3 - Matplotlib_plt

October 14, 2021

Table of Contents: Matplotlib Part 1

1 TOC

1.1 Figures and Axes

1.2 Getting Started

1.3 Plot-specific options

1.3.1 Other plot components

1.4 Other types of Plots

1.4.1 Boxplot

1.4.2 Scatterplot

1.4.3 Bar

1.4.4 Bar (Horizontal)

1.5 Exercise 1 - Create a histogram (5 minutes)

1.5.1 Plot with Dates

1.6 Multiple datasets in a single plot

1.7 Exercise 2 - Create two plots 5 minutes

1.8 Figure level modifications

1.8.1 Plot Styles

1.8.2 Experiment

1.8.3 Figure size

1.8.4 Facecolor

1.8.5 Saving to a file

1.8.6 Experiment

1.9 Subplots

1.10 Just for a little fun….

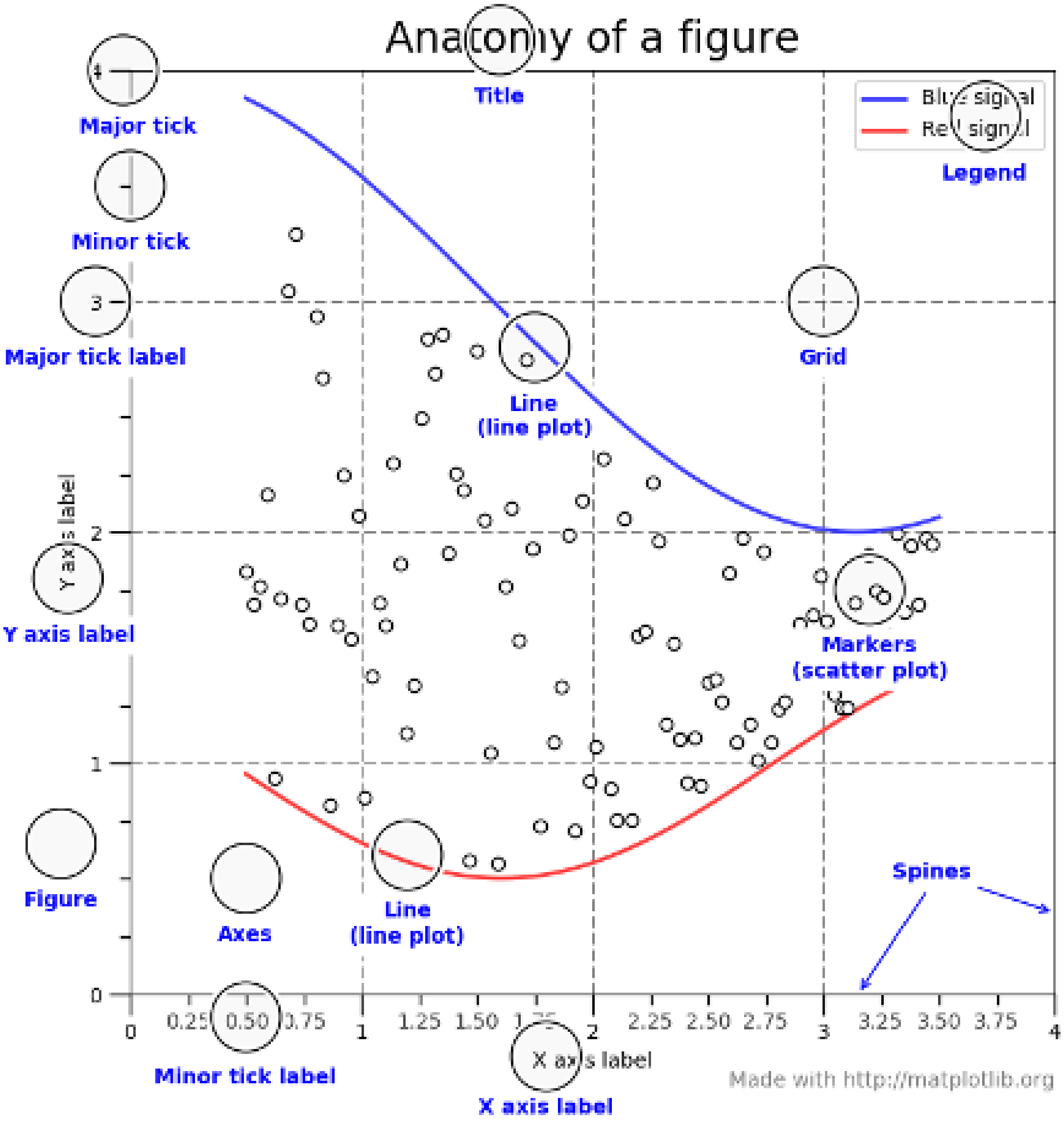
1.11 Exercise 3 - Create a figure with 4 subplots - 10 minutes

1 TOC

1.1 Figures and Axes

Think of the **Figure** as your workspace or canvas. It is the top level container in a plot hierarchy. You can have multiple independent figures and Figures can contain multiple Axes.

Plotting occurs on an **Axes** (not Axis). It is the plot and its associated details (labels, tick marks, grids, etc.)



Click here for matplotlib documentation - - matplotlib.org

Go here for the life cycle of a plot

1.2 Getting Started

[6]:

```
import matplotlib.pyplot as plt
import matplotlib as mpl
import numpy as np
#mpl.rcParams['lines.linewidth'] = 2
#mpl.rcParams['lines.linestyle'] = '--'


#import os
#for dirname, _, filenames in os.walk('/kaggle/input'):
#    for filename in filenames:
#        print(os.path.join(dirname, filename))

data = np.random.randn(100)
```

[7]:

```
data

# matplotlib works wit data in an array
```

[7]:

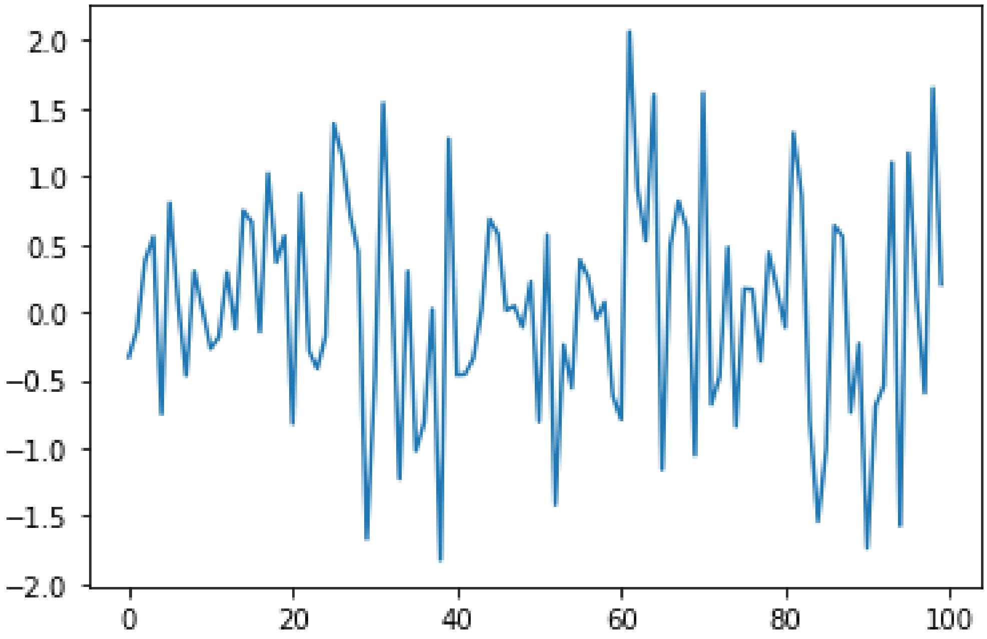
```
array([-0.33220731, -0.13232009,  0.37809939,  0.55292853, -0.74979521,
        0.80345443,  0.0819748 , -0.46725023,  0.30005585,  0.01917684,
       -0.2674023 , -0.1845754 ,  0.2917652 , -0.12488748,  0.74592295,
        0.66074983, -0.14398569,  1.01848213,  0.36599483,  0.55868709,
       -0.82080249,  0.87100167, -0.28794816, -0.41664645, -0.17819048,
        1.38693337,  1.14899622,  0.71402008,  0.43843703, -1.66838089,
       -0.49002417,  1.53640116,  0.29426454, -1.22803376,  0.30059671,
       -1.01950416, -0.81437656,  0.02078258, -1.82416504,  1.27562599,
       -0.46370529, -0.45764106, -0.34121312,  0.01523291,  0.6829625 ,
        0.57479888,  0.01219115,  0.04455287, -0.11068535,  0.2236469 ,
       -0.80664333,  0.56830558, -1.41765335, -0.24000198, -0.55806497,
        0.38430062,  0.25675145, -0.05374373,  0.0737512 , -0.61435577,
       -0.78873896,  2.06140037,  0.90573467,  0.52366153,  1.59854644,
       -1.15901438,  0.48736333,  0.81639358,  0.61493112, -1.05389321,
        1.61242193, -0.67916116, -0.4763373 ,  0.47790683, -0.83906673,
        0.17249314,  0.16559785, -0.36134734,  0.43822979,  0.17303841,
       -0.11378623,  1.3188806 ,  0.85557374, -0.80085921, -1.54107296,
       -1.00393909,  0.63583715,  0.55416824, -0.73839517, -0.23183669,
       -1.73794905, -0.68859751, -0.54369058,  1.10113345, -1.57215836,
        1.17223977,  0.112857  , -0.59649975,  1.64840311,  0.20611242])
```

