

# **DSP505: Programming Lab for Data Science and Artificial Intelligence**

## **TPL616: Advanced Programming for DSAI**

### **(Matplotlib Tutorial)**



**Vishwesh Jatala**

Assistant Professor

Department of CSE

Indian Institute of Technology Bhilai

[vishwesh@iitbhilai.ac.in](mailto:vishwesh@iitbhilai.ac.in)

# Acknowledgement

---

Today's lecture are borrowed from:

<http://jake-feldman.squarespace.com/data-science-python-osc-m400c>

# Plotting in Pandas

- We will start by seeing how to make basic plots in pandas from data stored in a pandas dataframe.
- We will then see how to add the following features to our graphs:
  - Change titles/axis titles
  - Annotate the graph
  - Change the window of the graph
  - Customizing axis labels
  - Create multiple plots
- In a future lecture, we will then investigate more advanced plotting tools.

# Sleep Data

We have the following data on how hours of sleep is related to GPA.

```
df_sleep = pd.read_csv("Data/Sleep.csv")  
df_sleep.head()
```

	Gender	Age	Year in College	Hours of Sleep	GPA
0	Female	22	4	7.0	3.80
1	Male	18	1	4.0	3.60
2	Male	19	2	9.0	3.50
3	Female	27	3	7.0	3.00
4	Female	37	3	5.0	3.61

# Basic Scatterplot

Makes plot show in Jupyter notebook

Type of plot given as string

```
%matplotlib inline  
df_sleep.plot(kind = "scatter", x = "Hours of Sleep", \n               y = "GPA")
```

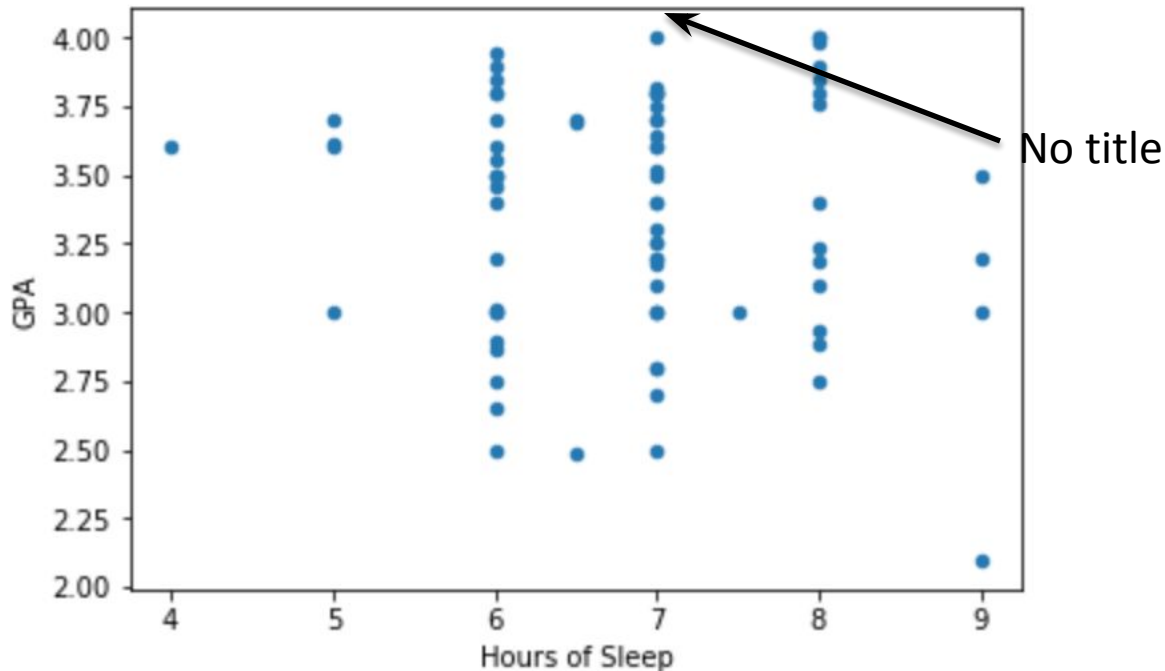
Given x,y axis

# Basic Scatterplot

```
%matplotlib inline
```

```
df_sleep.plot(kind = "scatter", x = "Hours of Sleep", \n                y = "GPA")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x10c30f080>
```



No title

Axis labels are column names

# Other Types of Plots

## pandas.DataFrame.plot

`DataFrame.plot(x=None, y=None, kind='line', ax=None, subplots=False, sharex=None, sharey=False, layout=None, figsize=None, use_index=True, title=None, grid=None, legend=True, style=None, logx=False, logy=False, loglog=False, xticks=None, yticks=None, xlim=None, ylim=None, rot=None, fontsize=None, colormap=None, table=False, yerr=None, xerr=None, secondary_y=False, sort_columns=False, **kws)` [\[source\]](#)

Make plots of DataFrame using matplotlib / pylab.

*New in version 0.17.0:* Each plot kind has a corresponding method on the `DataFrame.plot` accessor:

`df.plot(kind='line')` is equivalent to `df.plot.line()`.

**data** : *DataFrame*

**x** : *label or position, default None*

**y** : *label or position, default None*

Allows plotting of one column versus another

**kind** : *str*

- 'line' : line plot (default)
- 'bar' : vertical bar plot
- 'barh' : horizontal bar plot
- 'hist' : histogram
- 'box' : boxplot
- 'kde' : Kernel Density Estimation plot
- 'density' : same as 'kde'
- 'area' : area plot
- 'pie' : pie plot
- 'scatter' : scatter plot
- 'hexbin' : hexbin plot

# Basic Barplot

I have computed the average GPA for each. I will show you later on how to do this.

```
df_gpa_gender
```

	Gender	Avg_GPA
0	Female	3.427000
1	Male	3.285175



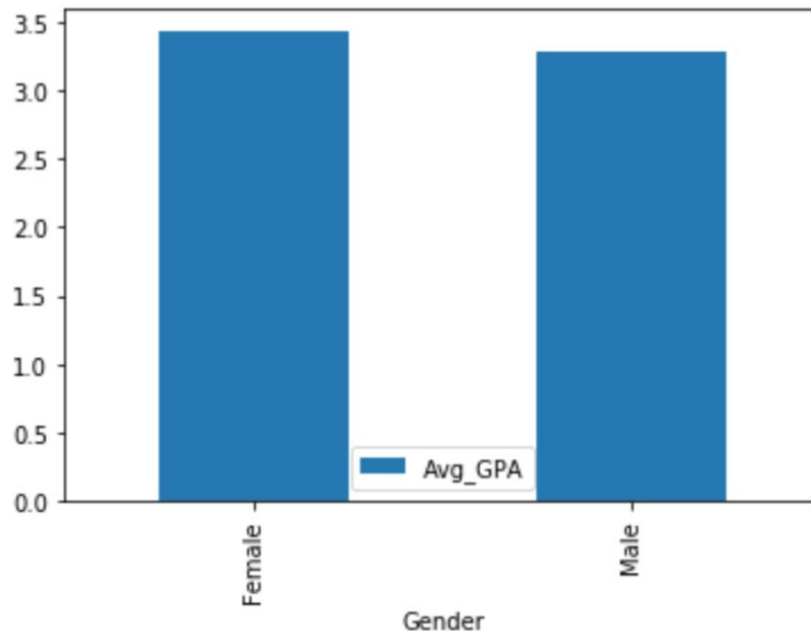
# Basic Barplot

```
df_gpa_gender
```

	Gender	Avg_GPA
0	Female	3.427000
1	Male	3.285175

```
df_gpa_gender.plot(kind="bar" , x = "Gender", y = "Avg_GPA")
```

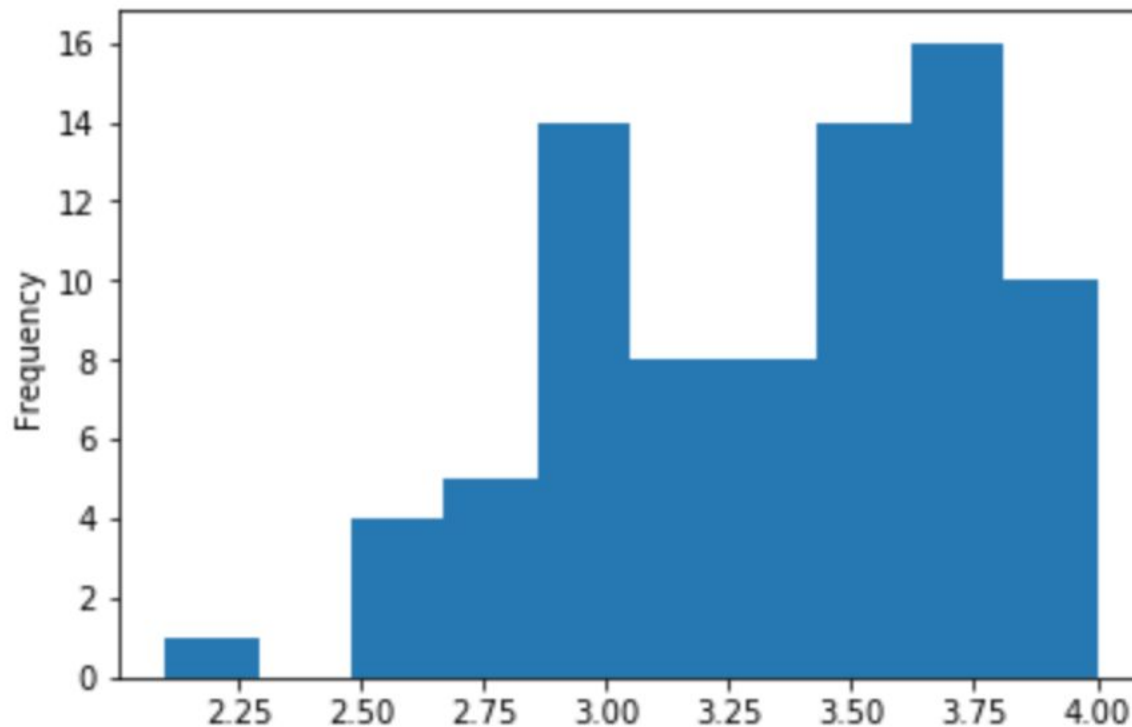
```
<matplotlib.axes._subplots.AxesSubplot at 0x11a6bdfd0>
```



