))
ax.hist(age, bins = [0, 33, 67, 100])
```

Out[ ]:
```
(array([12., 16.,  2.]),
 array([  0,  33,  67, 100]),
 <BarContainer object of 3 artists>)
```



In [ ]:
```
fig, ax = plt.subplots(figsize =(10, 7))
ax.hist(age, bins = [0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100])
```

Out[ ]:
```
(array([2., 1., 6., 8., 5., 4., 3., 0., 1., 0.]),
 array([  0,  10,  20,  30,  40,  50,  60,  70,  80,  90, 100]),
 <BarContainer object of 10 artists>)
```



The type of histogram is "unimodal" even with different bin sizes.

Question 2 - A Coach tracked the number of points that each of his 30 players on the team had in one game. The points scored by each player is given below. Visualize the data using ordered stem-leaf plot and also detect the outliers and shape of the distribution. 22, 21, 24, 19, 27, 28, 24, 25, 29, 28, 26, 31, 28, 27, 22, 39, 20, 10, 26, 24, 27, 28, 26, 28, 18, 32, 29, 25, 31, 27.

In [ ]:
```
import sys
!{sys.executable} -m pip install stemgraphic
```

```
Collecting stemgraphic
  Downloading stemgraphic-0.9.1-py3-none-any.whl (61 kB)
Requirement already satisfied: matplotlib in c:\users\hp\anaconda3\lib\site-pa
ckages (from stemgraphic) (3.3.4)
Requirement already satisfied: seaborn in c:\users\hp\anaconda3\lib\site-packa
ges (from stemgraphic) (0.11.1)
Collecting docopt
  Downloading docopt-0.6.2.tar.gz (25 kB)
Requirement already satisfied: pandas in c:\users\hp\anaconda3\lib\site-packag
es (from stemgraphic) (1.2.4)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\hp\anaconda3\lib\
site-packages (from matplotlib->stemgraphic) (1.3.1)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.3 in c:\
users\hp\anaconda3\lib\site-packages (from matplotlib->stemgraphic) (2.4.7)
Requirement already satisfied: python-dateutil>=2.1 in c:\users\hp\anaconda3\l
ib\site-packages (from matplotlib->stemgraphic) (2.8.1)
Requirement already satisfied: numpy>=1.15 in c:\users\hp\anaconda3\lib\site-p
ackages (from matplotlib->stemgraphic) (1.20.1)
Requirement already satisfied: cycler>=0.10 in c:\users\hp\anaconda3\lib\site-
packages (from matplotlib->stemgraphic) (0.10.0)
Requirement already satisfied: pillow>=6.2.0 in c:\users\hp\anaconda3\lib\site
-packages (from matplotlib->stemgraphic) (8.2.0)
Requirement already satisfied: six in c:\users\hp\anaconda3\lib\site-packages
(from cycler>=0.10->matplotlib->stemgraphic) (1.15.0)
Requirement already satisfied: pytz>=2017.3 in c:\users\hp\anaconda3\lib\site-
packages (from pandas->stemgraphic) (2021.1)
Requirement already satisfied: scipy>=1.0 in c:\users\hp\anaconda3\lib\site-pa
ckages (from seaborn->stemgraphic) (1.6.2)
Building wheels for collected packages: docopt
  Building wheel for docopt (setup.py): started
  Building wheel for docopt (setup.py): finished with status 'done'
  Created wheel for docopt: filename=docopt-0.6.2-py2.py3-none-any.whl size=13
705 sha256=e7dd94fa6f08f25f99489cb5090e51648c3b0970762f418df119aa9b301789b8
  Stored in directory: c:\users\hp\appdata\local\pip\cache\wheels\56\ea\58\ead
137b087d9e326852a851351d1debf4ada529b6ac0ec4e8c
Successfully built docopt
Installing collected packages: docopt, stemgraphic
Successfully installed docopt-0.6.2 stemgraphic-0.9.1
```

In [ ]:
```
import stemgraphic
from matplotlib import pyplot as plt
```

In [ ]:
```
points = [22, 21, 24, 19, 27, 28, 24, 25, 29, 28, 26, 31, 28, 27, 22, 39, 20,
```

In [ ]:
```
stemgraphic.stem_graphic(points, scale = 10)
```

Out[ ]:
```
(<Figure size 540x108 with 1 Axes>,
 <matplotlib.axes._axes.Axes at 0x164d96cfd90>)
```

```
  39
30 | 3 1129
26 | 2 01224445566677778888899
 3 | 1 089
  10
```

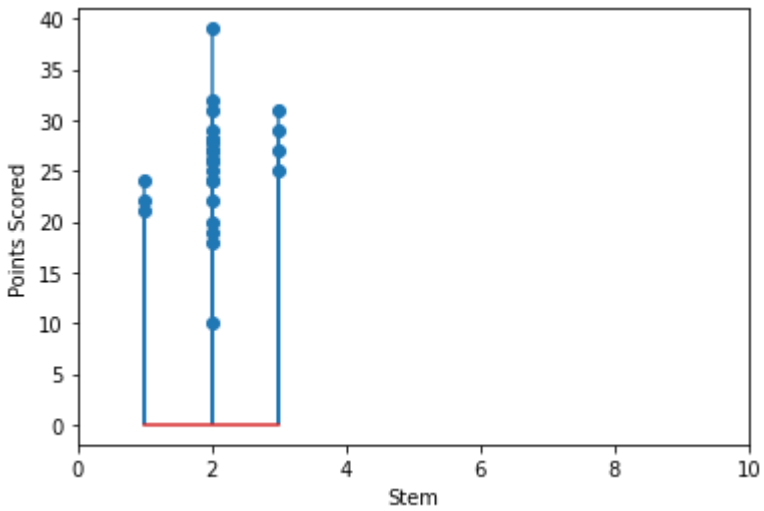
```
Key: aggr|stem|leaf
» |3 |1  =  3.1 x10 = 31.0
```

The leftmost column in the above plot is the frequency count. Then there are two values, one at bottom most we have 10. While at the topmost we have 39, these values are nothing but the minimum and maximum values respectively in the given data set. After that, we have the stem values and following that we have leaves values separating stem by the vertical line.