[8]:

```
# Create our first plot (plot is the function to use for a lineplot)

plt.plot(data)

# Behind the scenes, pyplot created the: figure, axes, plot, x-axis and y-axis
```

[8]:

```
[<matplotlib.lines.Line2D at 0x7ff0606e3100>]
```



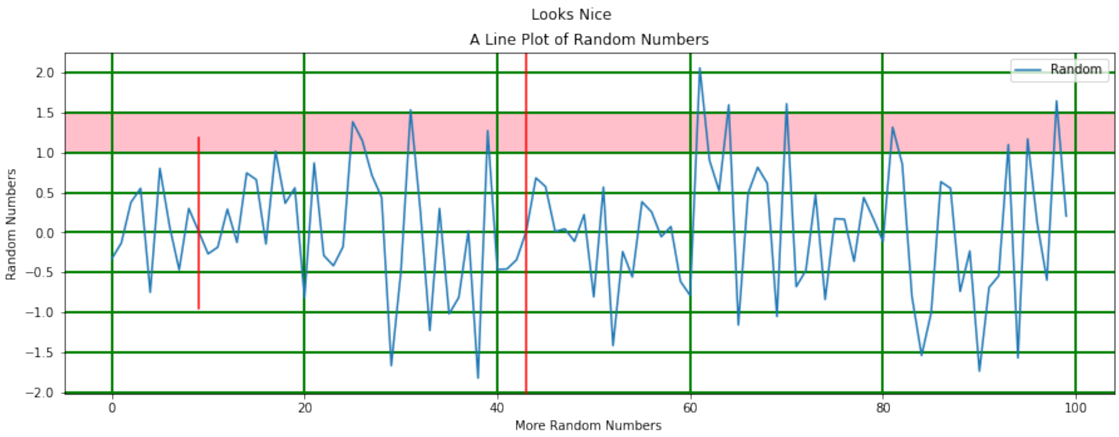
1.3 Plot-specific options

1.3.1 Other plot components

- Title
- Axis labels
- Legend
- Grid
- Reference lines

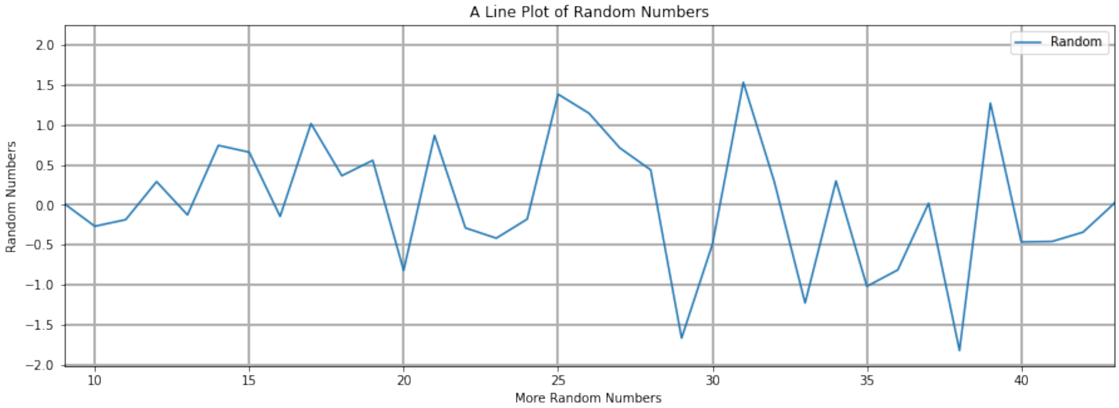
[9]:

```
plt.figure(figsize = (15,5))
plt.plot(data, label='Random')

plt.ylabel('Random Numbers')
plt.xlabel('More Random Numbers')
plt.title('A Line Plot of Random Numbers')
plt.legend()
plt.grid(color = 'green',linestyle='-', linewidth=2)
plt.axvline(x=43, color= 'r')
plt.axvline(x=9, ymin=0.25, ymax=0.75, color = 'r')
plt.axhspan(1,1.5, color = 'pink')
plt.suptitle('Looks Nice')
plt.show()    # removes that little extra line of output
```



[10]:

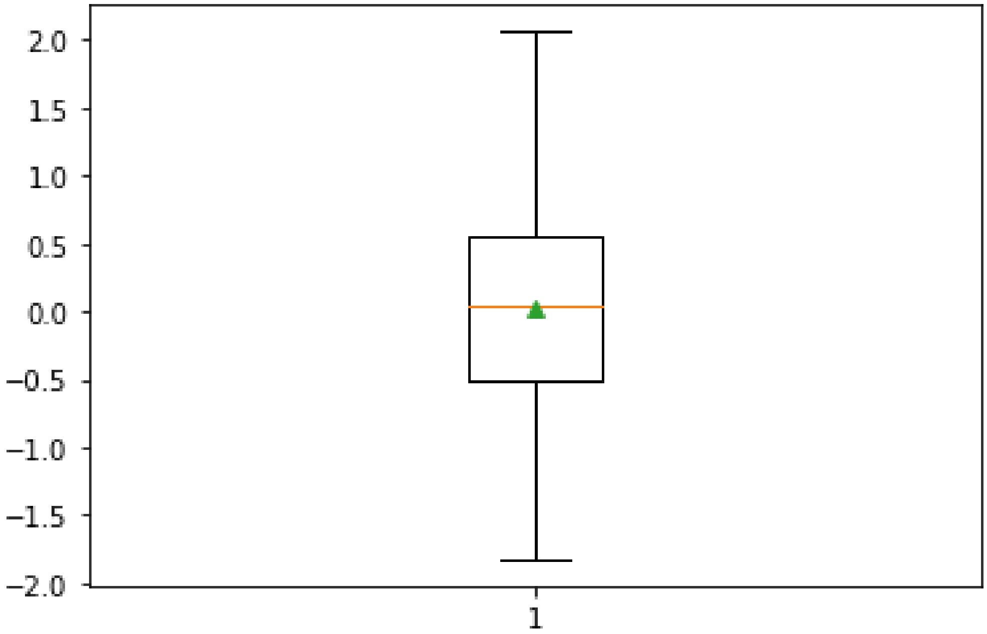
```
# In the sample above, reference lines were placed at 9 and 43.
# xlim (or ylim) can be used to control the range of the axis.

plt.figure(figsize = (15,5))
plt.plot(data, label='Random')
plt.ylabel('Random Numbers')
plt.xlabel('More Random Numbers')
plt.title('A Line Plot of Random Numbers')
plt.legend()
plt.grid(linestyle='-', linewidth=2)
plt.xlim(9,43)
plt.show()
```



