# Instructor 1 - Matplotlib 2022

June 20, 2022

## 1 Figures, Axes & Artists

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.)

### 1.1 Two Big Plotting Concepts

Click here for matplotlib documentation - - matplotlib.org

Go here for the life cycle of a plot

## 2 Types of Input Used by Matplotlib

## 3 Coding Interfaces

Matplotlib has three interfaces (ways to write code) - pyplot: hides the complexity of object-oriented coding. - object oriented: provides access to more functions and control over the visualizations - pylab (not recommended for use)

[1]:

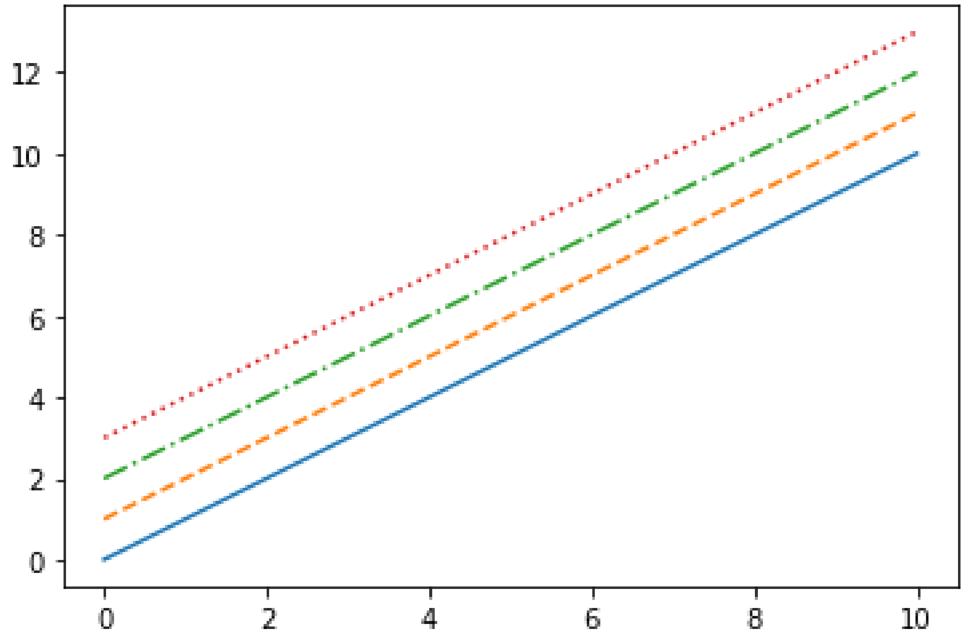
```python
import numpy as np
import matplotlib.pyplot as plt
import random
import math
```

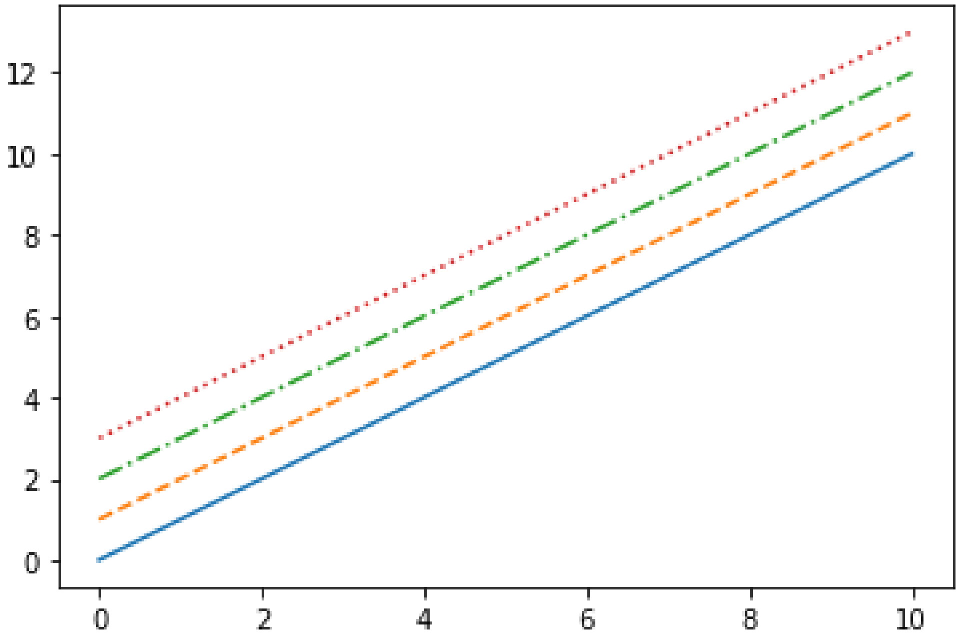
[2]:

```python
# Showing subtle different between pyploy and OO approach
x = np.linspace(0, 10, 1000)

# pyplot approach

plt.plot(x, x + 0, linestyle = 'solid')  # plot creates a line plot
plt.plot(x, x + 1, linestyle='dashed')
plt.plot(x, x + 2, linestyle='dashdot')
plt.plot(x, x + 3, linestyle='dotted')

# Object-oriented approach
```

```
fig = plt.figure()
ax1 = fig.add_subplot(1,1,1) # the same as fig.add_subplot(111)

ax1.plot(x, x + 0, linestyle='solid')
ax1.plot(x, x + 1, linestyle='dashed')
ax1.plot(x, x + 2, linestyle='dashdot')
ax1.plot(x, x + 3, linestyle='dotted')
```

```
[2]: [<matplotlib.lines.Line2D at 0x7fc330dd4850>]
```





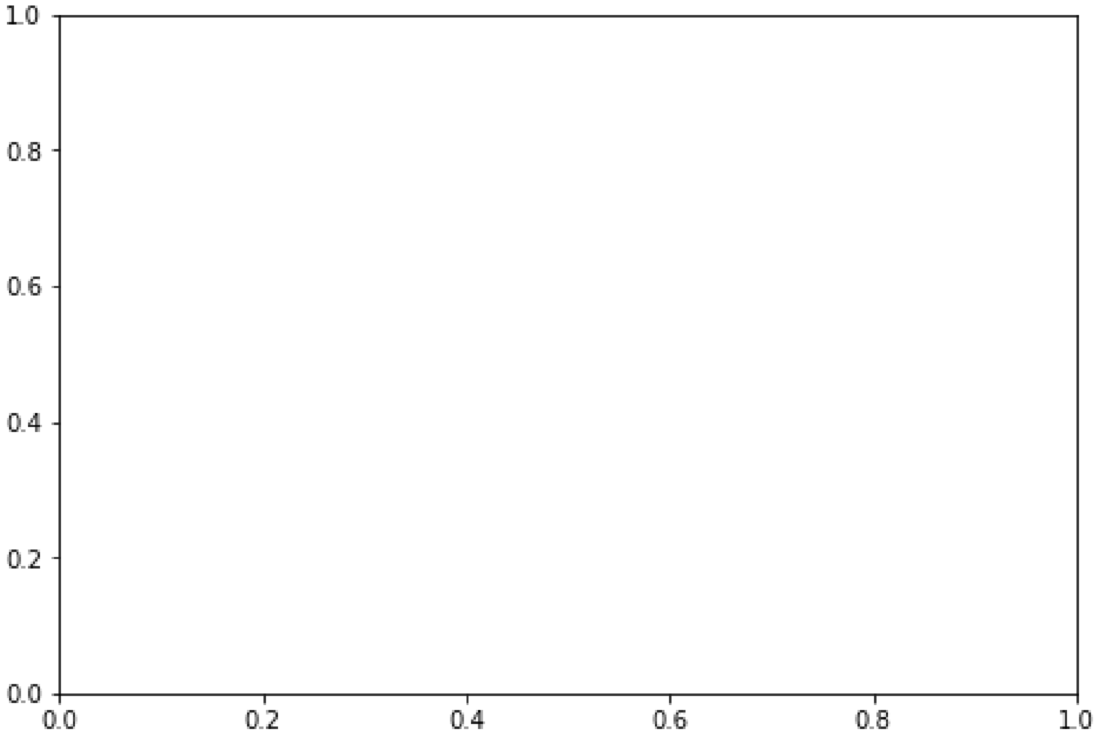
## 4 The Lifecycle of a Plot

- create the figure
- add the axes
- add the plot
- customize the plot
- customize the figure
- save figure to a file

[3]:
```
delivery_num = [1, 4, 9, 16, 25,36,49, 64]
delivered = [1, 16, 30, 42,55, 68, 77,88]
administered = [1,6,12,18,28, 40, 52, 65]
```

#### 4.0.1 Create the figure and the axes

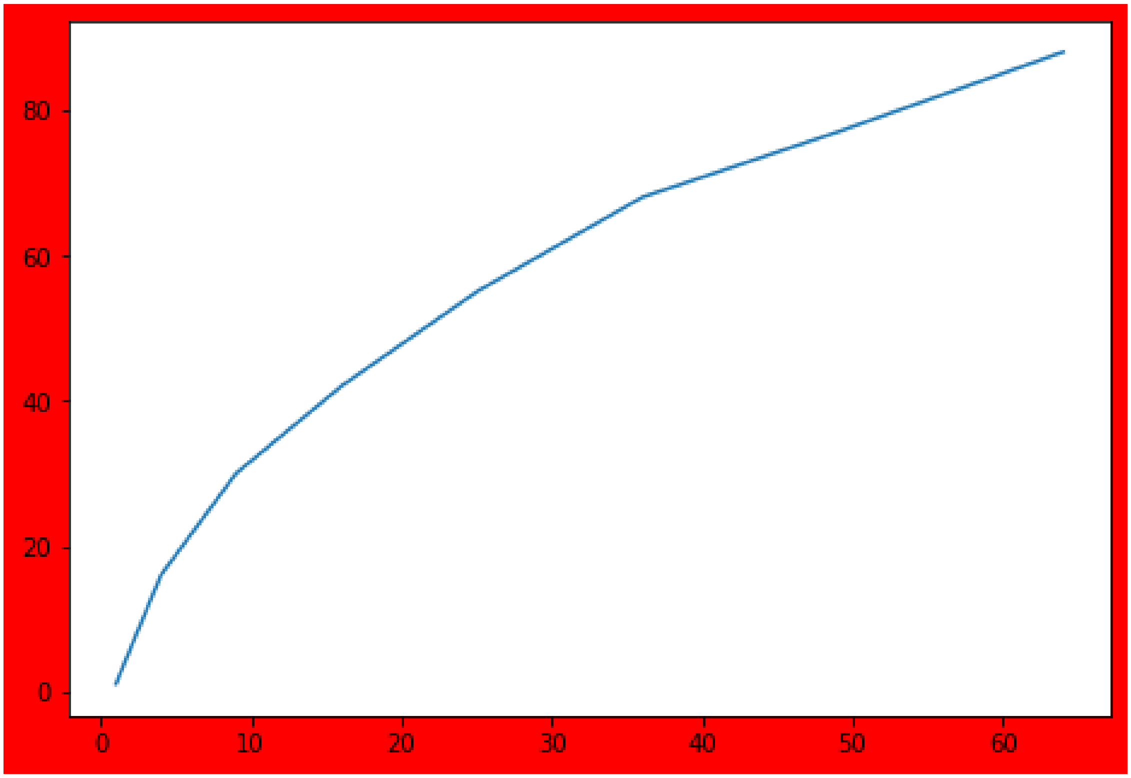
[4]:
```
#plt.rcParams.update(plt.rcParamsDefault)
fig = plt.figure()              # plt.figure calls matplotlib.figure
ax1 = fig.add_axes([0,0,1,1])   # left, bottom, width, height
```