In [ ]:
```
stems = [1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
```

In [ ]:
```
plt.ylabel('Points Scored')
plt.xlabel('Stem')
plt.xlim(0, 10)
plt.stem(stems, points)
```

Out[ ]:
```
<StemContainer object of 3 artists>
```



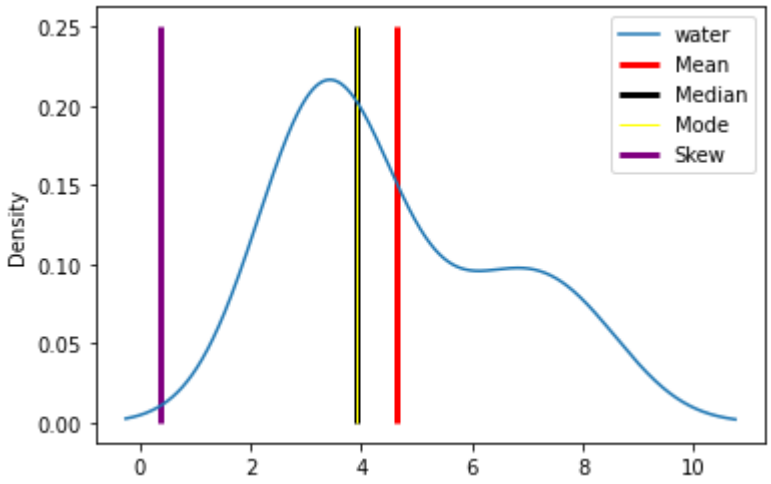
As we can see from the plot, the values 10 and 19 are the outliers. The shape of the stem-leaf plot distribution is bell shaped.

Question 3 - For a sample space of 15 people, a statistician wanted to know the consumption of water and other beverages. He collected their average consumption of water and beverages for 30 days (in litres). Help him to visualize the data using density plot, rug plot and identify the mean, median, mode and skewness of the data from the plot. WATER 3.2, 3.5, 3.6, 2.5, 2.8, 5.9, 2.9, 3.9, 4.9, 6.9, 7.9, 8.0, 3.3, 6.6, 4.4 BEVERAGES 2.2, 2.5, 2.6, 1.5, 3.8, 1.9, 0.9, 3.9, 4.9, 6.9, 0.1, 8.0, 0.3, 2.6, 1.4

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

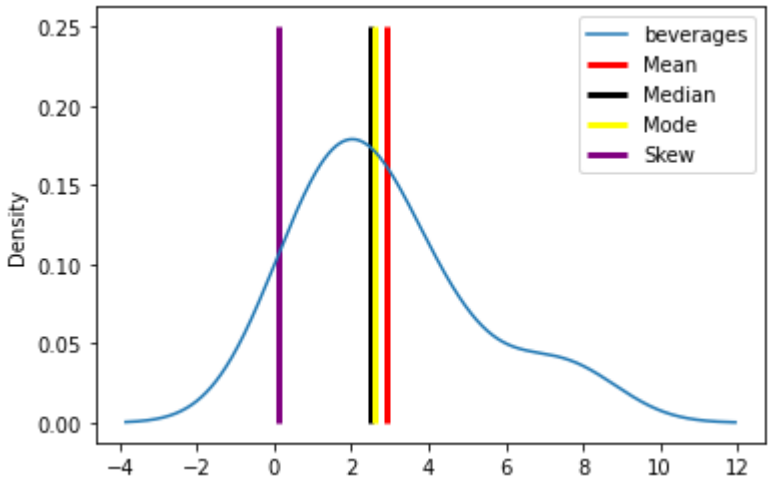
In [ ]:
```
water = [3.2, 3.5, 3.6, 2.5, 2.8, 5.9, 2.9, 3.9, 3.9, 6.9, 7.9, 8.0, 3.3, 6.6,
beverages = [2.2, 2.5, 2.6, 1.5, 3.8, 1.9, 0.9, 3.9, 4.9, 6.9, 0.1, 8.0, 0.3,
```

In [ ]:
```
df = pd.DataFrame(water, columns =  ['water'])
df.plot(kind = 'density')
Mean = np.mean(water)
Median = np.median(water)
Mode = df['water'].mode()[0]
Stddev = df['water'].std()
Skew = (Mean - Mode)/Stddev
plt.vlines(Mean, 0, 0.25, color = 'red', linestyle='solid', label = 'Mean', li
plt.vlines(Median, 0, 0.25, color = 'black', linestyle='solid', label = 'Media
plt.vlines(Mode, 0, 0.25, color = 'yellow', linestyle='solid', label = 'Mode',
plt.vlines(Skew, 0, 0.25, color = 'purple', linestyle='solid', label = 'Skew',
plt.legend()
plt.show()
```



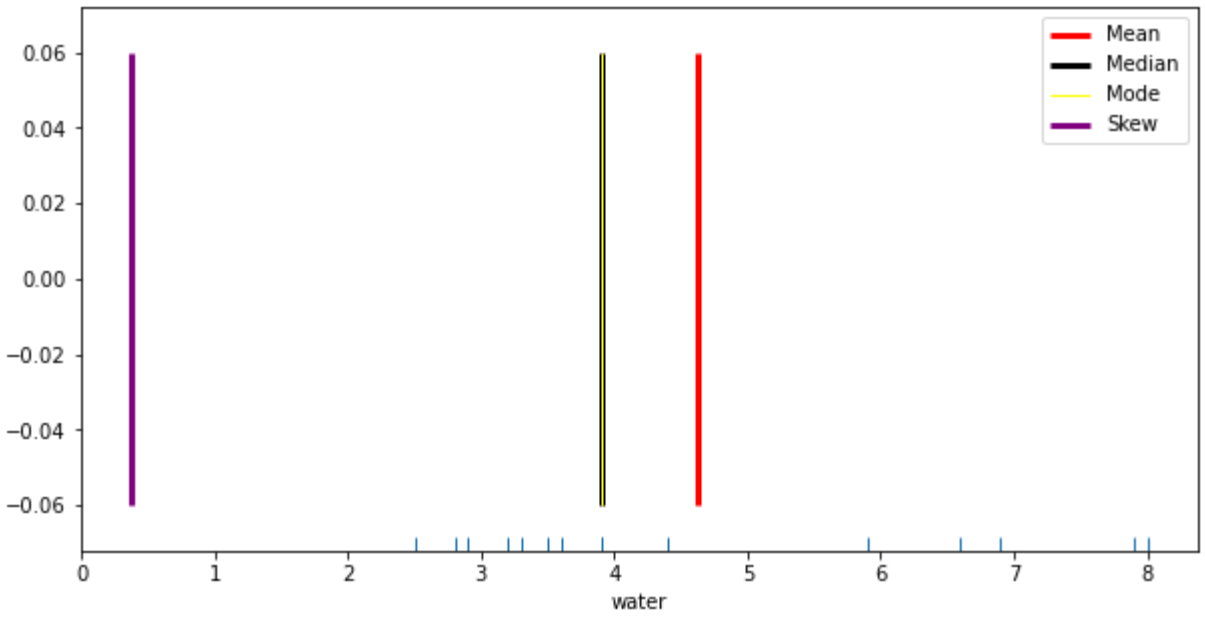
In [ ]:

```
df1 = pd.DataFrame(beverages, columns =  ['beverages'])
df1.plot(kind = 'density')
Mean = np.mean(beverages)
Median = np.median(beverages)
Mode = df1['beverages'].mode()[0]
Stddev = df1['beverages'].std()
Skew = (Mean - Mode)/Stddev
plt.vlines(Mean, 0, 0.25, color = 'red', linestyle='solid', label = 'Mean', li
plt.vlines(Median, 0, 0.25, color = 'black', linestyle='solid', label = 'Media
plt.vlines(Mode, 0, 0.25, color = 'yellow', linestyle='solid', label = 'Mode'
