3 - Matplotlib_plt

March 22, 2023

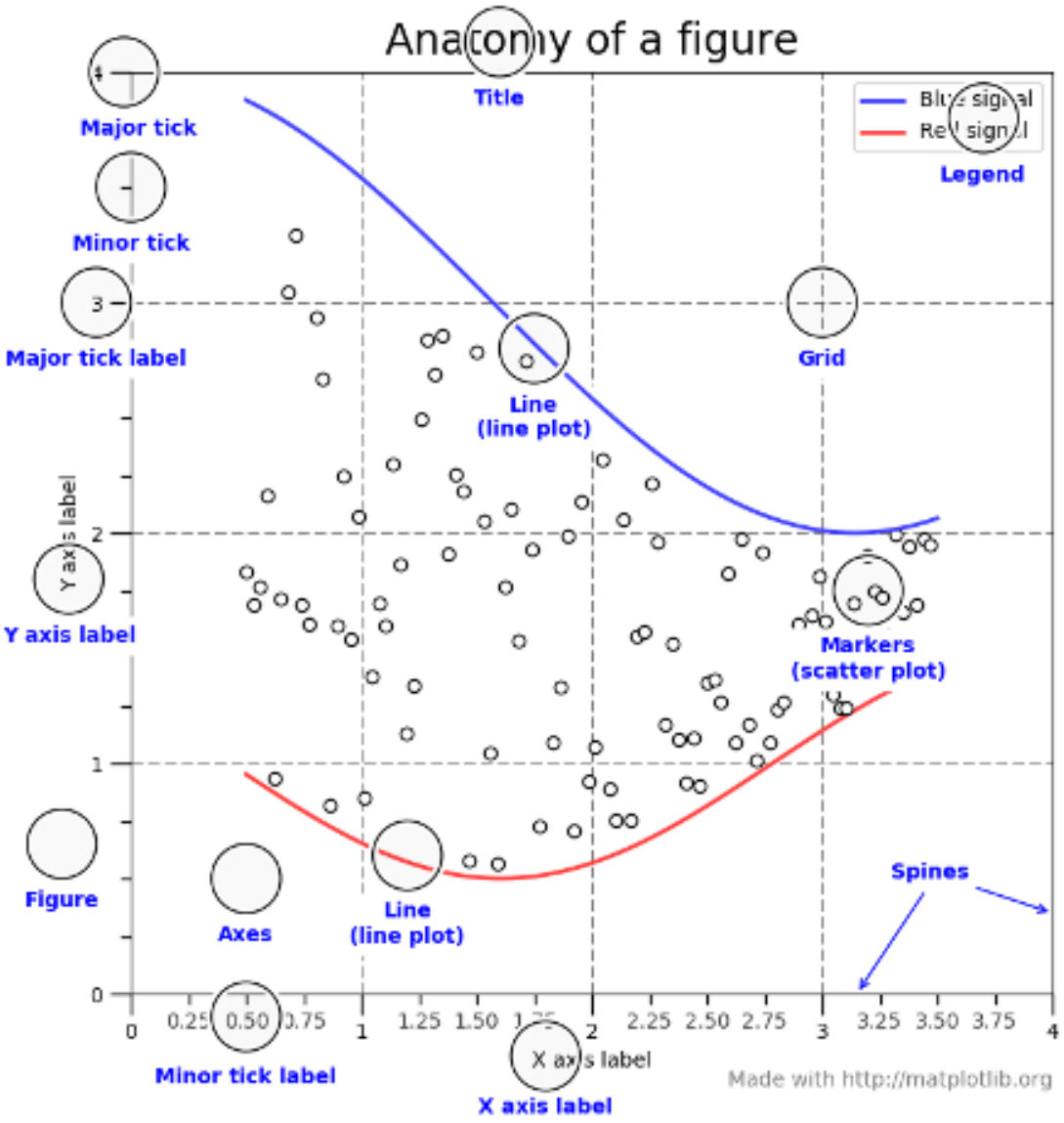
Table of Contents: Matplotlib Part 1

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

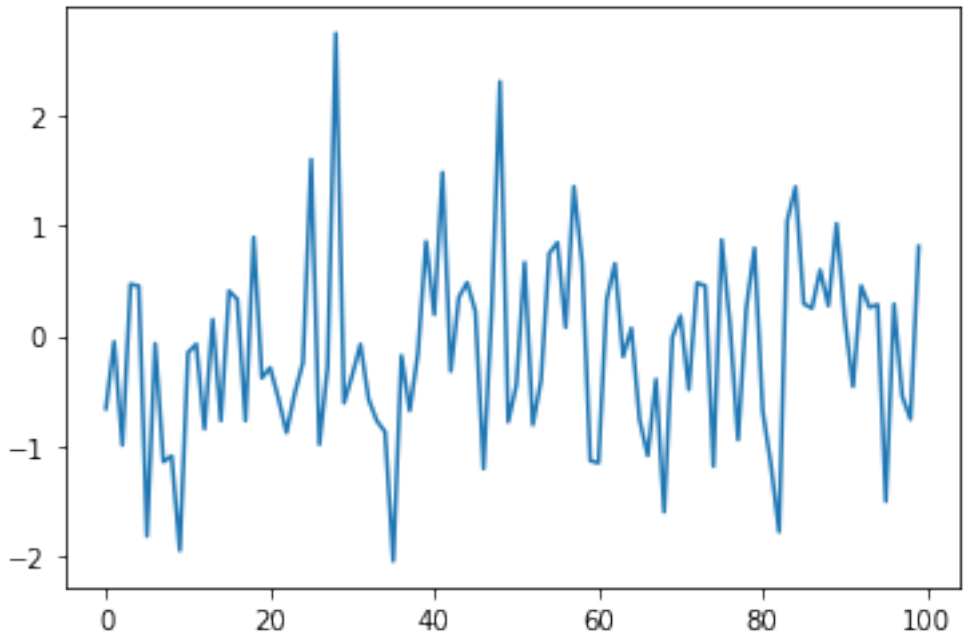
[1]:
```
import matplotlib.pyplot as plt
import matplotlib as mpl
import numpy as np
#mpl.rcParams['lines.linewidth'] = 2
#mpl.rcParams['lines.linestyle'] = '--'


#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)
```