### 4.0.2 Add a plot to the axes

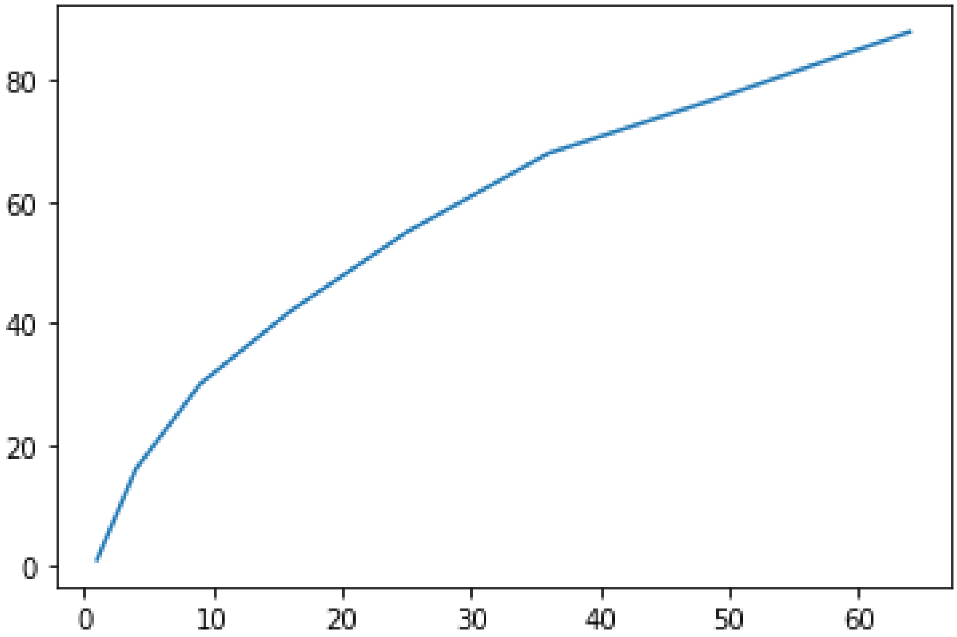
```
[5]: fig = plt.figure(facecolor = 'red') # fig is just a name.  It can be anbything␣
     ↪you want.
     ax = fig.add_axes([0,0,1,1])        # left, bottom, width, height
     ax.plot(delivery_num,delivered)
```

```
[5]: [<matplotlib.lines.Line2D at 0x7fc33116df70>]
```



```
[6]: # A little shortcut

     fig, ax = plt.subplots()
     ax.plot(delivery_num,delivered)
```

```
[6]: [<matplotlib.lines.Line2D at 0x7fc330d68130>]
```

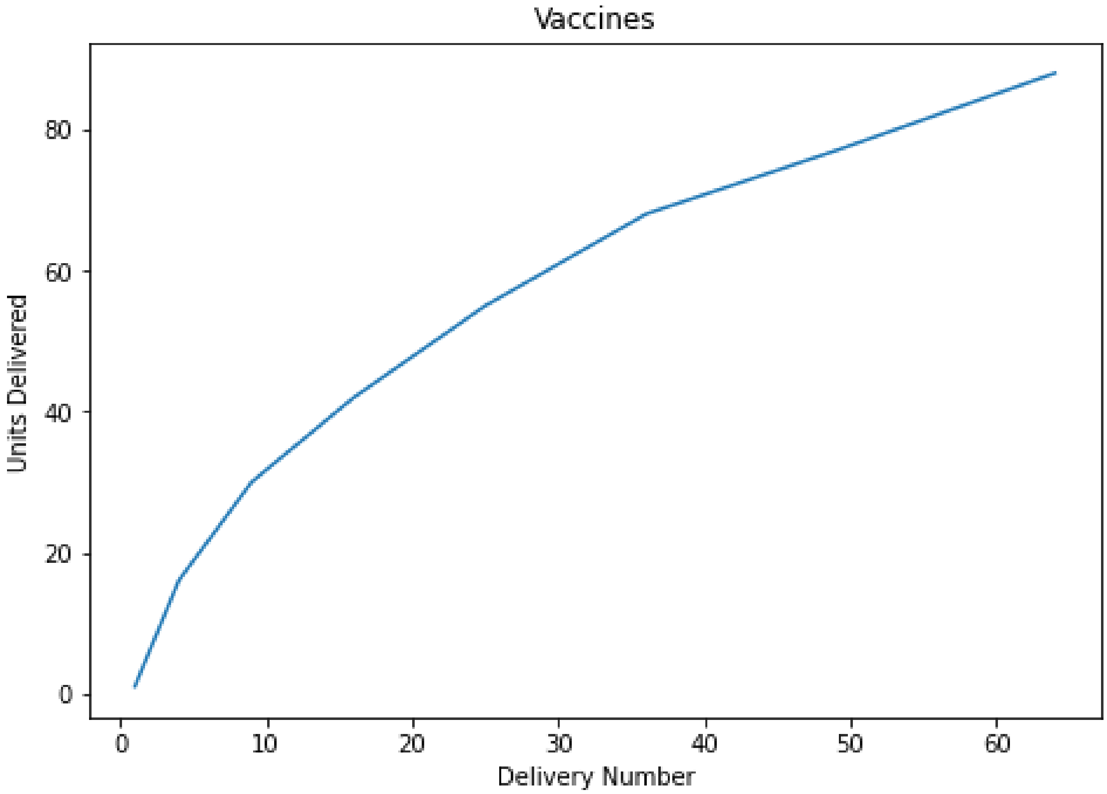


#### 4.0.3 Customize the plot

- Labels
- Legend
- Axis limits
- Colors and marks

[7]:
```
# Set axis labels
fig = plt.figure()
ax = fig.add_axes([0,0,1,1])
ax.plot(delivery_num,delivered)

ax.set_title("Vaccines")
ax.set_xlabel('Delivery Number')
ax.set_ylabel('Units Delivered')
```

[7]:
```
Text(0, 0.5, 'Units Delivered')
```

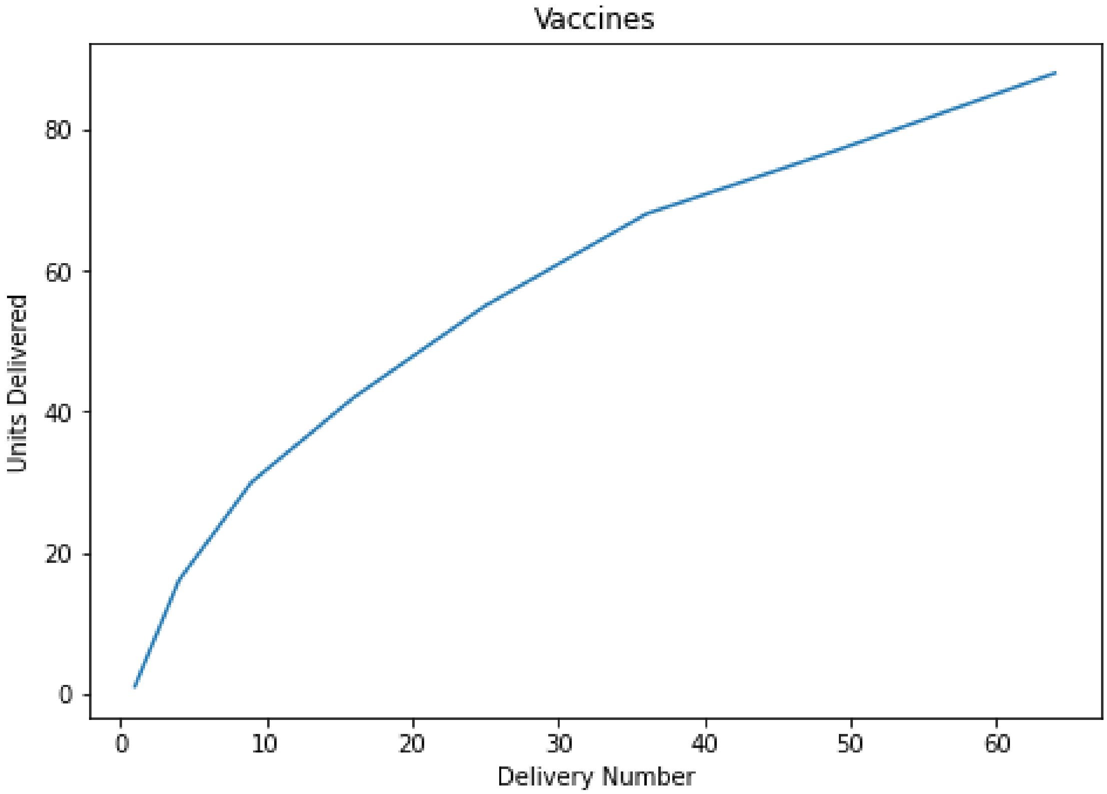


[8]:

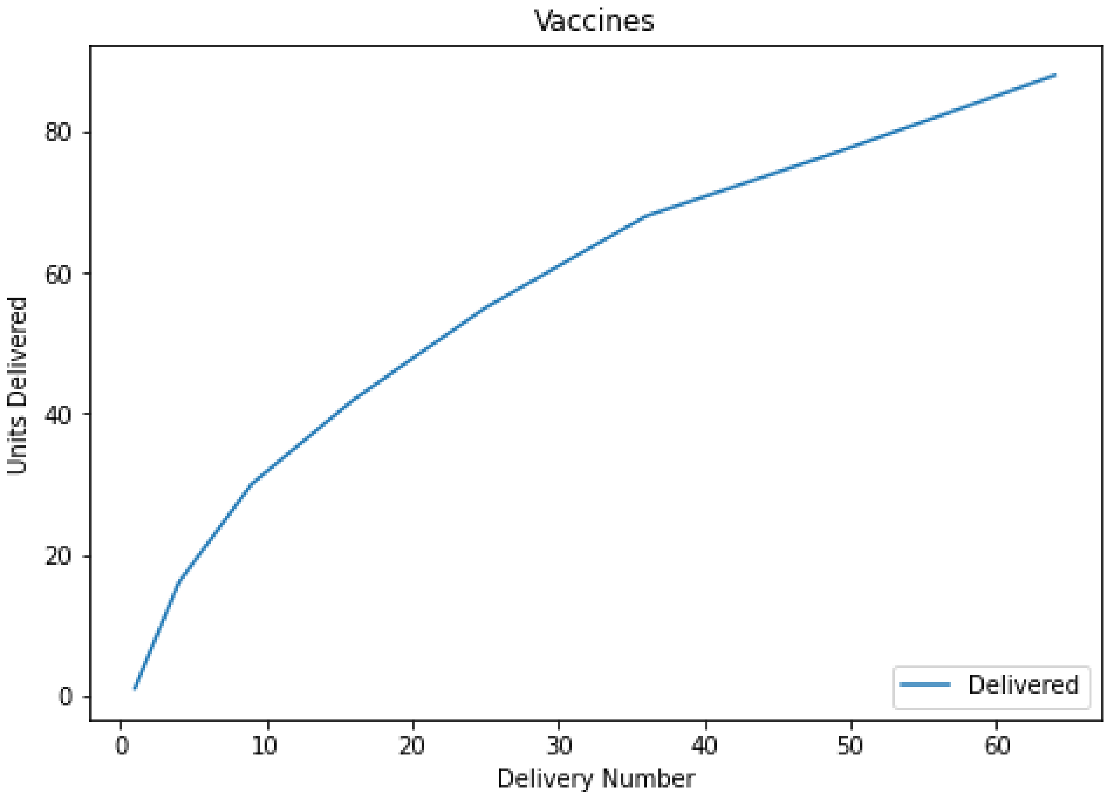
```
# Set line labels
fig = plt.figure()
ax = fig.add_axes([0,0,1,1])
ax.plot(delivery_num,delivered, label = 'Delivered')

ax.set_title("Vaccines")
ax.set_xlabel('Delivery Number')
ax.set_ylabel('Units Delivered')
```