plt.vlines(Skew, 0, 0.25, color = 'purple', linestyle='solid', label = 'Skew'
plt.legend()
plt.show()
```



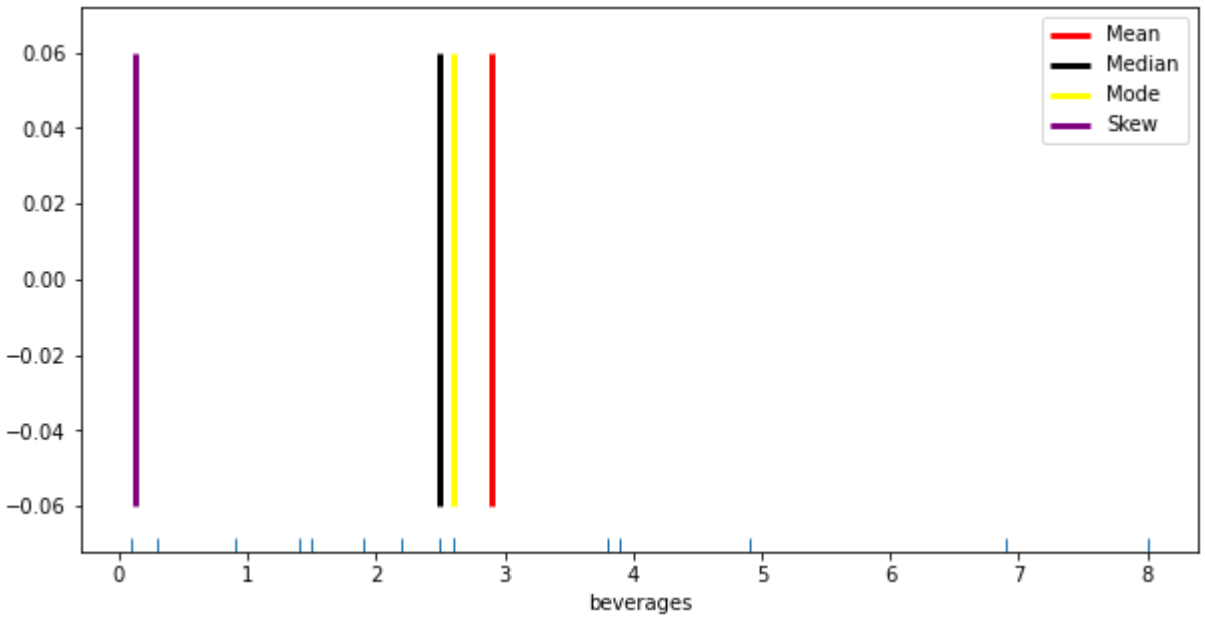
In [ ]:

```
plt.figure(figsize=(10,5))
sns.rugplot(data=df, x ="water")
Mean = np.mean(water)
Median = np.median(water)
Mode = df['water'].mode()[0]
Stddev = df['water'].std()
Skew = (Mean - Mode)/Stddev
plt.vlines(Mean, -0.06, 0.06, color = 'red', linestyle='solid', label = 'Mean
plt.vlines(Median, -0.06, 0.06, color = 'black', linestyle='solid', label = 'M
plt.vlines(Mode, -0.06, 0.06, color = 'yellow', linestyle='solid', label = 'Mo
plt.vlines(Skew, -0.06, 0.06, color = 'purple', linestyle='solid', label = 'Sk
plt.legend()
plt.show()
```



In [ ]:

```
plt.figure(figsize=(10,5))
sns.rugplot(data=df1, x ="beverages")
Mean = np.mean(beverages)
Median = np.median(beverages)
Mode = df1['beverages'].mode()[0]
Stddev = df1['beverages'].std()
Skew = (Mean - Mode)/Stddev
plt.vlines(Mean, -0.06, 0.06, color = 'red', linestyle='solid', label = 'Mean
plt.vlines(Median, -0.06, 0.06, color = 'black', linestyle='solid', label = 'M
plt.vlines(Mode, -0.06, 0.06, color = 'yellow', linestyle='solid', label = 'Mo
plt.vlines(Skew, -0.06, 0.06, color = 'purple', linestyle='solid', label = 'Sk
plt.legend()
plt.show()
```

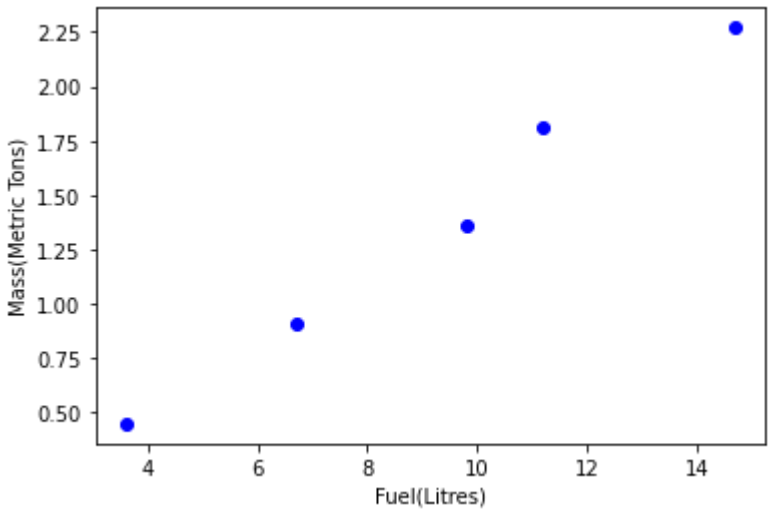


Question 4 - A car company wants to predict how much fuel different cars will use based on their masses. They took a sample of cars, drove each car 100km, and measured how much fuel was used in each case (in litres). Visualize the data using scatterplot and also find co-relation between the 2 variables (eg. Positive//Negative, Linear/ Non- linear co-relation) The data is summarized in the table below. (Use a reasonable scale on both axes and put the explanatory variable on the x-axis.) Fuel used (L) 3.6 6.7 9.8 11.2 14.7 Mass (metric tons) 0.45 0.91 1.36 1.81 2.27

In [ ]:
```
from matplotlib import pyplot as plt
import pandas as pd
import numpy as np
from scipy.stats import pearsonr
```

In [ ]:
```
fuel = [3.6, 6.7, 9.8, 11.2, 14.7]
mass = [0.45, 0.91, 1.36, 1.81, 2.27]
```

In [ ]:
```
plt.scatter(fuel, mass, c="blue")
plt.xlabel("Fuel(Litres)")
plt.ylabel("Mass(Metric Tons)")
plt.show()
```



In [ ]:
```
Correlation, _ = pearsonr(fuel, mass)
```

In [ ]:
```
Correlation
```

Out[ ]:
```
0.9938681082455859
```

