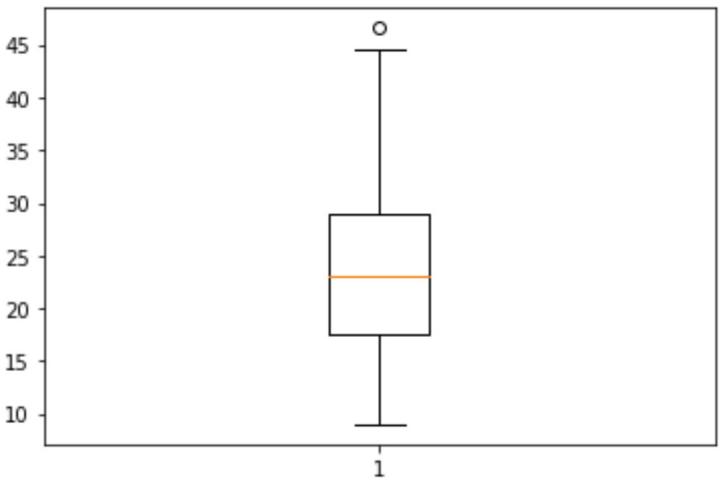
CH-2 Data Visualization Using Python

Box Plot:

- Whisker's plot

In [1]:
```
1 import numpy as np
2 import pandas as pd
3 import matplotlib.pyplot as plt
4 df = pd.read_csv("auto-mpg.csv")
5 df
```

Out[1]:
```
{'whiskers': [<matplotlib.lines.Line2D at 0x22577f70fd0>,
  <matplotlib.lines.Line2D at 0x225781de370>],
 'caps': [<matplotlib.lines.Line2D at 0x225781de6d0>,
  <matplotlib.lines.Line2D at 0x225781dea30>],
 'boxes': [<matplotlib.lines.Line2D at 0x22577f70c70>],
 'medians': [<matplotlib.lines.Line2D at 0x225781ded90>],
 'fliers': [<matplotlib.lines.Line2D at 0x225781ea130>],
 'means': []}
```

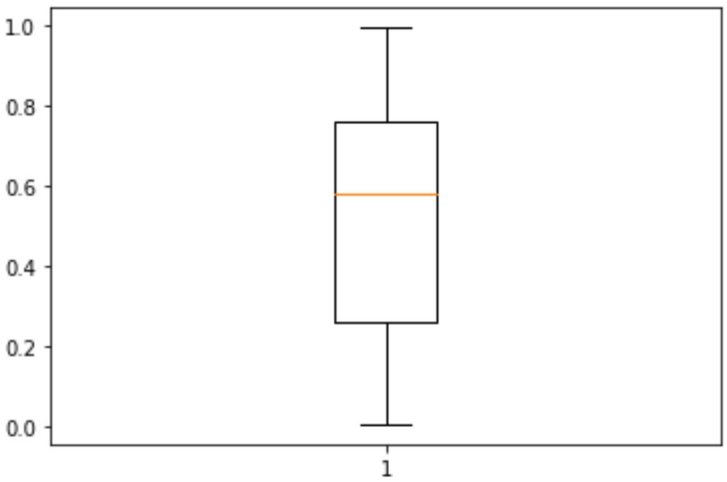


In [2]:
```
help(plt.boxplot)
# width : 0.5 or 0.15
```

```
Help on function boxplot in module matplotlib.pyplot:

boxplot(x, notch=None, sym=None, vert=None, whis=None, positions=None, widths
=None, patch_artist=None, bootstrap=None, usermedians=None, conf_intervals=No
ne, meanline=None, showmeans=None, showcaps=None, showbox=None, showfliers=No
ne, boxprops=None, labels=None, flierprops=None, medianprops=None, meanprops=
None, capprops=None, whiskerprops=None, manage_ticks=True, autorange=False, z
order=None, *, data=None)
    Make a box and whisker plot.

    Make a box and whisker plot for each column of *x* or each
    vector in sequence *x*.  The box extends from the lower to
    upper quartile values of the data, with a line at the median.
    The whiskers extend from the box to show the range of the
    data.  Flier points are those past the end of the whiskers.

    Parameters
    ----------
    x : Array or a sequence of vectors.
        The input data
```

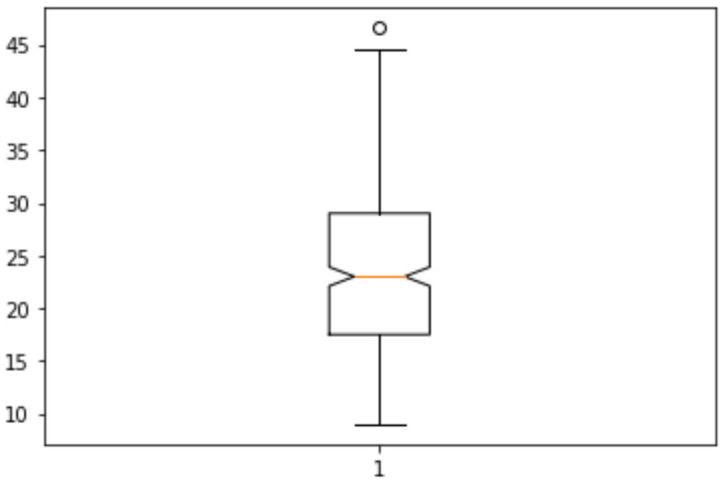
In [3]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