[8]:

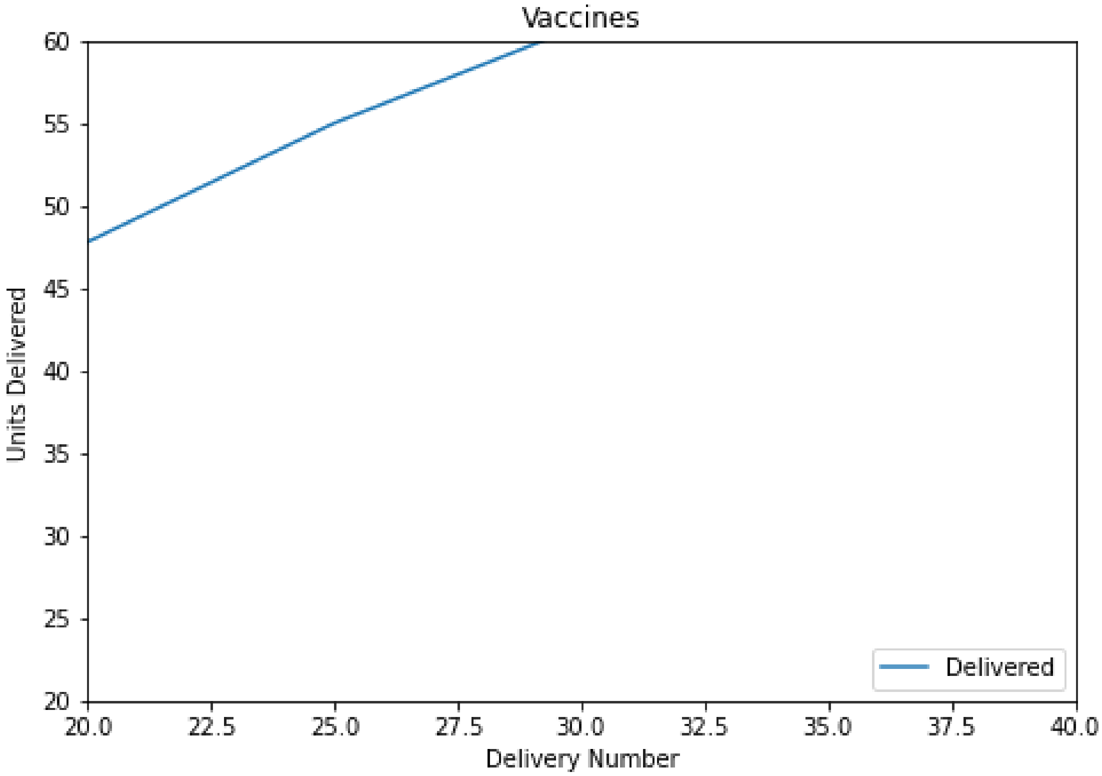
```
Text(0, 0.5, 'Units Delivered')
```



[9]:
```
# Set the legend

fig = plt.figure()
ax = fig.add_axes([0,0,1,1])
ax.plot(delivery_num,delivered, label = 'Delivered')

ax.set_title("Vaccines")
ax.set_xlabel('Delivery Number')
ax.set_ylabel('Units Delivered')

ax.legend(loc = 'lower right') # legend placed at lower right
plt.show() # This removes that little piece of output above the chart
```

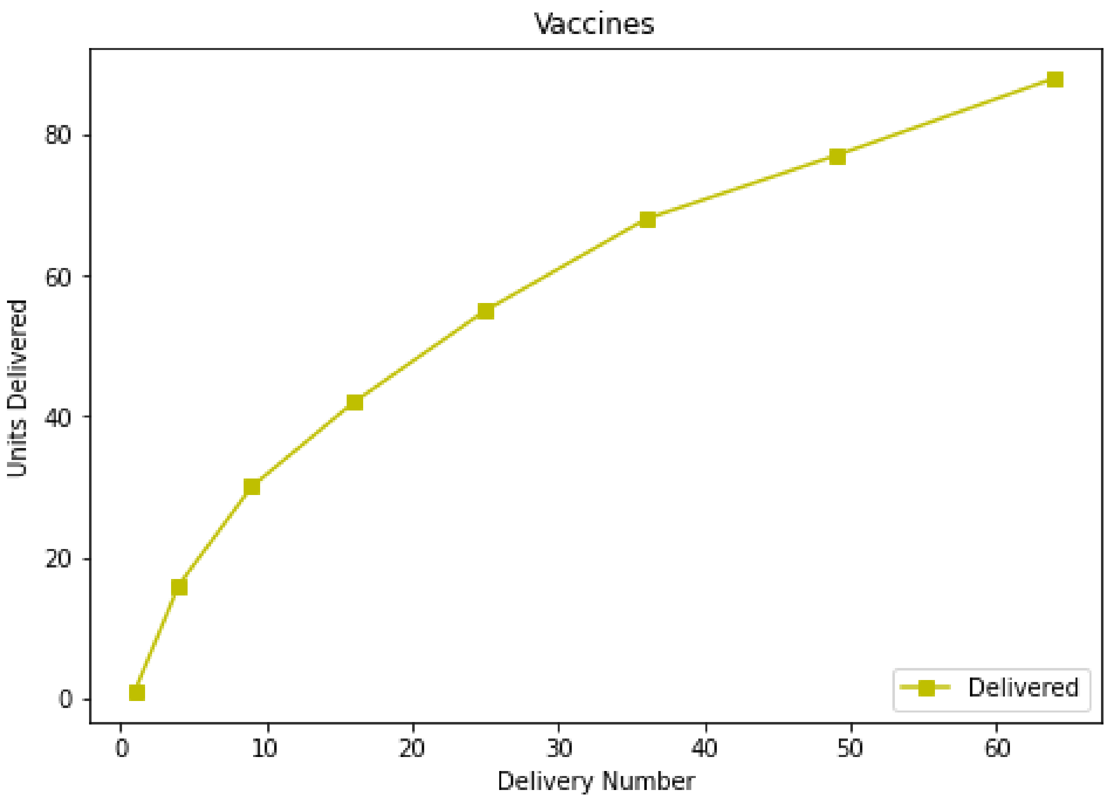


```
[10]: # Set axis limits
      fig = plt.figure()
      ax = fig.add_axes([0,0,1,1])
      ax.plot(delivery_num,delivered, label = 'Delivered')

      ax.set_title("Vaccines")
      ax.set_xlabel('Delivery Number')
      ax.set_ylabel('Units Delivered')

      ax.legend(loc = 'lower right')

      ax.set_xlim(20,40)
      ax.set_ylim(20,60)
      plt.show()
```



[11]:

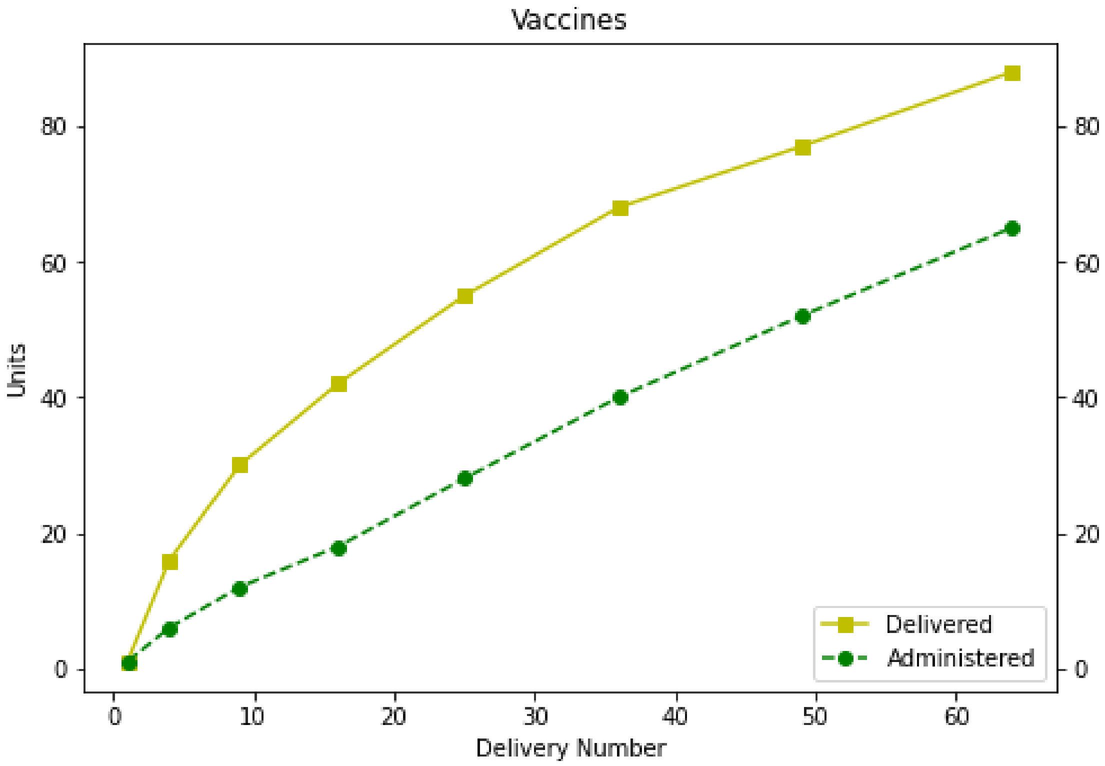
```
# Color and marks
fig = plt.figure()
ax = fig.add_axes([0,0,1,1])
ax.plot(delivery_num,delivered,'ys-', label = 'Delivered')
# ax.plot(delivery_num,delivered, label = 'Delivered','ys-') This fails.

ax.set_title("Vaccines")
ax.set_xlabel('Delivery Number')
ax.set_ylabel('Units Delivered')

ax.legend(loc = 'lower right')

plt.show()
```



### Add a second plot to the axes

[12]:

```
# Add a second line (a second plot)
fig = plt.figure()
ax = fig.add_axes([0,0,1,1])
ax.plot(delivery_num,delivered,'ys-', label = 'Delivered')
ax.plot(delivery_num,administered,'go--', label = 'Administered') # This is the␣
 ↪2nd line
ax.set_title("Vaccines")
ax.set_xlabel('Delivery Number')
ax.set_ylabel('Units')                    # Y label is changed
ax.secondary_yaxis('right')

ax.legend(loc = 'lower right')

plt.show()
```



### 4.0.4 Customize the figure

[13]:

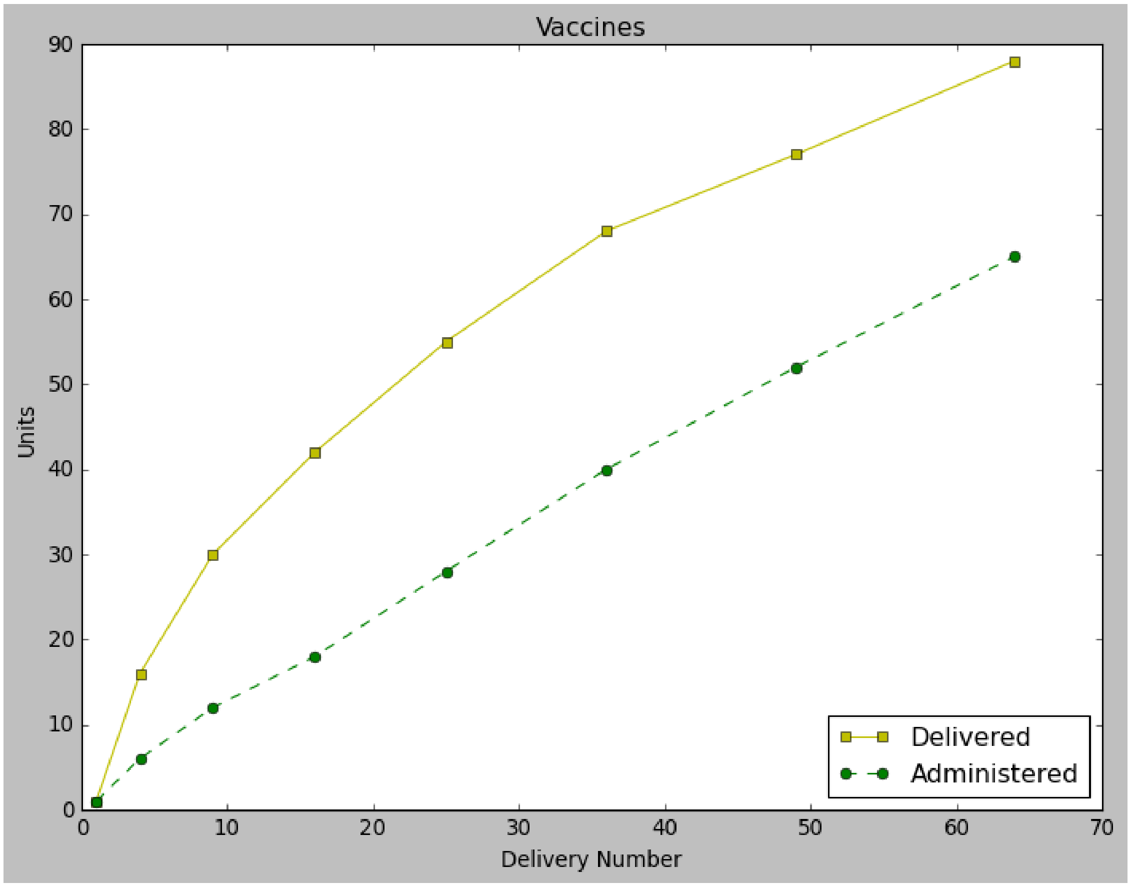
```
# Plot styles
plt.style.available
```

[13]:

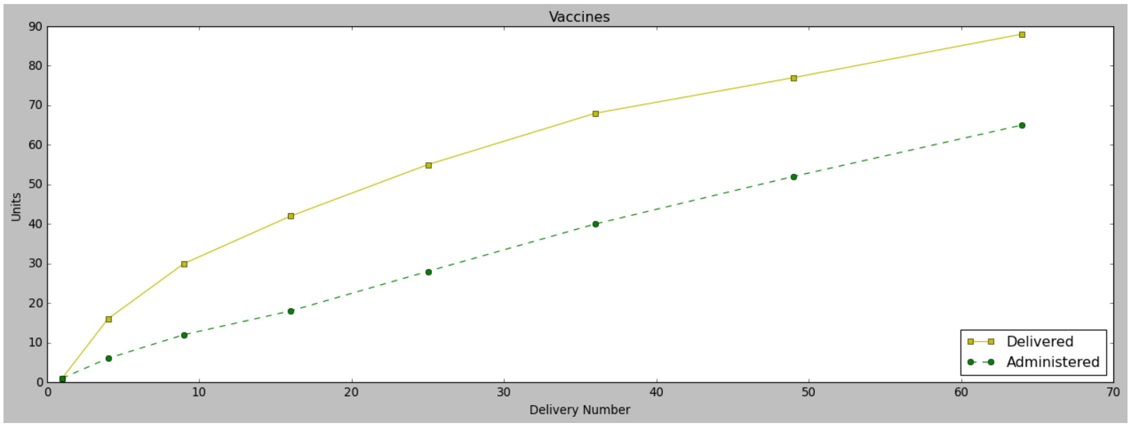
```
['Solarize_Light2',
 '_classic_test_patch',
 'bmh',
 'classic',
 'dark_background',
 'fast',
 'fivethirtyeight',
 'ggplot',
 'grayscale',
 'seaborn',
 'seaborn-bright',
 'seaborn-colorblind',
 'seaborn-dark',
 'seaborn-dark-palette',
 'seaborn-darkgrid',
 'seaborn-deep',
 'seaborn-muted',
 'seaborn-notebook',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

[14]:

```
# Plot styles
plt.style.use('classic')
fig = plt.figure()
ax = fig.add_axes([0,0,1,1])
ax.plot(delivery_num,delivered,'ys-', label = 'Delivered')
ax.plot(delivery_num,administered,'go--', label = 'Administered')
ax.set_title("Vaccines")
ax.set_xlabel('Delivery Number')
ax.set_ylabel('Units')

ax.legend(loc = 'lower right')

plt.show()
```



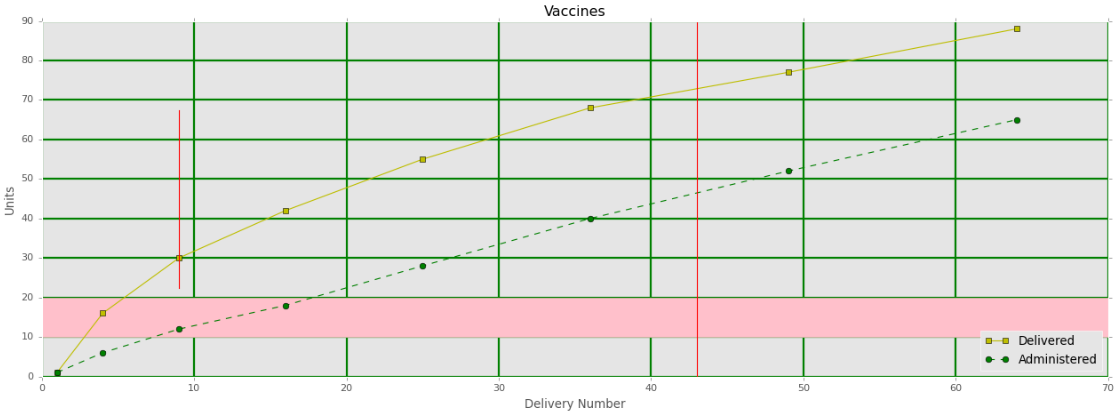
[15]:
```
# Figure size and color
fig = plt.figure( figsize = (15,5))
#fig = plt.figure( figsize = (15,5), facecolor = 'red')
ax = fig.add_axes([0,0,1,1])
ax.plot(delivery_num,delivered,'ys-', label = 'Delivered')
ax.plot(delivery_num,administered,'go--', label = 'Administered')
ax.set_title("Vaccines")
ax.set_xlabel('Delivery Number')
ax.set_ylabel('Units')
ax.legend(loc = 'lower right')
plt.show()
```



### 4.0.5 A little more customization

```
[16]: # Reference lines and reference bars
      plt.style.use('ggplot')
      fig = plt.figure( figsize = (15,5))
      ax = fig.add_axes([0,0,1,1])
      ax.plot(delivery_num,delivered,'ys-', label = 'Delivered')
      ax.plot(delivery_num,administered,'go--', label = 'Administered')
      ax.set_title("Vaccines")
      ax.set_xlabel('Delivery Number')
      ax.set_ylabel('Units')
      ax.legend(loc = 'lower right')
      ax.grid(color = 'green',linestyle='-', linewidth=2)
      ax.axvline(x=43, color= 'r')
      ax.axvline(x=9, ymin=0.25, ymax=0.75, color = 'r')
      ax.axhspan(10,20, color = 'pink')
```

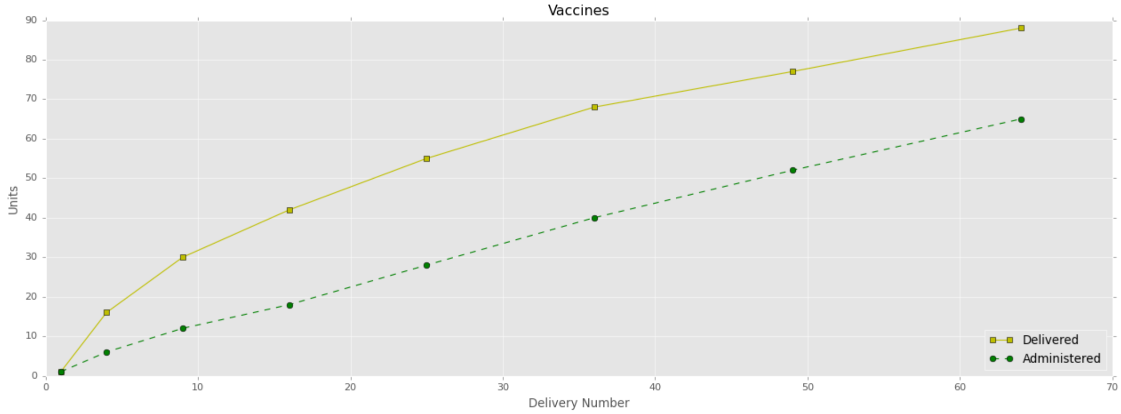
```
[16]: <matplotlib.patches.Polygon at 0x7fc3327429d0>
```



### 4.0.6 Save to a file

[17]:

```
fig = plt.figure( figsize = (15,5))
ax = fig.add_axes([0,0,1,1])
ax.plot(delivery_num,delivered,'ys-', label = 'Delivered')
ax.plot(delivery_num,administered,'go--', label = 'Administered')
ax.set_title("Vaccines")
ax.set_xlabel('Delivery Number')
ax.set_ylabel('Units')
ax.legend(loc = 'lower right')
plt.savefig('Deliveries', transparent=True)
plt.savefig('Deliveries') # default is .png
plt.savefig('Deliveries.jpg')
plt.savefig('Deliveries.svg')
plt.savefig('Deliveries.pdf')
plt.show()
```



Go here for the life cycle of a plot

## 4.1 Exercise 1 - 10 minutes

Links: - Horizontal bar chart: https://matplotlib.org/stable/gallery/lines_bars_and_markers/barh.html#sphx-glr-gallery-lines-bars-and-markers-barh-py - Figures: https://matplotlib.org/stable/api/figure_api.html

To do: - Create a horizontal bar chart using the data below. - Make the chart width approx. 3x the height. - Make the bar color green. - Only display the x-axis from 80 to 140 - Read about annotating - https://matplotlib.org/stable/tutorials/text/annotations.html#sphx-glr-tutorials-text-annotations-py

[18]:

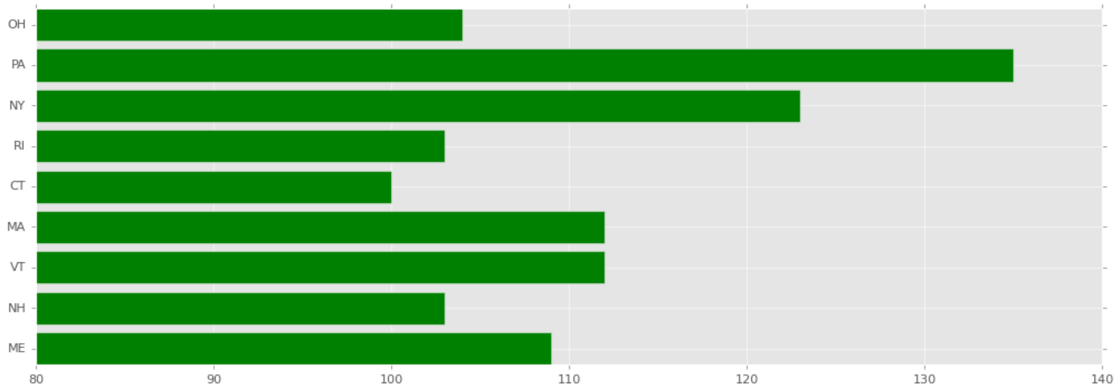
```
# Exercise data
data = {'ME': 109,
        'NH': 103,
        'VT': 112,
        'MA': 112,
        'CT': 100,
        'RI': 103,
        'NY': 137,
        'NY': 123,
        'PA': 135,
        'OH': 104}
food_illness = list(data.values())
state = list(data.keys())
state_mean = np.mean(food_illness)
```

[19]:

```
fig = plt.figure(figsize = (12,4))
ax = fig.add_axes([0,0,1,1])
ax.barh(state,food_illness, color = 'g')
ax.set_xlim(80,140)
```

[19]:

```
(80.0, 140.0)
```



## 5 A Survey of Plot Types

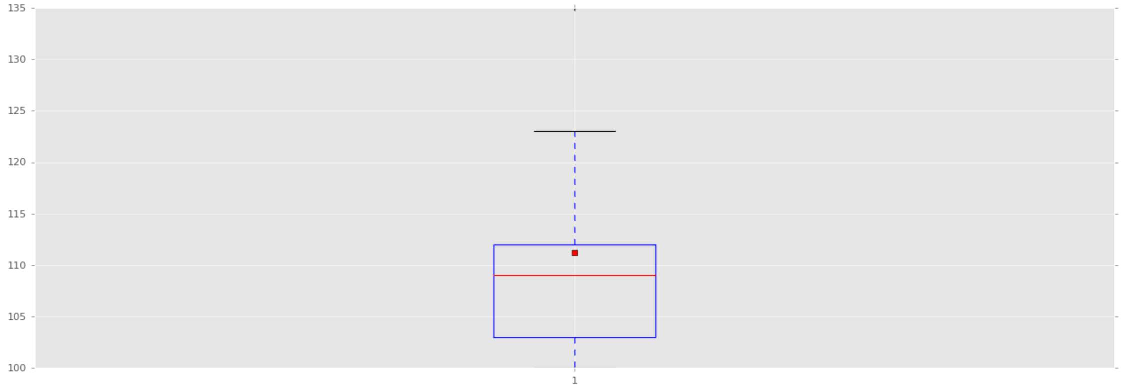
[20]:

```
# 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)]
```

[21]:

```
# OO Inteface
fig = plt.figure( figsize = (15,5))
ax = fig.add_axes([0,0,1,1])
ax.boxplot(food_illness, showmeans = True)
```

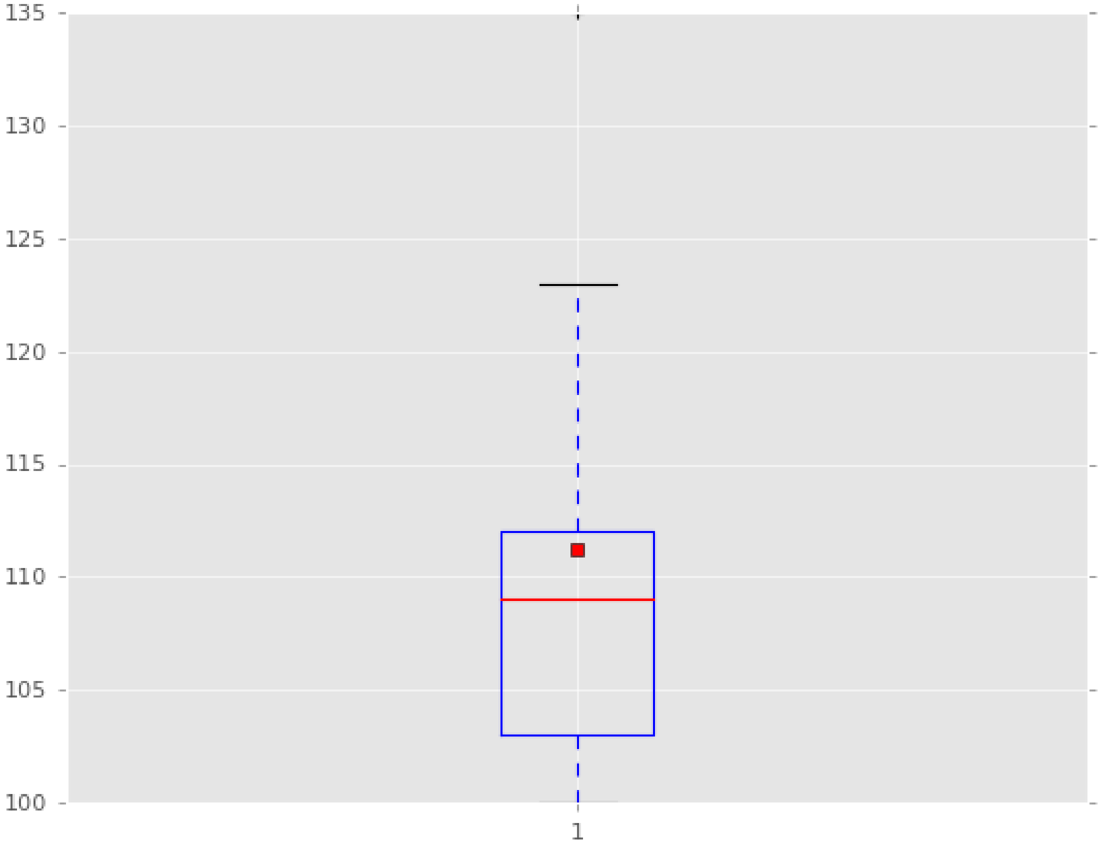
```
plt.show()
```



## 5.1 Using the pyplot interface

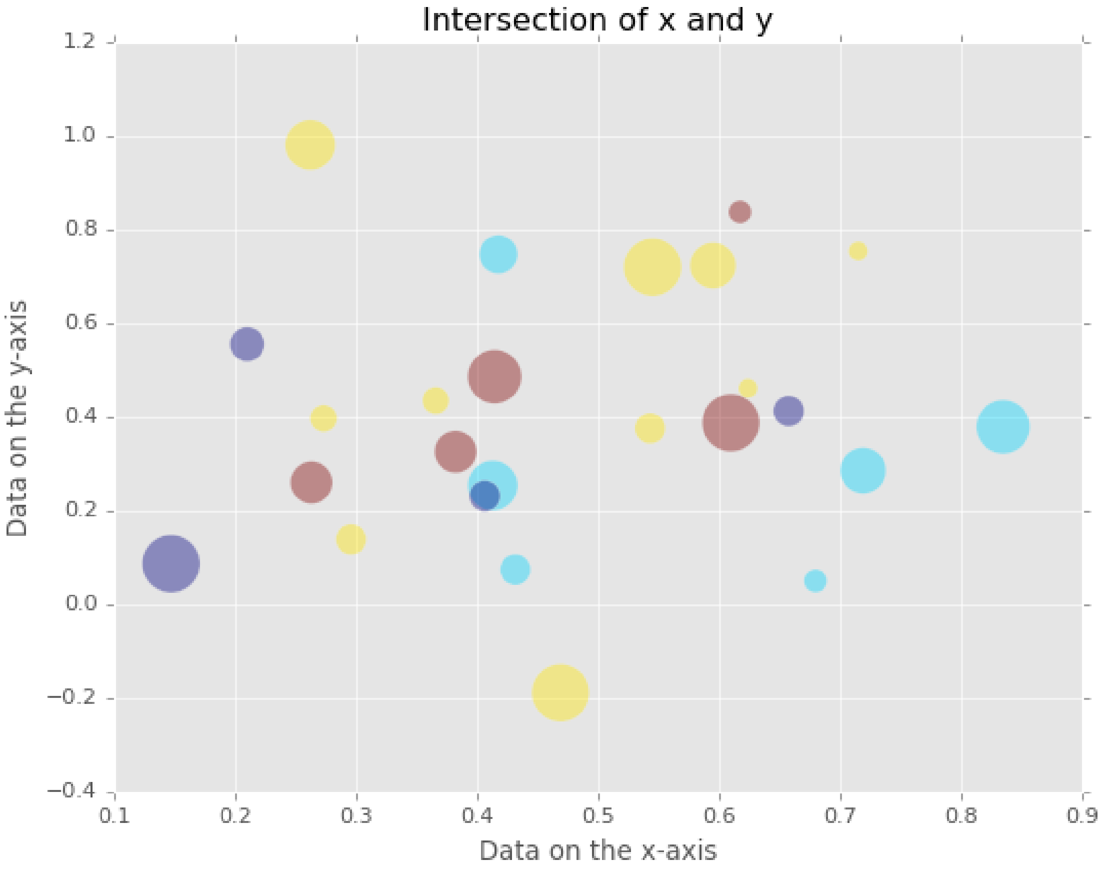
### Boxplot

[22]:
```
plt.boxplot(food_illness, showmeans=True)
plt.show()
```



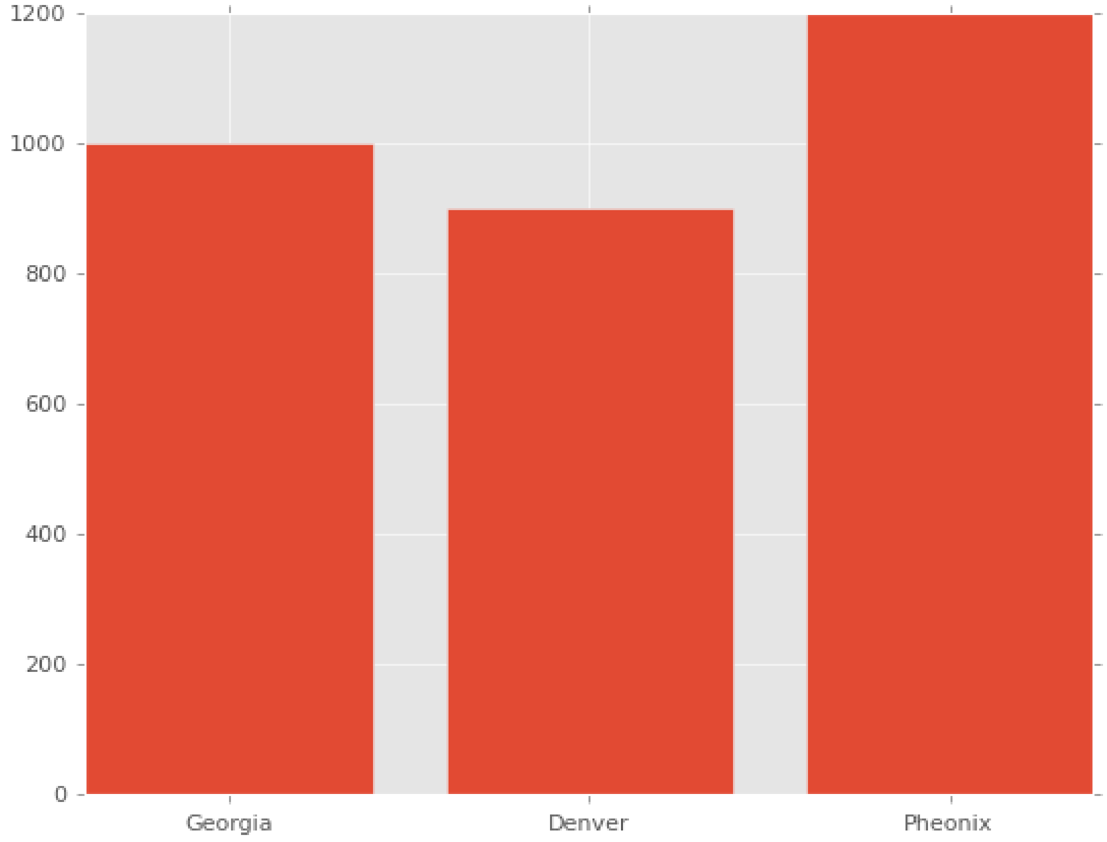
### Scatterplot

[23]:
```
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()
```



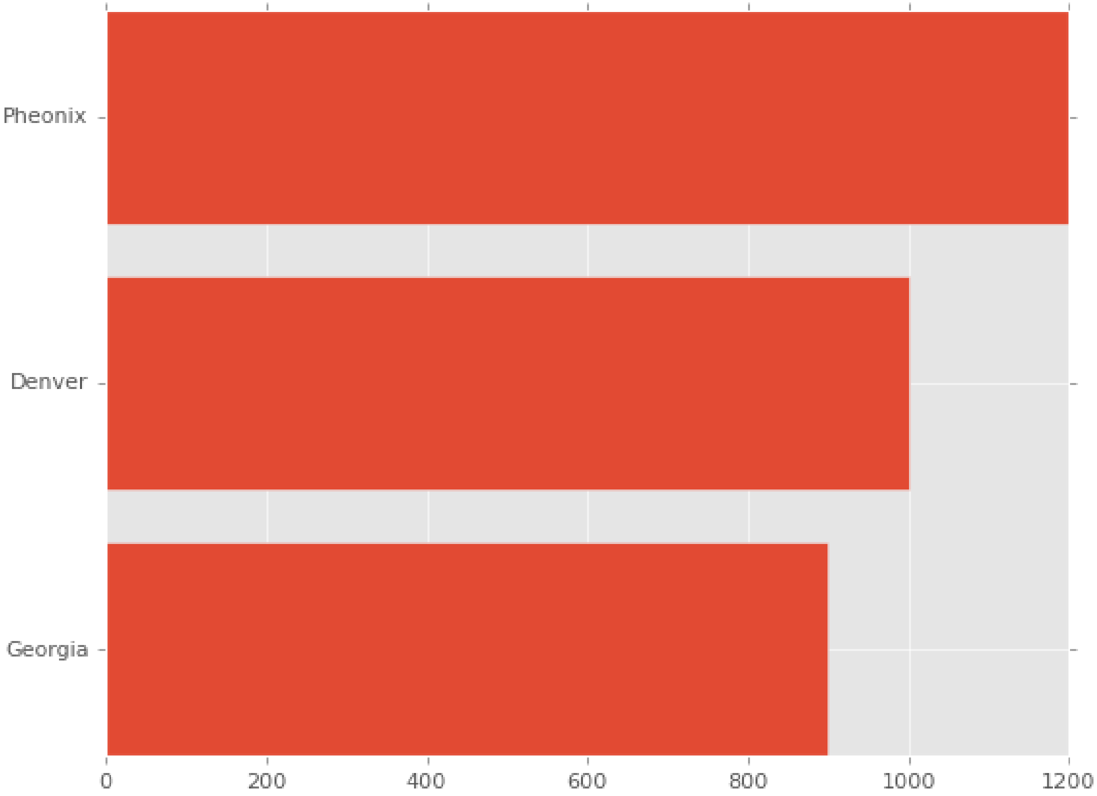
### Bar

[24]:
```
names = ['Georgia', 'Denver', 'Pheonix']
values = [1000, 900, 1200]