1.4 Other types of Plots

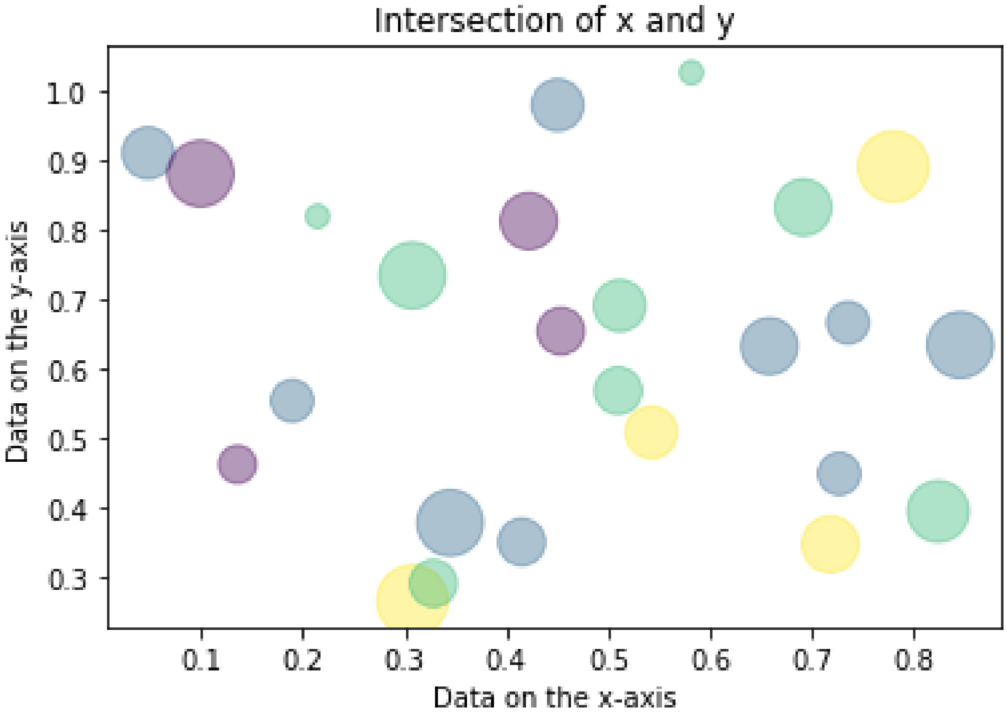
1.4.1 Boxplot

[11]:
```
plt.boxplot(data, showmeans=True)
plt.show()
```



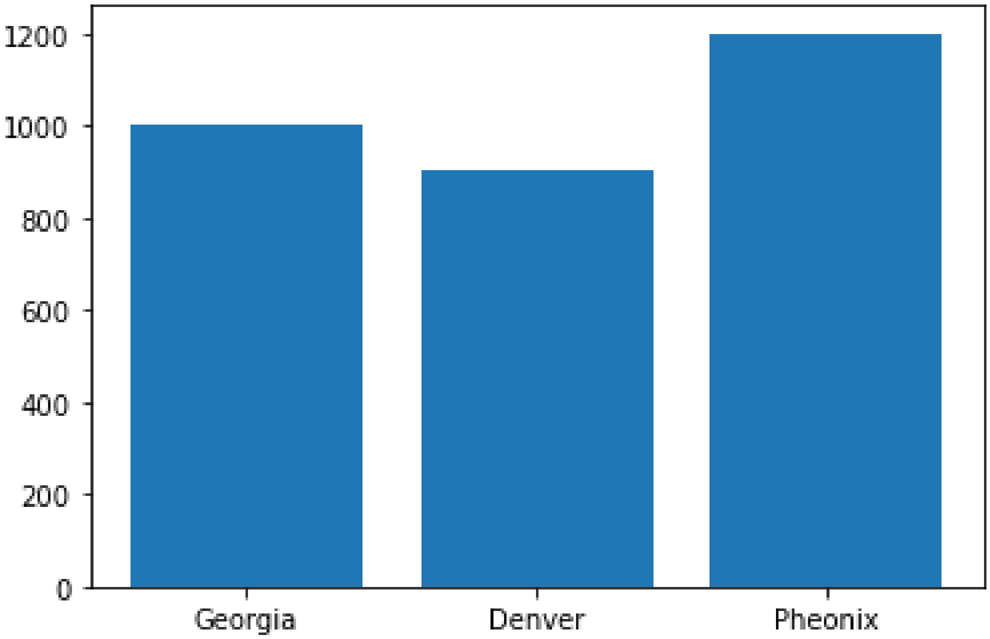
1.4.2 Scatterplot

[12]:
```
import math
import random

# create random data
no_of_points = 25
x = [random.triangular() for i in range(no_of_points)]
y = [random.gauss(0.5, 0.25) for i in range(no_of_points)]
colors = [random.randint(1, 4) for i in range(no_of_points)]
areas = [math.pi * random.randint(5, 15)**2 for i in range(no_of_points)]

plt.scatter(x,y, s=areas, c=colors, alpha=0.4)
plt.title('Intersection of x and y')
plt.xlabel('Data on the x-axis')
plt.ylabel('Data on the y-axis')
plt.show()
```



1.4.3 Bar

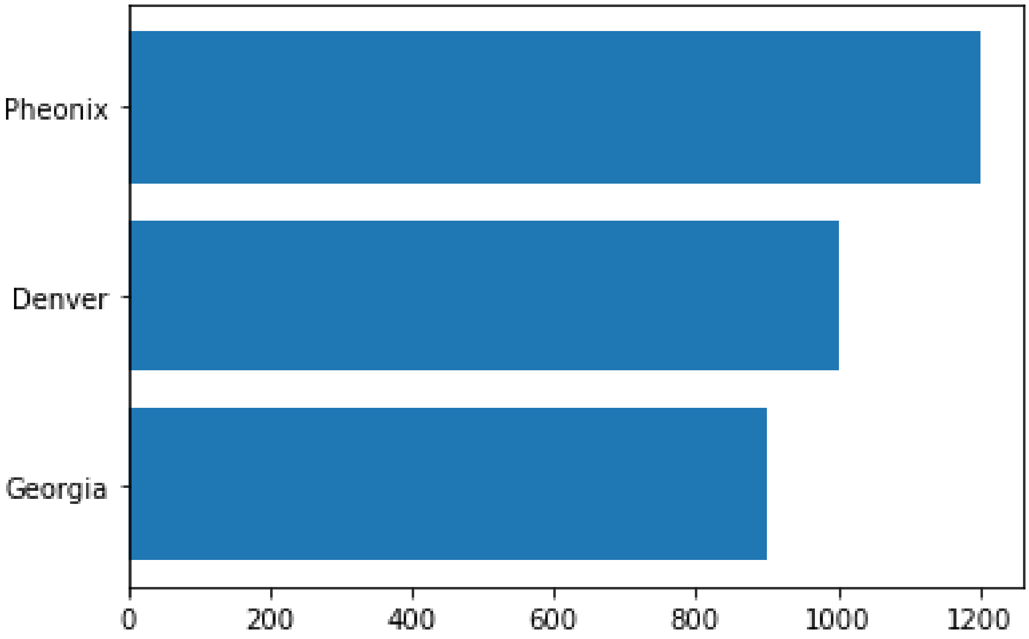
[13]:
```
names = ['Georgia', 'Denver', 'Pheonix']
values = [1000, 900, 1200]

plt.bar(names, values)
plt.show()
```