The correlation between the 2 variables is positive. As the correlation coefficient is very close to 1, this suggests a highly linear relationship.

Question 5 - The data below represents the number of chairs in each class of a government high school. Create a box plot and swarm plot (add jitter) and find the number of data points that are outliers. 35, 54, 60, 65, 66, 67, 69, 70, 72, 73, 75, 76, 54, 25, 15, 60, 65, 66, 67, 69, 70, 72, 130, 73, 75, 76

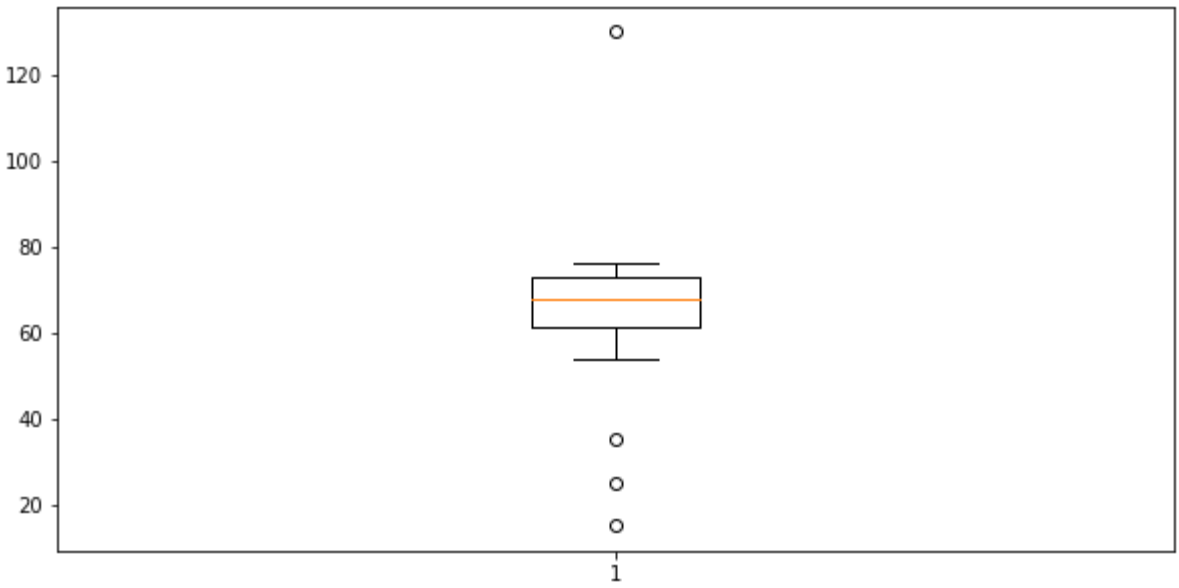
In [ ]:
```
from matplotlib import pyplot as plt
import numpy as np
import seaborn as sns
import pandas as pd
```

In [ ]:
```
chairs = [35, 54, 60, 65, 66, 67, 69, 70, 72, 73, 75, 76, 54, 25, 15, 60, 65,
```

In [ ]:
```
df = pd.DataFrame(chairs, columns =  ['chairs'])
```

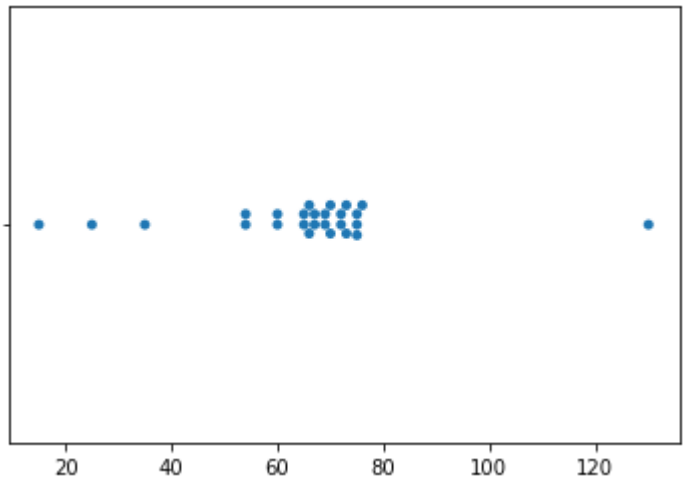
In descriptive statistics, a box plot or boxplot is a method for graphically demonstrating the locality, spread and skewness groups of numerical data through their quartiles. In addition to the box on a box plot, there can be lines (which are called whiskers) extending from the box indicating variability outside the upper and lower quartiles, thus, the plot is also termed as the box-and-whisker plot and the box-and-whisker diagram. Outliers that differ significantly from the rest of the dataset may be plotted as individual points beyond the whiskers on the box-plot. Box plots are non-parametric: they display variation in samples of a statistical population without making any assumptions of the underlying statistical distribution (though Tukey's boxplot assumes symmetry for the whiskers and normality for their length). The spacings in each subsection of the box-plot indicate the degree of dispersion (spread) and skewness of the data, which are usually described using the five-number summary. In addition, the box-plot allows one to visually estimate various L-estimators, notably the interquartile range, midhinge, range, mid-range, and trimean. Box plots can be drawn either horizontally or vertically.

In [ ]:
```
fig = plt.figure(figsize=(10, 5))
plt.boxplot(chairs)
plt.show()
```



In [ ]:
```
sns.swarmplot(x=chairs)
```

Out[ ]: <AxesSubplot:>



Boxplot and swarmplot together with added jitter will look like the following

In [ ]:
```
ax = sns.boxplot(x='chairs', data=df)
ax = sns.swarmplot(x='chairs', data=df, color="grey")
plt.show()
```

Question 6 - Generate random numbers from the following distribution and visualize the data using violin plot. (i) Standard-Normal distribution. (ii) Log-Normal distribution.

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

In [ ]:
```
sn = np.random.normal(size=30)
ln = np.random.lognormal(3, 1, 30)
```

In [ ]:
```
sn
```

Out[ ]:
```
array([ 0.28541713, -0.22531605,  1.44666482, -0.57507284, -0.77195104,
       -0.18313683, -0.77114821, -1.6690251 , -1.09280601,  0.86323237,
       -0.93826435,  0.39453643,  1.47119839, -0.42240933,  0.81215559,
       -2.69948229, -0.65921609,  1.13511661,  0.50853072,  0.23665931,
        0.41525593,  0.91644692,  0.32870781,  0.46482661,  0.07584104,
        0.62611901,  0.59746301, -1.12369596, -1.45343376, -1.19338371])
```

In [ ]:
```
ln
```

Out[ ]:
```
array([ 59.24987514,  29.86976173,   9.70561493,   2.31917173,
         5.77165856,  20.52971338,   6.08510627,  58.81020413,
        29.30329401,   5.25904488,   3.14932901,  32.08563382,
        33.02622521,   9.3130104 ,  49.66563834,  12.66931881,
        16.16901541, 163.3886737 ,   5.10535737,  13.81986365,
        29.11496413,   0.874934  ,   2.43336628,  32.64475358,
        15.71152048,   9.50992083,  12.6915328 ,   3.38226099,
        14.4103571 ,  55.8233054 ])
```

In [ ]:
```
df = pd.DataFrame(sn, columns =  ['Standard Normal'])
df1 = pd.DataFrame(ln, columns =  ['Log Normal'])
```

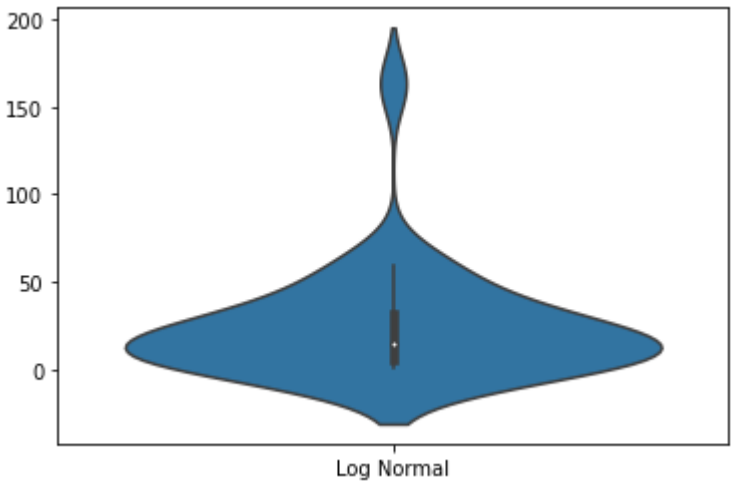
Here are the violin plots for the distributions given above :-

For the Log Normal Distribution; In probability theory, a log-normal (or lognormal) distribution is a continuous probability distribution of a random variable whose logarithm is normally distributed. Thus, if the random variable X is log-normally distributed, then Y = ln(X) has a normal distribution.Equivalently, if Y has a normal distribution, then the exponential function of Y, X = exp(Y), has a log-normal distribution. A random variable which is log-normally distributed takes only positive real values. It is a convenient and useful model for measurements in exact and engineering sciences, as well as medicine, economics and other topics (e.g., energies, concentrations, lengths, financial returns and other metrics).

The distribution is occasionally referred to as the Galton distribution or Galton's distribution, after Francis Galton. The log-normal distribution has also been associated with other names, such as McAlister, Gibrat and Cobb–Douglas.

In [ ]:
```
sns.violinplot(data = df1)
```

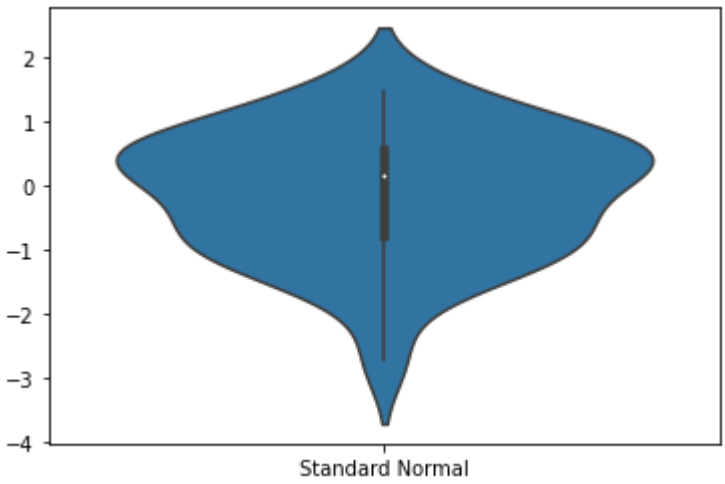
Out[ ]: <AxesSubplot:>



For the Standard Normal Distribution; In probability theory, a normal (or Gaussian or Gauss or Laplace–Gauss) distribution is a type of continuous probability distribution for a real-valued random variable. Normal distributions are important in statistics and are often used in the natural and social sciences to represent real-valued random variables whose distributions are not known. Their importance is partly due to the central limit theorem. It states that, under some conditions, the average of many samples (observations) of a random variable with finite mean and variance is itself a random variable—whose distribution converges to a normal distribution as the number of samples increases. Therefore, physical quantities that are expected to be the sum of many independent processes, such as measurement errors, often have distributions that are nearly normal.

In [ ]:
```
sns.violinplot(data = df)
```

Out[ ]: <AxesSubplot:>



Question 7 - An Advertisement agency develops new ads for various clients (like Jewellery shops, Textile shops).The Agency wants to assess their performance, for which they want to know the number of ads they developed in each quarter for different shop category. Help them to visualize data using radar/spider charts. ShopCategory Quarter 1 Quarter 2 Quarter 3 Quarter 4 Textile 10 6 8 13 Jewellery 5 5 2 4 CleaningEssentials 15 20 16 15 Cosmetics 14 10 21 11

In [ ]:
```
from matplotlib import pyplot as plt
import pandas as pd
from math import pi
```

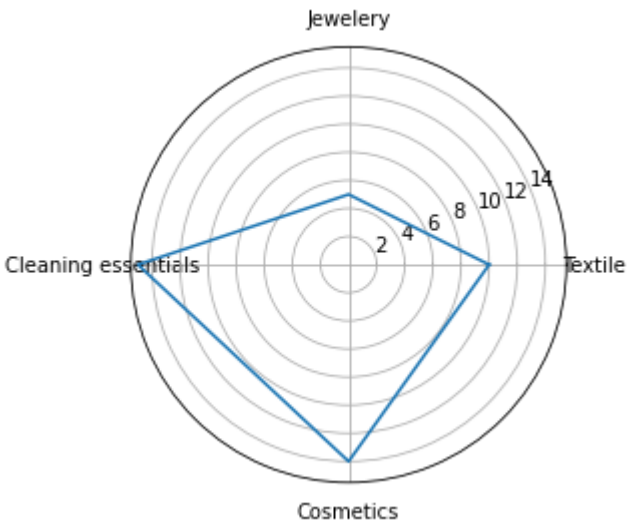
In [ ]:
```
shop_category = ['Textile', 'Jewelery', 'Cleaning essentials', 'Cosmetics']
N = len(shop_category)
```

QUARTER 1

In [ ]:
```
values1 = [10, 5, 15, 14]
values1 += values1[:1]
```

In [ ]:
```
angles1 = [n/float(N)*2*pi for n in range(N)]
angles1 += angles1[:1]
```

In [ ]:
```
plt.polar(angles1, values1)
plt.xticks(angles1[:-1], shop_category)
plt.show()
```

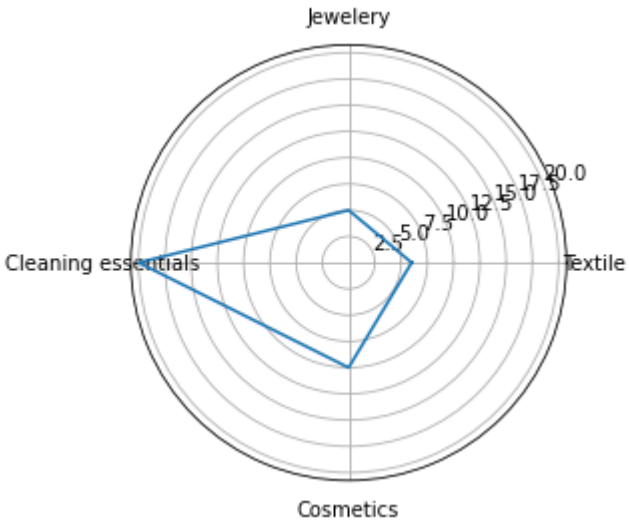


QUARTER 2

In [ ]:
```
values2 = [6, 5, 20, 10]
values2 += values2[:1]
```

In [ ]:
```
angles2 = [n/float(N)*2*pi for n in range(N)]
angles2 += angles2[:1]
```

In [ ]:
```
plt.polar(angles2, values2)
plt.xticks(angles2[:-1], shop_category)
plt.show()
```

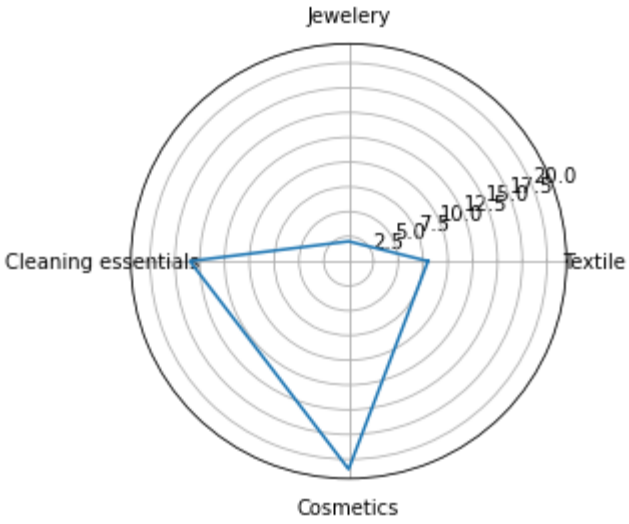


QUARTER 3

In [ ]:
```
values3 = [8, 2, 16, 21]
values3 += values3[:1]
```

In [ ]:
```