# Basic Histogram

```
df_sleep.GPA.plot(kind="hist")
```


```
<matplotlib.axes._subplots.AxesSubplot at 0x11a7ed748>
```



# Basic Histogram

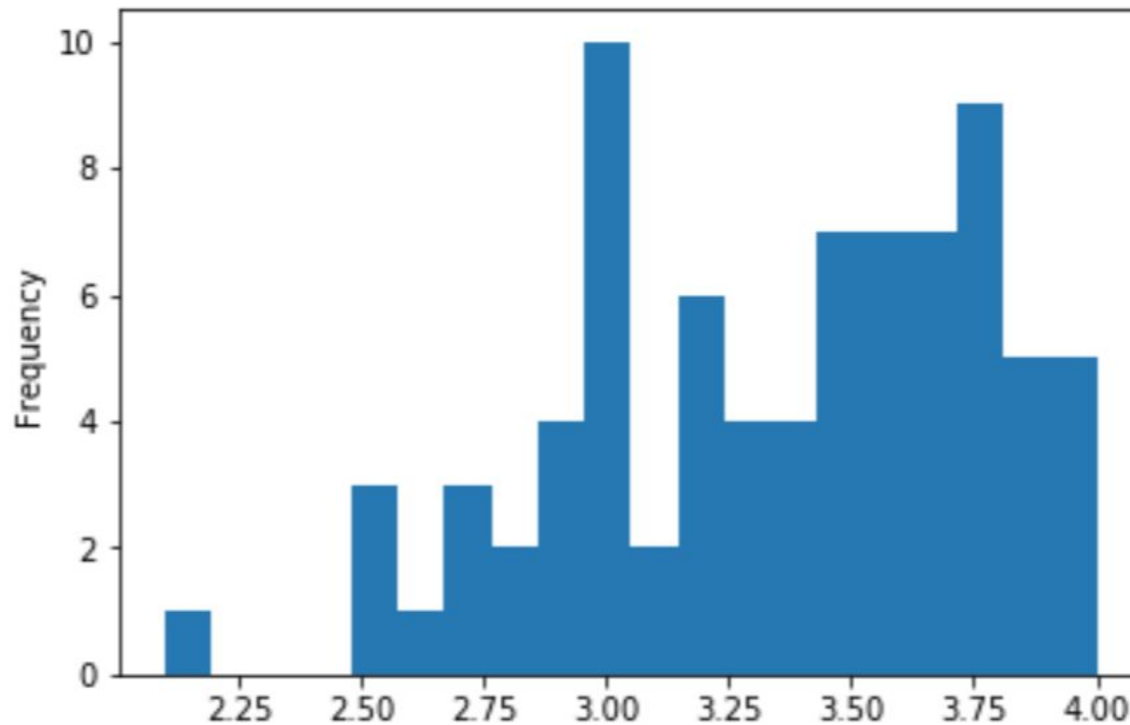
```
df_sleep.GPA.plot(kind="hist", bins = 20)
```

Number of bins to use

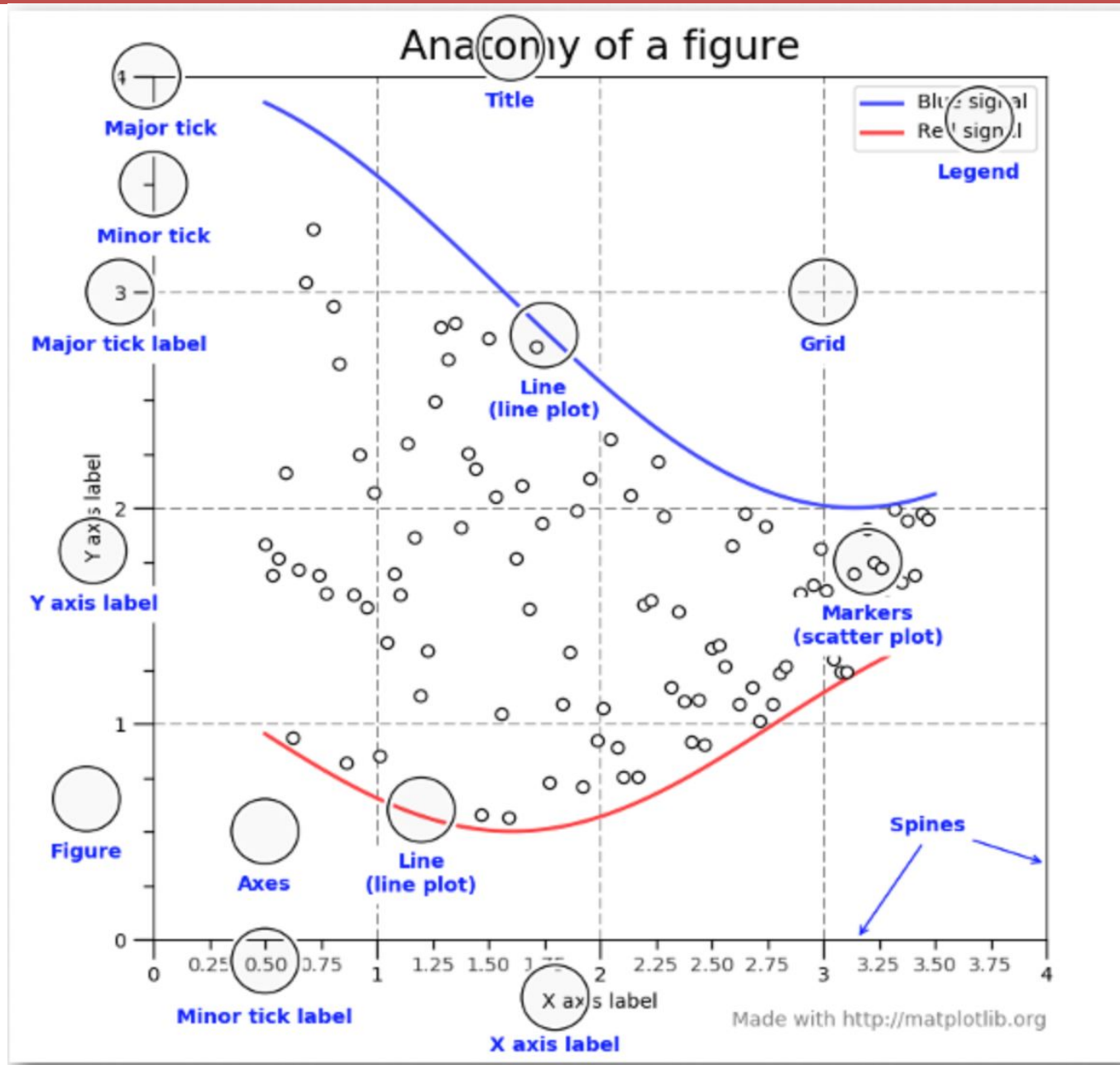


# Basic Histogram

```
df_sleep.GPA.plot(kind="hist", bins = 20)
```



# Customizing the Plots



# Sales Data

Consider the data from the top 10 customers:

```
df_top_ten
```

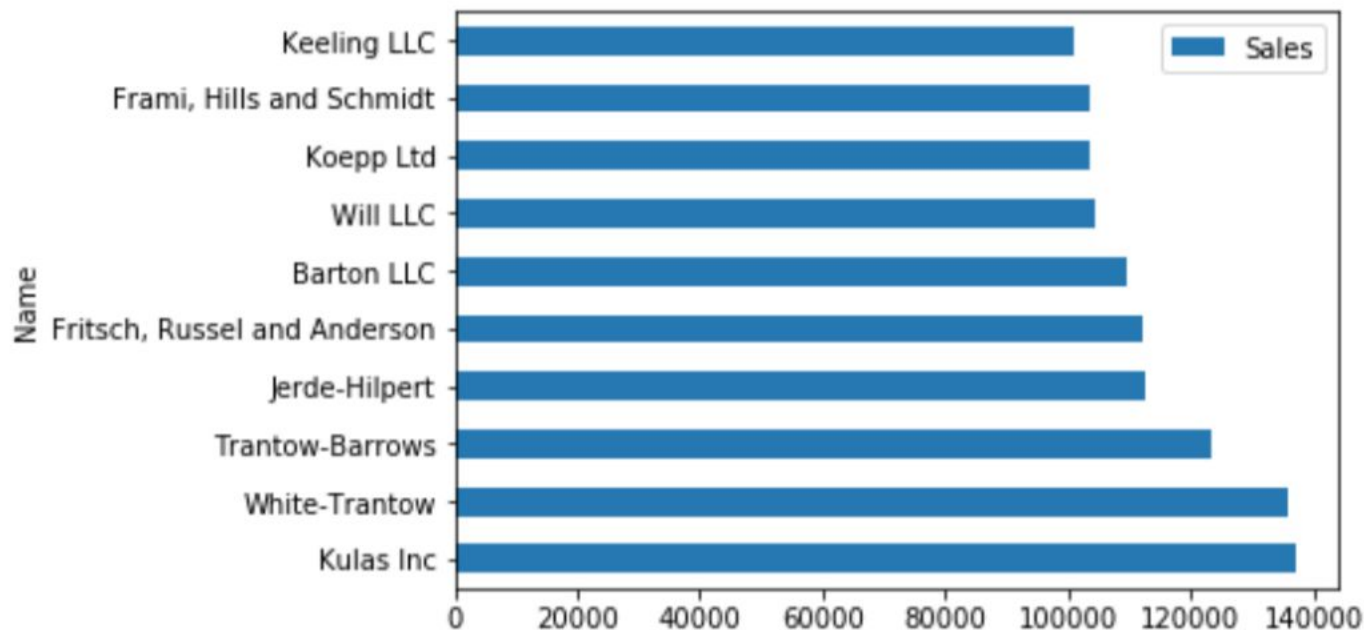
	Name	Sales	Purchases
0	Kulas Inc	137351.96	94
1	White-Trantow	135841.99	86
2	Trantow-Barrows	123381.38	94
3	Jerde-Hilpert	112591.43	89
4	Fritsch, Russel and Anderson	112214.71	81
5	Barton LLC	109438.50	82
6	Will LLC	104437.60	74
7	Koepp Ltd	103660.54	82
8	Frami, Hills and Schmidt	103569.59	72
9	Keeling LLC	100934.30	74

# Sales Data

Let's use pandas to create a basic bar plot and then see how we can manipulate it with matplotlib.

```
df_top_ten.plot(kind="barh", y = "Sales", x = "Name")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x11b269cf8>
```



# Matplotlib

We need the following import statement when we want to use matplotlib

```
import matplotlib.pyplot as plt
```

We can use matplotlib to change the styling of our plots:

```
#View available styles
```

```
print(plt.style.available)
```

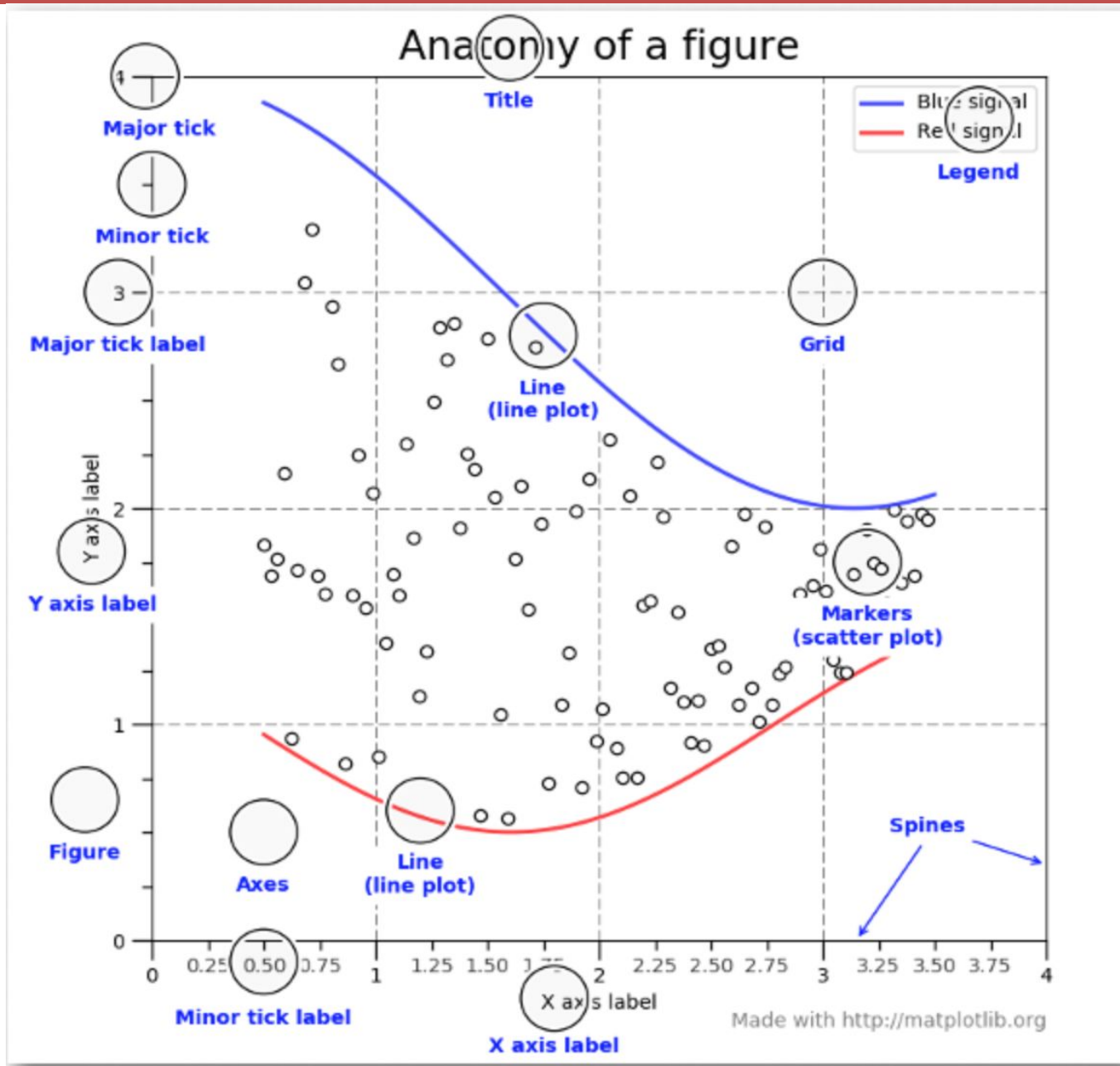
```
['bmh', 'classic', 'dark_background', 'fivethirtyeight', 'ggplot', 'grayscale', 'seaborn-bright', 'seaborn-colorblind', 'seaborn-dark-palette', 'seaborn-dark', 'seaborn-darkgrid', 'seaborn-deep', 'seaborn-muted', 'seaborn-notebook', 'seaborn-paper', 'seaborn-pastel', 'seaborn-poster', 'seaborn-talk',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

We can change the style as follows:

```
plt.style.use("ggplot")
```



# Creating Figure and Axes



# Creating Figure + Axis

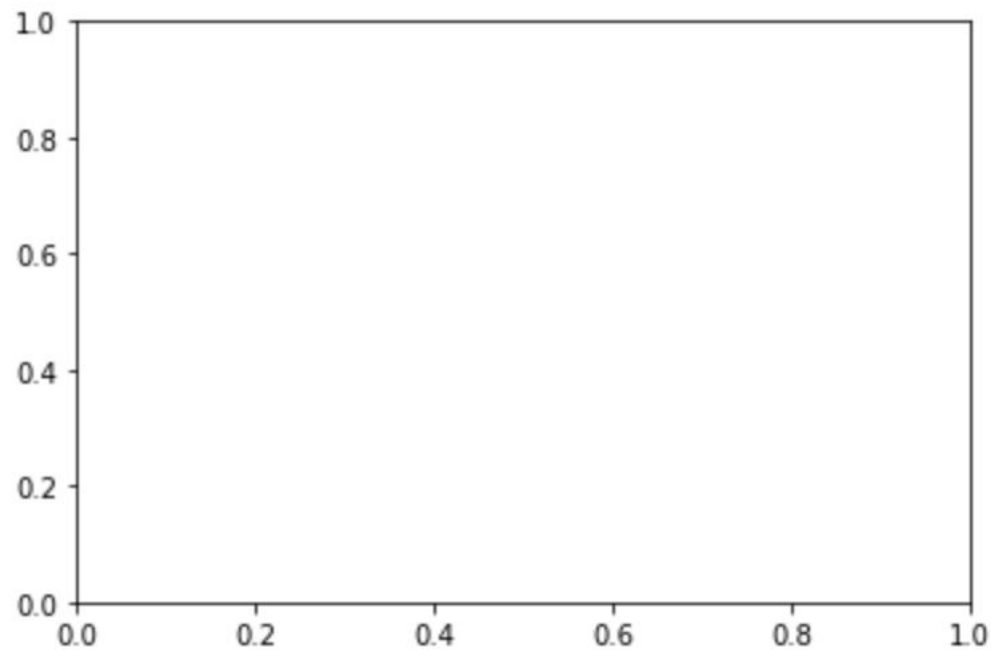
You will place something like this line of code before every plot that you create. Creates the following to elements of the visualization you want to produce:

- The variable fig will be the figure, which represents the entire visualization. A figure could contain multiple plots.
  - Adjust size
  - Number of plots
  - Borders
- The variable ax represents the axes of a single plot
  - Change axis limit/labels/ticks
  - Add multiple plots
  - Change titles

```
▼ #Create figure and axes  
fig, ax = plt.subplots()
```

# Creating Figure + Axis

```
▼ #Create fig with blank axis  
fig, ax = plt.subplots()
```



# Adding Barplot to Figure

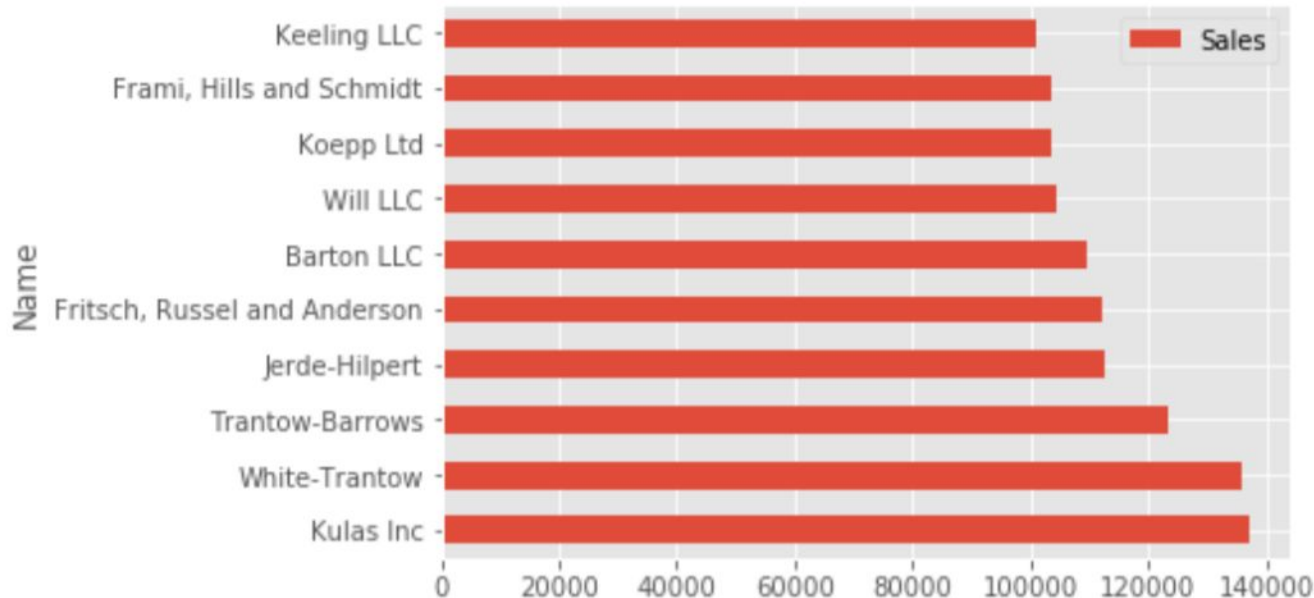
```
#Create figure and axes
```