[2]:
```
data

# matplotlib works with data in an array
```

[2]:
```
array([-0.66230409, -0.0528785 , -0.99427031,  0.47196466,  0.45391882,
       -1.8233921 , -0.07017794, -1.14683679, -1.09236095, -1.95414364,
       -0.15222187, -0.07543837, -0.84621457,  0.15046766, -0.77261829,
        0.40938832,  0.33313624, -0.77247625,  0.8978054 , -0.38265872,
       -0.28998865, -0.5652571 , -0.8804084 , -0.52151547, -0.24182888,
        1.60657126, -0.98824036, -0.28458281,  2.755008  , -0.61024853,
       -0.35648584, -0.07756419, -0.57755368, -0.77332496, -0.87297293,
       -2.04953858, -0.17903422, -0.68066019, -0.16191081,  0.86095939,
        0.19472453,  1.48895515, -0.32137869,  0.35447771,  0.48544429,
        0.22359133, -1.20936783,  0.25151713,  2.31705599, -0.78229426,
       -0.44566806,  0.66857681, -0.80896747, -0.40340108,  0.75192189,
        0.85022177,  0.07777088,  1.36130684,  0.6673949 , -1.13659919,
       -1.15559504,  0.33010242,  0.65865567, -0.18778942,  0.07029405,
       -0.76165523, -1.09047169, -0.39593365, -1.60265775, -0.01086848,
        0.18286819, -0.48820961,  0.48179535,  0.45582819, -1.18670888,
        0.87533629,  0.1155582 , -0.9467183 ,  0.24874587,  0.79969583,
       -0.68109112, -1.16916737, -1.78358508,  1.05052317,  1.35729212,
        0.29313354,  0.25182493,  0.60053265,  0.27472055,  1.02184554,
        0.16175054, -0.46100141,  0.45587678,  0.26092114,  0.28423023,
       -1.50654094,  0.29023959, -0.54833166, -0.75855698,  0.81446268])
```

[3]:
```
# 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
```

[3]: [<matplotlib.lines.Line2D at 0x10bbe7fd0>]