1.4.4 Bar (Horizontal)

[14]:

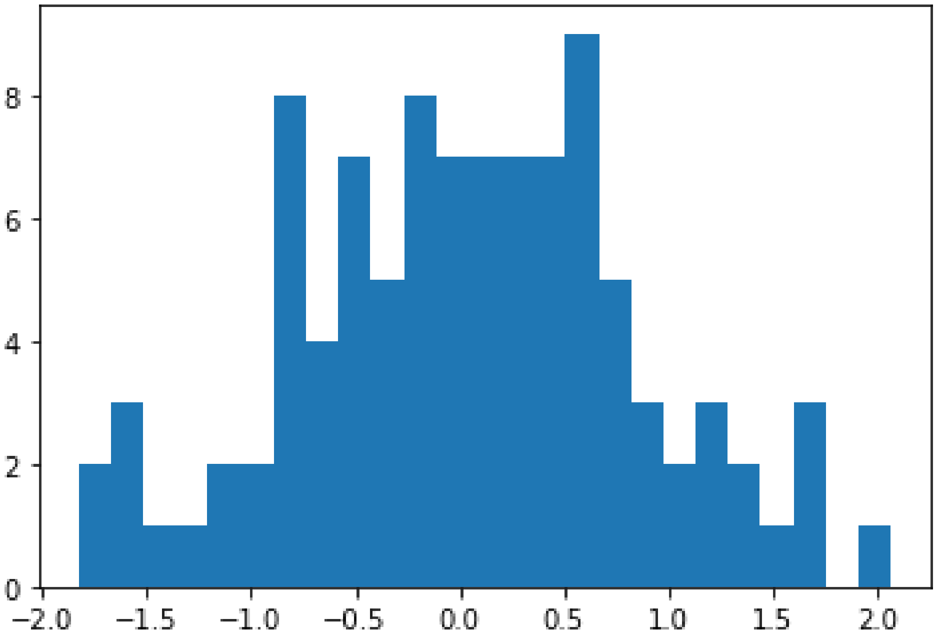
```
names = ['Georgia', 'Denver', 'Pheonix']
values = [1000, 900, 1200]

plt.barh(names, sorted(values))
plt.show()
```



1.5 Exercise 1 - Create a histogram (5 minutes)

```
[15]: # Using 'data', create a histogram.
      # Do 20 bins make a bettter presentation?

      plt.hist(data, bins=25)
```

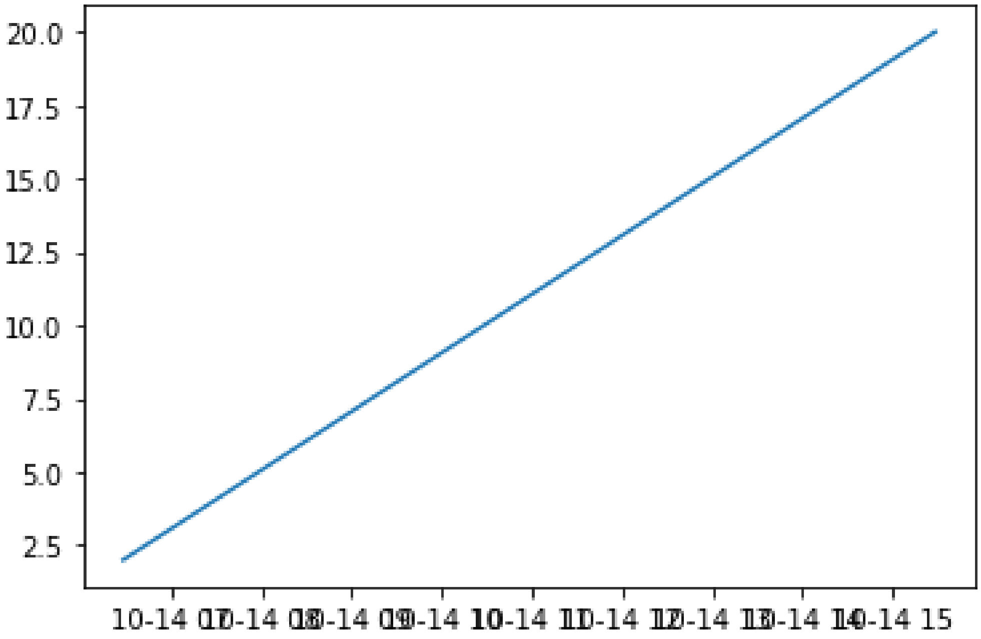
```
[15]: (array([2., 3., 1., 1., 2., 2., 8., 4., 7., 5., 8., 7., 7., 7., 7., 9., 5.,
              3., 2., 3., 2., 1., 3., 0., 1.]),
       array([-1.82416504, -1.66874242, -1.51331981, -1.35789719, -1.20247457,
              -1.04705196, -0.89162934, -0.73620673, -0.58078411, -0.42536149,
              -0.26993888, -0.11451626,  0.04090636,  0.19632897,  0.35175159,
               0.5071742 ,  0.66259682,  0.81801944,  0.97344205,  1.12886467,
               1.28428728,  1.4397099 ,  1.59513252,  1.75055513,  1.90597775,
               2.06140037]),
       <BarContainer object of 25 artists>)
```



1.5.1 Plot with Dates

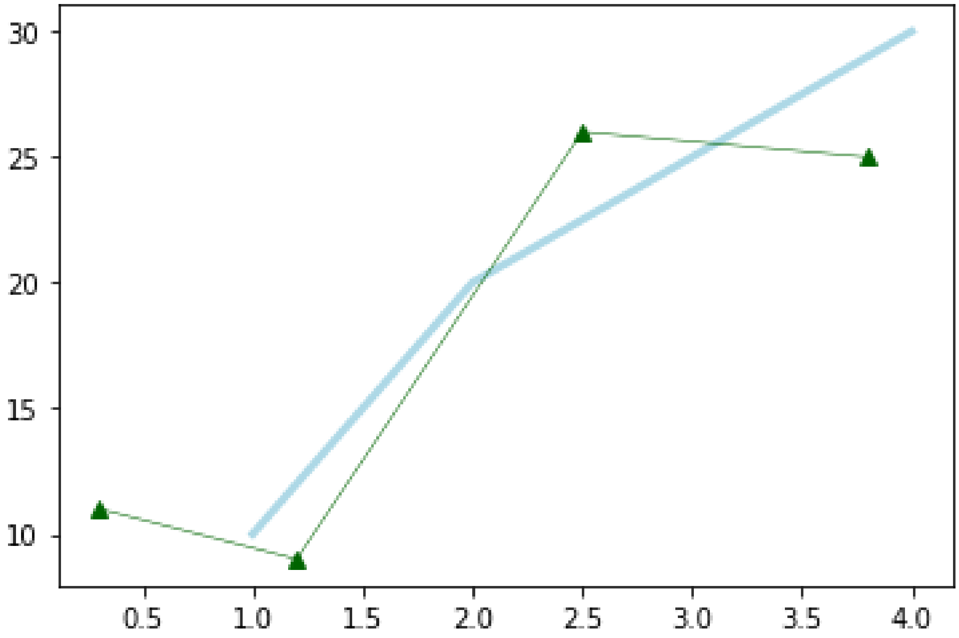
[16]:

```python
import matplotlib
import matplotlib.pyplot as plt
import numpy as np
import datetime



y = [ 2,4,6,8,10,12,14,16,18,20 ]
x = [datetime.datetime.now() + datetime.timedelta(hours=i) for i in range(len(y))]

plt.plot(x,y)
#plt.gcf().autofmt_xdate()
plt.show()
```