data = np.random.rand(100)
print(data)
plt.boxplot(data)
```

```
[0.82968505 0.36057129 0.77777012 0.99485711 0.09041484 0.02015761
 0.78447233 0.54911749 0.47192166 0.5039709  0.65684149 0.57989187
 0.66969307 0.86011954 0.79360955 0.25138736 0.73910521 0.75328555
 0.25355241 0.60185618 0.51699512 0.1511366  0.70297697 0.57782559
 0.19270642 0.03375608 0.63210205 0.70051695 0.38242372 0.37607912
 0.73799644 0.15745896 0.64224013 0.29722641 0.00484674 0.98538454
 0.50274953 0.7786209  0.44280167 0.9111781  0.87911    0.20922115
 0.2220702  0.92322527 0.86429382 0.74036905 0.72261893 0.52133567
 0.73519142 0.50923867 0.06610264 0.92777178 0.00835018 0.16251521
 0.93572285 0.43234105 0.36156816 0.04180432 0.92999235 0.24484973
 0.50634794 0.88613529 0.49354593 0.58239639 0.95108824 0.12576052
 0.7949263  0.67981929 0.85237318 0.2628744  0.56698735 0.32083499
 0.17407702 0.04591731 0.24654033 0.74427732 0.48213794 0.37291116
 0.44719405 0.47532649 0.58868093 0.74021452 0.19633756 0.73681538
 0.09695305 0.3042131  0.12901199 0.20886688 0.84444848 0.72561328
 0.65529822 0.72749107 0.67522356 0.98806165 0.65852323 0.87991082
 0.91902461 0.9862638  0.04359083 0.8894698 ]
```



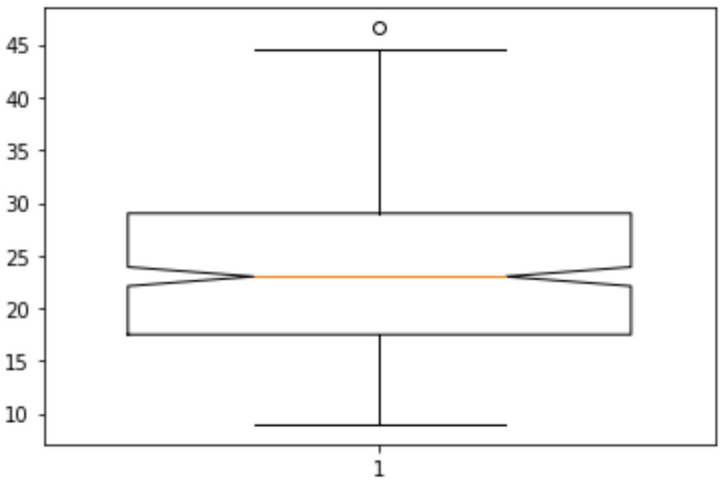
In [4]:
```
1 import numpy as np
2 import pandas as pd
3 import matplotlib.pyplot as plt
4 df = pd.read_csv("auto-mpg.csv")
5 df
```

Out[4]:
```
{'whiskers': [<matplotlib.lines.Line2D at 0x225783302b0>,
  <matplotlib.lines.Line2D at 0x22578330610>],
 'caps': [<matplotlib.lines.Line2D at 0x22578330970>,
  <matplotlib.lines.Line2D at 0x22578330cd0>],
 'boxes': [<matplotlib.lines.Line2D at 0x22578323f10>],
 'medians': [<matplotlib.lines.Line2D at 0x2257833f070>],
 'fliers': [<matplotlib.lines.Line2D at 0x2257833f3d0>],
 'means': []}
```



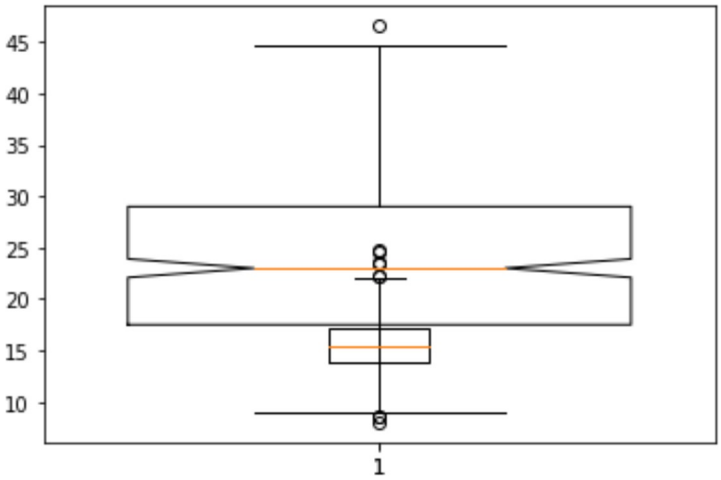
In [5]:
```
1 import numpy as np
2 import pandas as pd
3 import matplotlib.pyplot as plt
4 df = pd.read_csv("auto-mpg.csv")
5 df
```

Out[5]:
```
{'whiskers': [<matplotlib.lines.Line2D at 0x22578391910>,
  <matplotlib.lines.Line2D at 0x22578391c70>],
 'caps': [<matplotlib.lines.Line2D at 0x22578391fd0>,
  <matplotlib.lines.Line2D at 0x2257839f370>],
 'boxes': [<matplotlib.lines.Line2D at 0x225783915b0>],
 'medians': [<matplotlib.lines.Line2D at 0x2257839f6d0>],
 'fliers': [<matplotlib.lines.Line2D at 0x2257839fa30>],
 'means': []}
```



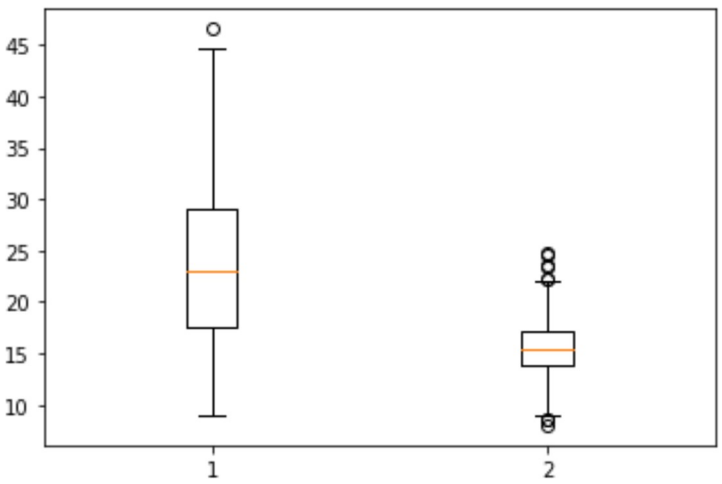
In [6]:
```
1 import numpy as np
2 import pandas as pd
3 import matplotlib.pyplot as plt
4 df = pd.read_csv("auto-mpg.csv")
5 df
6 plt.boxplot(df['mpg'],notch=True,widths=0.75)
```

Out[6]:
```
{'whiskers': [<matplotlib.lines.Line2D at 0x2257840b310>,
  <matplotlib.lines.Line2D at 0x2257840b670>],
 'caps': [<matplotlib.lines.Line2D at 0x2257840b9d0>,
  <matplotlib.lines.Line2D at 0x2257840bd30>],
 'boxes': [<matplotlib.lines.Line2D at 0x22578402f70>],
 'medians': [<matplotlib.lines.Line2D at 0x225784170d0>],
 'fliers': [<matplotlib.lines.Line2D at 0x22578417430>],
 'means': []}
```



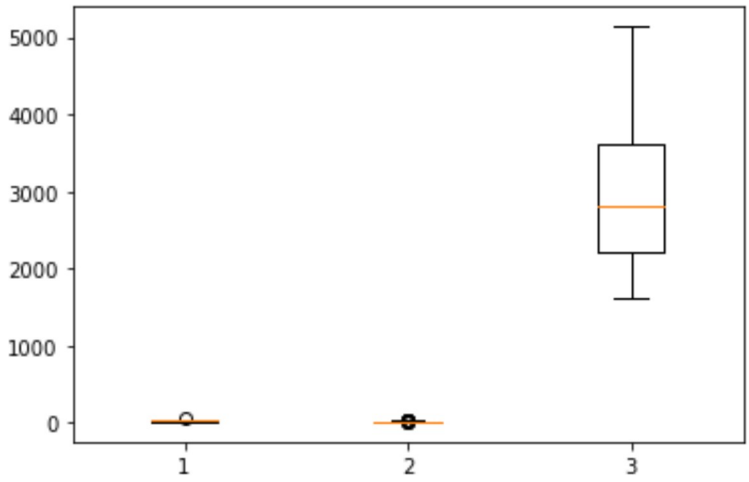
In [8]:
```
1 import numpy as np
2 import pandas as pd
3 import matplotlib.pyplot as plt
4 df = pd.read_csv("auto-mpg.csv")
5 df
6 plt.boxplot(df[['mpg','acceleration']])
```

Out[8]:
```
{'whiskers': [<matplotlib.lines.Line2D at 0x22578432a60>,
  <matplotlib.lines.Line2D at 0x225783b2e80>,
  <matplotlib.lines.Line2D at 0x2257843b760>,
  <matplotlib.lines.Line2D at 0x225782cffd0>],
 'caps': [<matplotlib.lines.Line2D at 0x2257842b3d0>,
  <matplotlib.lines.Line2D at 0x2257842ba30>,
  <matplotlib.lines.Line2D at 0x225782cf100>,
  <matplotlib.lines.Line2D at 0x225782cfd30>],
 'boxes': [<matplotlib.lines.Line2D at 0x22578432280>,
  <matplotlib.lines.Line2D at 0x2257822c730>],
 'medians': [<matplotlib.lines.Line2D at 0x2257863dbe0>,
  <matplotlib.lines.Line2D at 0x225782cf3a0>],
 'fliers': [<matplotlib.lines.Line2D at 0x2257863d1c0>,
  <matplotlib.lines.Line2D at 0x225782cfbb0>],
 'means': []}
```



In [9]:
```
1 import numpy as np
2 import pandas as pd
3 import matplotlib.pyplot as plt
4 df = pd.read_csv("auto-mpg.csv")
5 df
```

Out[9]:
```
{'whiskers': [<matplotlib.lines.Line2D at 0x22578689250>,
  <matplotlib.lines.Line2D at 0x225786895b0>,
  <matplotlib.lines.Line2D at 0x225786969d0>,
  <matplotlib.lines.Line2D at 0x22578696d30>,
  <matplotlib.lines.Line2D at 0x225786ad1f0>,
  <matplotlib.lines.Line2D at 0x225786ad550>],
 'caps': [<matplotlib.lines.Line2D at 0x22578689910>,
  <matplotlib.lines.Line2D at 0x22578689c70>,
  <matplotlib.lines.Line2D at 0x225786a20d0>,
  <matplotlib.lines.Line2D at 0x225786a2430>,
  <matplotlib.lines.Line2D at 0x225786ad8b0>,
  <matplotlib.lines.Line2D at 0x225786adc10>],
 'boxes': [<matplotlib.lines.Line2D at 0x2257867beb0>,
  <matplotlib.lines.Line2D at 0x225786966d0>,
  <matplotlib.lines.Line2D at 0x225786a2e50>],
 'medians': [<matplotlib.lines.Line2D at 0x22578689fd0>,
  <matplotlib.lines.Line2D at 0x225786a2790>,
  <matplotlib.lines.Line2D at 0x225786adf70>],
 'fliers': [<matplotlib.lines.Line2D at 0x22578696370>,
  <matplotlib.lines.Line2D at 0x225786a2af0>,
  <matplotlib.lines.Line2D at 0x225786b9310>],
 'means': []}
```



Scatterplot:

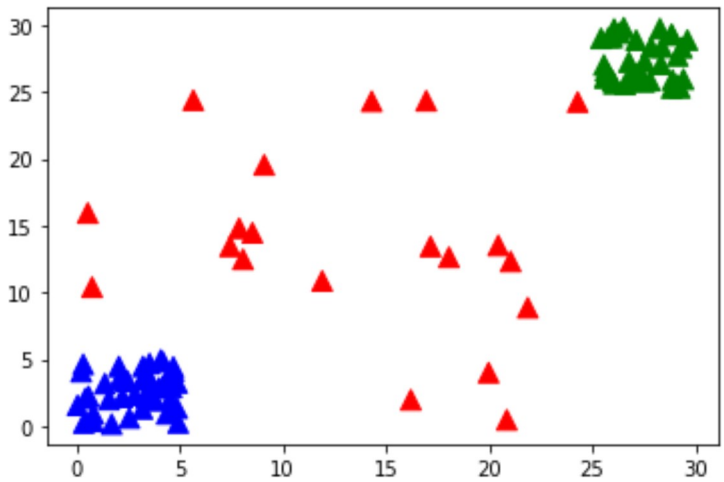
In [14]:

```
Help on function scatter in module matplotlib.pyplot:

scatter(x, y, s=None, c=None, marker=None, cmap=None, norm=None, vmin=None, v
max=None, alpha=None, linewidths=None, verts=<deprecated parameter>, edgecolo
rs=None, *, plotnonfinite=False, data=None, **kwargs)
    A scatter plot of *y* vs. *x* with varying marker size and/or color.

    Parameters
    ----------
    x, y : float or array-like, shape (n, )
        The data positions.

    s : float or array-like, shape (n, ), optional
        The marker size in points**2.
        Default is ``rcParams['lines.markersize'] ** 2``.

    c : array-like or list of colors or color, optional
        The marker colors. Possible values:
```

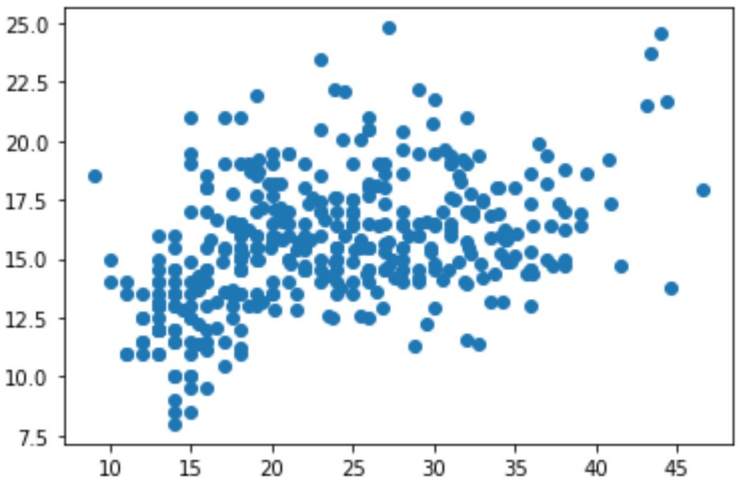
In [13]:

```
import numpy as np
import pandas as pd
x1 = 5*np.random.rand(40)
print(x1)
x2 = 5*np.random.rand(40)+25
print(x2)
x3 = 25*np.random.rand(20)
print(x3)
x = np.concatenate((x1,x2,x3))
y1 = 5*np.random.rand(40)
print(y1)
y2 = 5*np.random.rand(40)+25
print(y2)
y3 = 25*np.random.rand(20)
print(y3)
y = np.concatenate((y1,y2,y3))
color_array = ['b']*40 + ['g']*40 + ['r']*20

plt.scatter(x,y,s=[100],marker="^",c=color_array)
```

```
[2.49941886 4.55104575 3.39874515 4.09793137 3.22047741 2.83809128
 4.62881089 2.09525774 3.2360434  4.70467929 2.2387315  3.38063968
 4.19684361 4.890737   3.20917618 0.34104993 0.82938879 4.72086741
 4.94040537 3.551776   1.72011814 0.36797168 2.06726643 0.24436204
 0.06444463 1.58951574 1.3879297  0.82005413 4.20013507 3.49198092
 4.38462326 2.57377525 4.00673848 2.39738847 4.36652882 0.45286676
 0.60116404 3.82111906 4.88670945 3.52465593]
[29.24240064 27.13127322 26.94366926 27.12874843 28.32824268 26.05889767
 26.04861776 27.51362196 28.27934377 25.42538273 25.64239681 28.3755955
 29.60871314 25.56502779 28.85025733 25.98832195 28.84964973 25.64557648
 27.79505347 26.52370703 26.59565503 25.97112308 25.7967817  25.61025317
 28.84163757 26.86701675 29.41541759 27.26034482 28.8923575  29.140336
 25.79189854 27.84912744 26.07018873 27.35346682 26.32943359 26.77919585
 25.73052366 29.33932224 29.12594398 27.53899139]
[20.44810658 14.3159099  24.29321285 21.87689231  0.56042473 19.97816347
  8.5342266  11.91688897 18.05862878  7.89322276 16.21121688 16.95852636
  5.67359461  8.08087113  0.76932858 20.86015785  7.45625532 21.04739468
  9.10745551 17.16503142]
[3.33522478 1.9931285  2.12607955 4.93459013 3.42309688 2.30034613
 2.87973034 3.51177661 4.4569968  4.42139687 2.15128323 3.99293696
 4.80261788 1.32987344 1.27733834 4.61443883 0.90899463 4.11457638
 0.22600831 4.66924927 0.13371207 0.21842398 4.44838774 4.06498165
 1.5456426  1.97557172 3.16894775 0.33865928 2.97785638 2.73042765
 0.90673246 0.59718917 2.84016388 3.3649912  3.31150266 2.07398723
 2.21254327 1.89639619 3.22524344 3.53533837]
[25.3133137  26.37397809 26.39329553 28.88631025 27.07337159 29.71392135
 25.61888944 25.76397201 29.7454568  29.01904689 26.5907974  28.36297035
 28.89433596 27.02712302 29.29720794 25.83999942 25.29701881 26.11132228
 25.86045518 29.81738079 25.57174002 29.0891141  29.09072599 26.06782777
 29.29906638 25.84810213 25.96037888 26.45857751 25.91065254 27.71106573
 26.28235127 28.36374755 29.56490962 25.95946005 29.42132897 27.24446887
 26.41585202 28.32567986 25.51147441 27.28778181]
[13.51711201 24.30897884 24.25255473  8.85248749 15.94608785  3.98839908
 14.46355326 10.8556047  12.63403601 14.80219708  1.95482742 24.3509419
 24.39052745 12.48110695 10.39262847  0.4525155  13.44721435 12.31863807
 19.54764064 13.4355945 ]
```



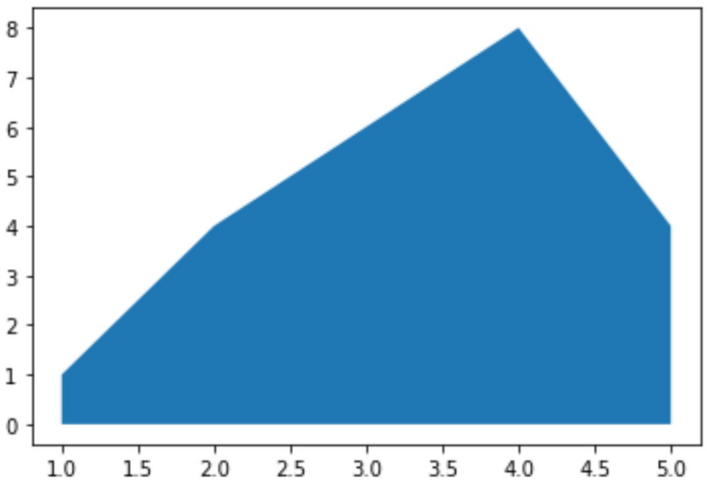
In [11]:
```
1 import numpy as np
2 import pandas as pd
3 import matplotlib.pyplot as plt
4 df = pd.read_csv("auto-mpg.csv")
5 plt.scatter(df["mpg"],df['acceleration'])
```



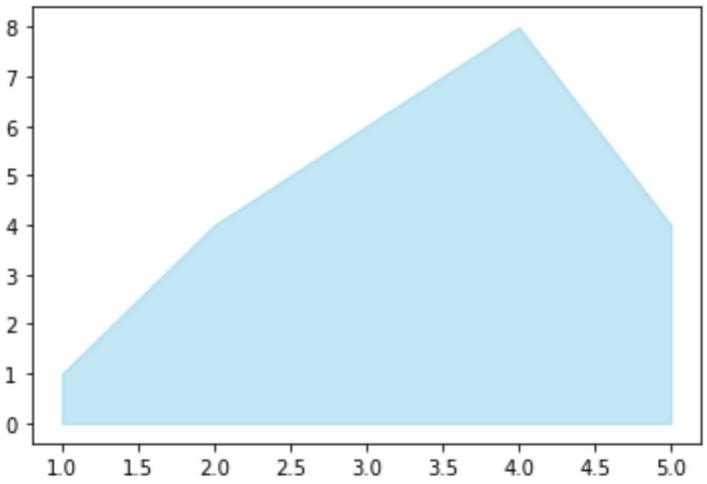
Area Plot:

In [15]:
```
Help on function fill_between in module matplotlib.pyplot:

fill_between(x, y1, y2=0, where=None, interpolate=False, step=None, *, data=N
one, **kwargs)
    Fill the area between two horizontal curves.

    The curves are defined by the points (*x*, *y1*) and (*x*,
    *y2*).  This creates one or multiple polygons describing the filled
    area.

    You may exclude some horizontal sections from filling using *where*.

    By default, the edges connect the given points directly.  Use *step*
    if the filling should be a step function, i.e. constant in between
    *x*.

    Parameters
    ----------
    x : array (length N)
```

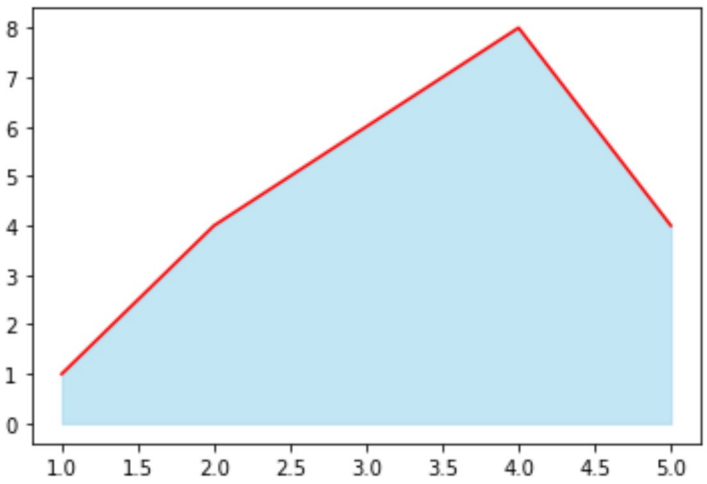
In [16]:
```
1 import numpy as np
2 import pandas as pd
3 import matplotlib.pyplot as plt
4
5 x = range(1,6)
6 y = [1,4,6,8,4]
7 plt.fill_between(x,y)
```



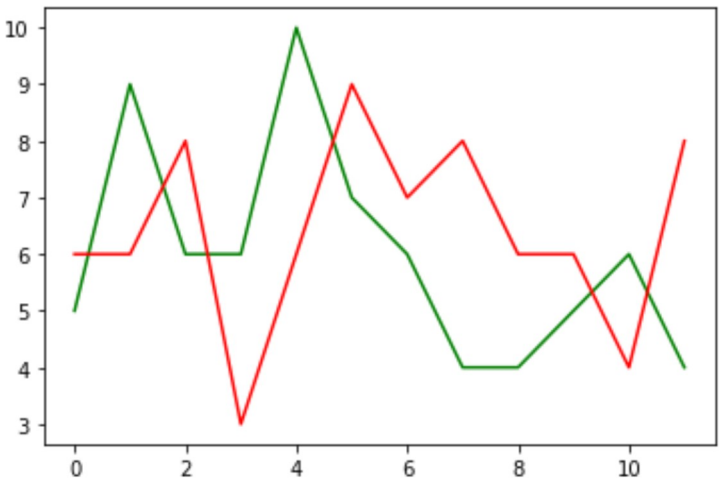
In [18]:
```
1 import numpy as np
2 import pandas as pd
3 import matplotlib.pyplot as plt
4
5 x = range(1,6)
6 y = [1,4,6,8,4]
7 plt.fill_between(x,y,color="skyblue",alpha=0.5)
```



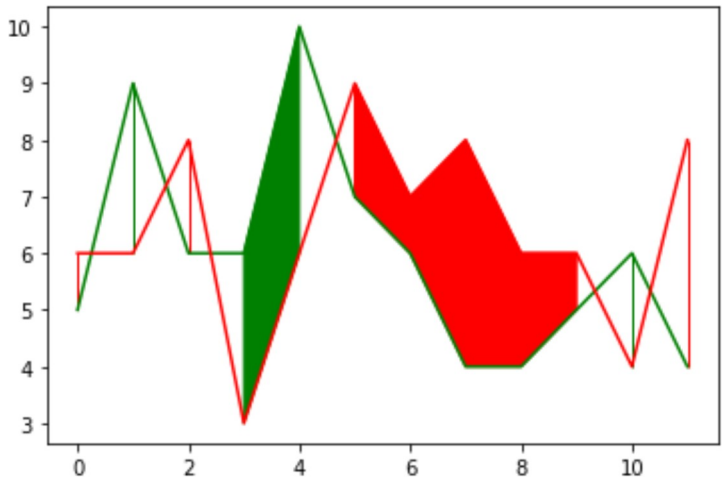
In [20]:
```
1 import numpy as np
2 import pandas as pd
3 import matplotlib.pyplot as plt
4
5 x = range(1,6)
6 y = [1,4,6,8,4]
7 plt.fill_between(x,y,color="skyblue",alpha=0.5)
8 plt.plot(x,y,color="red")
```



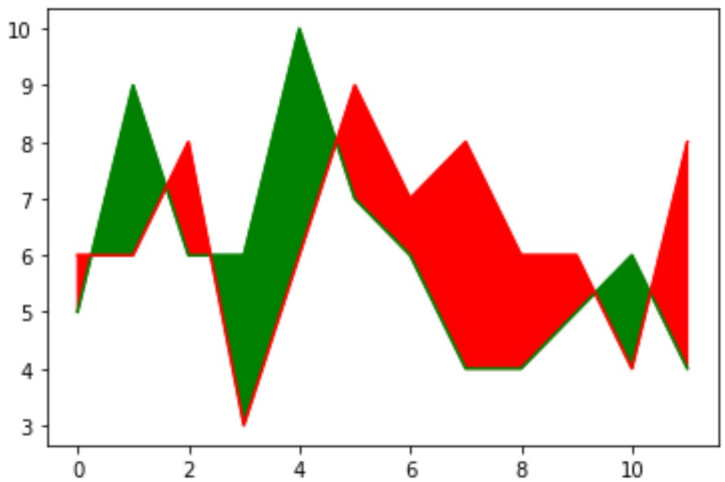
In [22]:
```
1 import numpy as np
2 import pandas as pd
3 import matplotlib.pyplot as plt
4
5 time = np.arange(12)
6 income = np.array([5,9,6,6,10,7,6,4,4,5,6,4])
7 expense = np.array([6,6,8,3,6,9,7,8,6,6,4,8])
8 plt.plot(time,income,color="green")
9 plt.plot(time,expense,color="red")
```



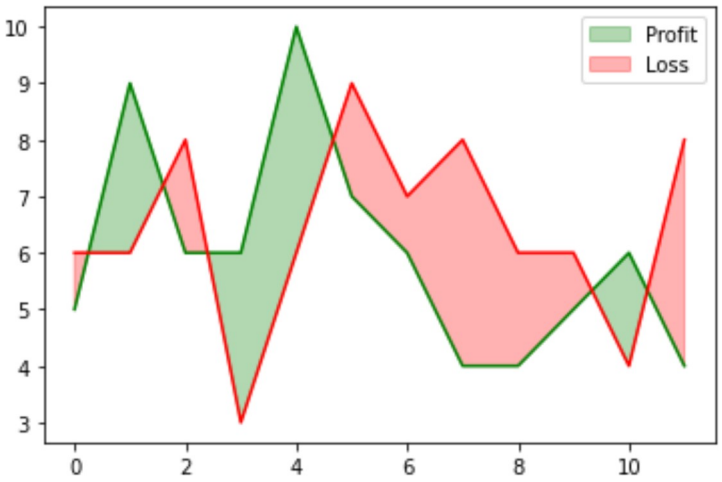
In [26]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

time = np.arange(12)
income = np.array([5,9,6,6,10,7,6,4,4,5,6,4])
expense = np.array([6,6,8,3,6,9,7,8,6,6,4,8])
plt.plot(time,income,color="green")
plt.plot(time,expense,color="red")
plt.fill_between(time,income,expense,where=(income>expense),color="green")
plt.fill_between(time,income,expense,where=(income<=expense),color="red")
```



