Visualization Libraries

Matplotlib:

- It is a Python library used for plotting graphs with the help of other libraries like Numpy and Pandas. It is a powerful tool for visualizing data in Python.
- It is used for creating statical interferences and plotting 2D graphs of arrays.
- It was first introduced by John D. Hunter in 2002.
- It uses Pyplot for providing MATLAB like interface free and open-source.
- It is capable of dealing with various operating systems and their graphical backends.
- Matplotlib is a Python 2D plotting library which produces publication-quality figures in a variety of hardcopy formats and interactive environments across platforms.

Sl.No.	Functions	Description
1.	import matplotlib.pyplot as plt	Loading matplotlib into notebook
2.	plt.plot(x,y)	simple line plot
3.	plt.xlabel('str')	set x label
4.	plt.ylabel('str')	set y label
5.	plt.title('str')	set title
6.	plt.subplot(r,c,1) plt.plot(x, y) plt.subplot(r,c,2) plt.plot(y, x, 'g*-')	create multiplot
7.	subplot()	This command requires to specify the number of row and column we want to print the plots, and the third parameter specify what of the graph we are going to handle.
8.	fig = plt.figure()	create canvas
9.	ax = fig.add_axes([0,0,1,1])	create axes*

10.	ax.plot(x, y, 'b')	create plot
11.	ax.set_xlabel(''str)	set x label
12.	ax.set_ylabel('str')	set y label
13.	ax.set_title(''str)	set title
14.	fig, ax = plt.subplots(r,c)	create an array of axes in the figure
15.	axes[0].plot(x,y)	create pl ax1
16.	axes[1].plot(x,y)	create pl ax2
17.	axes[0].set_title('str')	set plot 1 title
18.	axes[1].set_title(''str)	set plot 2 title
19.	plt.tight_layout()	avoid overlap
20.	plt.fig(figsize=(x,x))	set figure size
21.	plt.fig(figsize=(x,x), dpi=x)	set dpi
22.	fig.savefig("name.png")	save figure
23.	fig.savefig("name", dpi=200)	set dpi
24.	ax.plot(x, y, label="str")	set legend
25.	ax.legend()	show legend
26.	ax.legend(loc=0)	best
27.	ax.legend(loc=1)	upper right
28.	ax.legend(loc=2)	upper left
29.	ax.legend(loc=3)	lower left
30.	ax.legend(loc=4)	lower right
31.	fig = plt.figure()	creates a figure (but with no axes in it)
32.	ax = fig.add_axes([0,0,1,1])	----
33.	ax.plot (x,y	----
	color='#xxxxxx',	set color
	lw=x,	set linewidth
	alpha=x,	set alpha

	ls='',	set line style
	marker='',	set marker type
	markersize=x,	set marker size
	markerfacecolor='',	set mark color
	markeredgecolor='',	set external col
	markeredgewidth=x)	set marker width
34.	ax.set_xlim([0,1])	set x axes limit
35.	ax.set_ylim([0,1])	set y axes limit
36.	ax.plot(x, y, 'r--')	MATLAB style
37.	plt.tight_layout()	Adjust subplot params
38.	plt.ylim(0,100)	Adjust the limits of the y-axis
39.	plt.xlim(0,10)	Adjust the limits of the x-axis

Subplots():

- This utility wrapper makes it convenient to create common layouts of subplots, including the enclosing figure object, in a single call.
- That means you can use this single function to create a figure with several subplots with only one line of code. For example, the code below will return both fig which is the figure object, and axes which is a 2x3 array of axes objects which allows you to easily access each subplot:

```
fig, axes = plt.subplots(nrows=2, ncols=3)
```

subplot():

- In contrast, matplotlib.pyplot.subplot() creates only a single subplot axes at a specified grid position. This means it will require several lines of code to achieve the same result as matplot.pyplot.subplots() did in a single line of code above:

```
# First you have to make the figure
fig = plt.figure(1)
# Now you have to create each subplot individually
ax1 = plt.subplot(231)
ax2 = plt.subplot(232)
ax3 = plt.subplot(233)
ax4 = plt.subplot(234)
```

```
ax5 = plt.subplot(235)
ax6 = plt.subplot(236)
```

axes: ([left, bottom, width, height])
fig, axes allow you to auto-manage axis, you don't have to create them. Axes, now, will be an array of axis. We could use for loop to populate labels on axis.