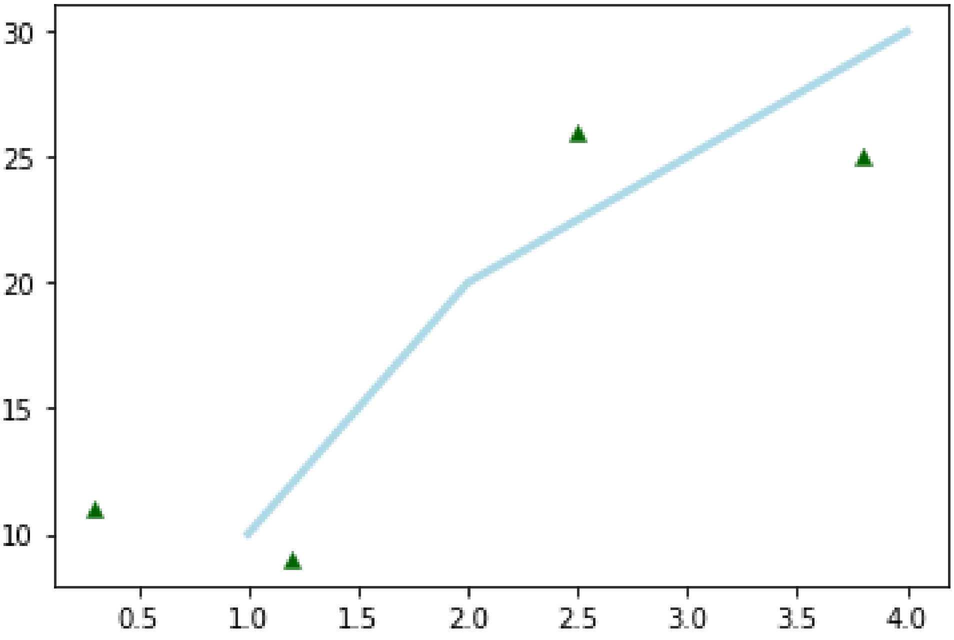


1.6 Multiple datasets in a single plot

[17]:
```
plt.plot([1, 2, 3, 4], [10, 20, 25, 30], color='lightblue', linewidth=3)
plt.plot([0.3, 1.2, 2.5, 3.8], [11, 9, 26, 25], color='darkgreen', marker='^',␣
 ↪linewidth = 0.5)
plt.show()
```



```
[18]: plt.plot([1, 2, 3, 4], [10, 20, 25, 30], color='lightblue', linewidth=3)
      plt.scatter([0.3, 1.2, 2.5, 3.8], [11, 9, 26, 25], color='darkgreen',␣
       ↪marker='^', linewidth = 0.5)
      plt.show()
```



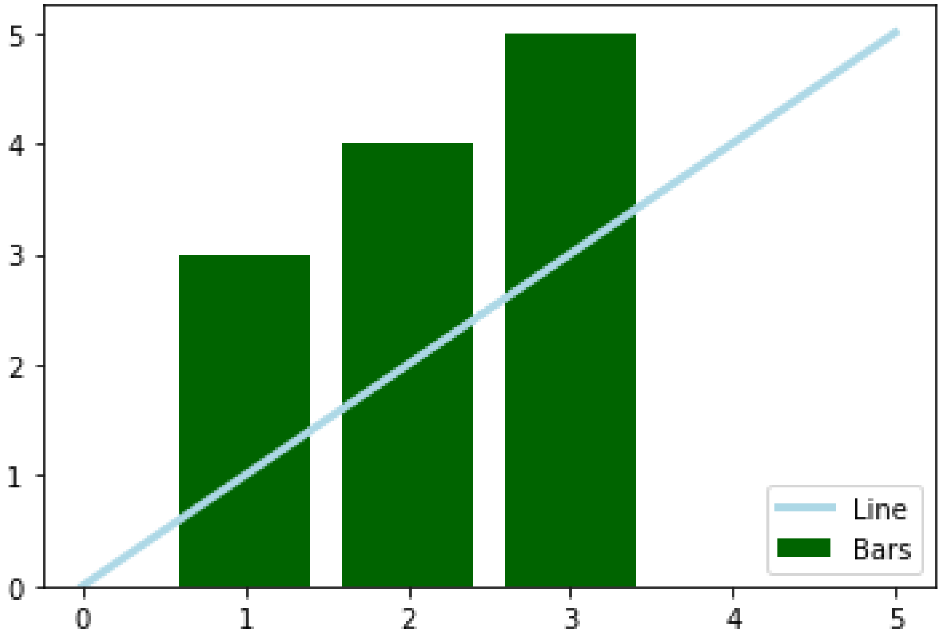
1.7 *Exercise 2 - Create two plots* 5 minutes

- Create a new workbook
- Remove all of the code added by kaggle.
- Import the required packages
- In the same plot, plot the data below.
    - x = np.linspace(0, 5, 5)
    - y = np.linspace(0, 5, 5)
    - a = [1,2,3]
    - b = [3,4,5]
- Plot x and y as a blue line
- Plot a and b as a bar chart with green bars
- Include a legend in the bottom right hand corner (use the documentation)

```
[19]: x = np.linspace(0, 5, 5)
      y = np.linspace(0, 5, 5)
      a = [1,2,3]
      b = [3,4,5]
```

```
plt.plot(x,y, color='lightblue', linewidth=3, label = 'Line')
plt.bar(a,b, color='darkgreen', label = 'Bars')
plt.legend(loc='lower right')
```

```
[19]: <matplotlib.legend.Legend at 0x7ff05fcdfaf0>
```



1.8 Figure level modifications

- Changing the sytle
- Changing the figure size
- Changing the facecolor

1.8.1 Plot Styles

```
[20]: plt.style.available
```

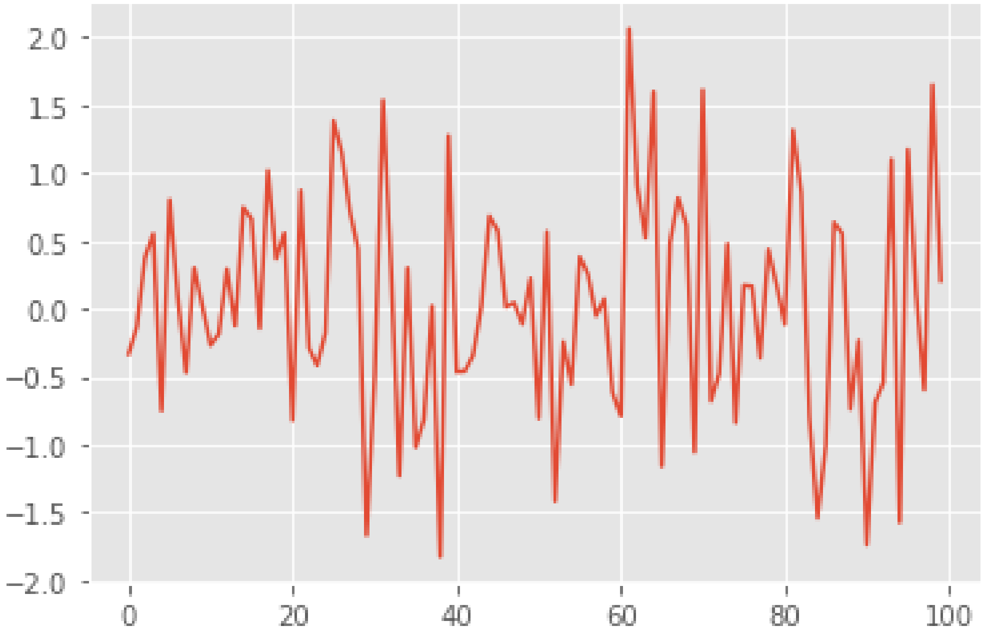
```
[20]: ['Solarize_Light2',
       '_classic_test_patch',
       'bmh',
       'classic',
       'dark_background',
       'fast',
       'fivethirtyeight',
       'ggplot',
       'grayscale',
       'seaborn',
       'seaborn-bright',
       'seaborn-colorblind',
       'seaborn-dark',
       'seaborn-dark-palette',
       'seaborn-darkgrid',
       'seaborn-deep',
       'seaborn-muted',
       'seaborn-notebook',
       'seaborn-paper',
       'seaborn-pastel',
       'seaborn-poster',
       'seaborn-talk',
       'seaborn-ticks',
       'seaborn-white',
       'seaborn-whitegrid',
       'tableau-colorblind10']
```