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



### Bar Horizontal

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

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



### 5.2 Exercise 2 - 5 minutes

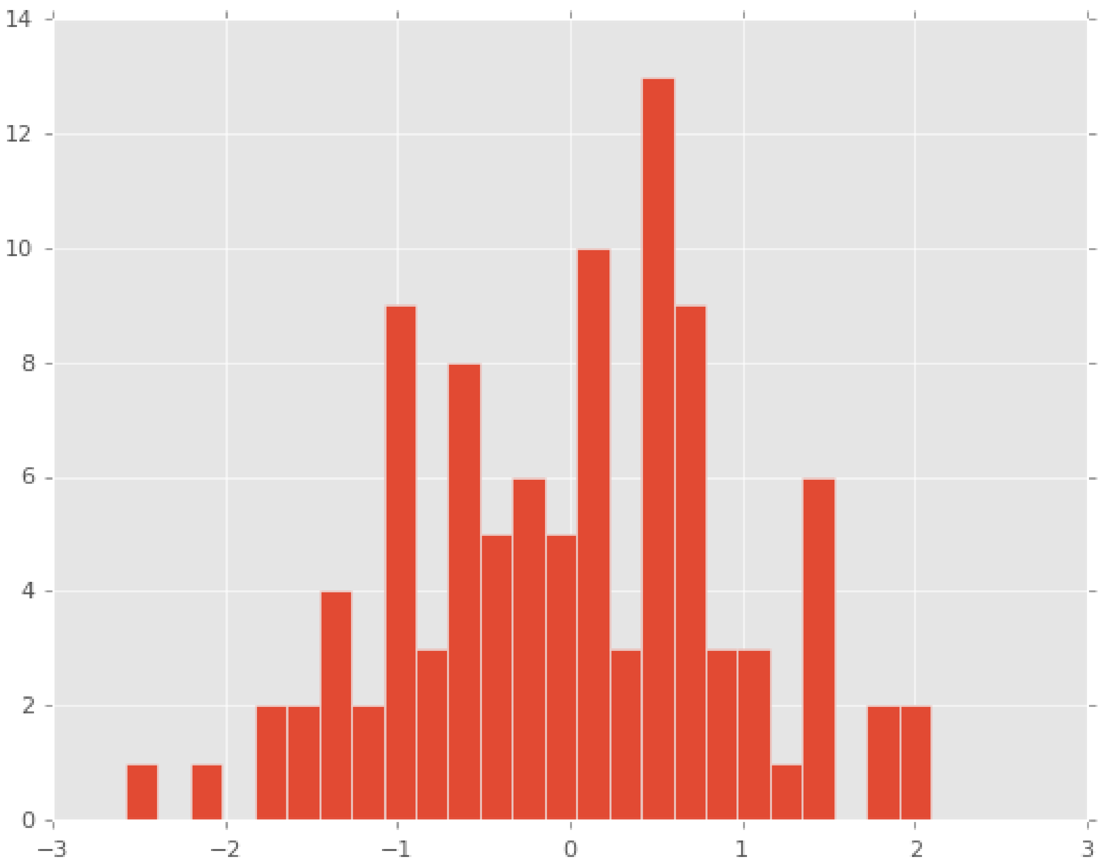
- Using the data below (variable is called data), create a histogram
- Can you change the number of bins?

[26]:

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

[27]:

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



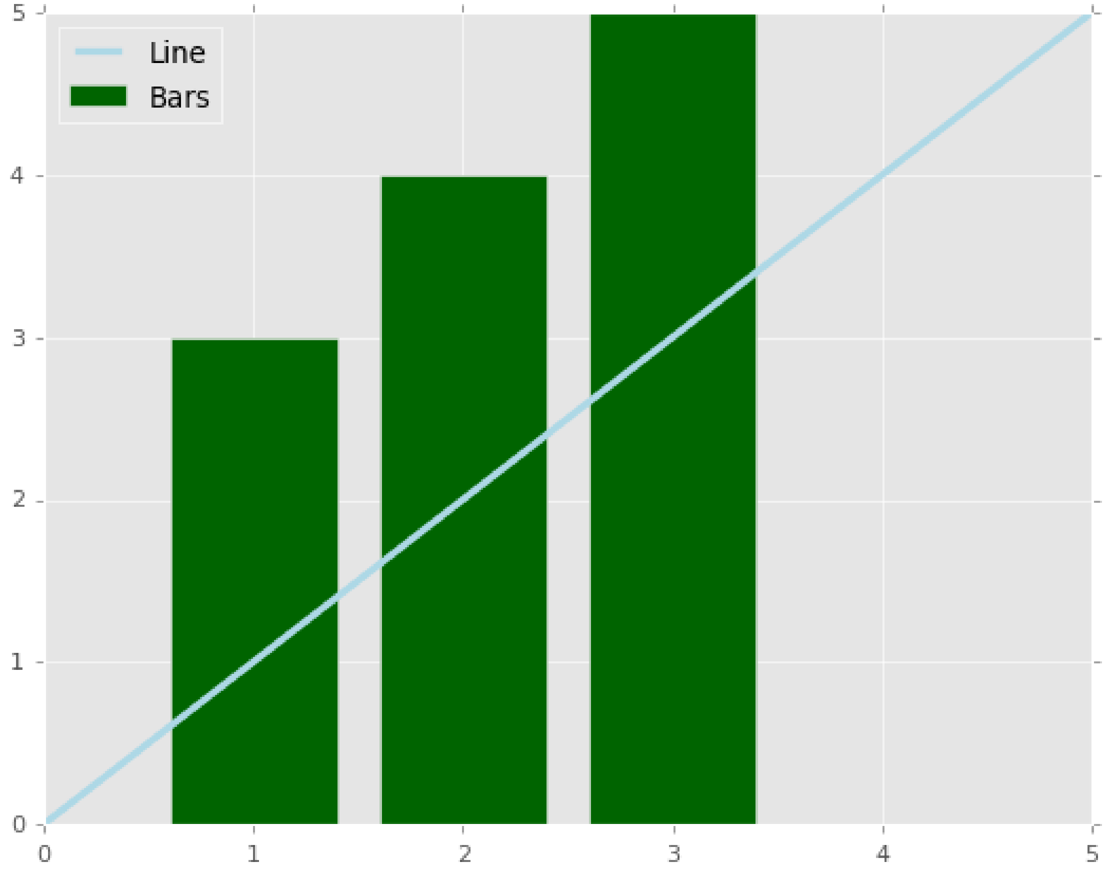
### 5.3 Exercise 3 - 10 minutes

- Recreate the chart below.
- The line chart shows c and d.
- The bar chart shows a and b.
- Use the documentation to place the legend.

[28]:
```
c = np.linspace(0, 5, 5)
d = np.linspace(0, 5, 5)
a = [1,2,3]
b = [3,4,5]
```

[29]:
```
plt.plot(c,d, color='lightblue', linewidth=3, label = 'Line')
plt.bar(a,b, color='darkgreen', label = 'Bars')
plt.legend(loc='upper left')
```

[29]:
```
<matplotlib.legend.Legend at 0x7fc3319eea00>
```



# 6 Multiple Axes in a Figure

## 6.1 add_axes() vs. add_subplot()

The difference between fig.add_axes() and fig.add_subplot() is the mechanisms used. For add_axes() a list is passd in specifiying the the position of the axes (left, botton,width, height). This means that the axes object is positioned in **absolute coordinates**.

The add_subplot() function does not permit explicit positioning. Rather, the axes is positioned according to a subplot grid.

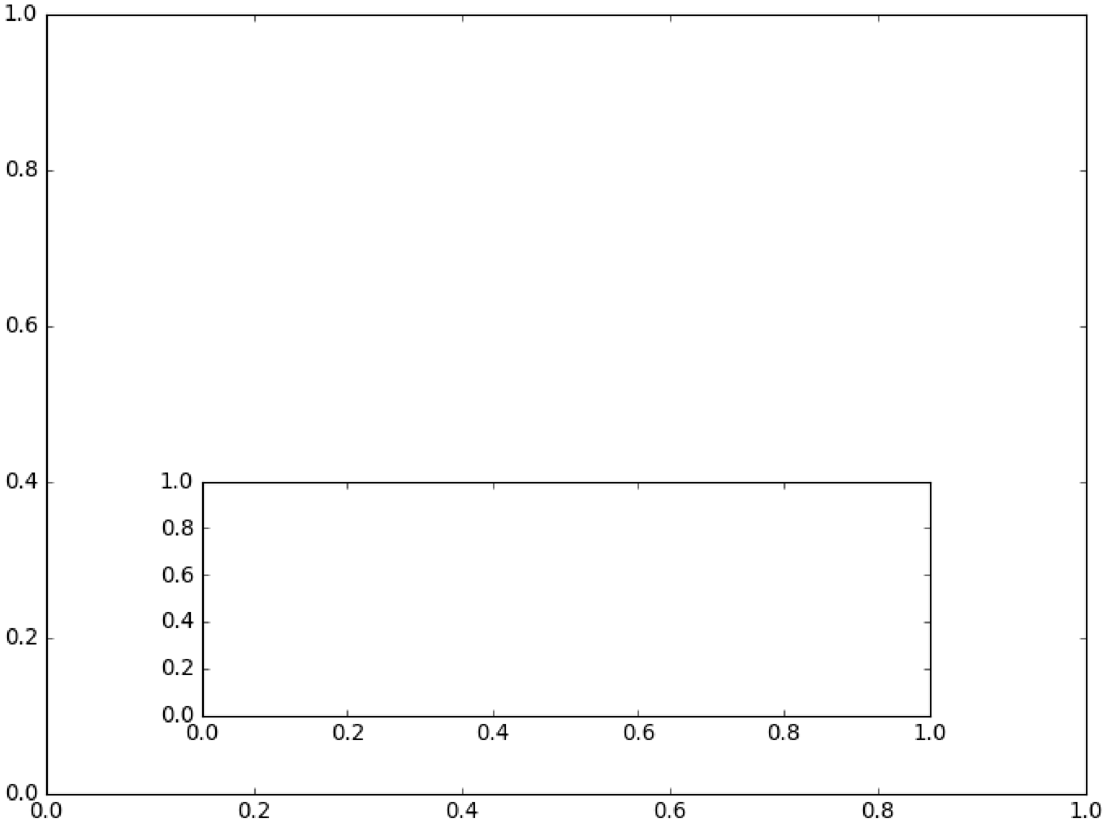
In most cases, add_subplot() is used. In cases where positioning matters, add_axes() is used. fig =

https://matplotlib.org/stable/tutorials/intermediate/arranging_axes.html#sphx-glr-tutorials-intermediate-arranging-axes-py

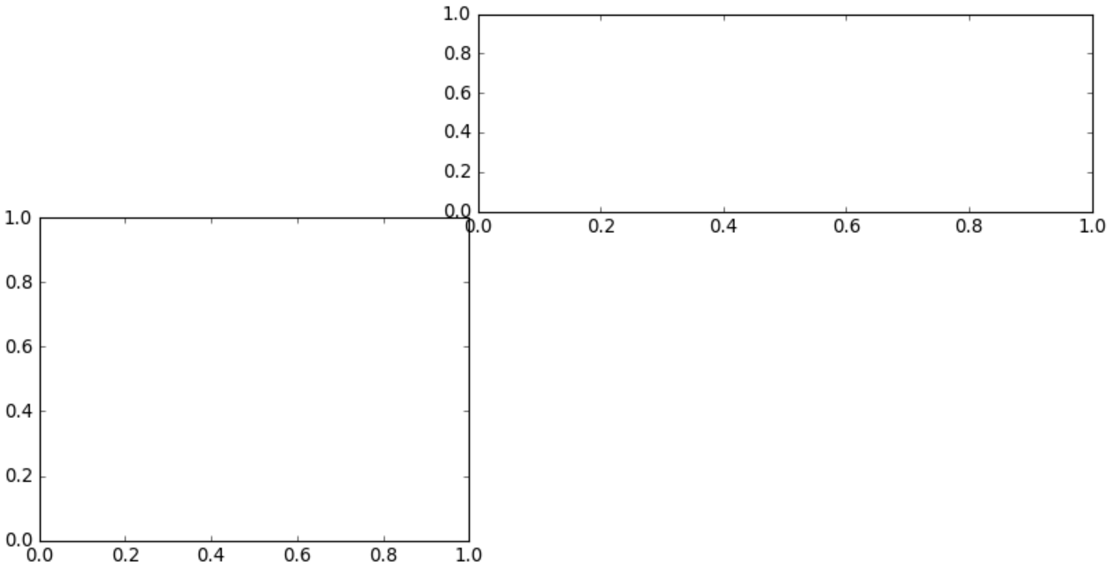
## 6.2 Using .add_axes( )

[30]:
```
plt.style.use('classic')
```

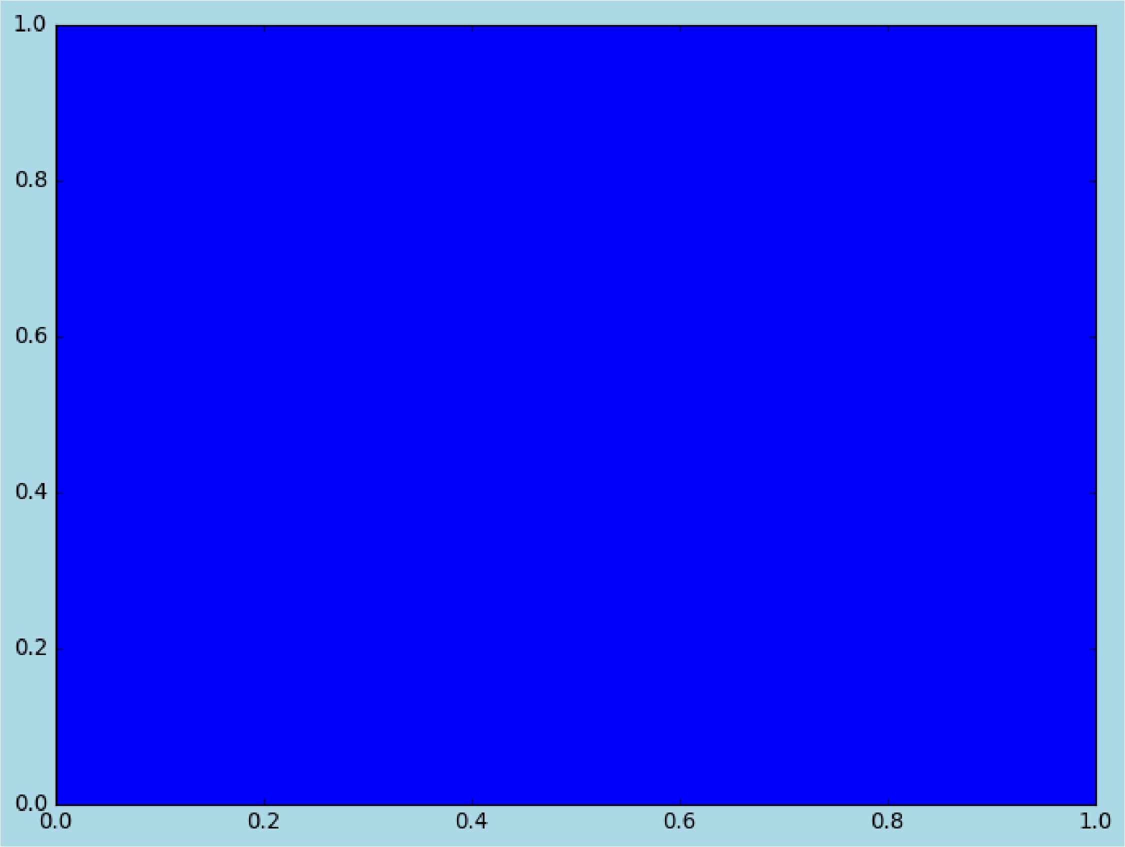
[31]:
```
fig = plt.figure(facecolor='white')
ax1 = fig.add_axes([0,0,1,1])  # left, bottom, width, height
ax2 = fig.add_axes([0.15, 0.1, 0.7, 0.3])
```



[32]:
```
fig = plt.figure(facecolor='white')
ax1 = fig.add_axes([0,0,0.49,0.49])  # left, bottom, width, height
ax2 = fig.add_axes([0.5, 0.5, 0.7, 0.3])
```



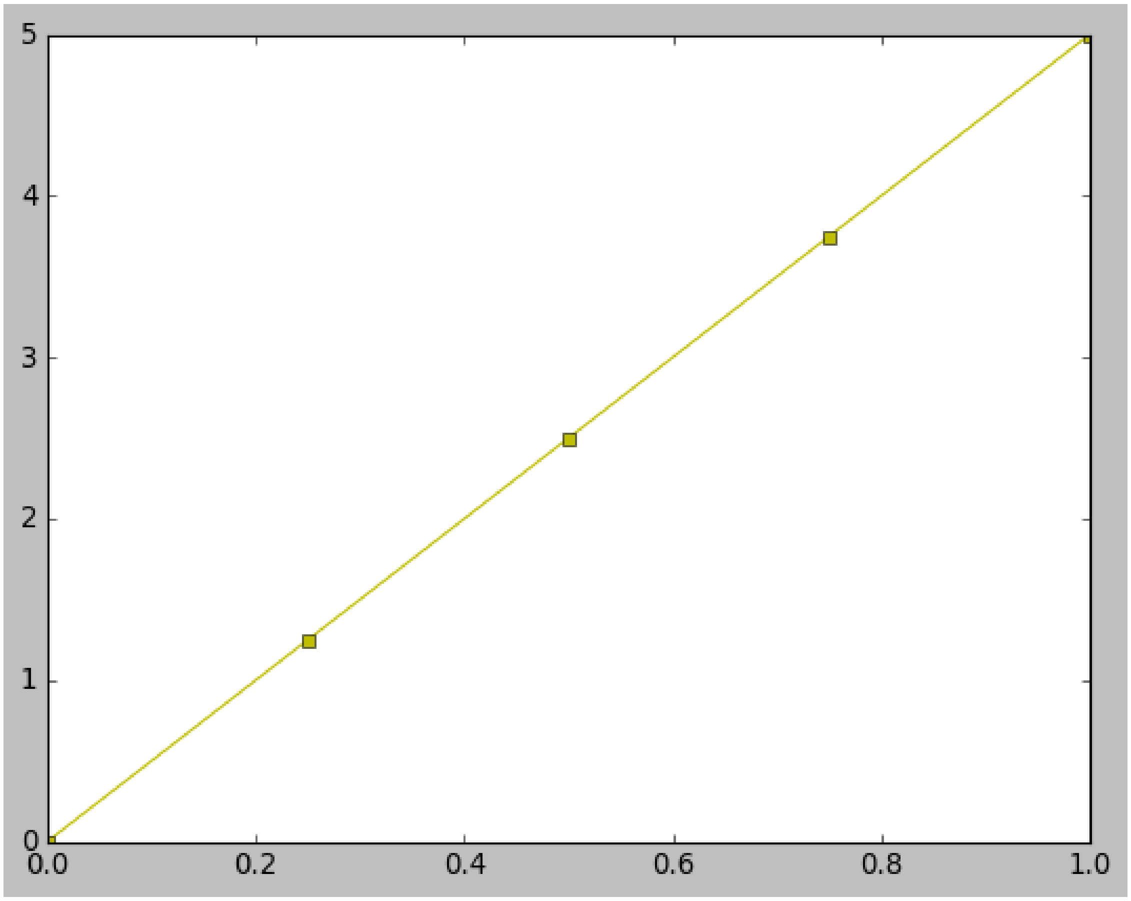
[33]:
```
# Where is the axes on the figure?

fig = plt.figure()
fig.set_facecolor('lightblue')

ax = fig.add_axes([0,0,1,1])      # left, bottom, width, height
ax.set_facecolor('blue')

# fig = plt.figure(facecolor = 'lightblue')  # The pyplot syntax can still be␣
 ↪used
```