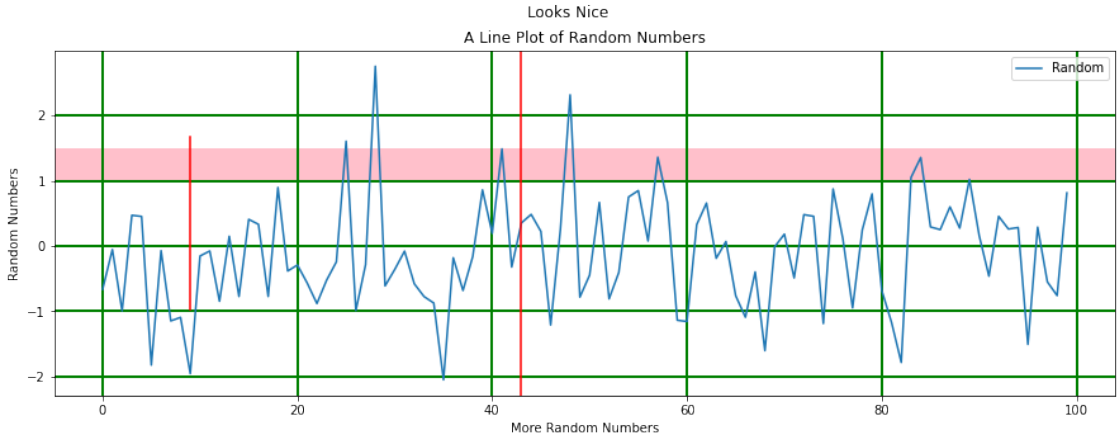


1.3 Plot-specific options

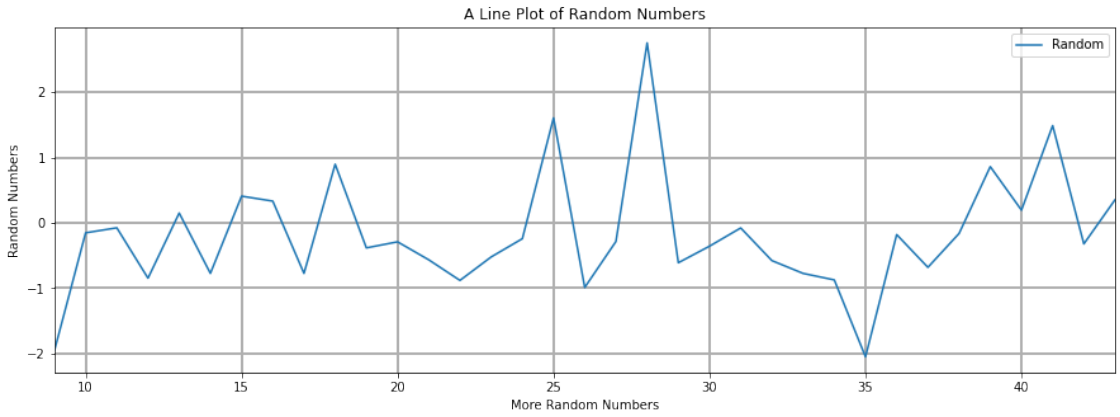
1.3.1 Other plot components

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

[4]:
```
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
```



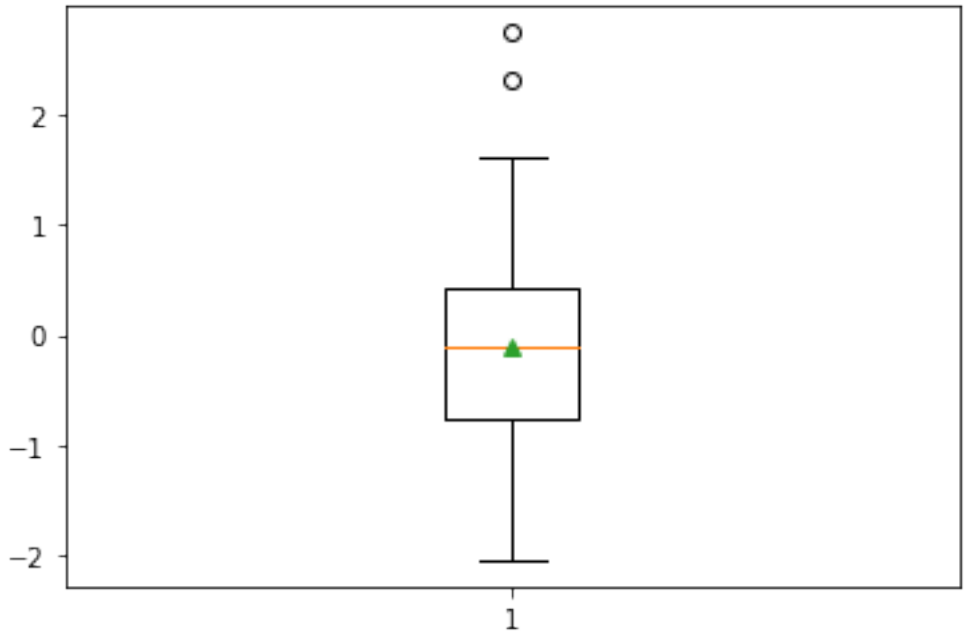
[5]:
```
# In the sample above, reference lines were placed at 9 and 43.
# xlim (or ylim) can be used to control the range of the axis.

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