[21]:

```
# style can be universal or plot specific

plt.style.use('ggplot')
plt.plot(data)
```

[21]:

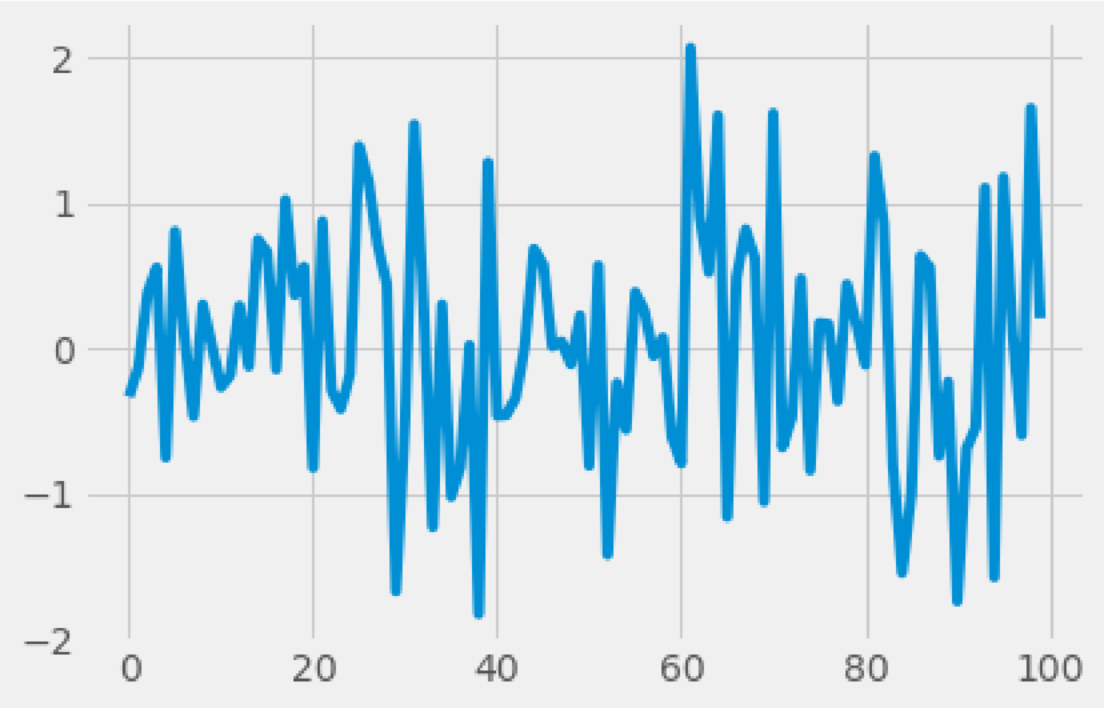
```
[<matplotlib.lines.Line2D at 0x7ff060546070>]
```



[22]:

```
plt.style.use('fivethirtyeight')
plt.plot(data)
```

```
[22]: [<matplotlib.lines.Line2D at 0x7ff060598730>]
```

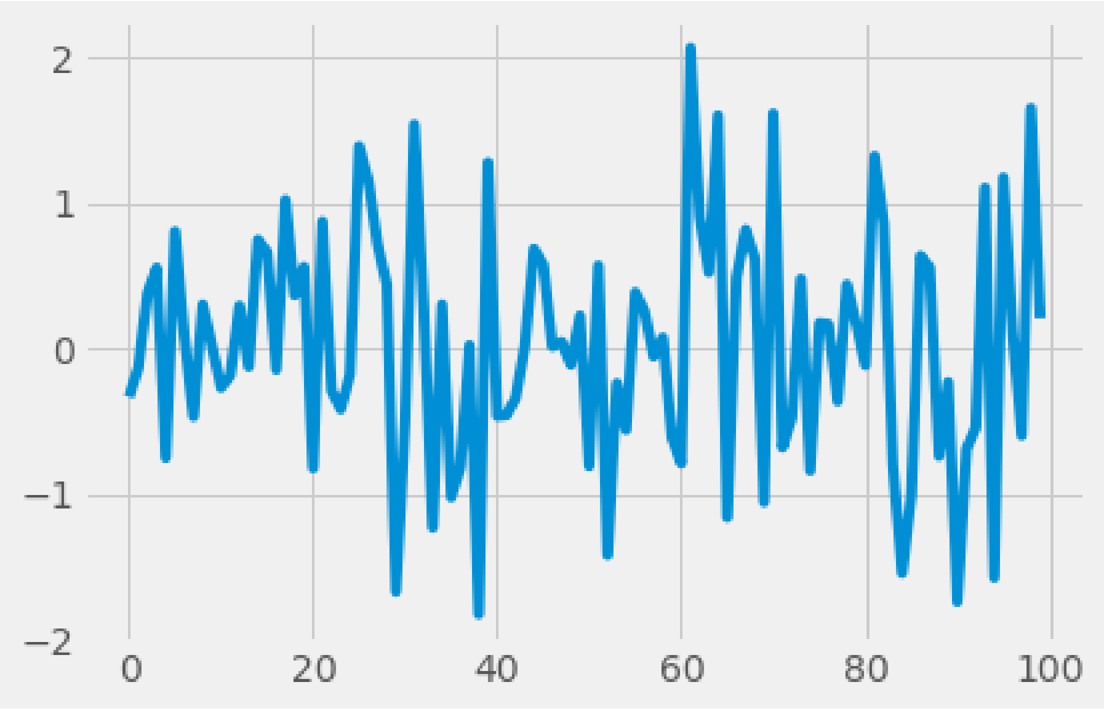


1.8.2 *Experiment*

Try out a couple of different styles to find one you like.

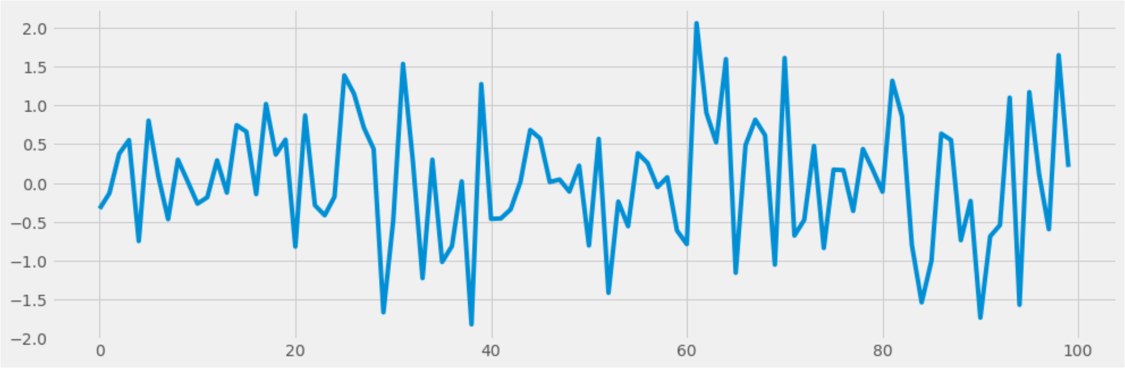
```
[23]: # plt.style.use('xxxxxxxx')
      plt.plot(data)
```

```
[23]: [<matplotlib.lines.Line2D at 0x7ff0605d7760>]
```



1.8.3 Figure size

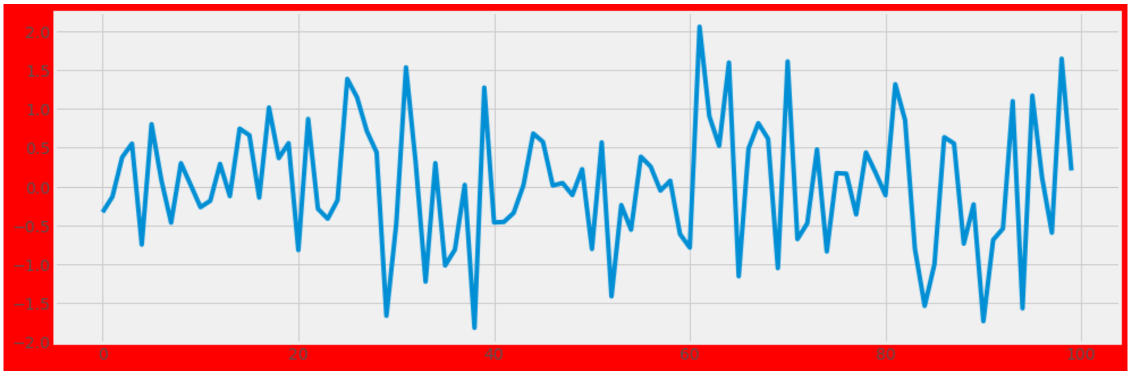
[24]:
```
plt.style.use('fivethirtyeight')
plt.figure(figsize = (15,5))
plt.plot(data)
plt.show()    # removes that little extra line of output
```



1.8.4 Facecolor

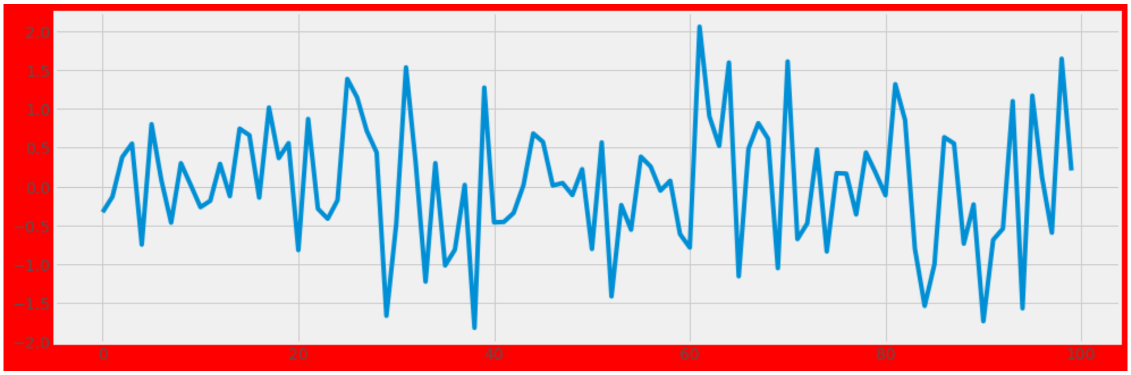
[25]:
```
plt.style.use('fivethirtyeight')
plt.figure(figsize = (15,5), facecolor='red')
plt.plot(data)
```

[25]:
```
[<matplotlib.lines.Line2D at 0x7ff0600f1190>]
```



1.8.5 Saving to a file

[26]:
```
plt.style.use('fivethirtyeight')
plt.figure(figsize = (15,5), facecolor='red')
plt.plot(data)
plt.savefig('new data', transparent=True)
```



1.8.6 *Experiment*

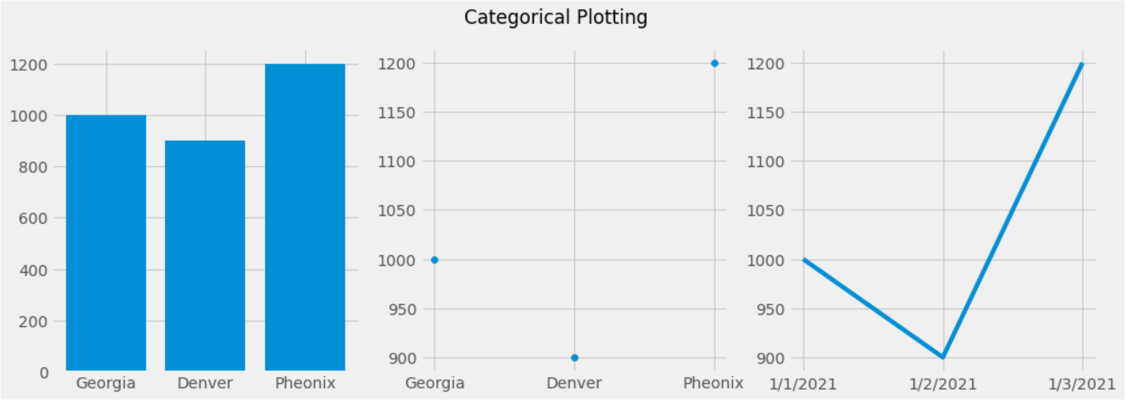
Using the empty code line below, try changing the facecolor and the figure size. Ave the plot to a file

[ ]:
```

```

1.9 Subplots

[27]:

```
names = ['Georgia', 'Denver', 'Pheonix']
values = [1000, 900, 1200]
dts = ['1/1/2021', '1/2/2021','1/3/2021']

plt.figure(figsize=(15, 5))

plt.subplot(131)
# plt.subplot(131, facecolor = 'r', frameon = True, title = 'xyz', ylabel = 'Employee Count')
plt.bar(names, values, label = 'values')
plt.subplot(132)
plt.scatter(names, values)
plt.subplot(133)
plt.plot(dts, values)
plt.suptitle('Categorical Plotting')
plt.show()
```



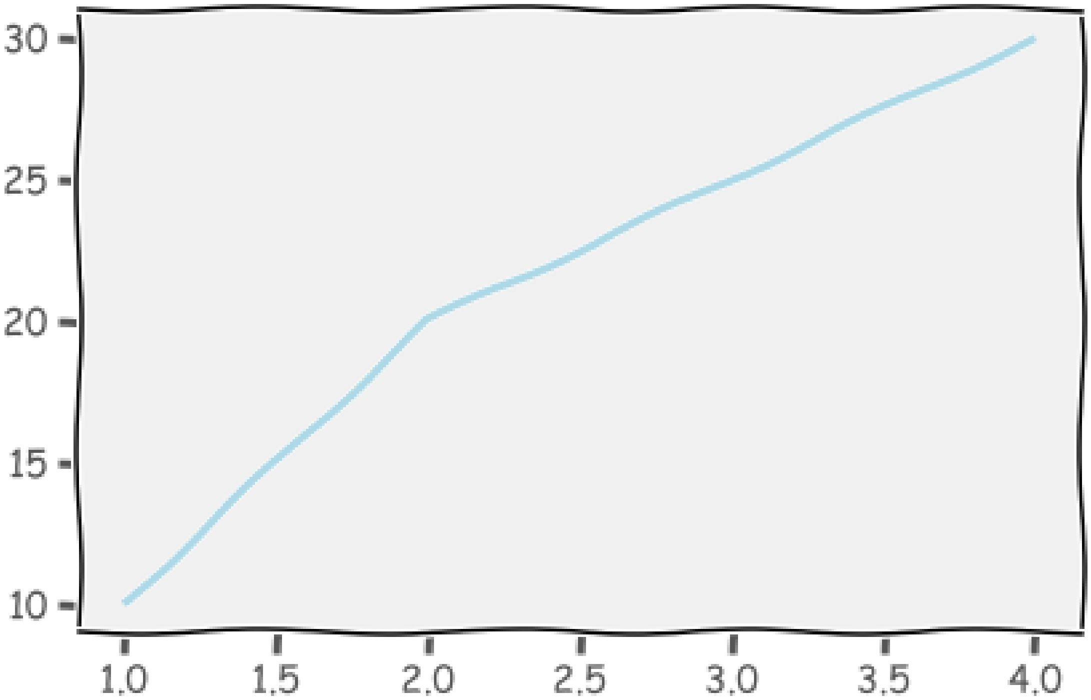
1.10 Just for a little fun....