```
fig,ax = plt.subplots()
```

```
#Add plot to figure
```

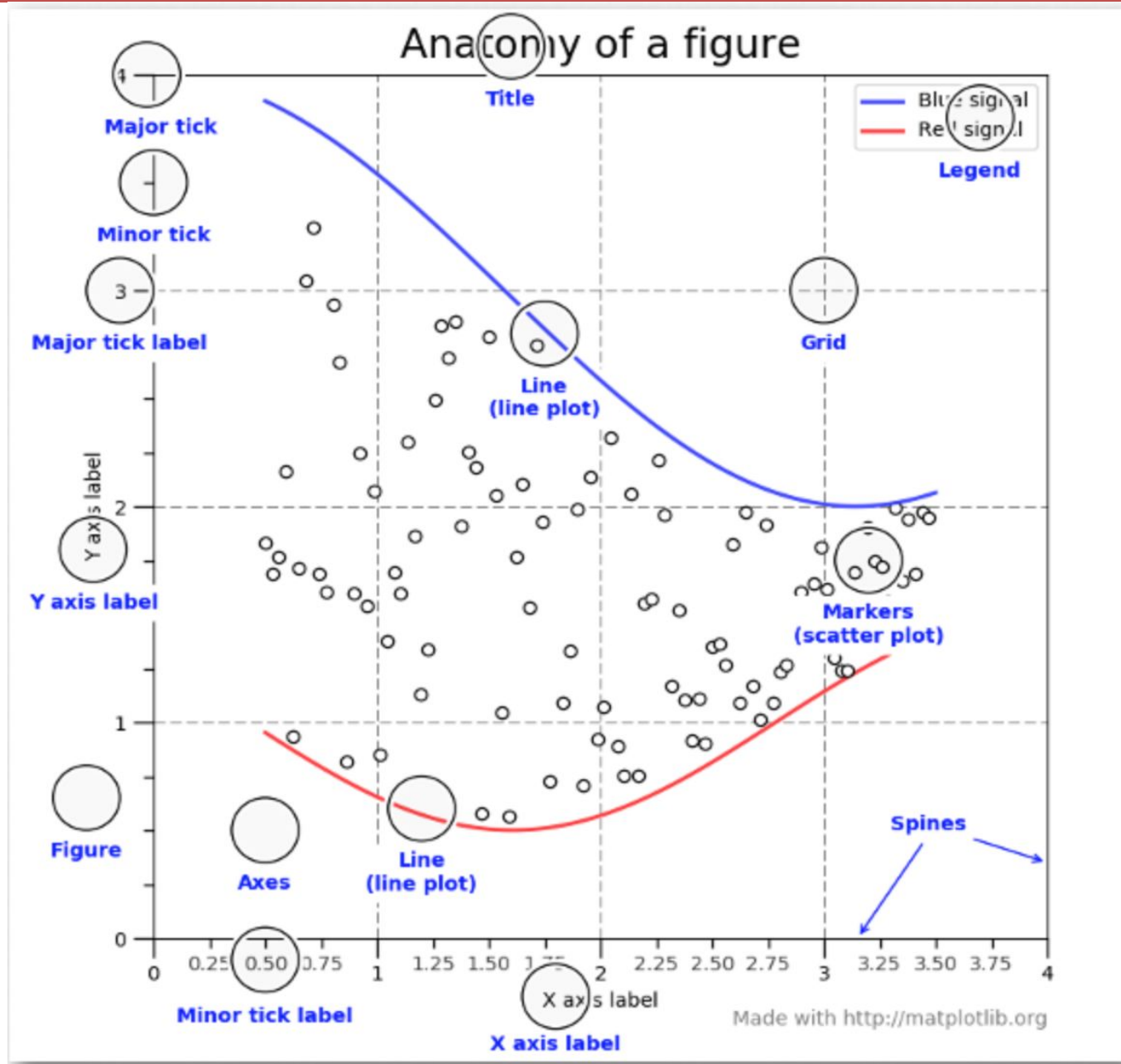
```
df_top_ten.plot(kind="barh", y = "Sales", x = "Name", ax=ax )
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x11b607b00>
```



Same plot, but now we can change the plot using attributes of the variables fig and ax.

# Changing Features of Axes



# Changing Features of Axes

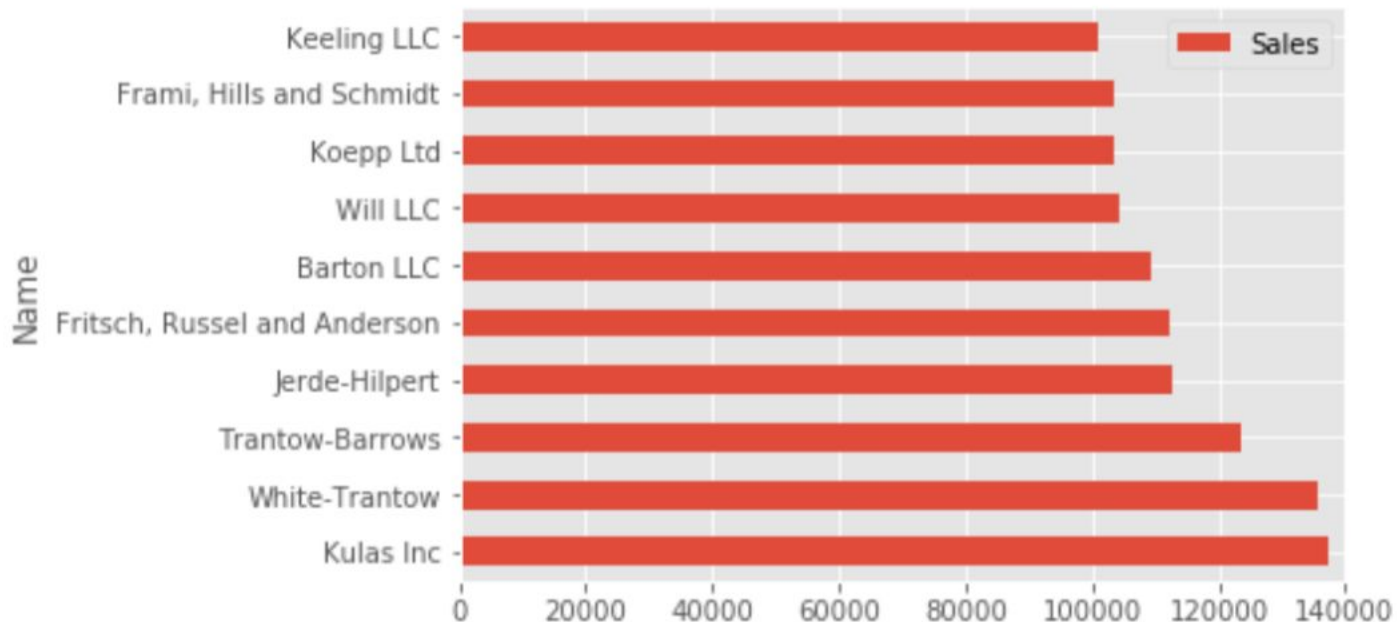
```
#Change x limit  
ax.set_xlim([0,140000])
```

The variable that stores our  
axes

List of x limits

# Changing Features of Axes

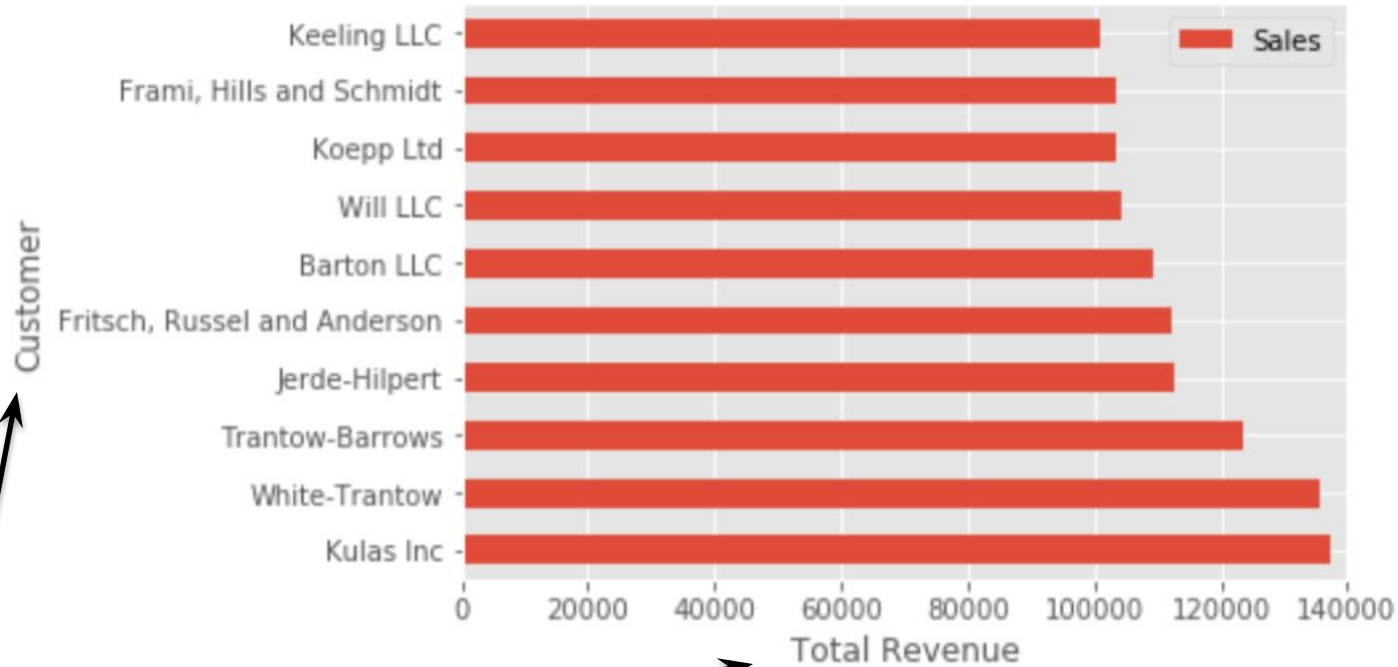
```
#Change x limit  
ax.set_xlim([0,140000])  
  
#View updated plot  
fig
```



x-axis only goes up to 140000

# Changing Features of Axes

```
▼ #Set axis labels  
ax.set_xlabel("Total Revenue")  
ax.set_ylabel("Customer")  
  
fig
```



New axis labels

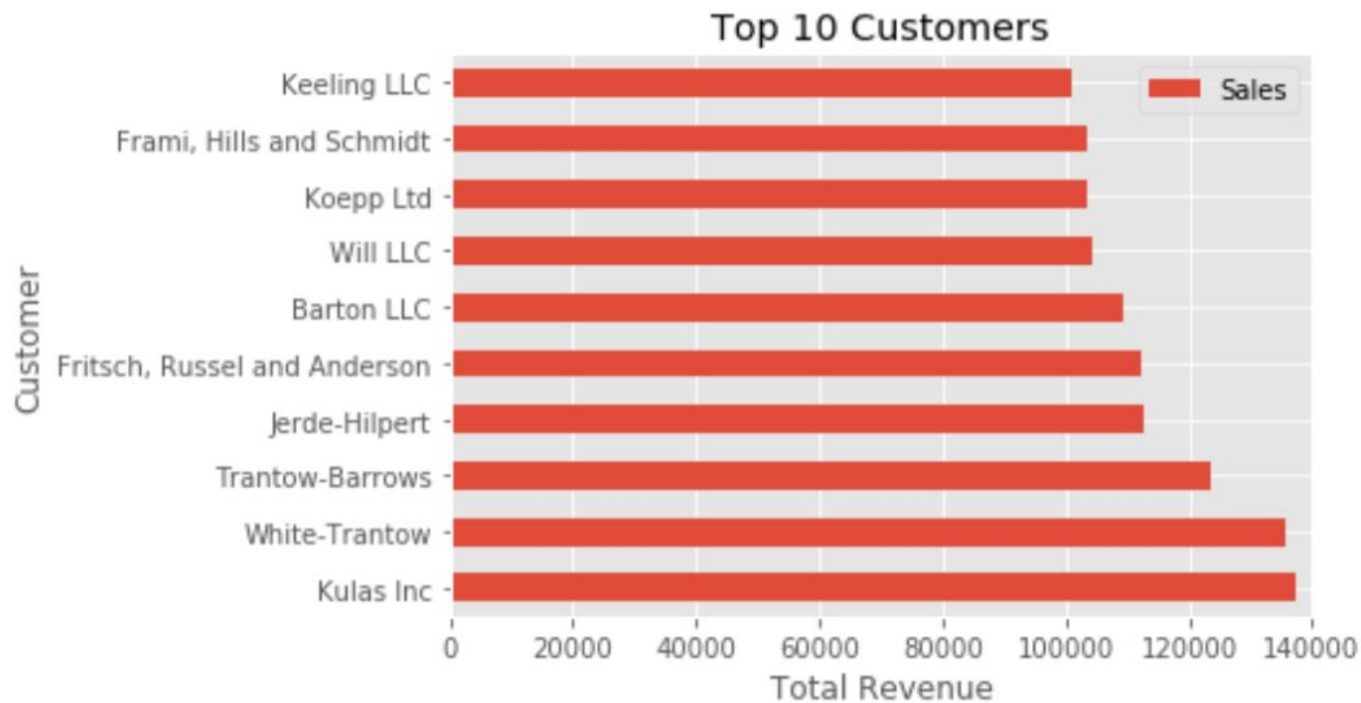


# Changing Features of Axes

```
#Adding labels all at once  
ax.set(title = "Top 10 Customers", xlabel="Total Revenue",  
        ylabel = "Customer")
```

fig

Title



# Change Figure Size

Give list of length of length and width in inches



```
#Change figure size  
fig,ax = plt.subplots(figsize = [5,6])  
#Add plot to figure  
df_top_ten.plot(kind="barh", y = "Sales", x = "Name", ax=ax )
```

# Change Figure Size

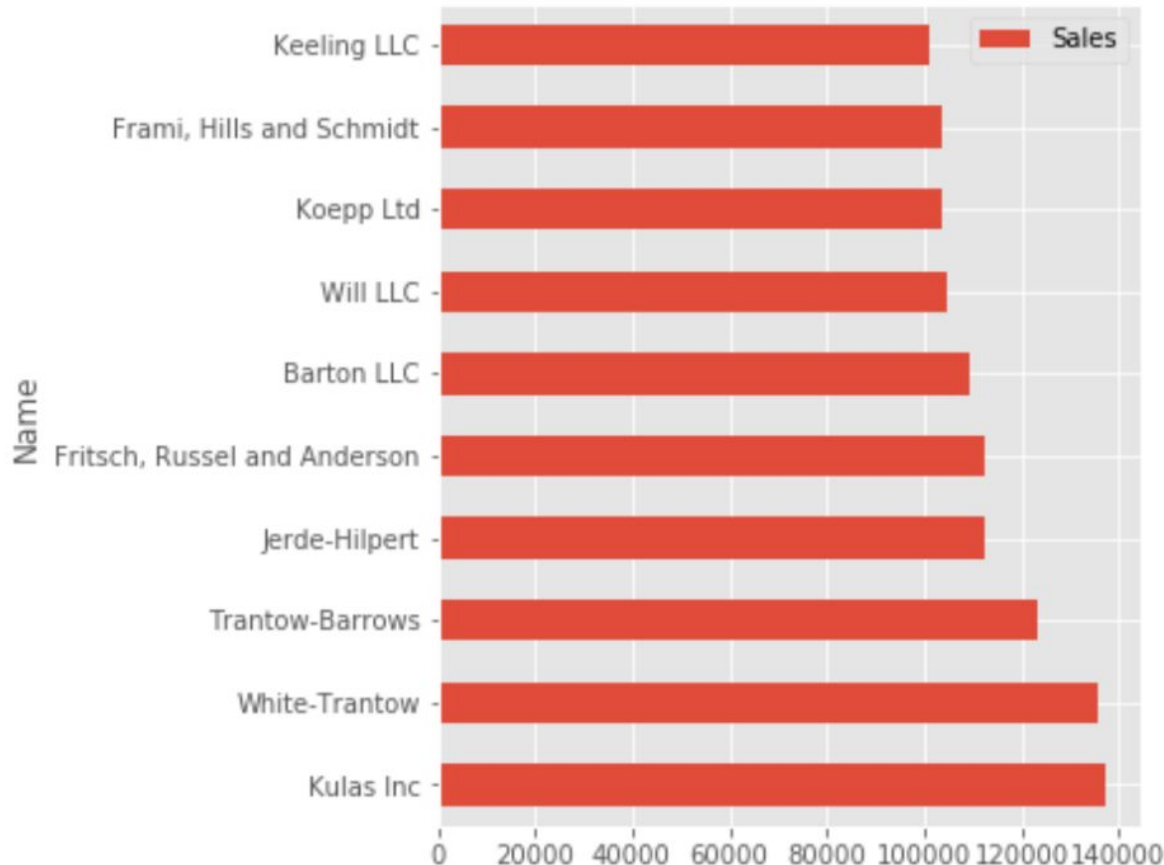
*#Change figure size*

```
fig,ax = plt.subplots(figsize = [5,6])
```

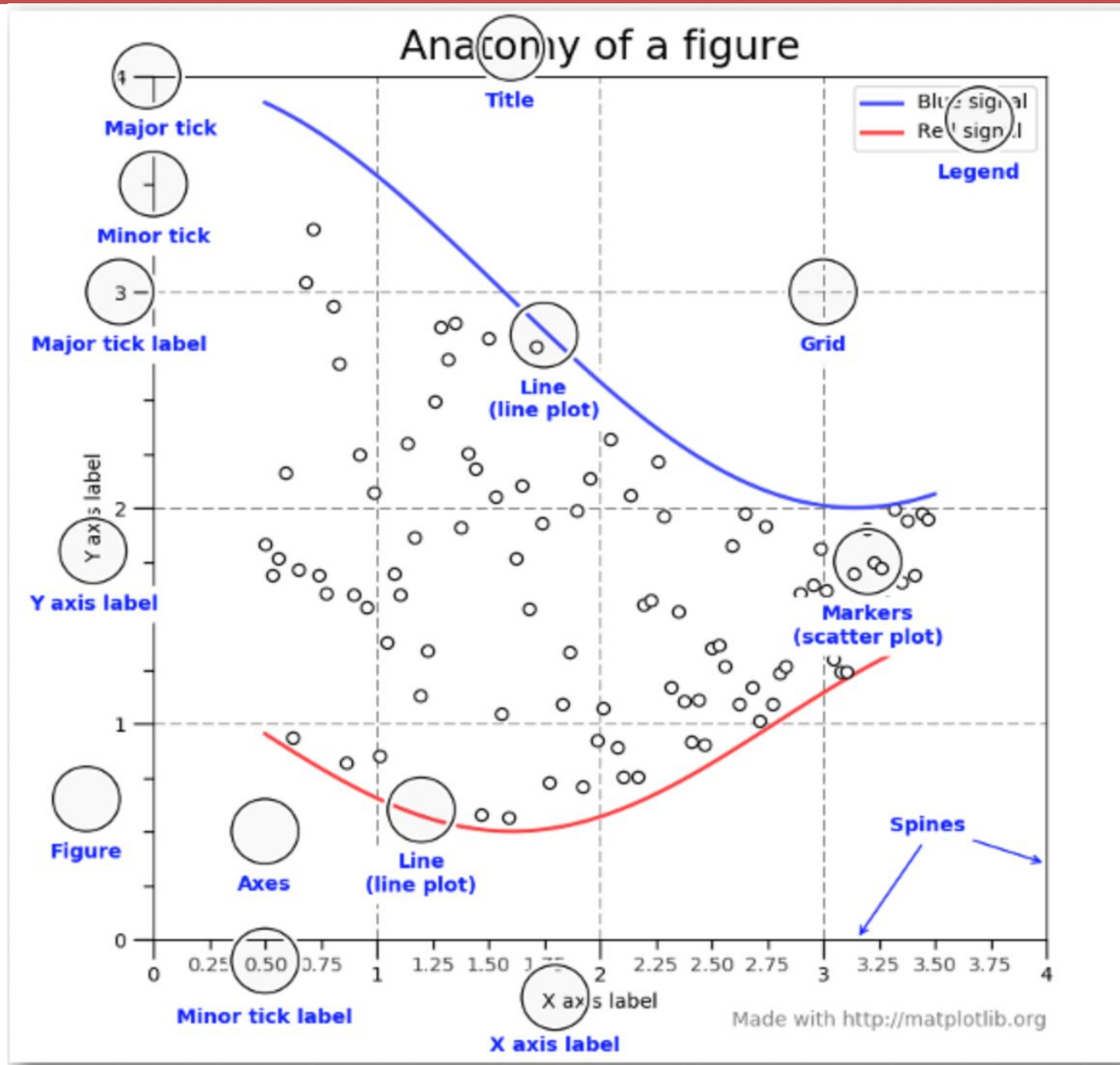
*#Add plot to figure*

```
df_top_ten.plot(kind="barh", y = "Sales", x = "Name", ax=ax )
```

<matplotlib.axes.\_subplots.AxesSubplot at 0x11bc2b748>




# Move Legend



# Move Legend

```
#Move Legend  
ax.legend(loc = 5)  
#View updated figure  
fig
```

loc parameter moves  
legend



Location String	Location Code
'best'	0
'upper right'	1
'upper left'	2
'lower left'	3
'lower right'	4
'right'	5
'center left'	6
'center right'	7
'lower center'	8
'upper center'	9
'center'	10

# Move Legend

```
▼ #Move Legend  
ax.legend(loc = 5)  
#View updated figure  
fig
```



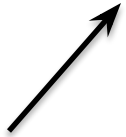
# Move Legend

```
▼ #Move Legend  
ax.legend(loc = 5)  
#View updated figure  
fig
```



# Annotate the Plot – Vertical Line

```
#Compute mean revenue  
mean_rev = df_top_ten.Sales.mean()  
#Add vertical line representing this mean  
ax.axvline(x=mean_rev, color='b', label='Average', linestyle='--', linewidth=1)
```



Where to draw  
vertical line



What do put in  
legend



# Annotate the Plot – Vertical Line

```
#Compute mean revenue
```

```
mean_rev = df_top_ten.Sales.mean()
```

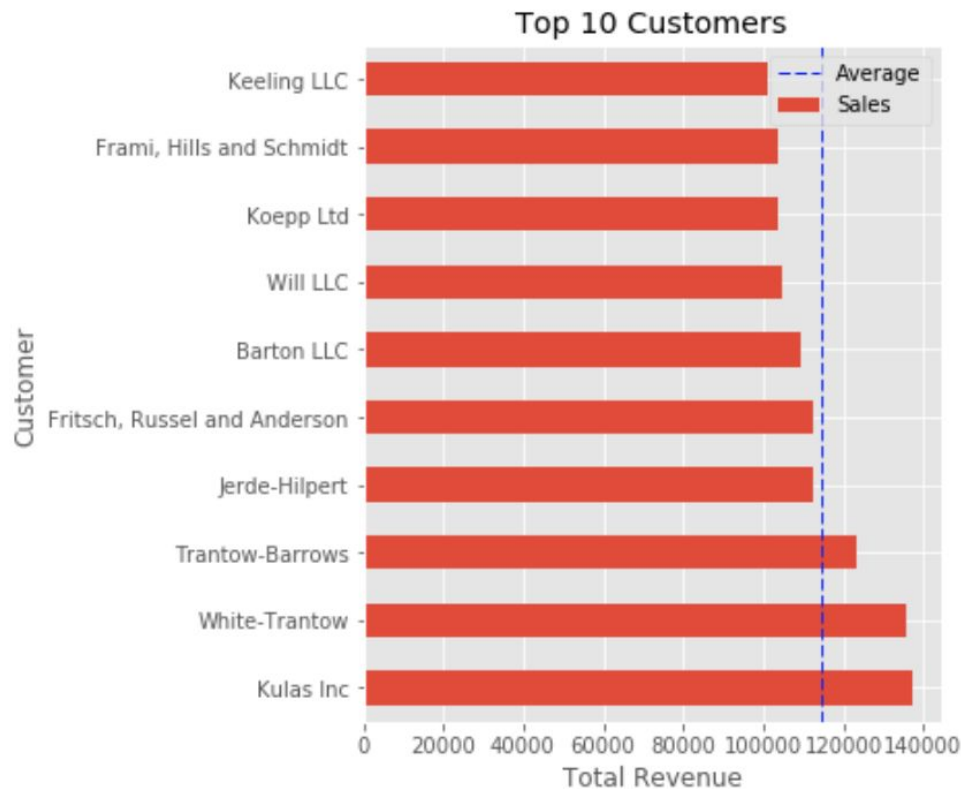
```
#Add vertical line representing this mean
```

```
ax.axvline(x=mean_rev, color='b', label='Average', linestyle='--', linewidth=1)
```

```
#Set legend to best location
```

```
ax.legend(loc=0)
```

```
fig
```



# Annotate the Plot – Add Text

```
▼ #Add Text  
ax.text(115000, 5, "New Customer")  
fig
```

(x,y) for where to place text

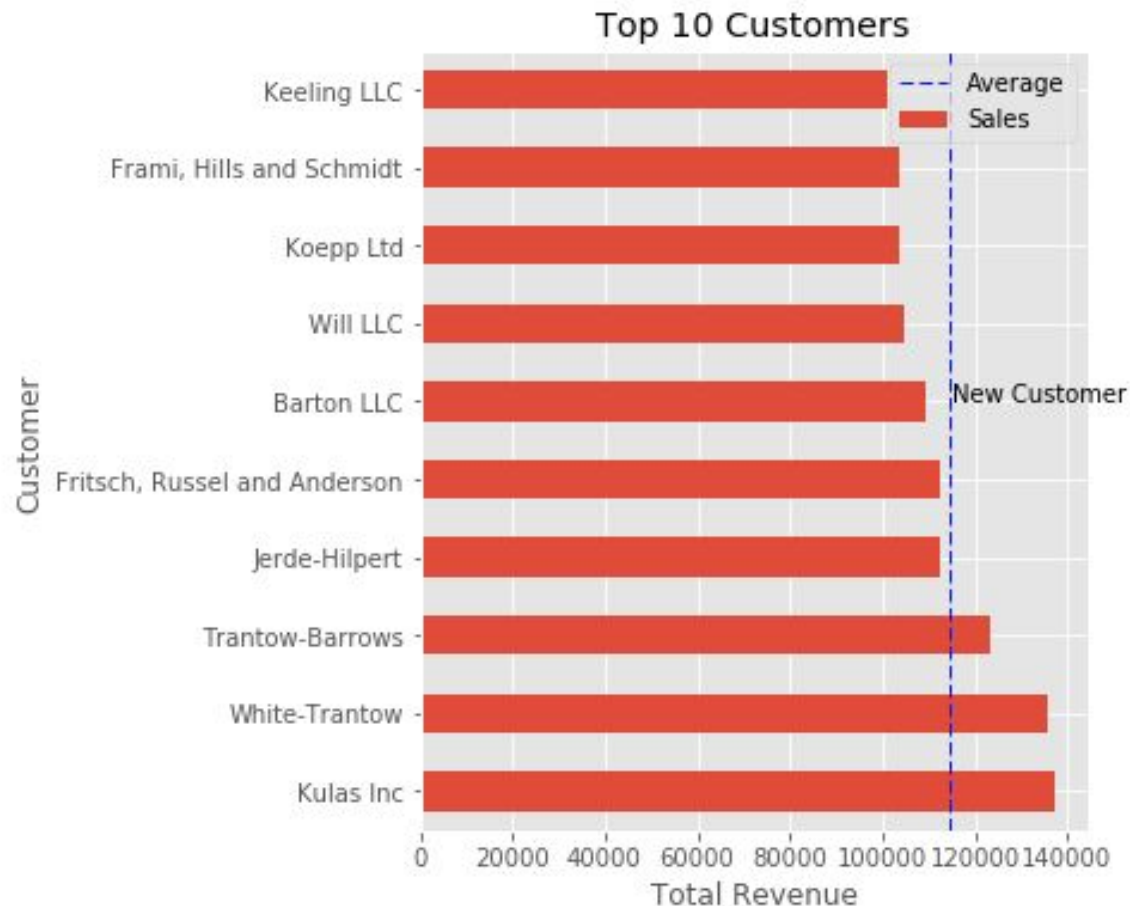


Text to place

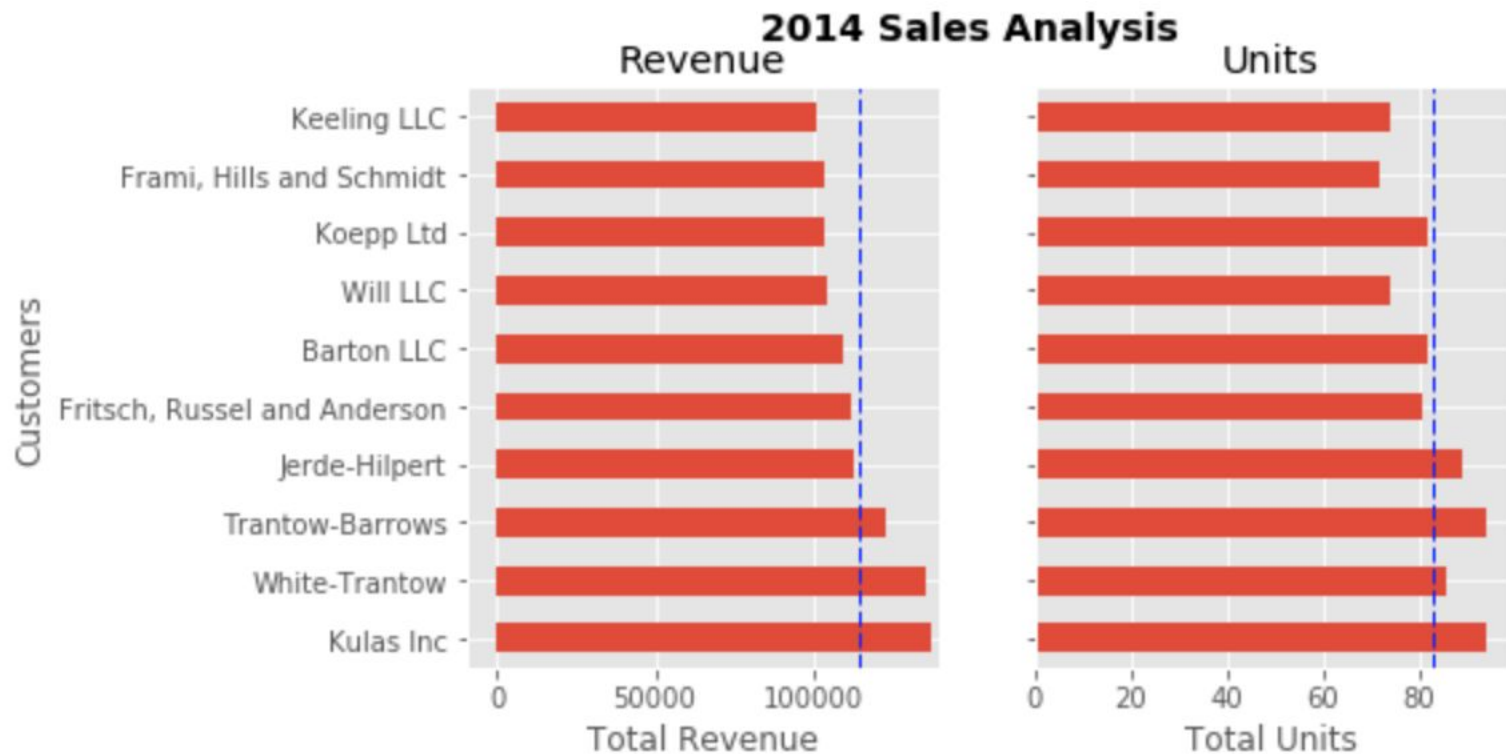


# Annotate the Plot – Add Text

```
▼ #Add Text  
ax.text(115000, 5, "New Customer")  
fig
```

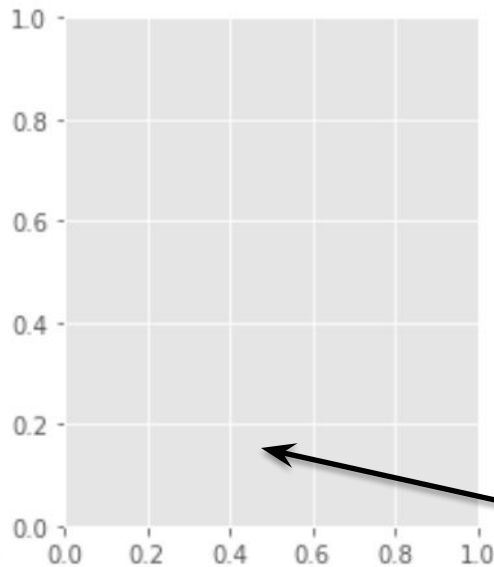
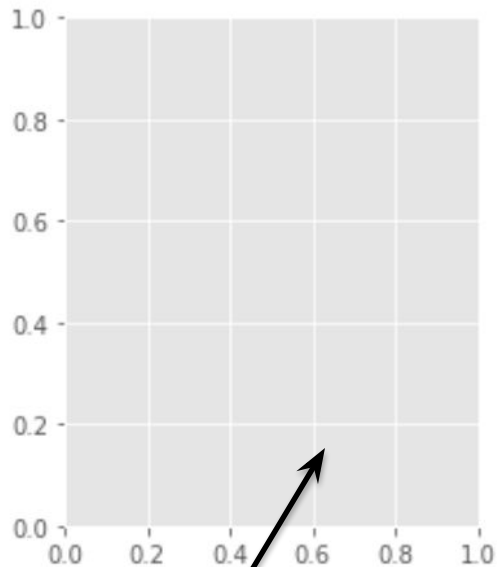


# Creating Multiple Plots



# Creating Multiple Plots

```
fig, (ax0, ax1) = plt.subplots(nrows=1, ncols=2, figsize=(7, 4))
```



7x4 Figure, with 1  
row and 2 columns of  
subplots

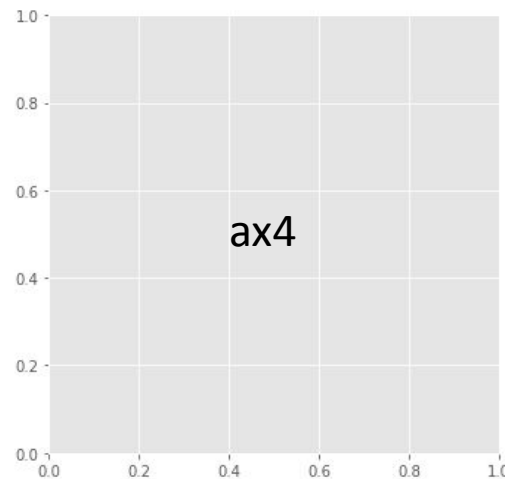
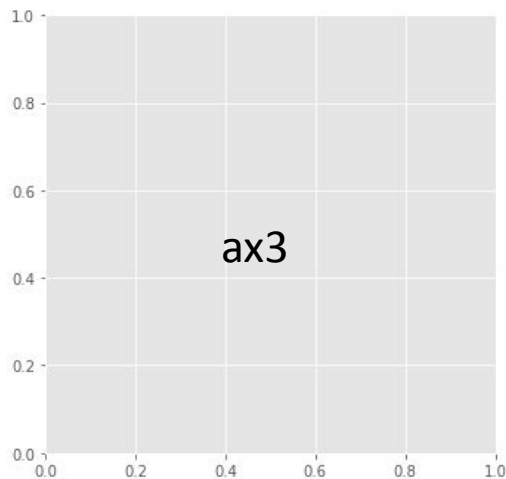
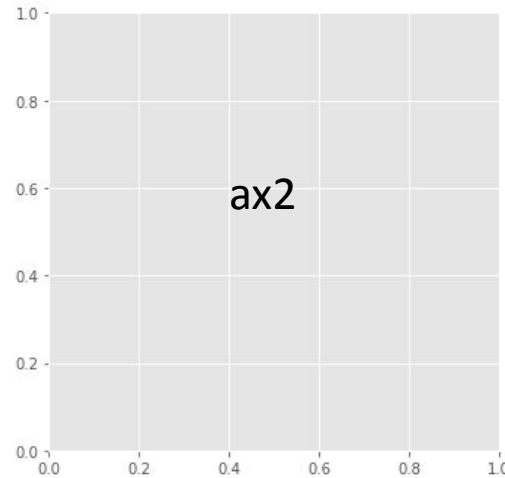
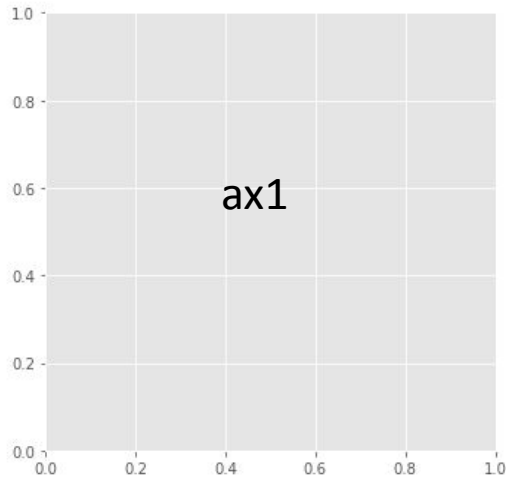
ax0

ax1

# Creating Multiple Plots

List of lists

```
fig, [[ax1, ax2], [ax3, ax4]] = plt.subplots(nrows=2, ncols=2, figsize = (12,12))
```

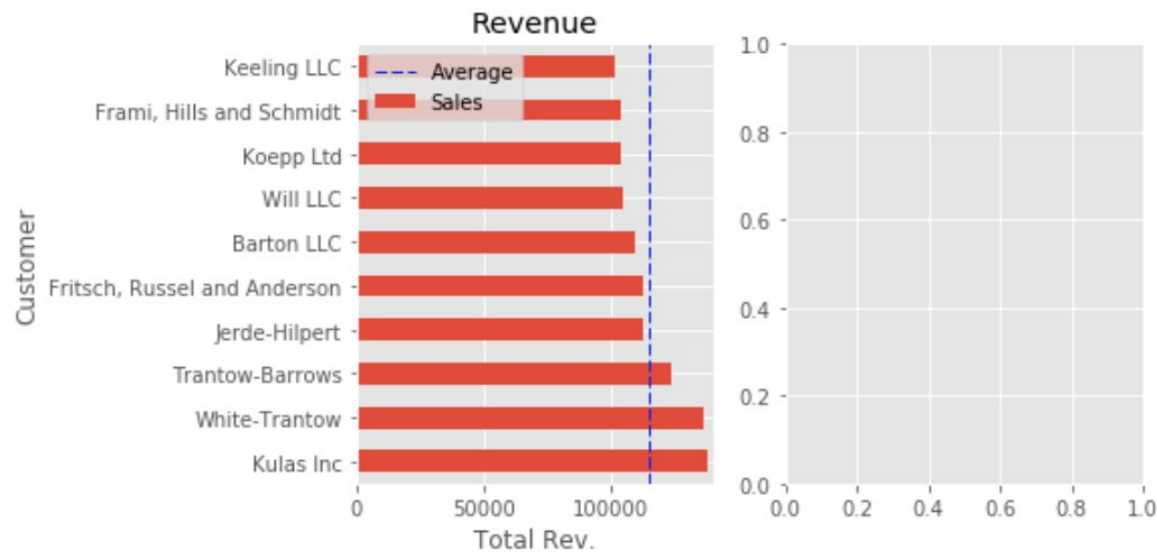


12x12 Figure, with 2 rows and 2 columns of subplots

# Creating Multiple Plots

```
#Make first bar plot
df_top_ten.plot(kind="barh", x = "Name", y = "Sales", ax = ax0)
ax0.set(title = "Revenue", xlabel="Total Rev.", ylabel="Customer")
ax0.set_xlim([0,140000])
avg=df_top_ten.Sales.mean()
ax0.axvline(x = avg, color='b', label="Average", linestyle='--', linewidth=1)
ax0.legend(loc=0)
```

<matplotlib.legend.Legend at 0x118d6e438>

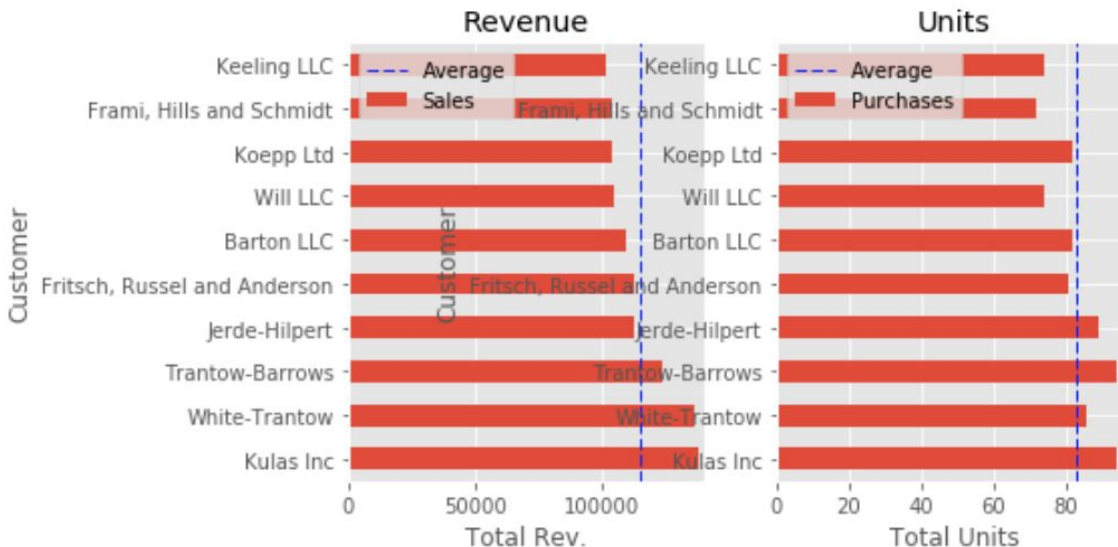


Notice that all references are made to ax0.

# Creating Multiple Plots

*#Make second bar plot*

```
df_top_ten.plot(kind="barh", x = "Name", y = "Purchases", ax = ax1)
ax1.set(title = "Units", xlabel="Total Units", ylabel="Customer")
avg=df_top_ten.Purchases.mean()
ax1.axvline(x = avg, color='b', label="Average", linestyle='--', linewidth=1)
ax1.legend(loc=0)
fig
```



Uh oh! Plots are overlapping. Let's two options for fixing this.

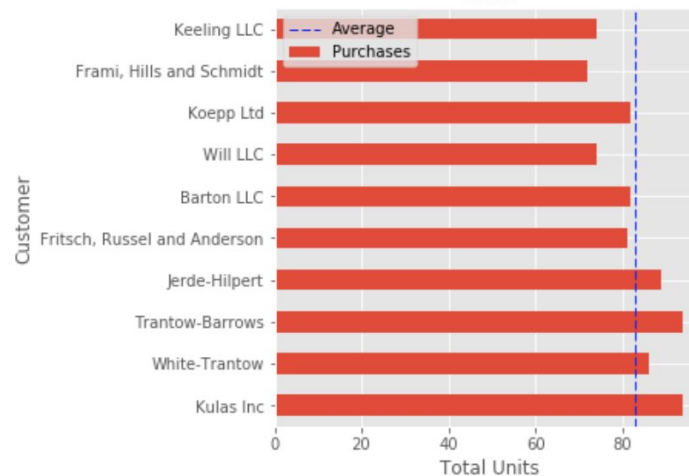
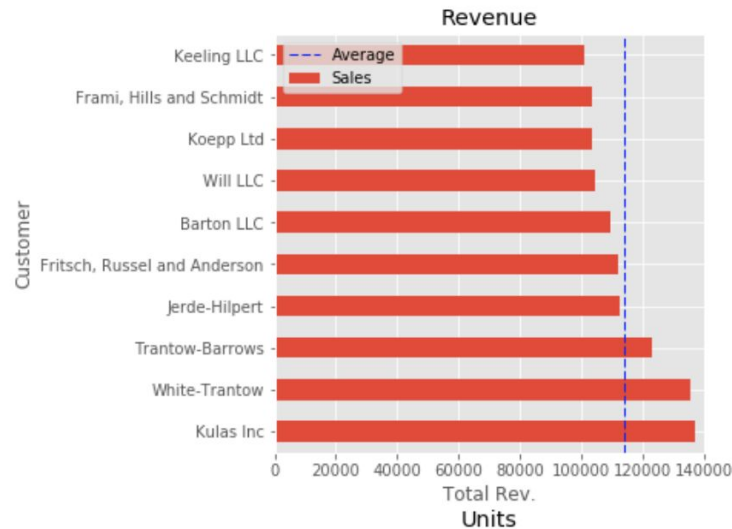


# Creating Multiple Plots

*#Option 1 - Stack them*

```
fig, (ax0, ax1) = plt.subplots(nrows=2, ncols=1, figsize=(5, 11))
```

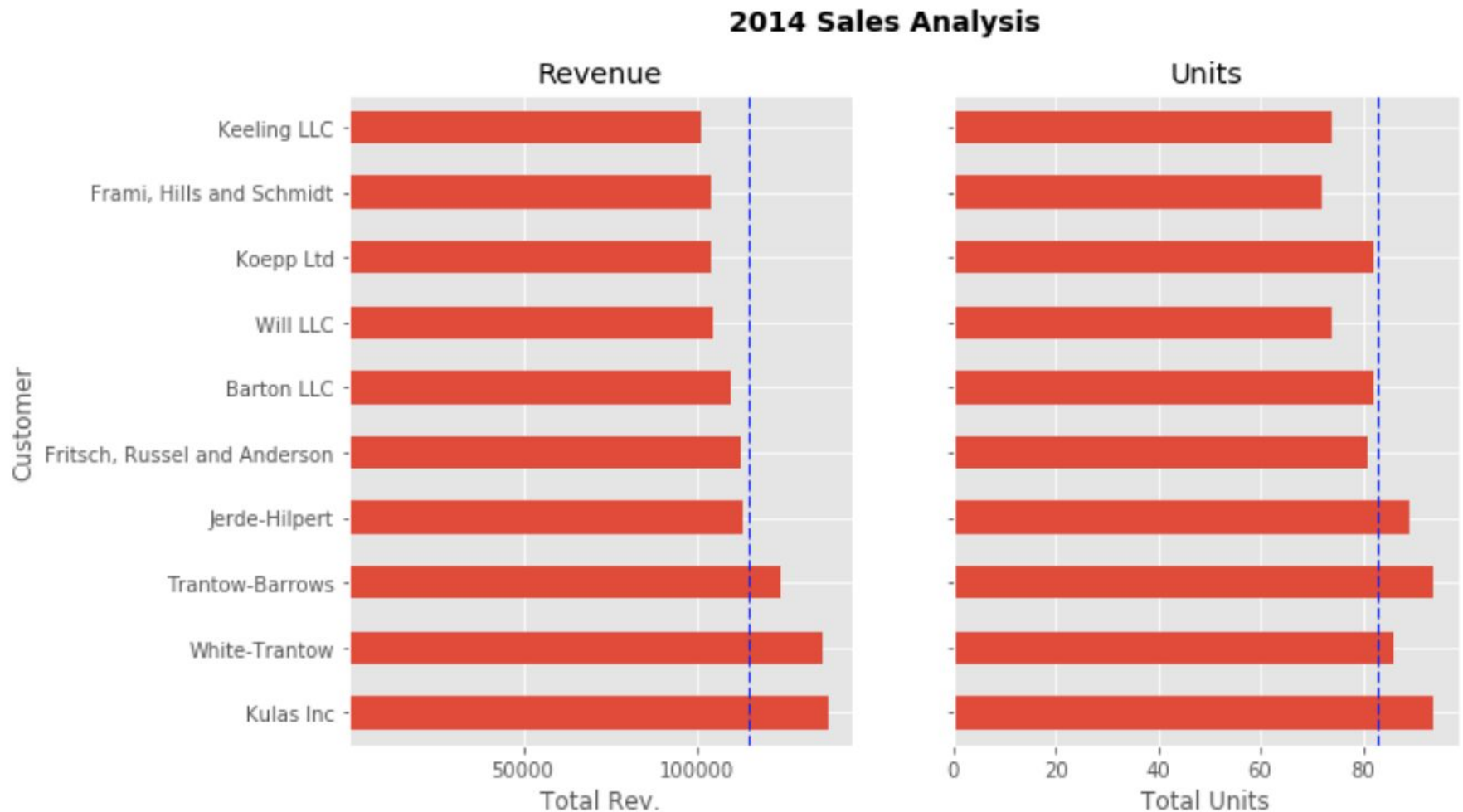
updated



# Creating Multiple Plots

*#Option 2 - Have them share y axis*

```
fig, (ax0, ax1) = plt.subplots(nrows=1, ncols=2, sharey=True, figsize=(10, 6))
```



# All of the Code

*#Option 2 - Have them share y axis*

```
fig, (ax0, ax1) = plt.subplots(nrows=1, ncols=2, sharey=True, figsize=(10, 6))
```

*#Make first bar plot*

```
df_top_ten.plot(kind="barh", x = "Name", y = "Sales", ax = ax0)
```

```
ax0.set(title = "Revenue", xlabel="Total Rev.", ylabel="Customer")
```

```
ax0.set_xticks([50000,100000])
```

```
avg=df_top_ten.Sales.mean()
```

```
ax0.axvline(x = avg, color='b', label="Average",\
            linestyle='--', linewidth=1)
```

```
ax0.legend(loc=0)
```

*#Make second bar plot*

```
df_top_ten.plot(kind="barh", x = "Name", y = "Purchases", ax = ax1)
```

```
ax1.set(title = "Units", xlabel="Total Units", ylabel="Customer")
```

```
avg=df_top_ten.Purchases.mean()
```

```
ax1.axvline(x = avg, color='b', label="Average", linestyle='--', linewidth=1)
```

```
ax1.legend(loc=0)
```

*# Title the figure*

```
fig.suptitle('2014 Sales Analysis', fontsize=14, fontweight='bold');
```

*# Hide the legends*

```
ax1.legend().set_visible(False)
```

```
ax0.legend().set_visible(False)
```