In [27]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

time = np.arange(12)
income = np.array([5,9,6,6,10,7,6,4,4,5,6,4])
expense = np.array([6,6,8,3,6,9,7,8,6,6,4,8])
plt.plot(time,income,color="green")
plt.plot(time,expense,color="red")
plt.fill_between(time,income,expense,where=(income>expense),color="green",
plt.fill_between(time,income,expense,where=(income<=expense),color="red",
```



In [28]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

time = np.arange(12)
income = np.array([5,9,6,6,10,7,6,4,4,5,6,4])
expense = np.array([6,6,8,3,6,9,7,8,6,6,4,8])
plt.plot(time,income,color="green")
plt.plot(time,expense,color="red")
plt.fill_between(time,income,expense,where=(income>expense),color="green",
plt.fill_between(time,income,expense,where=(income<=expense),color="red",
plt.legend()
```



Stack Area Chart:

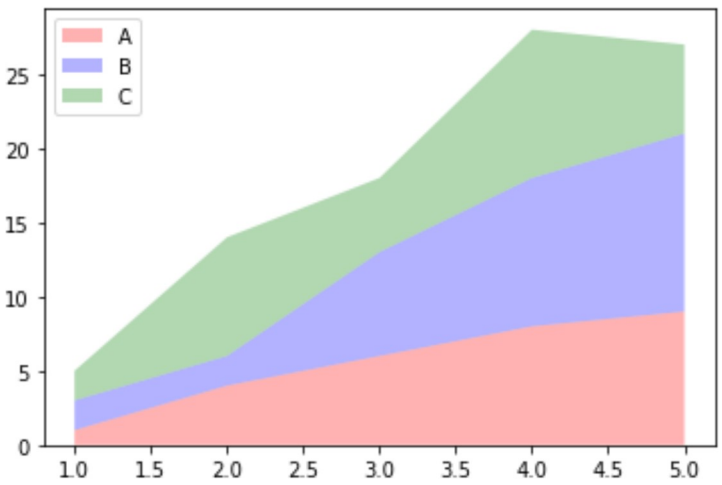
In [34]:

```
Help on function stackplot in module matplotlib.pyplot:

stackplot(x, *args, labels=(), colors=None, baseline='zero', data=None, **kwa
rgs)
    Draw a stacked area plot.

    Parameters
    ----------
    x : 1d array of dimension N

    y : 2d array (dimension MxN), or sequence of 1d arrays (each dimension 1x
N)

        The data is assumed to be unstacked. Each of the following
        calls is legal::

            stackplot(x, y)               # where y is MxN
            stackplot(x, y1, y2, y3, y4)  # where y1, y2, y3, y4, are all 1xN
m

    baseline : {'zero', 'sym', 'wiggle', 'weighted_wiggle'}
        Method used to calculate the baseline:

        - ``'zero'``: Constant zero baseline, i.e. a simple stacked plot.
        - ``'sym'``:  Symmetric around zero and is sometimes called
          'ThemeRiver'.
        - ``'wiggle'``: Minimizes the sum of the squared slopes.
        - ``'weighted_wiggle'``: Does the same but weights to account for
          size of each layer. It is also called 'Streamgraph'-layout. More
          details can be found at http://leebyron.com/streamgraph/. (http://l
eebyron.com/streamgraph/.)

    labels : Length N sequence of strings
        Labels to assign to each data series.

    colors : Length N sequence of colors
        A list or tuple of colors. These will be cycled through and used to
        colour the stacked areas.

    **kwargs
        All other keyword arguments are passed to `.Axes.fill_between`.

    Returns
    -------
    list of `.PolyCollection`
        A list of `.PolyCollection` instances, one for each element in the
        stacked area plot.

    Notes
    -----


    .. note::
        In addition to the above described arguments, this function can take
        a *data* keyword argument. If such a *data* argument is given,
        every other argument can also be string ``s``, which is
```

```
interpreted as ``data[s]`` (unless this raises an exception).

Objects passed as **data** must support item access (``data[s]``) and
membership test (``s in data``).
```

In [32]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

x = range(1,6)
y1 = [1,4,6,8,9]
y2 = [2,2,7,10,12]
y3 = [2,8,5,10,6]

plt.stackplot(x,y1,y2,y3,labels=["A","B","C"],colors=["red","blue","green"
plt.legend(loc="upper left")
```



seaborn library:

- Regression Plots

In [36]:
```
1 import seaborn as sns
```

```
Help on function regplot in module seaborn.regression:

regplot(*, x=None, y=None, data=None, x_estimator=None, x_bins=None, x_ci='ci
', scatter=True, fit_reg=True, ci=95, n_boot=1000, units=None, seed=None, ord
er=1, logistic=False, lowess=False, robust=False, logx=False, x_partial=None,
y_partial=None, truncate=True, dropna=True, x_jitter=None, y_jitter=None, lab
el=None, color=None, marker='o', scatter_kws=None, line_kws=None, ax=None)
    Plot data and a linear regression model fit.

    There are a number of mutually exclusive options for estimating the
    regression model. See the :ref:`tutorial <regression_tutorial>` for more
    information.