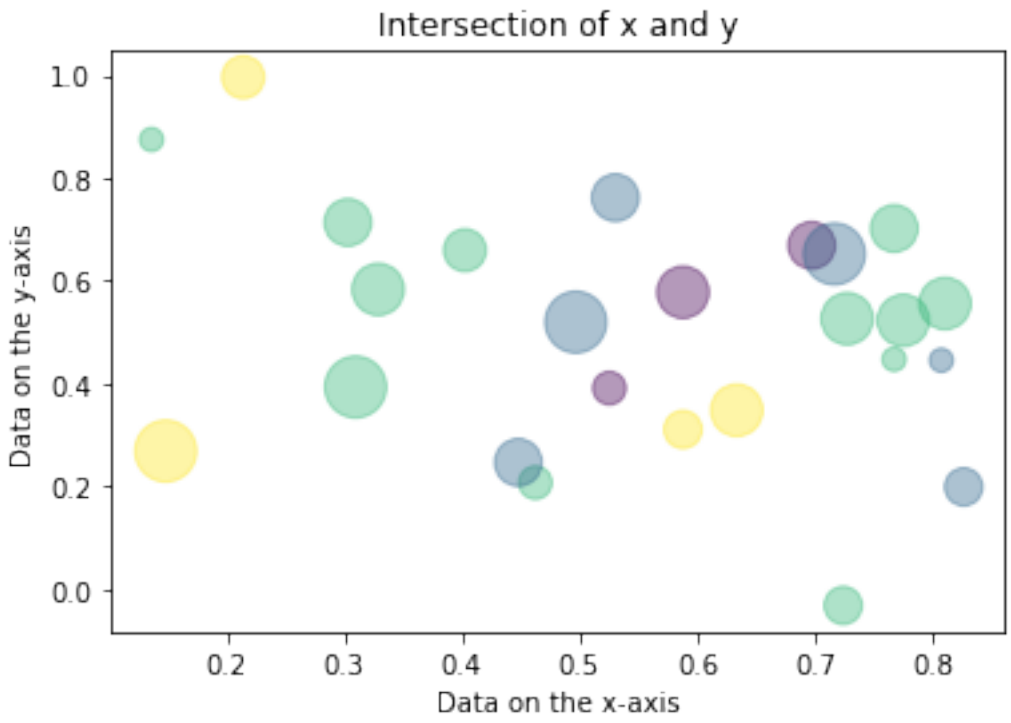
1.4.1 Boxplot

[6]:
```
plt.boxplot(data, showmeans=True)
plt.show()
```



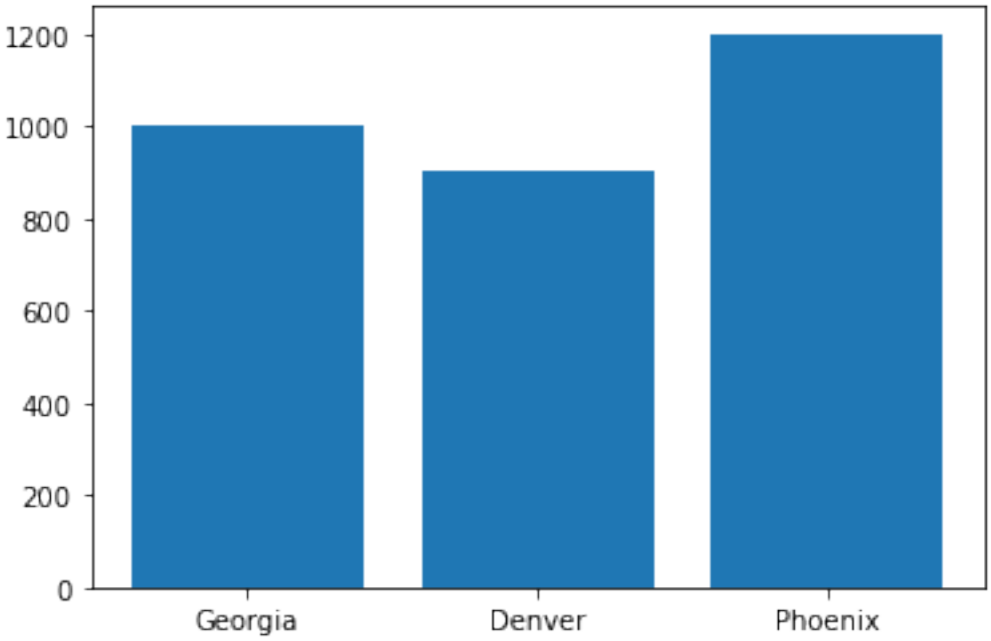
1.4.2 Scatterplot

[7]:
```
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()
```



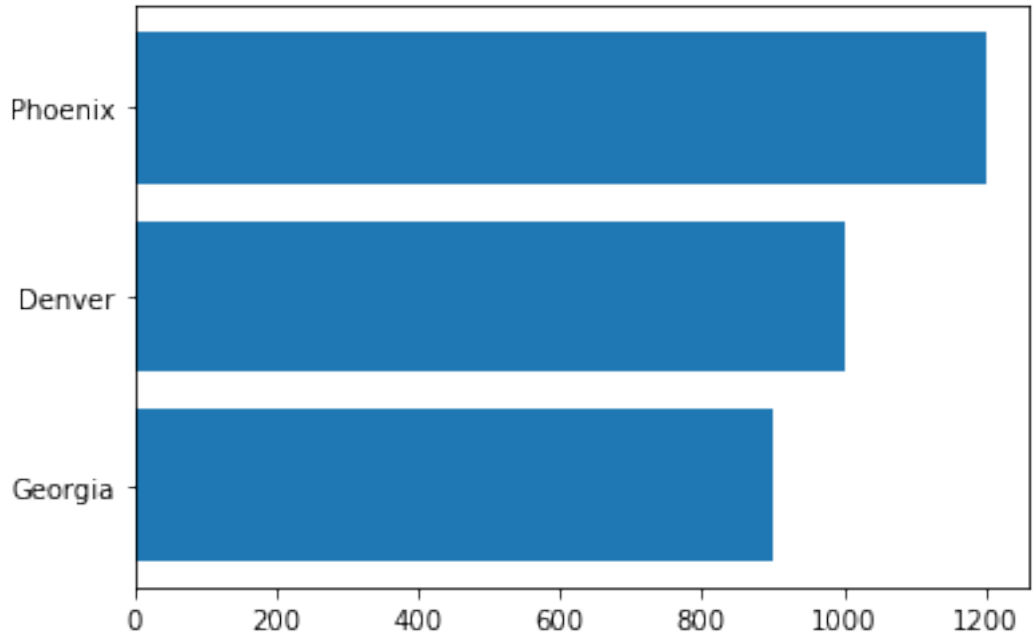
1.4.3 Bar

[8]:
```
names = ['Georgia', 'Denver', 'Phoenix']
values = [1000, 900, 1200]