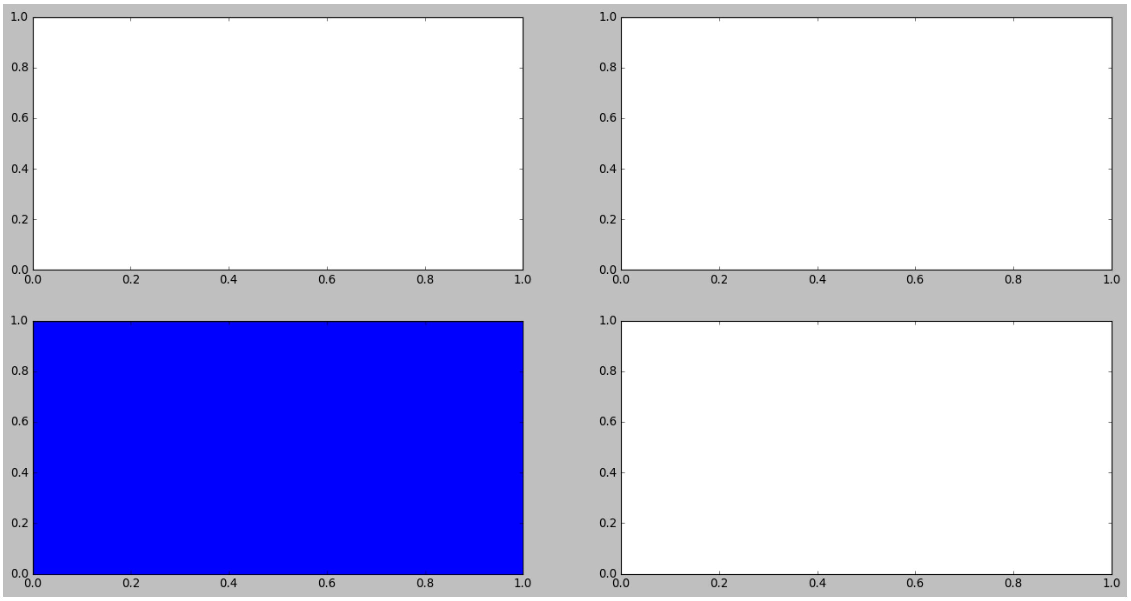
### 6.3 Using add_subplot( )

[34]:
```
x = np.linspace(0, 1, 5)
y = np.linspace(0, 5, 5)
```

[35]:
```
fig = plt.figure()
ax = fig.add_subplot(111)
ax = ax.plot(x,y,'ys-')
```

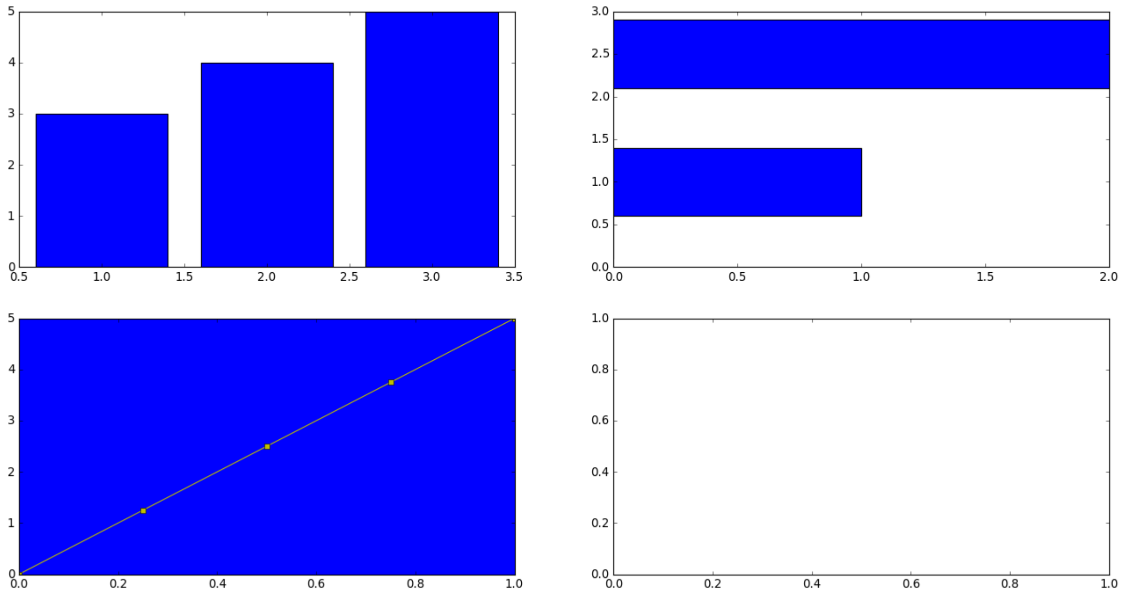


[36]:
```
# Initialize the plot
fig = plt.figure(figsize=(20,10))
ax1 = fig.add_subplot(2,2,1)
ax2 = fig.add_subplot(222)
ax3 = fig.add_subplot(223)
ax4 = fig.add_subplot(224)

ax3.set_facecolor('blue')
# Show the plot
# plt.show()
```



[37]:

```
# Initialize the plot
fig = plt.figure(figsize=(20,10),facecolor='white')
ax1 = fig.add_subplot(2,2,1)
ax2 = fig.add_subplot(222)
ax3 = fig.add_subplot(223)
ax4 = fig.add_subplot(224)

ax3.set_facecolor('blue')

# Plot the data
ax1.bar([1,2,3],[3,4,5])
ax2.barh([0.5,1,2.5],[0,1,2])
ax3.plot(x,y,'ys-')

plt.show()
```



### 6.4 Exercise 4 - 10 minutes

Using the data below, recreate the figure shown using subplots. Add plot titles.

- names = ‘Georgia’, ‘Denver’, ‘Pheonix’
- values = 1000, 900, 1200
- dts = ‘1/1/2021’, ‘1/2/2021’,‘1/3/2021’

[38]:

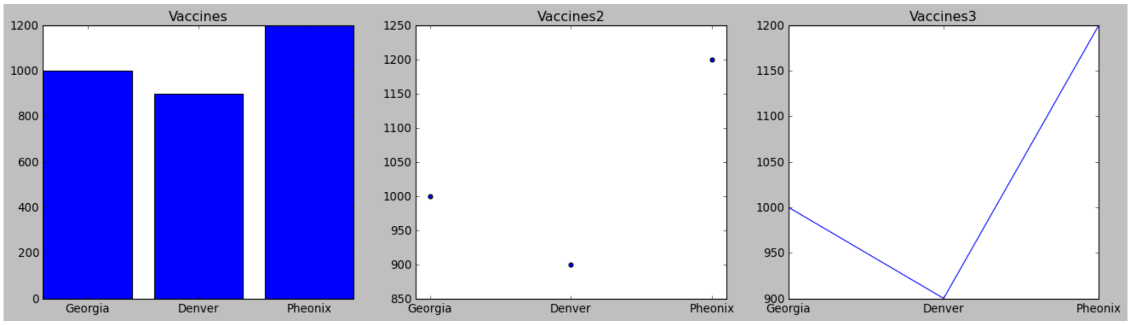
```
# Exercise 4 Solution Here.

import numpy as np
import matplotlib.pyplot as plt

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

[39]:

```
# Exercise 4 Solution Here.

# Initialize the plot
fig = plt.figure(figsize=(20,5))
ax1 = fig.add_subplot(1,3,1)
ax2 = fig.add_subplot(132)
ax3 = fig.add_subplot(133)



# Plot the data
ax1.bar(names,values, label = 'bar')
```

```
ax2.scatter(names,values,label = 'scatter')
ax3.plot(names,values,label = 'line')

ax1.set_title("Vaccines")
ax2.set_title("Vaccines2")
ax3.set_title("Vaccines3")

# Show the plot
# plt.show()
```

[39]:
```
Text(0.5, 1.0, 'Vaccines3')
```



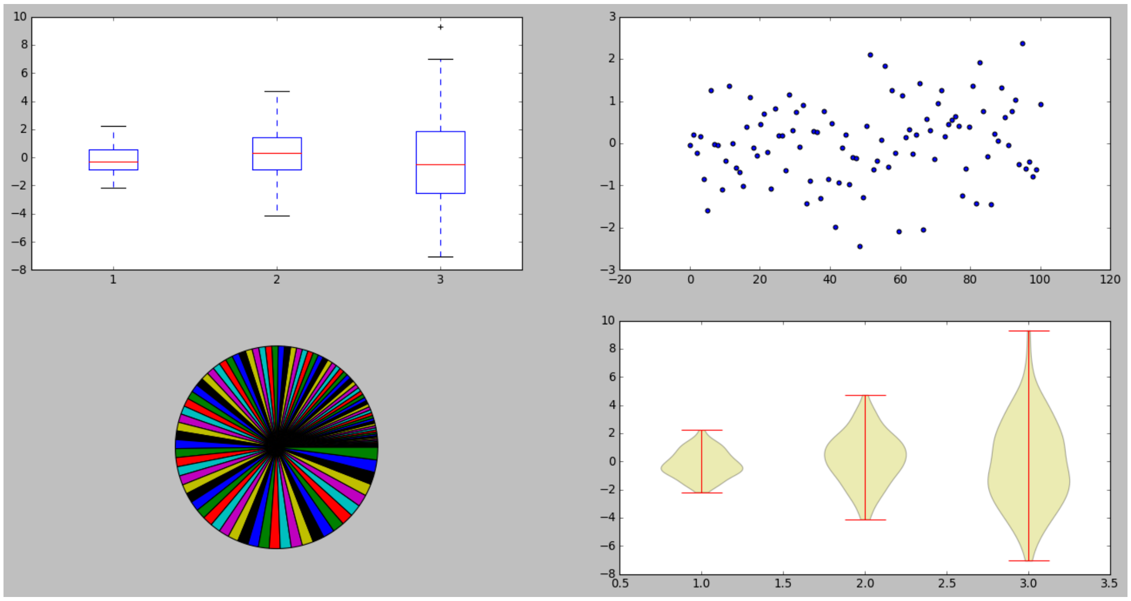
## 6.5 Exercise 5 - 15 minutes

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

[40]:
```
# Data for Exercise 5
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)
data = np.random.randn(100)
```

[41]:
```
# Initialize the plot
fig = plt.figure(figsize=(20,10))
ax1 = fig.add_subplot(221)
ax2 = fig.add_subplot(222)
ax3 = fig.add_subplot(223)
ax4 = fig.add_subplot(224)

# Plot the data
ax1.boxplot(y)
ax2.scatter(x,data)
ax3.pie(x)
ax4.violinplot(y)
```

```
# Show the plot
plt.show()
```


10
8
6
4
2
0
−2
−4
−6
−8
1
2
3
3
2
1
0
−1
−2
−3
−20
0
20
40
60
80
100
120
10
8
6
4
2
0
−2
−4
−6
−8
0.5
1.0
1.5
2.0
2.5
3.0
3.5