angles3 = [n/float(N)*2*pi for n in range(N)]
angles3 += angles3[:1]
```

In [ ]:
```
plt.polar(angles3, values3)
plt.xticks(angles3[:-1], shop_category)
plt.show()
```

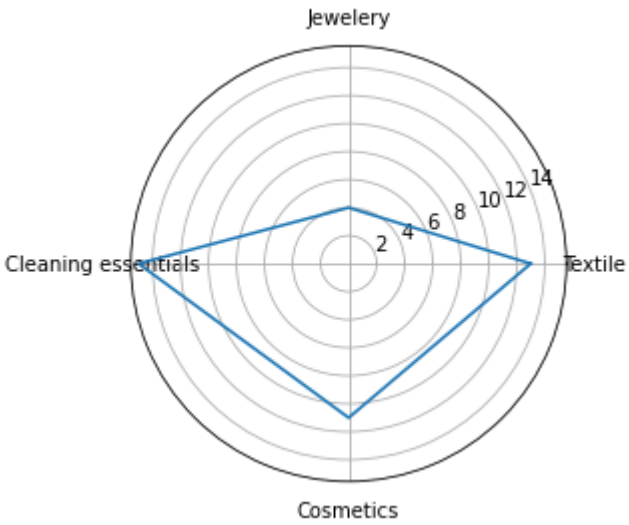


QUARTER 4

In [ ]:
```
values4 = [13, 4, 15, 11]
values4 += values4[:1]
```

In [ ]:
```
angles4 = [n/float(N)*2*pi for n in range(N)]
angles4 += angles4[:1]
```

In [ ]:
```
plt.polar(angles4, values4)
plt.xticks(angles4[:-1], shop_category)
plt.show()
```



Question 8 - An organization wants to calculate the % of time they spent on each process for their product development. Visualize the data using funnel chart with the data given below.

ProductDevelopmentSteps TimeSpent(in hours) RequirementElicitation 50 RequirementAnalysis 110 SoftwareDevelopment 250 Debugging&Testing 180 Others 70

In [ ]:
```
pip install pygal
```

```
Collecting pygal
  Downloading pygal-3.0.0-py2.py3-none-any.whl (129 kB)
Installing collected packages: pygal
Successfully installed pygal-3.0.0
Note: you may need to restart the kernel to use updated packages.
```

In [ ]:
```
pip install cairosvg
```

```
Collecting cairosvg
  Downloading CairoSVG-2.5.2-py3-none-any.whl (45 kB)
Collecting tinycss2
  Downloading tinycss2-1.1.1-py3-none-any.whl (21 kB)
Requirement already satisfied: defusedxml in c:\users\hp\anaconda3\lib\site-pa
ckages (from cairosvg) (0.7.1)
Collecting cairocffi
  Downloading cairocffi-1.3.0.tar.gz (88 kB)
Requirement already satisfied: pillow in c:\users\hp\anaconda3\lib\site-packag
es (from cairosvg) (8.2.0)
Collecting cssselect2
  Downloading cssselect2-0.4.1-py3-none-any.whl (13 kB)
Requirement already satisfied: cffi>=1.1.0 in c:\users\hp\anaconda3\lib\site-p
ackages (from cairocffi->cairosvg) (1.14.5)
Requirement already satisfied: pycparser in c:\users\hp\anaconda3\lib\site-pac
kages (from cffi>=1.1.0->cairocffi->cairosvg) (2.20)
Requirement already satisfied: webencodings in c:\users\hp\anaconda3\lib\site-
packages (from cssselect2->cairosvg) (0.5.1)
Building wheels for collected packages: cairocffi
  Building wheel for cairocffi (setup.py): started
  Building wheel for cairocffi (setup.py): finished with status 'done'
  Created wheel for cairocffi: filename=cairocffi-1.3.0-py3-none-any.whl size=
89666 sha256=597e73c09e250a4a8d032712fe341ef78bd3e1e538fc617695ddf9b9c4d74caa
  Stored in directory: c:\users\hp\appdata\local\pip\cache\wheels\f3\97\83\802
2b9237866102e18d1b7ac0a269769e6fccba0f63dceb9b7
Successfully built cairocffi
Installing collected packages: tinycss2, cssselect2, cairocffi, cairosvg
Successfully installed cairocffi-1.3.0 cairosvg-2.5.2 cssselect2-0.4.1 tinycss
2-1.1.1
Note: you may need to restart the kernel to use updated packages.
```

In [ ]:
```
import pygal
import cairosvg
```

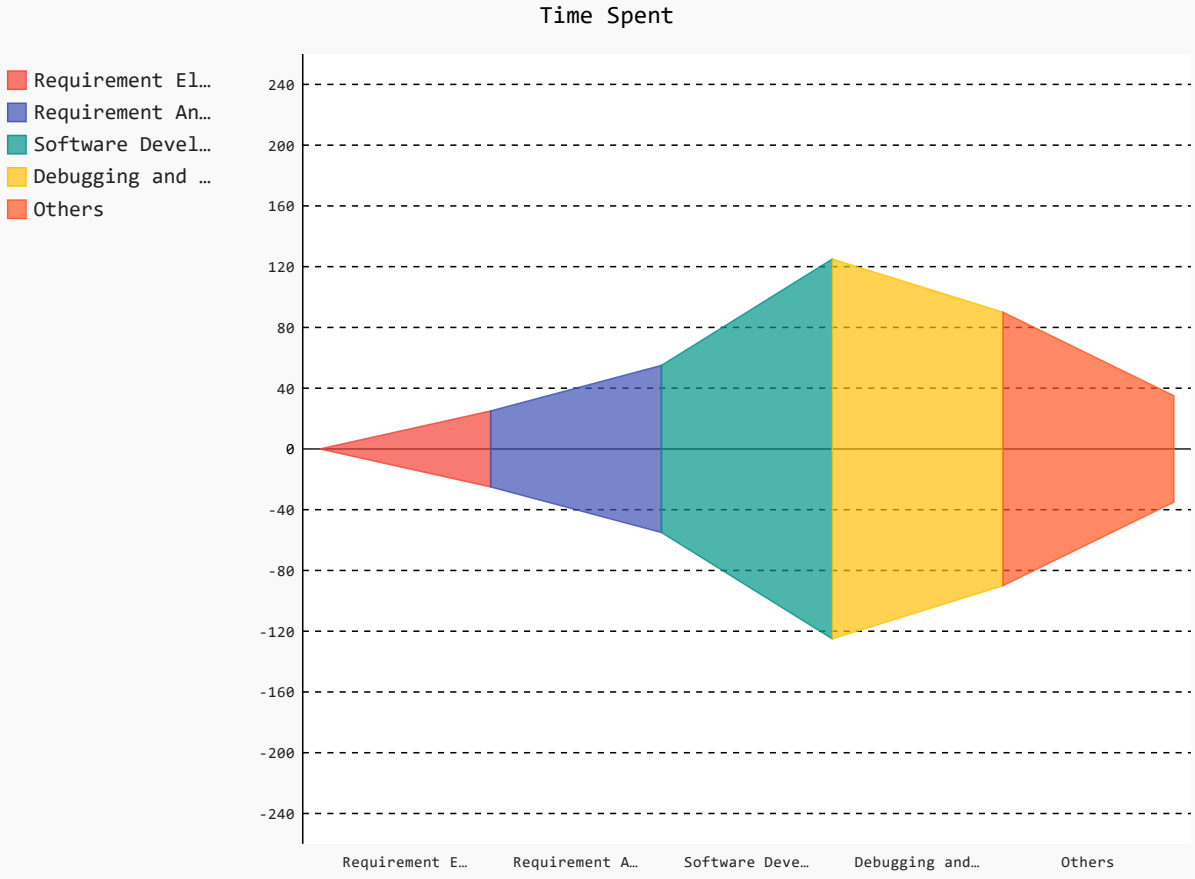
In [ ]:
```
funnel1 = pygal.Funnel()
```

In [ ]:
```
funnel1.title = 'Time Spent'
```

In [ ]:

```
funnel1.add('Requirement Elicitation', [50])
funnel1.add('Requirement Analysis', [110])
funnel1.add('Software Development', [250])
funnel1.add('Debugging and Testing', [180])
funnel1.add('Others', [70])
```

Out[ ]:



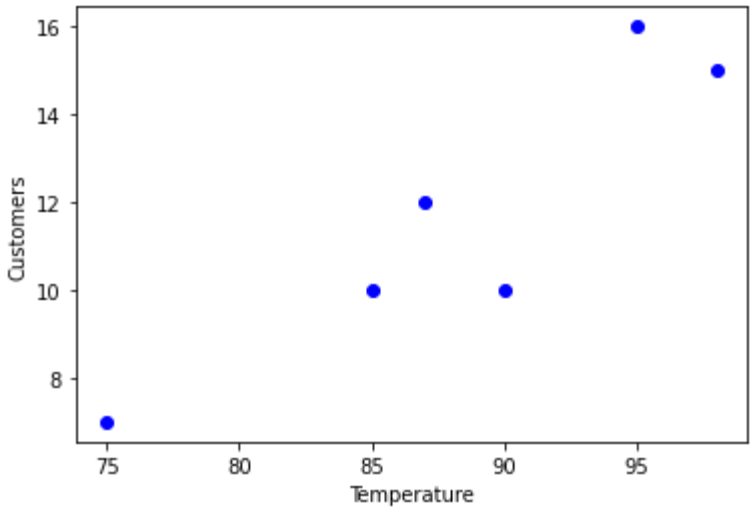
Question 9 - Let's say you are the new owner of a small ice-cream shop in a little village near the beach. You noticed that there was more business in the warmer months than the cooler months. Before you alter your purchasing pattern to match this trend, you want to be sure that the relationship is real. Help him to find the correlation between the data given.

Temperature Number of Customers 98 15 87 12 90 10 85 10 95 16 75 7

In [ ]:
```
from matplotlib import pyplot as plt
import pandas as pd
import numpy as np
from scipy.stats import pearsonr
```

In [ ]:
```
Temperature = [98, 87, 90, 85, 95, 75]
Customers = [15, 12, 10, 10, 16, 7]
```

In [ ]:
```
plt.scatter(Temperature, Customers, c="blue")
plt.xlabel("Temperature")
plt.ylabel("Customers")
plt.show()
```



In [ ]:
```
Correlation, _ = pearsonr(Temperature, Customers)
```

In [ ]:
```
Correlation
```

Out[ ]:
```
0.9117671365080744
```

The correlation between the 2 variables is positive. As the correlation coefficient is very close to 1, this suggests a highly linear relationship.