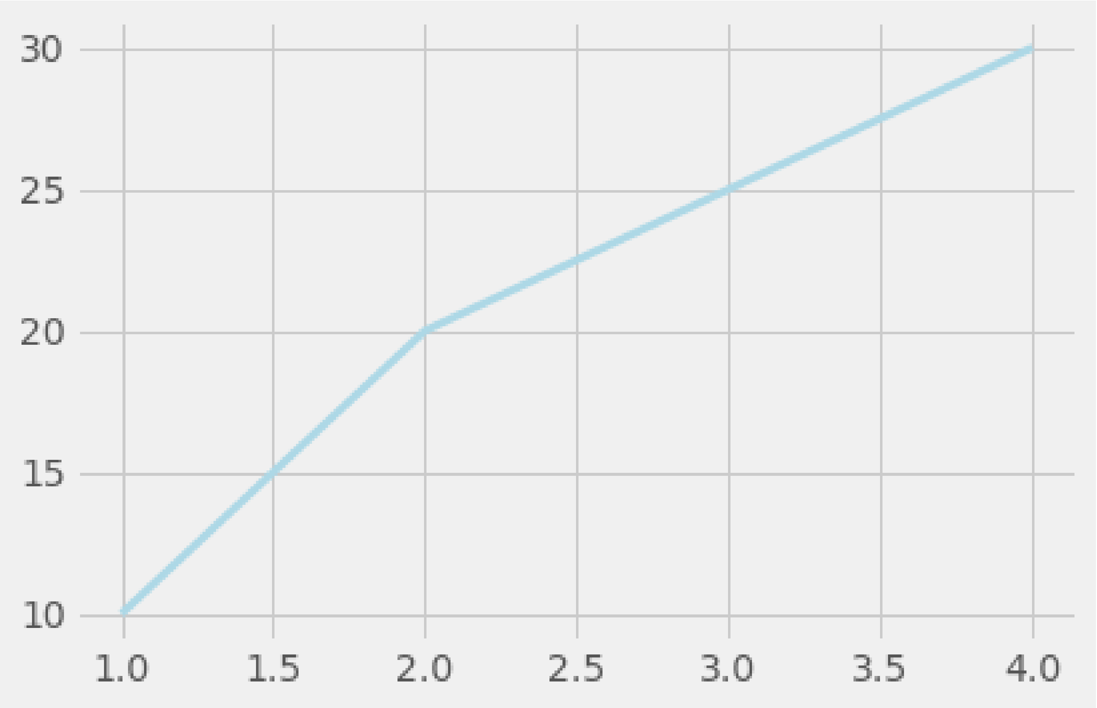
[28]:

```
with plt.xkcd():
    # This figure will be in XKCD-style
    fig1 = plt.figure()
    plt.plot([1, 2, 3, 4], [10, 20, 25, 30], color='lightblue', linewidth=3)
    # ...

# This figure will be in regular style
fig2 = plt.figure()
plt.plot([1, 2, 3, 4], [10, 20, 25, 30], color='lightblue', linewidth=3)
```

[28]:

```
[<matplotlib.lines.Line2D at 0x7ff06048b520>]
```


30
25
20
15
10
1.0
1.5
2.0
2.5
3.0
3.5
4.0


30
25
20
15
10
1.0
1.5
2.0
2.5
3.0
3.5
4.0

1.11 Exercise 3 - Create a figure with 4 subplots - 10 minutes

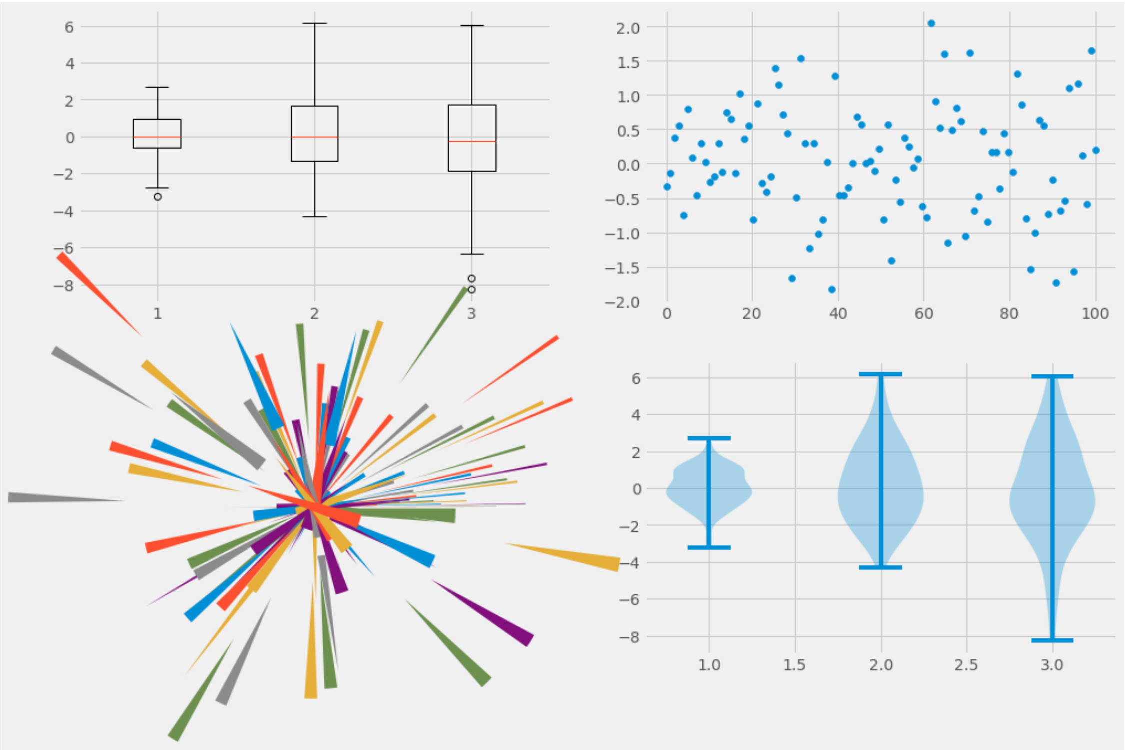
In - position 1 add a boxplot using y - position 2 add a scatterplot using x and data - position 3 add a pie chart of x - position 4 add a violin plot using y

[29]:

```
x = np.linspace(0, 100, 100)
y = [np.random.normal(0, std, size=100) for std in range(1, 4)]
z = np.linspace(100, 200, 100)
```

[30]:

```
plt.figure(figsize=(15,10))
plt.subplot(221)
plt.boxplot(y)
plt.subplot(222)
plt.scatter(x,data)
plt.subplot(223)
#plt.xcorr(data)
plt.pie(x,data)
plt.subplot(224)
plt.violinplot(y)
plt.show()
```



6
4
2
0
−2
−4
−6
−8
1
2
3
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
0
20
40
60
80
100
6
4
2
0
−2
−4
−6
−8
1.0
1.5
2.0
2.5
3.0