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



1.4.4 Bar (Horizontal)

[9]:
```
names = ['Georgia', 'Denver', 'Phoenix']
values = [1000, 900, 1200]

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

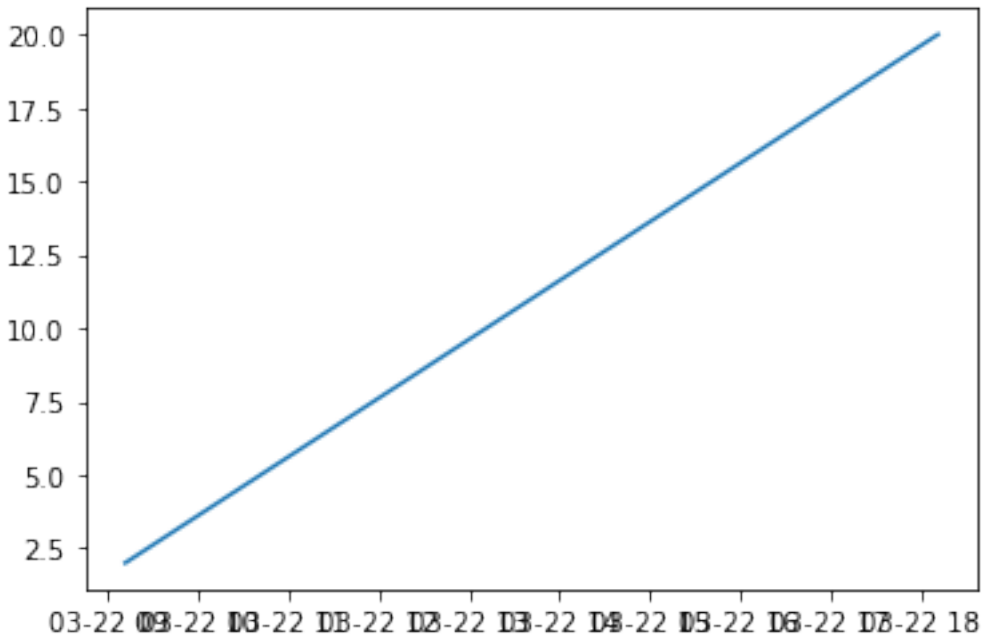


1.5 Exercise 1 - Create a histogram (5 minutes)

[10]:
```
# Using 'data', create a histogram.
# Do 20 bins make a bettter presentation?
```

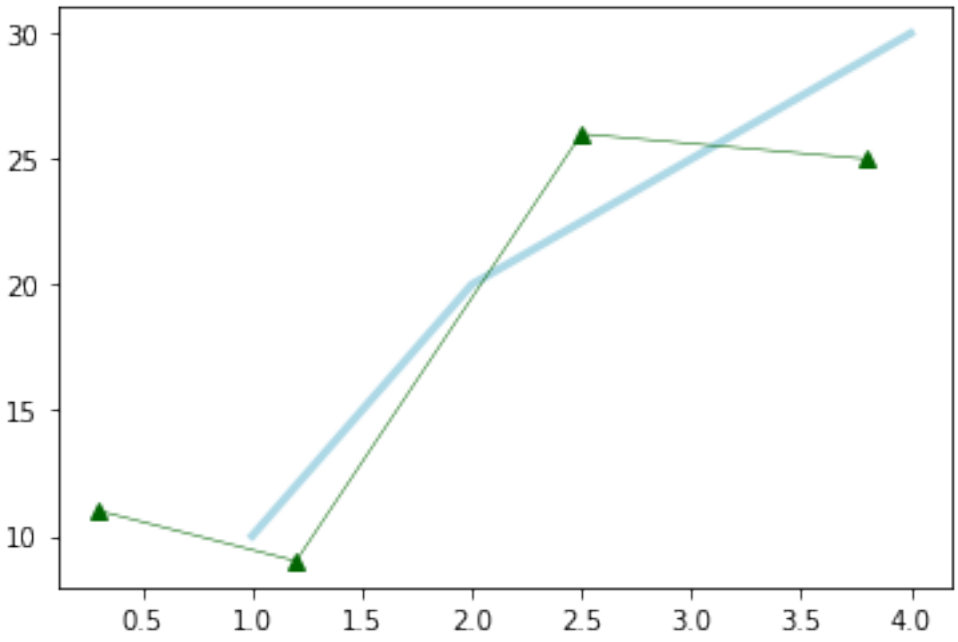
1.5.1 Plot with Dates

[11]:
```
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()
```

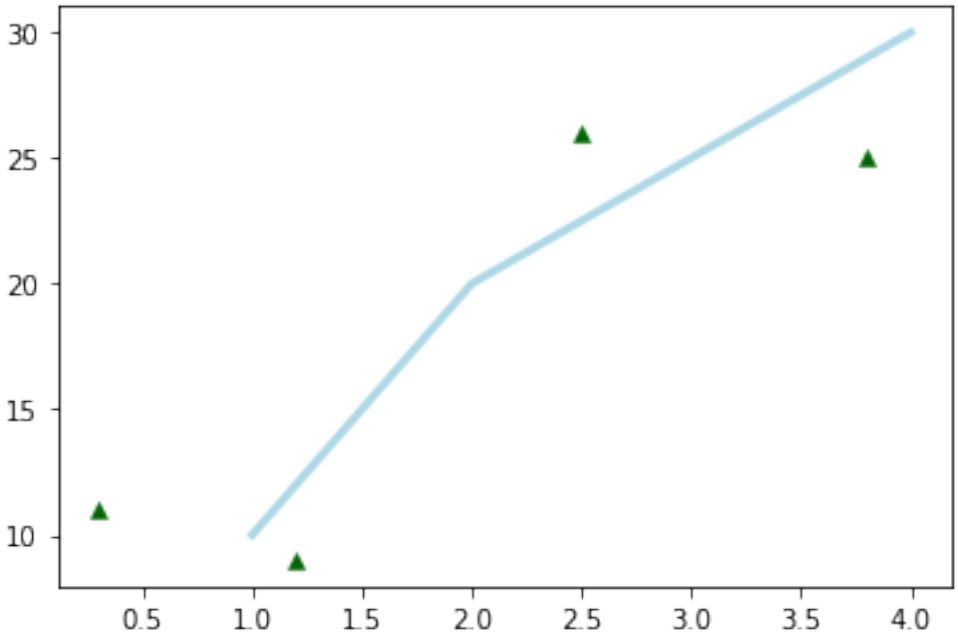


1.6 Multiple datasets in a single plot

[12]:
```
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()
```



[13]:
```
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)

[ ]:
```

```

1.8 Figure level modifications

- Changing the sytle

- Changing the figure size
- Changing the facecolor

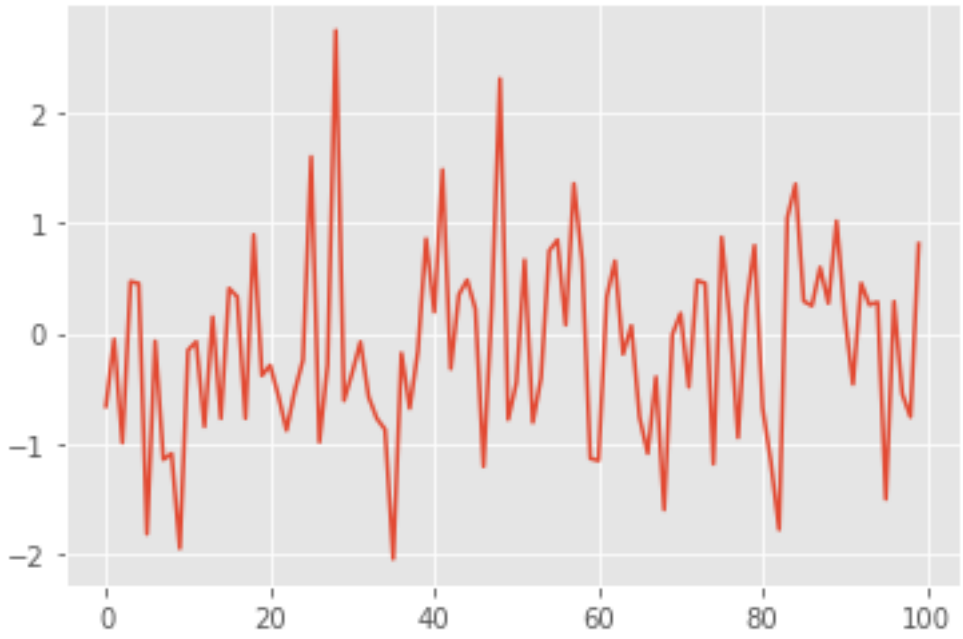
1.8.1 Plot Styles

[14]:
```
plt.style.available
```

[14]:
```
['Solarize_Light2',
 '_classic_test_patch',
 '_mpl-gallery',
 '_mpl-gallery-nogrid',
 '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']
```