    Parameters
    ----------
    x, y: string, series, or vector array
        Input variables. If strings, these should correspond with column name
s
        in ``data``. When pandas objects are used, axes will be labeled with
```

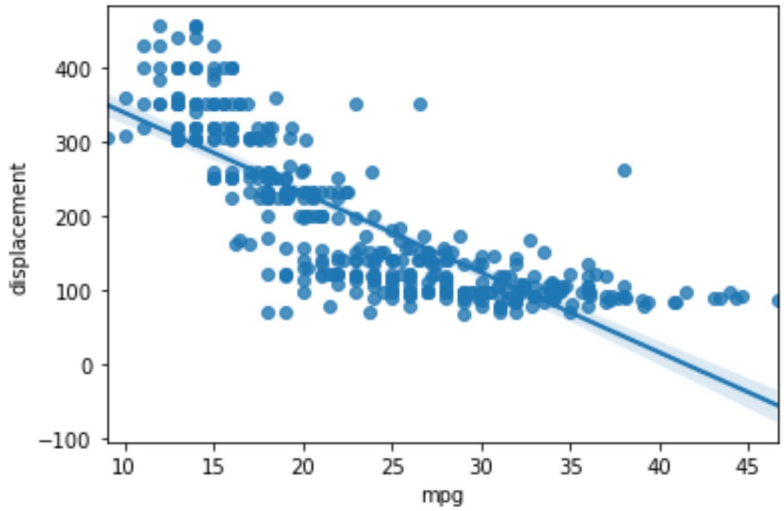
In [37]:
```
1 import numpy as np
2 import pandas as pd
3 import matplotlib.pyplot as plt
4 import seaborn as sns
5 df = pd.read_csv("auto-mpg.csv")
```

Out[37]:

	mpg	cylinders	displacement	weight	acceleration	model year	origin
mpg	1.000000	-0.775396	-0.804203	-0.831741	0.420289	0.579267	0.563450
cylinders	-0.775396	1.000000	0.950721	0.896017	-0.505419	-0.348746	-0.562543
displacement	-0.804203	0.950721	1.000000	0.932824	-0.543684	-0.370164	-0.609409
weight	-0.831741	0.896017	0.932824	1.000000	-0.417457	-0.306564	-0.581024
acceleration	0.420289	-0.505419	-0.543684	-0.417457	1.000000	0.288137	0.205873
model year	0.579267	-0.348746	-0.370164	-0.306564	0.288137	1.000000	0.180662
origin	0.563450	-0.562543	-0.609409	-0.581024	0.205873	0.180662	1.000000

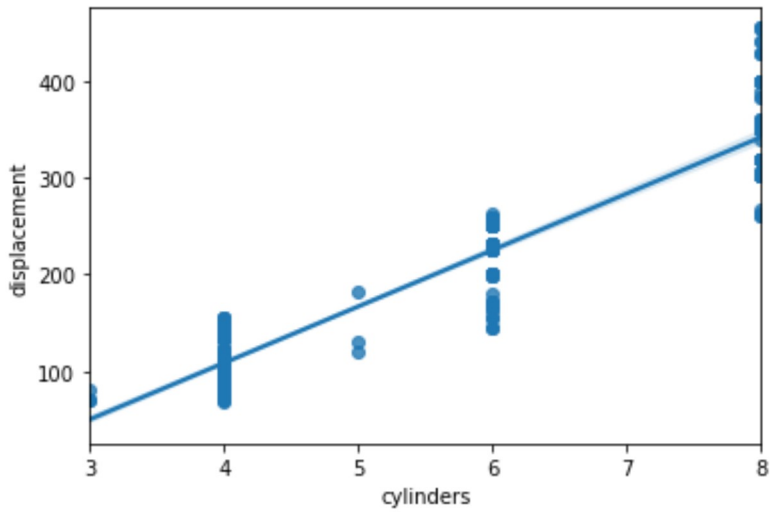
In [38]:
```
1 import numpy as np
2 import pandas as pd
3 import matplotlib.pyplot as plt
4 import seaborn as sns
5 df = pd.read_csv("auto-mpg.csv")
```

Out[38]: <AxesSubplot:xlabel='mpg', ylabel='displacement'>



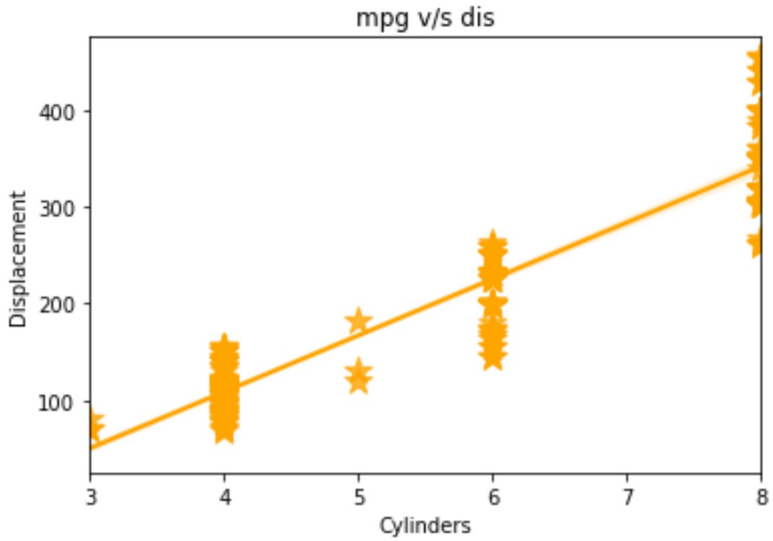
In [39]:
```
1 import numpy as np
2 import pandas as pd
3 import matplotlib.pyplot as plt
4 import seaborn as sns
5 df = pd.read_csv("auto-mpg.csv")
```

Out[39]: <AxesSubplot:xlabel='cylinders', ylabel='displacement'>



In [40]:
```
1 import numpy as np
2 import pandas as pd
3 import matplotlib.pyplot as plt
4 import seaborn as sns
5 df = pd.read_csv("auto-mpg.csv")
6 sns.regplot(x="cylinders",y="displacement",data=df,color="orange",marker='
7 plt.xlabel("Cylinders")
8 plt.ylabel("Displacement")
```

Out[40]: Text(0.5, 1.0, 'mpg v/s dis')



waffle charts

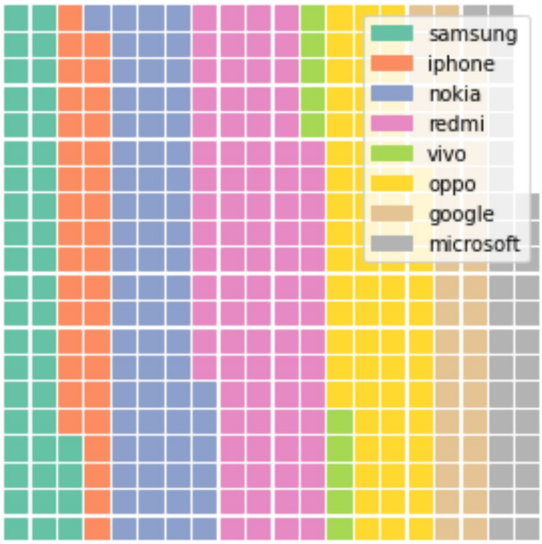
In [42]:

```
Requirement already satisfied: pywaffle in c:\programdata\anaconda3\lib\site-
packages (1.1.0)
Requirement already satisfied: fontawesomefree in c:\programdata\anaconda3\li
b\site-packages (from pywaffle) (6.5.1)
Requirement already satisfied: matplotlib in c:\programdata\anaconda3\lib\sit
e-packages (from pywaffle) (3.3.2)
Requirement already satisfied: numpy>=1.15 in c:\programdata\anaconda3\lib\si
te-packages (from matplotlib->pywaffle) (1.19.2)
Requirement already satisfied: cycler>=0.10 in c:\programdata\anaconda3\lib\s
ite-packages (from matplotlib->pywaffle) (0.10.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\programdata\anaconda3
\lib\site-packages (from matplotlib->pywaffle) (1.3.0)
Requirement already satisfied: python-dateutil>=2.1 in c:\programdata\anacond
a3\lib\site-packages (from matplotlib->pywaffle) (2.8.1)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.3 in c:
\programdata\anaconda3\lib\site-packages (from matplotlib->pywaffle) (2.4.7)
Requirement already satisfied: certifi>=2020.06.20 in c:\programdata\anaconda
3\lib\site-packages (from matplotlib->pywaffle) (2020.6.20)
Requirement already satisfied: pillow>=6.2.0 in c:\programdata\anaconda3\lib
\site-packages (from matplotlib->pywaffle) (8.0.1)
Requirement already satisfied: six in c:\programdata\anaconda3\lib\site-packa
ges (from cycler>=0.10->matplotlib->pywaffle) (1.15.0)
Note: you may need to restart the kernel to use updated packages.
```

In [43]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from pywaffle import Waffle

data = {"phone":["samsung","iphone","nokia","redmi","vivo","oppo","google"
        "stock":[44,35,67,89,10,69,45,34]}
df = pd.DataFrame(data)
```

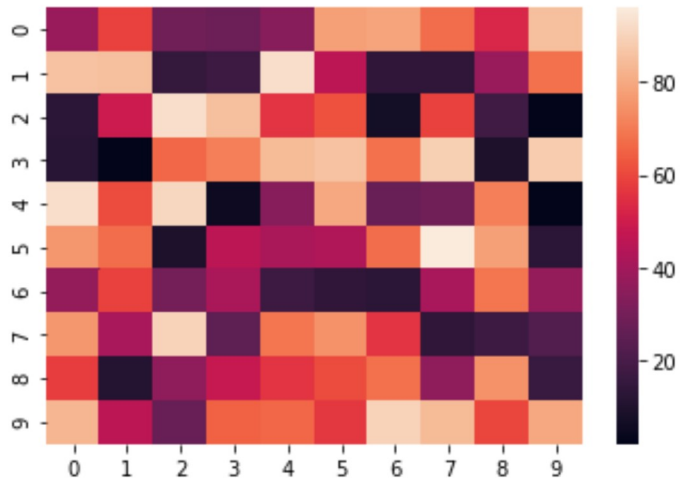
In [44]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from pywaffle import Waffle

data = {"phone":["samsung","iphone","nokia","redmi","vivo","oppo","google"
        "stock":[44,35,67,89,10,69,45,34]}
df = pd.DataFrame(data)
```



Heatmap:

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
data = np.random.randint(low=1,high=100,size=(10,10))
print(data)
```

```
[[38 59 29 28 34 78 79 67 53 85]
 [86 85 15 17 93 46 14 14 38 68]
 [13 49 93 85 56 62  7 59 18  2]
 [12  2 66 71 84 86 68 89  9 88]
 [93 61 91  5 34 80 27 29 71  2]
 [76 67  9 46 42 43 67 96 78 13]
 [37 59 30 42 17 14 13 41 69 37]
 [76 41 90 25 69 75 56 14 17 22]
 [58 11 36 48 56 61 68 36 75 16]
 [83 46 27 65 66 57 90 84 60 80]]
```

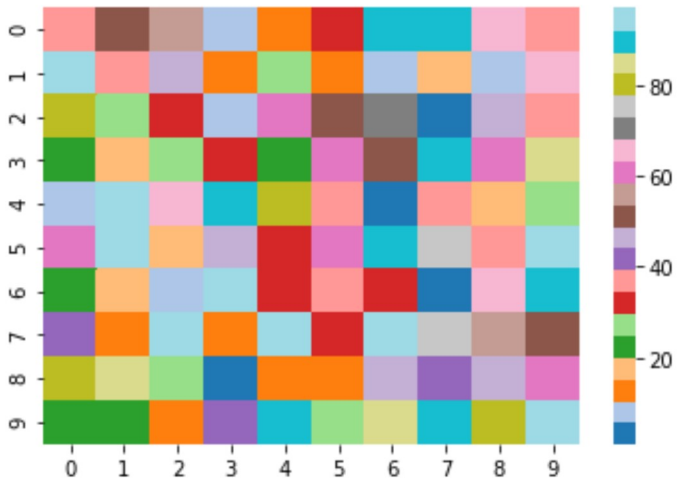


In [2]:
```
```

```
Help on function heatmap in module seaborn.matrix:

heatmap(data, *, vmin=None, vmax=None, cmap=None, center=None, robust=False,
annot=None, fmt='.2g', annot_kws=None, linewidths=0, linecolor='white', cbar=
True, cbar_kws=None, cbar_ax=None, square=False, xticklabels='auto', yticklab
els='auto', mask=None, ax=None, **kwargs)
    Plot rectangular data as a color-encoded matrix.

    This is an Axes-level function and will draw the heatmap into the
    currently-active Axes if none is provided to the ``ax`` argument.  Part o
f
    this Axes space will be taken and used to plot a colormap, unless ``cbar`
`
    is False or a separate Axes is provided to ``cbar_ax``.

    Parameters
    ----------
    data : rectangular dataset
        2D dataset that can be coerced into an ndarray. If a Pandas DataFrame
```