Prepare The Data:

1D Data

```
>>> import numpy as np
>>> x = np.linspace(0, 10, 100)
>>> y = np.cos(x)
>>> z = np.sin(x)
```

2D Data or Images

```
>>> data = 2 np.random.random((10, 10))
>>> data2 = 3 np.random.random((10, 10))
>>> Y, X = np.mgrid[-3:3:100j, -3:3:100j]
>>> U = -1 - X 2 + Y
>>> V = 1 + X - Y 2
>>> from matplotlib.cbook import get_sample_data
>>> img = np.load(get_sample_data('axes_grid/bivariate_normal.npy' ))
```

Create Plot

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

Figure

```
>>> fig = plt.figure()
>>> fig2 = plt.figure(figsize=plt.figaspect(2.0))
```

Axes

All plotting is done with respect to an Axes. In most cases, a subplot will fit your needs.
A subplot is an axes on a grid system.

```
>>> fig.add_axes()
>>> ax1 = fig.add_subplot(221) #row-col-num
>>> ax3 = fig.add_subplot(212)
>>> fig3, axes = plt.subplots(nrows=2,ncols=2)
```

```
>>> fig4, axes2 = plt.subplots(ncols=3)
```

Save Plot

```
>>> plt.savefig('foo.png' ) #Save figures
>>> plt.savefig( 'foo.png', transparent=True) #Save transparent figures
```

Show Plot

```
>>> plt.show()
```

Plotting Routines

1D Data

```
>>> fig, ax = plt.subplots()
>>> lines = ax.plot(x,y) #Draw points with lines or markers connecting them
>>> ax.scatter(x,y) #Draw unconnected points, scaled or colored
>>> axes[0,0].bar([1,2,3],[3,4,5]) #Plot vertical rectangles (constant width)
>>> axes[1,0].barh([0.5,1,2.5],[0,1,2]) #Plot horizontal rectangles (constant height)
>>> axes[1,1].axhline(0.45) #Draw a horizontal line across axes
>>> axes[0,1].axvline(0.65) #Draw a vertical line across axes
>>> ax.fill(x,y,color='blue') #Draw filled polygons
>>> ax.fill_between(x,y,color='yellow') #Fill between y-values and 0
```

2D Data

```
>>> fig, ax = plt.subplots()
>>> im = ax.imshow(img, #Colormapped or RGB arrays
cmap= 'gist_earth',
interpolation='nearest' ,vmin=-2,vmax=2)
>>> axes2[0].pcolor(data2) #Pseudocolor plot of 2D array
>>> axes2[0].pcolormesh(data) #Pseudocolor plot of 2D array
>>> CS = plt.contour(Y,X,U) #Plot contours
>>> axes2[2].contourf(data1) #Plot filled contours
>>> axes2[2]= ax.clabel(CS) #Label a contour plot
```

Data Distributions

```
>>> ax1.hist(y) #Plot a histogram
>>> ax3.boxplot(y) #Make a box and whisker plot
>>> ax3.violinplot(z) #Make a violin plot
```

Workflow

The basic steps to creating plots with matplotlib are:

1 Prepare Data

2 Create Plot

3 Plot

4 Customized Plot

5 Save Plot

6 Show Plot

```
>>> import matplotlib.pyplot as plt
>>> x = [1,2,3,4] #Step 1
>>> y = [10,20,25,30]
>>> fig = plt.figure()#Step 2
>>> ax = fig.add_subplot(111) #Step 3
>>> ax.plot(x, y, color= 'lightblue', linewidth=3) #Step 3, 4
>>> ax.scatter([2,4,6],[5,15,25],color='darkgreen' , marker= '^')
>>> ax.set_xlim(1, 6.5)
>>> plt.savefig('foo.png' )
>>> plt.show()
```

Close and Clear

```
>>> plt.cla()#Clear an axis
>>> plt.clf()#Clear the entire figure
>>> plt.close()#Close a window
```

Seaborn:

- It is also a Python library used for plotting graphs with the help of Matplotlib, Pandas, and Numpy.
- It is built on the roof of Matplotlib and is considered as a superset of the Matplotlib library.
- It helps in visualizing univariate and bivariate data. It uses beautiful themes for decorating Matplotlib graphics.
- It acts as an important tool in picturing Linear Regression Models.
- It serves in making graphs of statical Time-Series data.
- It eliminates the overlapping of graphs and also aids in their beautification.

The Python visualization library Seaborn is based on matplotlib and provides a high-level interface for drawing attractive statistical graphics.

```
>>> import matplotlib.pyplot as plt
>>> import seaborn as sns
```

The basic steps to creating plots with Seaborn are:

1. Prepare some data
2. Control figure aesthetics
3. Plot with Seaborn
4. Further customize your plot

```
>>> import matplotlib.pyplot as plt
>>> import seaborn as sns
>>> tips = sns.load_dataset("tips") #step 1
>>> sns.set_style("whitegrid") #step 2
>>> g = sns.lmplot(x="tip",y="total_bill",data=tips,aspect=2) #step 3
>>> g = (g.set_axis_labels("Tip","Total bill(USD)").set(xlim=(0,10),ylim=(0,100)))
>>> plt.title("title")#step 4
>>> plt.show(g) #step 5
```

Sl.No.	Functions	Description
40.	>>>f, ax = plt.subplots(figsize=(5,6))	Create a figure and one subplot
41.	>>> sns.set()	(Re)set the seaborn default
42.	>>> sns.set_style("whitegrid")	Set the matplotlib parameters
43.	>>> sns.set_style("ticks",{"xtick.major.size":8, "ytick.major.size":8})	Set the matplotlib parameters
44.	>>> sns.axes_style("whitegrid")	Return a dict of params or use with with to temporarily set the style
45.	>>> sns.set_palette("husl",3)	Define the color palette
46.	>>>sns.color_palette("husl")	Use with to temporarily set palette
47.	>>> g = sns.FacetGrid(titanic, col="survived", row="sex") >>> g = g.map(plt.hist,"age")	Subplot grid for plotting conditional relationships
48.	>>> sns.factorplot(x="pclass", y="survived", hue="sex", data=titanic)	Draw a categorical plot onto a Facetgrid
49.	>>> sns.lmplot(x="sepal_width", y="sepal_length", hue="species", data=iris)	Plot data and regression model fits across a FacetGrid
50.	>>> h = sns.PairGrid(iris) >>> h = h.map(plt.scatter)	Subplot grid for plotting pairwise relationships
51.	>>> sns.pairplot(iris)	Plot pairwise bivariate distributions

52.	>>> i = sns.JointGrid(x="x", y="y", data=data)
	
	>>> i = i.plot(sns.regplot, sns.distplot)
		Grid for bivariate plot with marginal univariate plots

53.	>>> sns.jointplot("sepal_length", "sepal_width", data=iris, kind='kde')	Plot bivariate distribution
54.	**Categorical Plots**	
55.	**Scatterplot**	
56.	>>> sns.stripplot(x="species", y="petal_length", data=iris)	Scatterplot with one categorical variable
57.	>>> sns.swarmplot(x="species", y="petal_length", data=iris)	Categorical scatterplot with non-overlapping points
58.	**Bar Chart** >>> sns.barplot(x="sex", y="survived", hue="class", data=titanic)	Show point estimates and confidence intervals with scatterplot glyphs
59.	**Count Plot** >>> sns.countplot(x="deck", data=titanic, palette="Greens_d")	Show count of observations
60.	**Point Plot**	Show point estimates and

	>>> sns.pointplot(x="class", y="survived", hue="sex", data=titanic, palette={"male":"g", "female":"m"},markers=["^","o"], linestyles=["-","--"])	confidence intervals as rectangular bars

61.	**Boxplot**
	>>> sns.boxplot(x="alive", y="age", hue="adult_male", data=titanic)
	
	>>> sns.boxplot(data=iris,orient="h")
		Boxplot
		
		Boxplot with wide-form data

62.	**Violinplot** >>> sns.violinplot(x="age", y="sex", hue="survived", data=titanic)	Violin plot
63.	**Regression Plots** >>> sns.regplot(x="sepal_width", y="sepal_length", data=iris,ax=ax)	Plot data and a linear regression model fit
64.	**Distribution Plots** >>> plot = sns.distplot(data.y,kde=False, color="b")	Plot univariate distribution
65.	**Matrix Plots** >>> sns.heatmap(uniform_data,vmin=0,vmax=1)	Heatmap
66.	>>> h.set(xlim=(0,5), ylim=(0,5), xticks=[0,2.5,5],yticks=[0,2.5,5])	Set the limit and ticks of the x-and y-axis

Sample Interview questions:

1. How do you plot a line chart?
2. How do you plot a bar chart?
3. How do you plot scatter chart?
4. How do you plot heat map?
5. How do you plot distribution chart?
6. How do you add x-label and y-label to the chart?
7. How do you add title to the chart?
8. How do you add legend to chart?