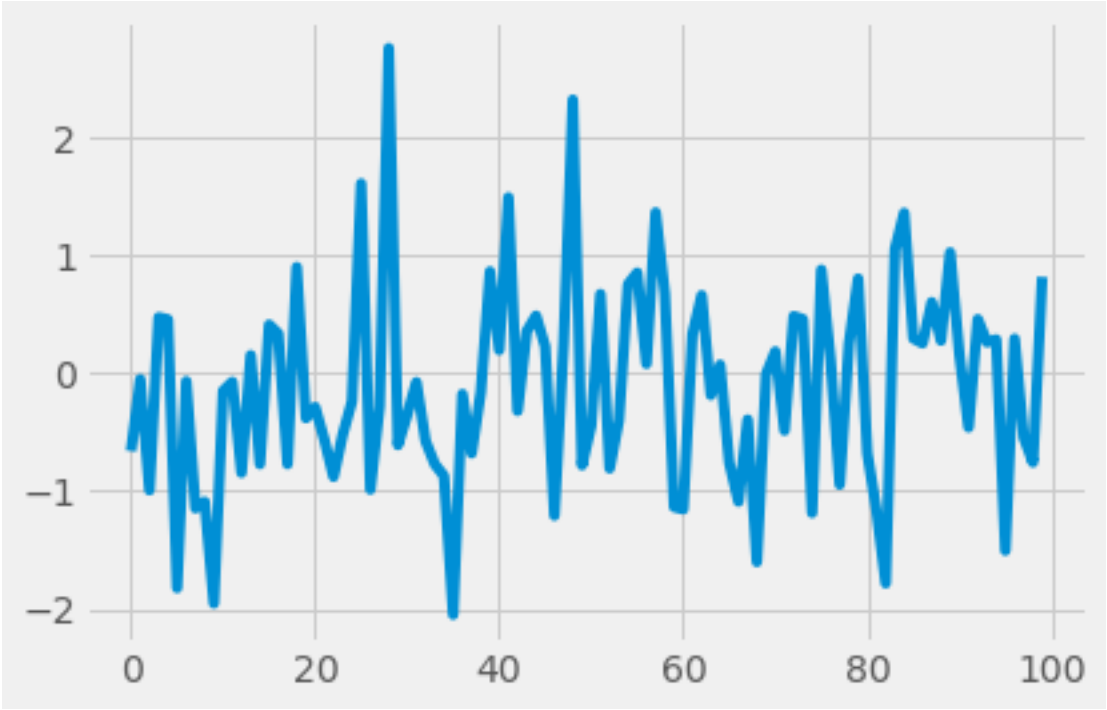
[15]:
```
# style can be universal or plot specific

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

[15]:
```
[<matplotlib.lines.Line2D at 0x10c0d1a60>]
```



[16]:
```
plt.style.use('fivethirtyeight')
plt.plot(data)
```

[16]: [<matplotlib.lines.Line2D at 0x10c136af0>]

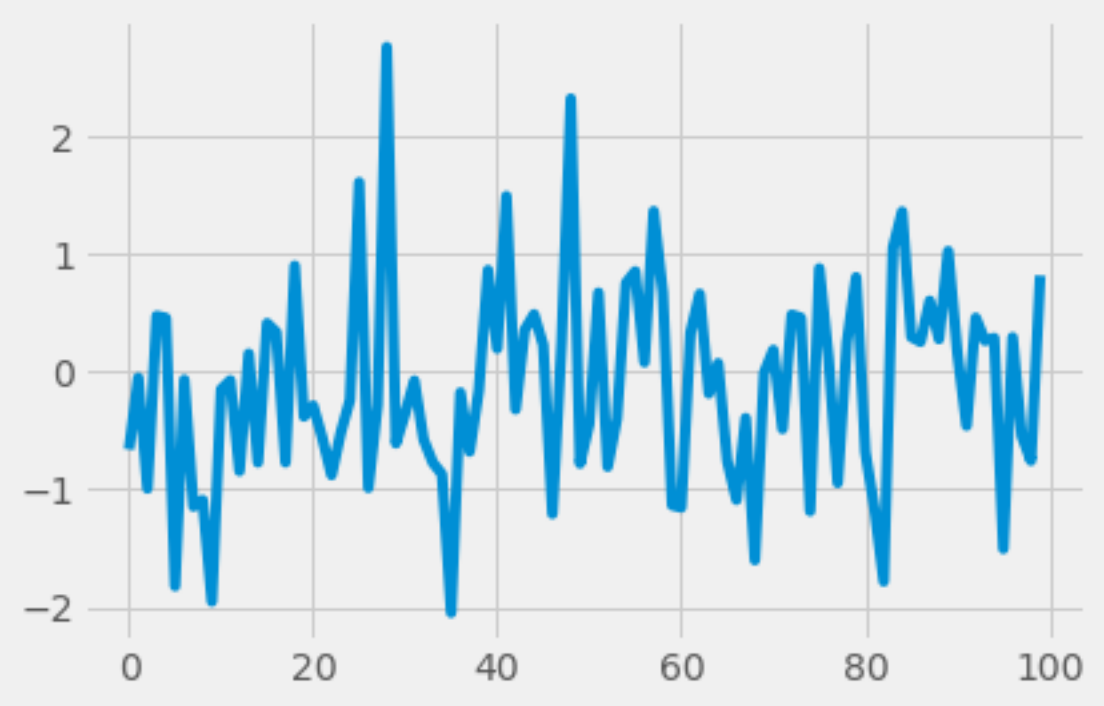


1.8.2 *Experiment*

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

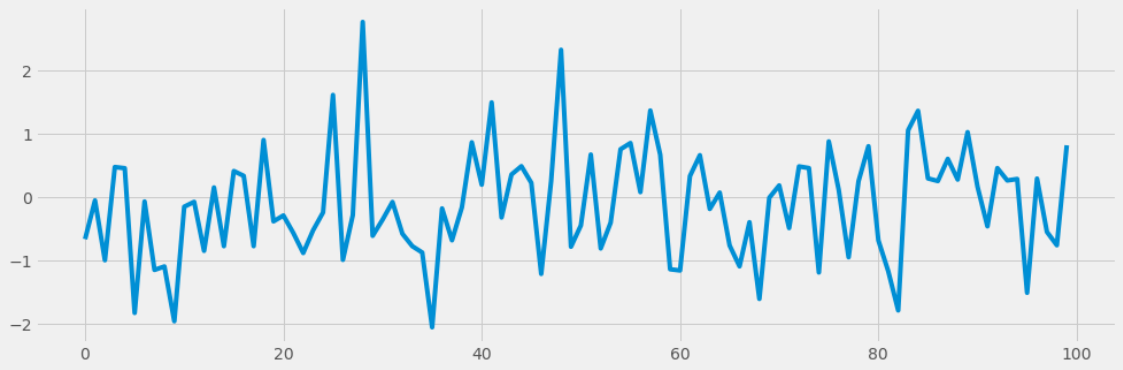
[17]:
```
# plt.style.use('xxxxxxxx')
plt.plot(data)
```

[17]:
```
[<matplotlib.lines.Line2D at 0x10c066d60>]
```



1.8.3 Figure size

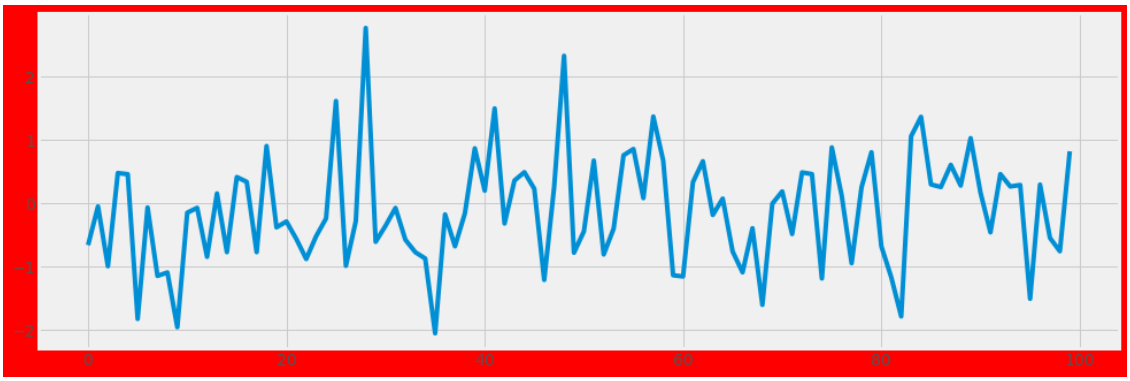
[18]:
```
plt.style.use('fivethirtyeight')
plt.figure(figsize = (15,5))
plt.plot(data)
plt.show()   # removes that little extra line of output
```



1.8.4 Facecolor

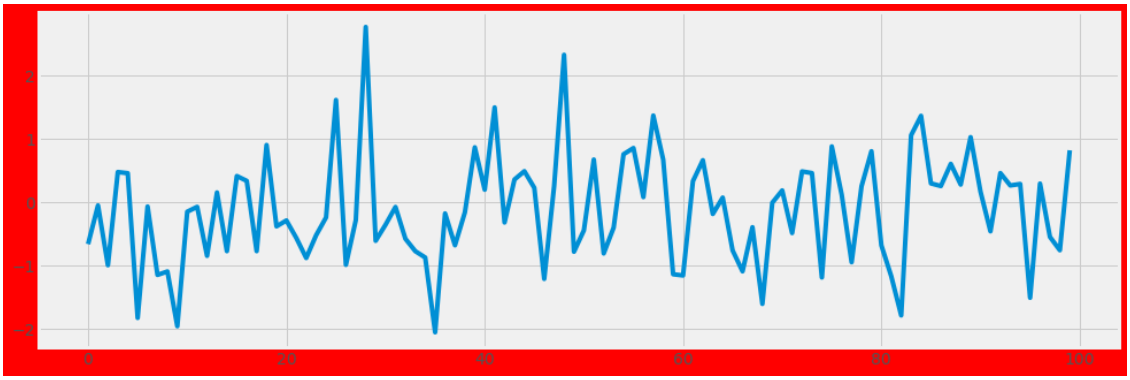
[19]:
```
plt.style.use('fivethirtyeight')
plt.figure(figsize = (15,5), facecolor='red')
plt.plot(data)
```

[19]:
```
[<matplotlib.lines.Line2D at 0x10beebbe0>]
```



1.8.5 Saving to a file

[20]:
```
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