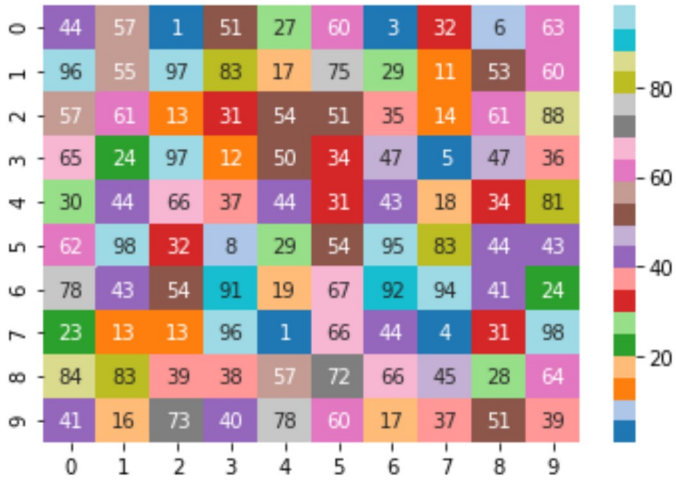
In [3]:
```
1 import pandas as pd
2 import matplotlib.pyplot as plt
3 import numpy as np
4 import seaborn as sns
5 data = np.random.randint(low=1,high=100,size=(10,10))
6 print(data)
```

```
[[39 51 57  9 14 33 92 92 67 36]
 [94 38 48 12 25 11 10 16 10 64]
 [81 27 32 10 60 51 71  2 46 39]
 [24 19 27 31 24 62 50 89 63 83]
 [10 97 68 91 82 37  1 35 17 25]
 [63 94 20 48 31 63 91 74 38 93]
 [24 16  7 93 33 35 32  3 67 92]
 [40 13 93 12 95 32 94 75 56 53]
 [79 86 25  2 12 12 45 42 48 63]
 [23 22 12 41 89 28 87 92 79 96]]
```



In [4]:
```
1 import pandas as pd
2 import matplotlib.pyplot as plt
3 import numpy as np
4 import seaborn as sns
5 data = np.random.randint(low=1,high=100,size=(10,10))
6 print(data)
```

```
[[44 57  1 51 27 60  3 32  6 63]
 [96 55 97 83 17 75 29 11 53 60]
 [57 61 13 31 54 51 35 14 61 88]
 [65 24 97 12 50 34 47  5 47 36]
 [30 44 66 37 44 31 43 18 34 81]
 [62 98 32  8 29 54 95 83 44 43]
 [78 43 54 91 19 67 92 94 41 24]
 [23 13 13 96  1 66 44  4 31 98]
 [84 83 39 38 57 72 66 45 28 64]
 [41 16 73 40 78 60 17 37 51 39]]
```



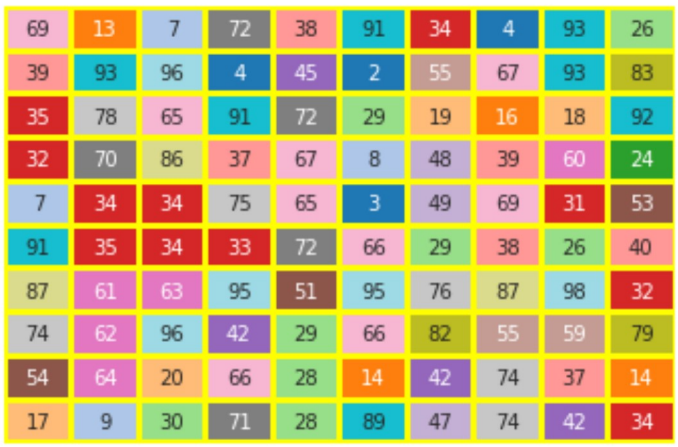
In [5]:
```
1 import pandas as pd
2 import matplotlib.pyplot as plt
3 import numpy as np
4 import seaborn as sns
5 data = np.random.randint(low=1,high=100,size=(10,10))
6 print(data)
```

```
[[54 52 99 52 75 24 69 82 74 29]
 [82 75 11 98 56 62 10 72 25 38]
 [61 83  4 63 57 95 52 49 71 21]
 [53 35 87 89 95 86 43 56 99 72]
 [21  6 21 77 17 12  7 56 46 43]
 [59 63 22  3 74  3 63 67 38 76]
 [18 38 65 30  8 74 73 93 77 71]
 [78 41 55 93 28 38 48 28 41 81]
 [54 92 77 86 95 52 48  3 25 43]
 [87 46 31 52  2 35 79  3 22 75]]
```

In [6]:

```
1 import pandas as pd
2 import matplotlib.pyplot as plt
3 import numpy as np
4 import seaborn as sns
5 data = np.random.randint(low=1,high=100,size=(10,10))
6 print(data)
7 hm = sns.heatmap(data=data,cmap="tab20",annot=True,linewidths=2,linecolor=
```

```
[[69 13  7 72 38 91 34  4 93 26]
 [39 93 96  4 45  2 55 67 93 83]
 [35 78 65 91 72 29 19 16 18 92]
 [32 70 86 37 67  8 48 39 60 24]
 [ 7 34 34 75 65  3 49 69 31 53]
 [91 35 34 33 72 66 29 38 26 40]
 [87 61 63 95 51 95 76 87 98 32]
 [74 62 96 42 29 66 82 55 59 79]
 [54 64 20 66 28 14 42 74 37 14]
 [17  9 30 71 28 89 47 74 42 34]]
```



In [7]:
```
1 import pandas as pd
2 import matplotlib.pyplot as plt
3 import numpy as np
4 import seaborn as sns
5 data = np.random.randint(low=1,high=100,size=(10,10))
6 print(data)
```

```
[[13 25 46 10 17 23 81 23 99 60]
 [10 50  4 97 21 86 56 92 19 94]
 [45 99 38 47  6 57 88  7 22  9]
 [26 43 53 10 32 29 20 15 41 86]
 [ 5 28 17 26 53 99 88  8 89 80]
 [32 47 17 74 23 22 48 71 32 30]
 [ 3  8 30 81 26 42 70 37 69 29]
 [99 99  9 93 98 49 51 84 46 80]
 [10 74 93 93 46 86 99 18 61 57]
 [23 56  5 94 45 38 45 60  6  8]]
```

In [8]:
```
1 import pandas as pd
2 import matplotlib.pyplot as plt
3 import numpy as np
4 import seaborn as sns
5 df = pd.read_csv("auto-mpg.csv")
```

Out[8]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model year	origin	car name
0	18.0	8	307.0	130	3504	12.0	70	1	chevrolet chevelle malibu
1	15.0	8	350.0	165	3693	11.5	70	1	buick skylark 320
2	18.0	8	318.0	150	3436	11.0	70	1	plymouth satellite
3	16.0	8	304.0	150	3433	12.0	70	1	amc rebel sst
4	17.0	8	302.0	140	3449	10.5	70	1	ford torino
...	...	...	...	...	...	...	...	...	...
393	27.0	4	140.0	86	2790	15.6	82	1	ford mustang gl
394	44.0	4	97.0	52	2130	24.6	82	2	vw pickup
395	32.0	4	135.0	84	2295	11.6	82	1	dodge rampage
396	28.0	4	120.0	79	2625	18.6	82	1	ford ranger
397	31.0	4	119.0	82	2720	19.4	82	1	chevy s-10

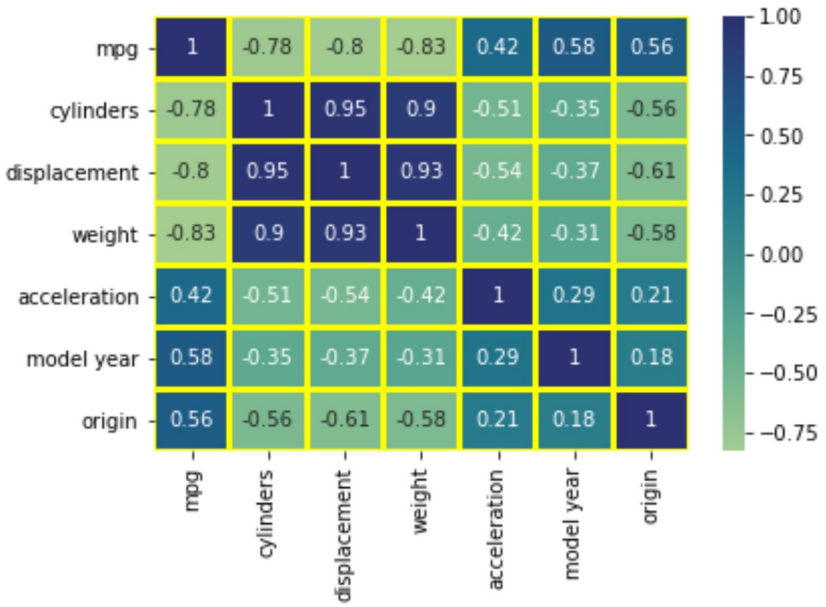
398 rows × 9 columns

In [9]:
```
1 import pandas as pd
2 import matplotlib.pyplot as plt
3 import numpy as np
4 import seaborn as sns
5 df = pd.read_csv("auto-mpg.csv")
```

Out[9]:

	mpg	cylinders	displacement	weight	acceleration	model year	origin
mpg	1.000000	-0.775396	-0.804203	-0.831741	0.420289	0.579267	0.563450
cylinders	-0.775396	1.000000	0.950721	0.896017	-0.505419	-0.348746	-0.562543
displacement	-0.804203	0.950721	1.000000	0.932824	-0.543684	-0.370164	-0.609409
weight	-0.831741	0.896017	0.932824	1.000000	-0.417457	-0.306564	-0.581024
acceleration	0.420289	-0.505419	-0.543684	-0.417457	1.000000	0.288137	0.205873
model year	0.579267	-0.348746	-0.370164	-0.306564	0.288137	1.000000	0.180662
origin	0.563450	-0.562543	-0.609409	-0.581024	0.205873	0.180662	1.000000

In [10]:
```
1 import pandas as pd
2 import matplotlib.pyplot as plt
3 import numpy as np
4 import seaborn as sns
5 df = pd.read_csv("auto-mpg.csv")
```



Networkx:

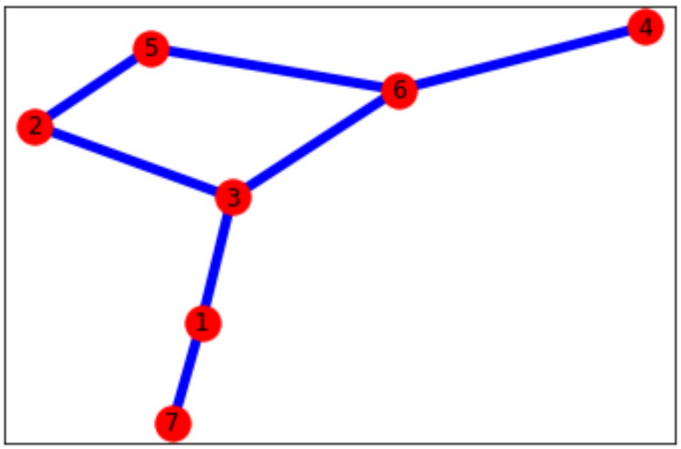
- directory graph
- undirectory graph

In [11]:

```
Requirement already satisfied: networkx in c:\programdata\anaconda3\lib\site-
packages (2.5)
Requirement already satisfied: decorator>=4.3.0 in c:\programdata\anaconda3\l
ib\site-packages (from networkx) (4.4.2)
Note: you may need to restart the kernel to use updated packages.
```

In [12]:

```
import networkx as nx
import matplotlib.pyplot as plt
g = nx.Graph()
g.add_node(1)
g.add_nodes_from([2,3])
g.add_nodes_from(range(4,7))
g.add_edge(1,3)
g.add_edge(2,5)
g.add_edge(1,7)
g.add_edge(3,3)
g.add_edges_from([(2,3),(3,6),(4,6),(5,6)])
nx.draw_networkx(g,node_size=300,node_color="red",edge_color="blue",width=
plt.figsize = [150,150]
```



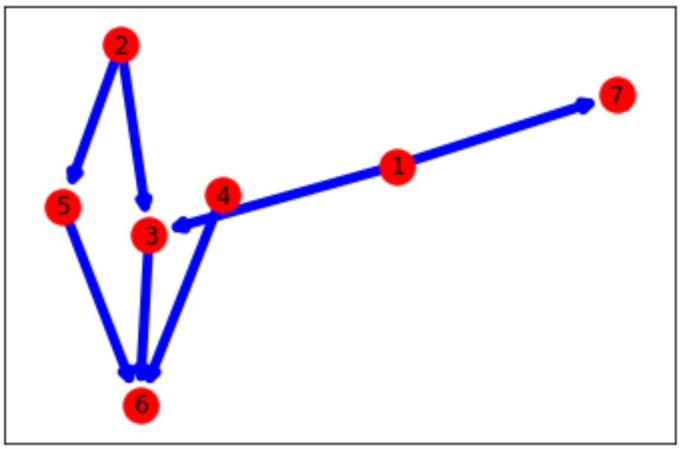
In [13]:

```
Help on function draw_networkx in module networkx.drawing.nx_pylab:

draw_networkx(G, pos=None, arrows=True, with_labels=True, **kwds)
    Draw the graph G using Matplotlib.

    Draw the graph with Matplotlib with options for node positions,
    labeling, titles, and many other drawing features.
    See draw() for simple drawing without labels or axes.

    Parameters
    ----------
    G : graph
       A networkx graph

    pos : dictionary, optional
       A dictionary with nodes as keys and positions as values.
       If not specified a spring layout positioning will be computed.
       See :py:mod:`networkx.drawing.layout` for functions that
       compute node positions.
```

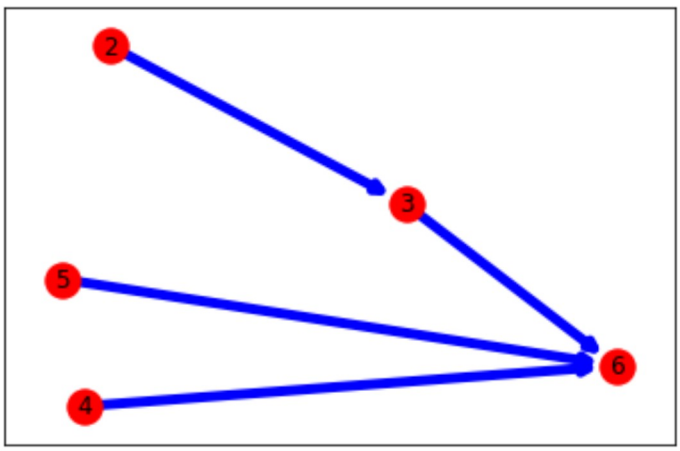
In [14]:

```
Help on module networkx.classes.graph in networkx.classes:

NAME
    networkx.classes.graph - Base class for undirected graphs.

DESCRIPTION
    The Graph class allows any hashable object as a node
    and can associate key/value attribute pairs with each undirected edge.

    Self-loops are allowed but multiple edges are not (see MultiGraph).

    For directed graphs see DiGraph and MultiDiGraph.

CLASSES
    builtins.object
        Graph

    class Graph(builtins.object)
     |  Graph(incoming_graph_data=None, **attr)
     |
```

In [15]:
```
1  import networkx as nx
2  import matplotlib.pyplot as plt
3  g = nx.DiGraph()
4  g.add_node(1)
5  g.add_nodes_from([2,3])
6  g.add_nodes_from(range(4,7))
7  g.add_edge(1,3)
8  g.add_edge(2,5)
9  g.add_edge(1,7)
10 g.add_edge(3,3)
11 g.add_edges_from([(2,3),(3,6),(4,6),(5,6)])
12 nx.draw_networkx(g,node_size=300,node_color="red",edge_color="blue",width=
13 plt.figsize = [150,150]
```

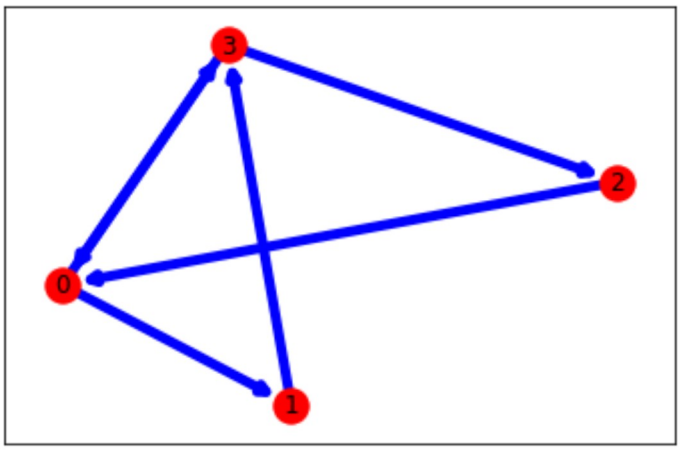


In [17]:
```
1  import networkx as nx
2  import matplotlib.pyplot as plt
3  g = nx.DiGraph()
4  g.add_edges_from([(2,3),(3,6),(4,6),(5,6)])
5  nx.draw_networkx(g,node_size=300,node_color="red",edge_color="blue",width=
6  plt.figsize = [150,150]
```



In [29]:

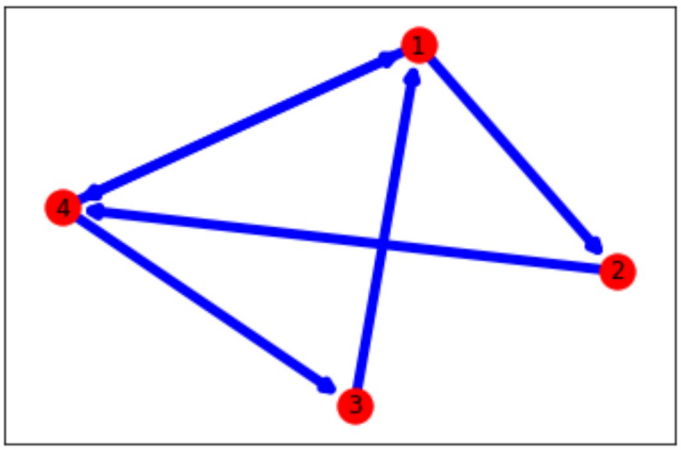
```
import networkx as nx
import matplotlib.pyplot as plt
G = nx.DiGraph()
G.add_nodes_from(range(4))
l = [[0,1,0,1],[0,0,0,1],[1,0,0,0],[1,0,1,0]]
for i in range(4):
    for j in range(4):
        if(l[i][j]==1):
            G.add_edge(i,j)
nx.draw_networkx(G,node_size=300,node_color="red",edge_color="blue",width=
plt.figsize=[150,150]
```



- node start with 1 to 4 logic

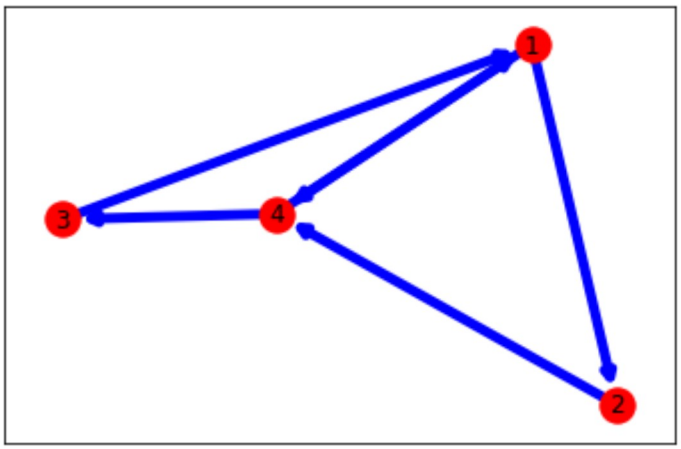
In [31]:

```
import networkx as nx
import matplotlib.pyplot as plt
G = nx.DiGraph()
G.add_nodes_from(range(1,5))
l = [[0,1,0,1],[0,0,0,1],[1,0,0,0],[1,0,1,0]]
for i in range(1,5):
    for j in range(1,5):
        if(l[i-1][j-1]==1):
            G.add_edge(i,j)
nx.draw_networkx(G,node_size=300,node_color="red",edge_color="blue",width=
plt.figsize=[150,150]
```



In [33]:

```
import networkx as nx
import matplotlib.pyplot as plt
G = nx.DiGraph()
G.add_nodes_from(range(1,5))
l = [[0,1,0,1],[0,0,0,1],[1,0,0,0],[1,0,1,0]]
for i in range(4):
    for j in range(4):
        if(l[i][j]==1):
            G.add_edge(i+1,j+1)
nx.draw_networkx(G,node_size=300,node_color="red",edge_color="blue",width=
plt.figsize=[150,150]
```



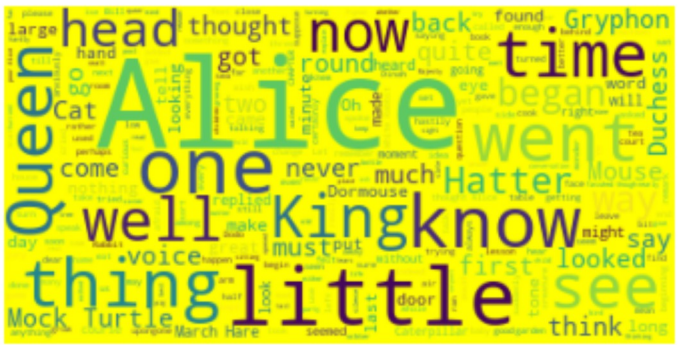
In [44]:
```
from wordcloud import WordCloud,STOPWORDS
stopwords = set(STOPWORDS)
stopwords.add("said")
```

```
{'her', 'into', 'same', 'you', 'with', 'he', 'by', 'com', 'other', 'hers', 'b
elow', "he'll", 'been', "he's", 'through', 'whom', "you've", "what's", 'if',
'at', 'how', 'and', "haven't", 'or', 'ever', "they'll", 'themselves', 'to', '
further', 'here', 'above', 'i', "we're", 'be', 'in', "you'll", 'get', 'my', "
we've", 'doing', 'as', "she'll", 'not', 'where', "let's", 'any', 'however', "
how's", 'their', 'until', 'them', 'http', 'am', 'then', 'being', 'who', 'of',
"when's", "didn't", 'more', 'when', "doesn't", 'they', 'can', 'are', 'myself'
, 'down', 'both', 'ought', "aren't", 'why', 'k', 'your', 'could', 'yours', 't
heirs', "why's", "couldn't", 'his', 'after', 'should', 'from', 'she', 'r', "h
asn't", 'him', 'while', 'too', "shan't", "i'm", 'have', 'nor', "i've", "there
's", "i'll", 'was', "who's", "i'd", 'having', "can't", 'over', 'therefore', '
very', "you're", 'also', 'few', 'some', "you'd", "don't", 'off', 'most', 'we'
, "won't", 'shall', 'did', 'otherwise', "she's", 'such', 'these', 'its', "the
y've", 'during', 'yourselves', 'else', 'ourselves', "they're", 'but', 'agains
t', 'me', 'out', 'because', 'hence', 'herself', 'yourself', 'said', 'each', '
once', 'between', 'only', 'again', 'there', "they'd", 'for', 'under', "where'
s", "hadn't", 'our', "here's", 'itself', 'this', "isn't", 'own', 'before', "t
hat's", 'a', 'on', "we'd", 'which', 'an', 'just', 'is', "she'd", 'had', "shou
ldn't", 'those', "weren't", 'www', "it's", 'all', "wouldn't", 'what', 'would'
, 'about', 'were', "he'd", 'since', 'the', 'has', 'so', 'does', "wasn't", 'do
', "mustn't", 'up', 'ours', 'cannot', 'than', 'himself', 'like', 'it', 'that'
, "we'll", 'no'}
```

In [48]:
```
f = open("alice_in_wonderland.txt","r")
alice = f.read()
alice_wc = WordCloud(background_color="yellow",max_words=2000,stopwords=st
```

Out[48]:
```
<wordcloud.wordcloud.WordCloud at 0x20d008d0af0>
```

In [49]:
```
import matplotlib.pyplot as plt
plt.imshow(alice_wc)
plt.axis("off")
```



q.folium:

In [51]:
```
1 import folium
2 world_map = folium.Map()
```

Out[51]: Make this Notebook Trusted to load map: File -> Trust Notebook

In [53]:
```
1 import folium
2 world_map = folium.Map(location=[22.9912,72.4884],zoom_start=12)
```

Out[53]: Make this Notebook Trusted to load map: File -> Trust Notebook

In [9]:
```
import folium
import pandas as pd
df_incidents = pd.read_csv("Police_Department_Incidents_-_Previous_Year__2
```

```
   IncidntNum      Category                                          Descript
\
0   120058272   WEAPON LAWS                         POSS OF PROHIBITED WEAPON
1   120058272   WEAPON LAWS  FIREARM, LOADED, IN VEHICLE, POSSESSION OR USE
2   141059263      WARRANTS                                  WARRANT ARREST
3   160013662  NON-CRIMINAL                                   LOST PROPERTY
4   160002740  NON-CRIMINAL                                   LOST PROPERTY

  DayOfWeek                     Date   Time  PdDistrict      Resolution  \
0    Friday  01/29/2016 12:00:00 AM  11:00    SOUTHERN  ARREST, BOOKED
1    Friday  01/29/2016 12:00:00 AM  11:00    SOUTHERN  ARREST, BOOKED
2    Monday  04/25/2016 12:00:00 AM  14:59     BAYVIEW  ARREST, BOOKED
3   Tuesday  01/05/2016 12:00:00 AM  23:50  TENDERLOIN            NONE
4    Friday  01/01/2016 12:00:00 AM  00:30     MISSION            NONE

                  Address           X          Y  \
0  800 Block of BRYANT ST -122.403405  37.775421
1  800 Block of BRYANT ST -122.403405  37.775421
2   KEITH ST / SHAFTER AV -122.388856  37.729981
3  JONES ST / OFARRELL ST -122.412971  37.785788
4    16TH ST / MISSION ST -122.419672  37.765050

                              Location            PdId
0   (37.775420706711, -122.403404791479)  12005827212120
1   (37.775420706711, -122.403404791479)  12005827212168
2  (37.7299809672996, -122.388856204292)  14105926363010
3  (37.7857883766888, -122.412970537591)  16001366271000
4  (37.7650501214668, -122.419671780296)  16000274071000
```

In [55]:
```
import folium
import pandas as pd
df_incidents = pd.read_csv("Police_Department_Incidents_-_Previous_Year__2
```

Out[55]: (150500, 13)

In [56]:
```
import folium
import pandas as pd
df_incidents = pd.read_csv("Police_Department_Incidents_-_Previous_Year__2
df = df_incidents.iloc[0:100,:]
```

Out[56]: (100, 13)

In [8]:
```
import folium
import pandas as pd
df_incidents = pd.read_csv("Police_Department_Incidents_-_Previous_Year__2
df = df_incidents.iloc[0:100,:]
sanfran_map = folium.Map(location=[37.77,-122.44],zoom_start=12)
incidents = folium.map.FeatureGroup()
for lat,lon in zip(df.Y,df.X):
    incidents.add_child(folium.features.CircleMarker([lat,lon],radius=5,cc
sanfran_map.add_child(incidents)
```

Out[8]: Make this Notebook Trusted to load map: File -> Trust Notebook

In [1]:
```
import folium
import pandas as pd
df_incidents = pd.read_csv("Police_Department_Incidents_-_Previous_Year__2
df = df_incidents.iloc[0:100,:]
sanfran_map = folium.Map(location=[37.77,-122.44],zoom_start=12)
incidents = folium.map.FeatureGroup()
for lat,lon,label in zip(df.Y,df.X,df.Category):
    incidents.add_child(folium.features.Marker([lat,lon],popup=label))
```

Out[1]: Make this Notebook Trusted to load map: File -> Trust Notebook

choropleth

In [2]:
```
import folium
import pandas as pd
state_unemp = pd.read_csv("US_Unemployment_Oct2012.csv")
state_geo = "us-states.json"
usa_state = folium.Map(location=[48,-102],zoom_start=3)
folium.Choropleth(geo_data=state_geo,name="choropleth",data=state_unemp,co
                  key_on="feature.id",fill_color="YlGnBu",fill_opacity=0.7
usa_state
```

Out[2]: Make this Notebook Trusted to load map: File -> Trust Notebook

PB:85

You have been hired as a network analyst by a company to analyze the social network of their employees. The company has

provided you with the following data:

There are 5 employees in the company, each identified by a unique ID from 1 to 5.

The following relationships exist between the employees:

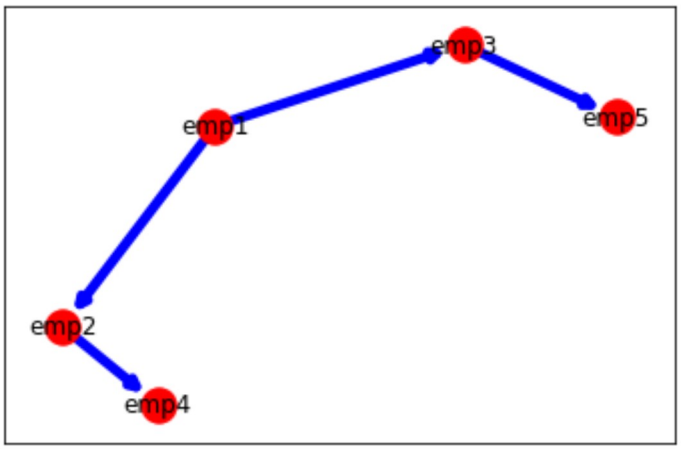
1. Employee 1 is friends with Employee 2 and Employee 3.

2. Employee 2 is friends with Employee 4.

3. Employee 3 is friends with Employee 5.

Your task is to create a NetworkX graph representing this social network and display it.

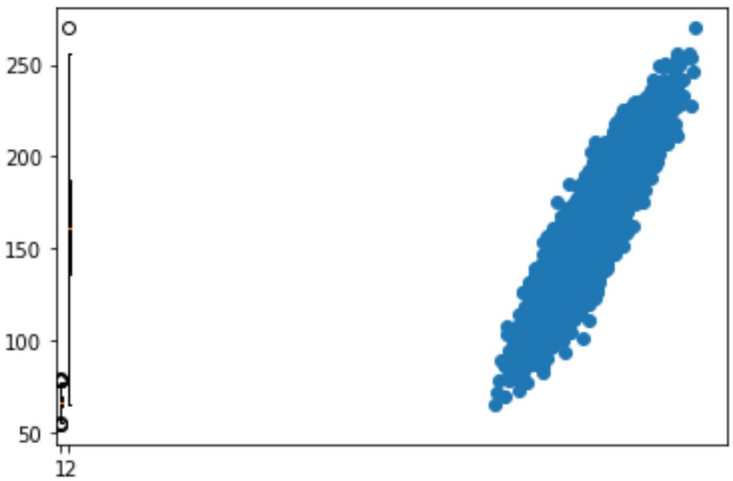
In [5]:
```
import matplotlib.pyplot as plt
import networkx as nx
G = nx.DiGraph()
# G.add_edges_from([(1,2),(1,3),(2,4),(3,5)])
G.add_edges_from([("emp1","emp2"),("emp1","emp3"),("emp2","emp4"),("emp3",
nx.draw_networkx(G,node_size=300,node_color="red",edge_color="blue",width=
plt.show()
```



PB:31

In [15]:
```
import pandas as pd
import  matplotlib.pyplot as plt

df = pd.read_csv("heights_weights.csv")
df.describe()
df.corr() # good corelation
plt.boxplot(df[["Height","Weight"]])
```

Out[15]: <matplotlib.collections.PathCollection at 0x1a682682640>



In [9]:
```
1 import pandas as pd
2
3 df = pd.read_csv("heights_weights.csv")
```

Out[9]:

	Height	Weight	Male
Height	1.000000	0.924756	0.691072
Weight	0.924756	1.000000	0.796723
Male	0.691072	0.796723	1.000000

In [ ]:
```
1
```