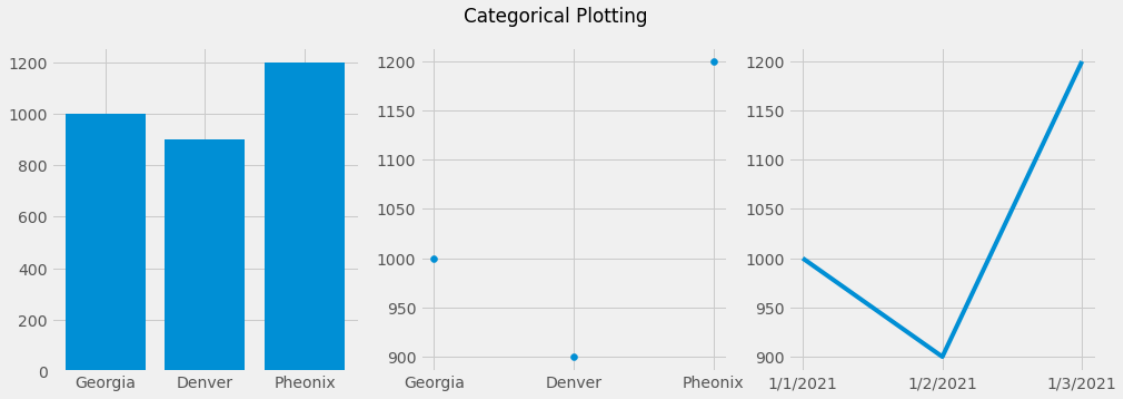
[ ]:

```
```

1.9 Subplots

[21]:

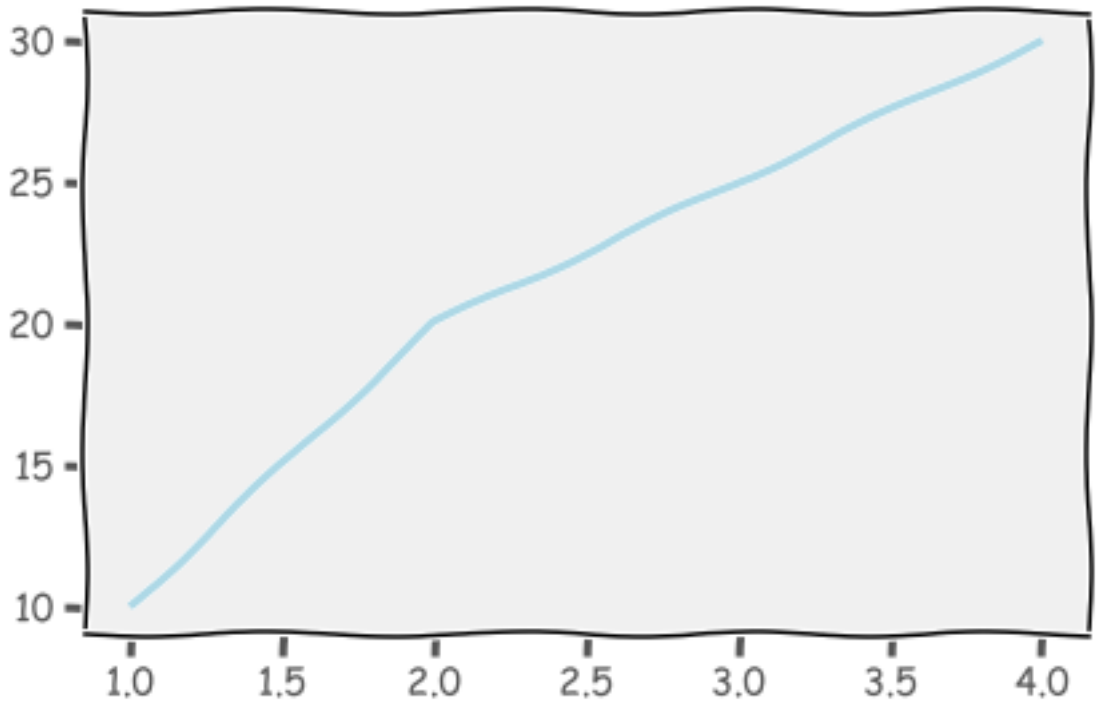
```
names = ['Georgia', 'Denver', 'Pheonix']
values = [1000, 900, 1200]
dts = ['1/1/2021', '1/2/2021','1/3/2021']

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()
```

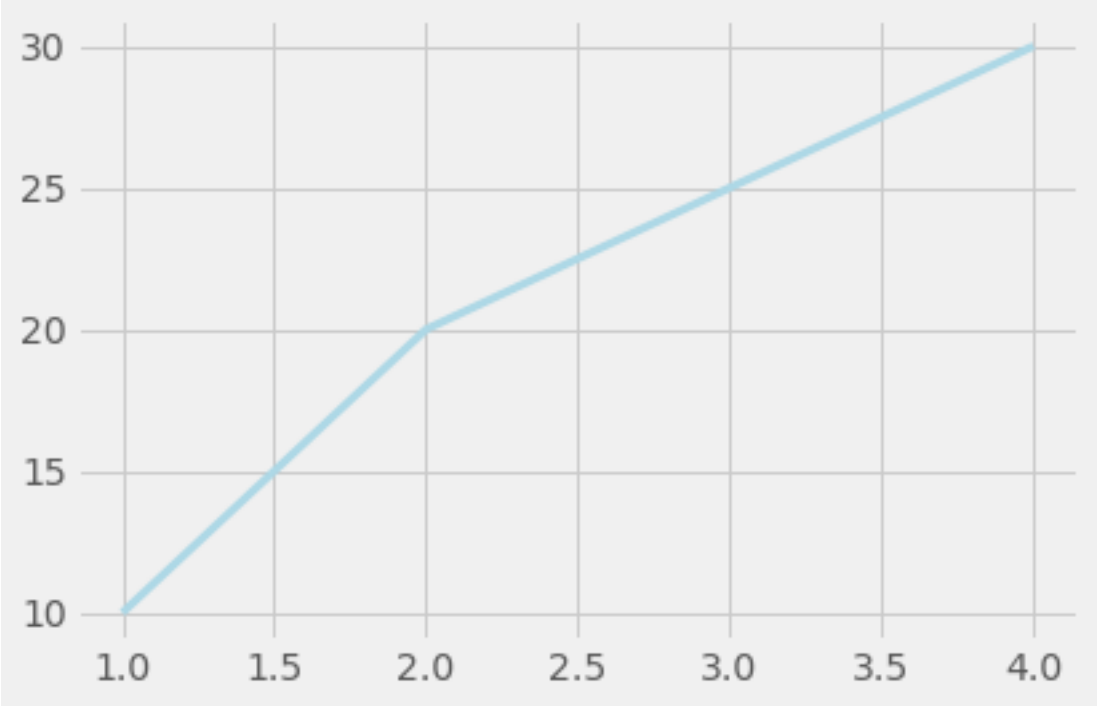


1.10 Just for a little fun....

[22]:
```
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)
```

[22]:
```
[<matplotlib.lines.Line2D at 0x10bd8cf40>]
```


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

[23]:
```
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)
```

[ ]:

[ ]: