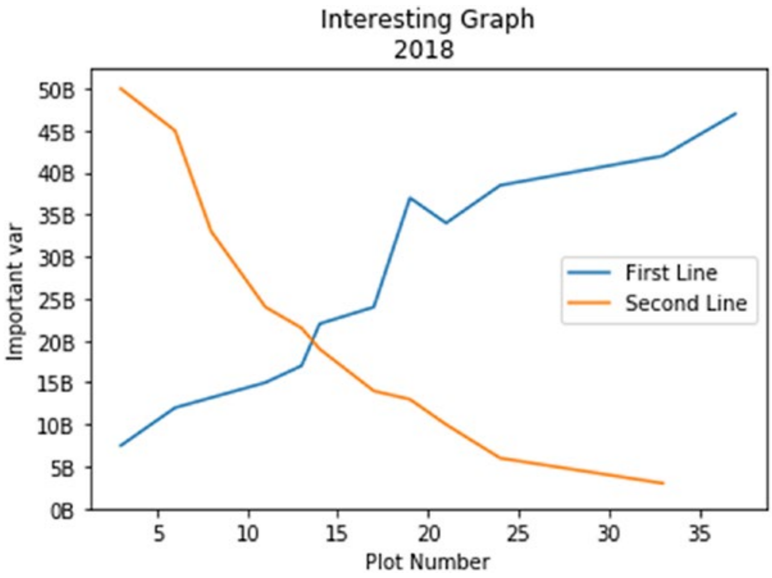
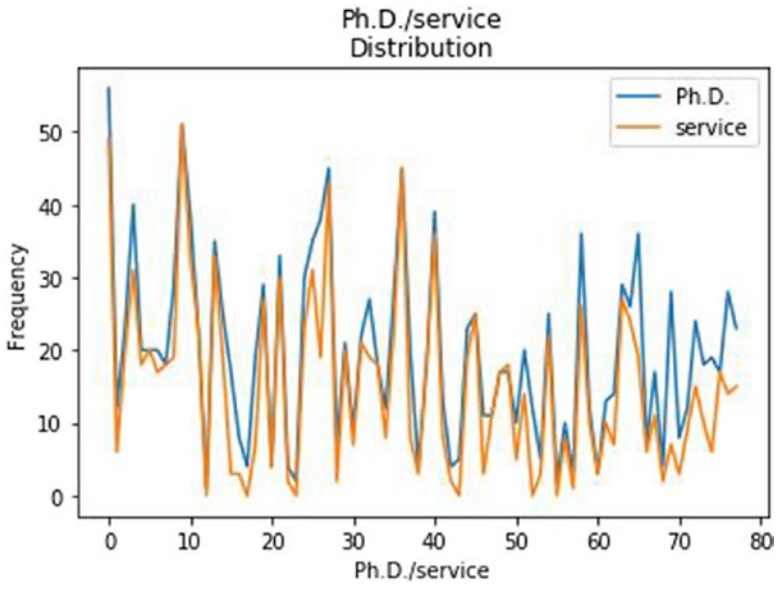
Listing 7-16. Matplotlib Line Plotting

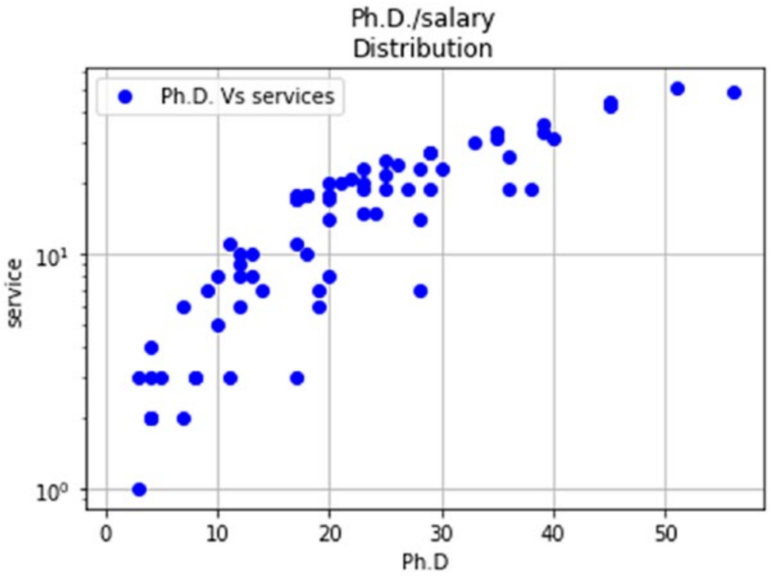
```
In [2]: import matplotlib.pyplot as plt
        x =[3,6,8,11,13,14,17,19,21,24,33,37]
        y = [7.5,12,13.2,15,17,22,24,37,34,38.5,42,47]

        x2 =[3,6,8,11,13,14,17,19,21,24,33]
        y2 = [50,45,33,24,21.5,19,14,13,10,6,3]
        plt.plot(x,y, label='First Line')
        plt.plot(x2, y2, label='Second Line')
        plt.xlabel('Plot Number')
        plt.ylabel('Important var')
        plt.title('Interesting Graph\n2018 ')
        plt.yticks([0,5,10,15,20,25,30,35,40,45,50],
                ['0B','5B','10B','15B','20B','25B','30B','35B',
                '40B','45B','50
        B'])
        plt.legend()
        plt.show()
```



```
In [13]: plt.plot(phd, label='Ph.D.')
         plt.plot(service, label='service')
         plt.xlabel('Ph.D./service')
         plt.ylabel('Frequency')
         plt.title('Ph.D./service\nDistribution')
         plt.legend()
         plt.show()
```



```
In [15]: plt.plot(phd, service, 'bo', label="Ph.D. Vs
services", lw=10)
         plt.grid()
         plt.legend()
         plt.xlabel('Ph.D')
         plt.ylabel('service')
         plt.title('Ph.D./salary\nDistribution')
         plt.yscale('log')
```

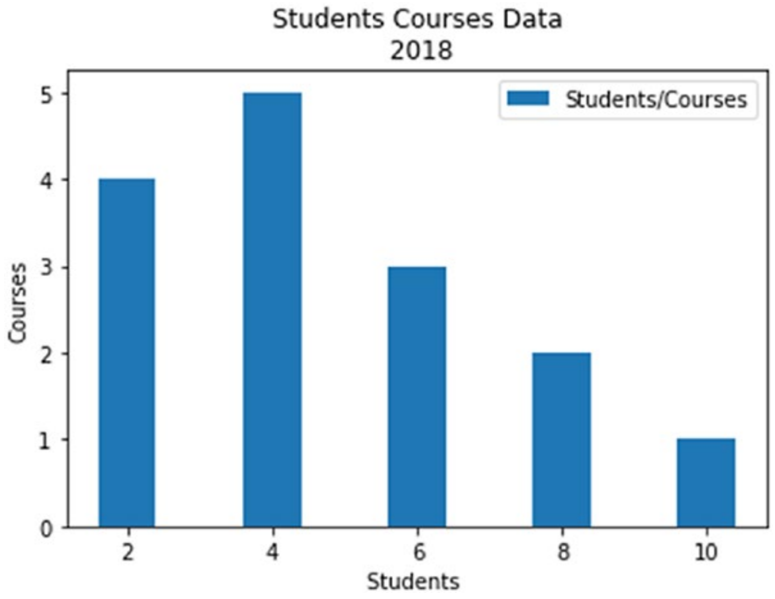


Bar Chart

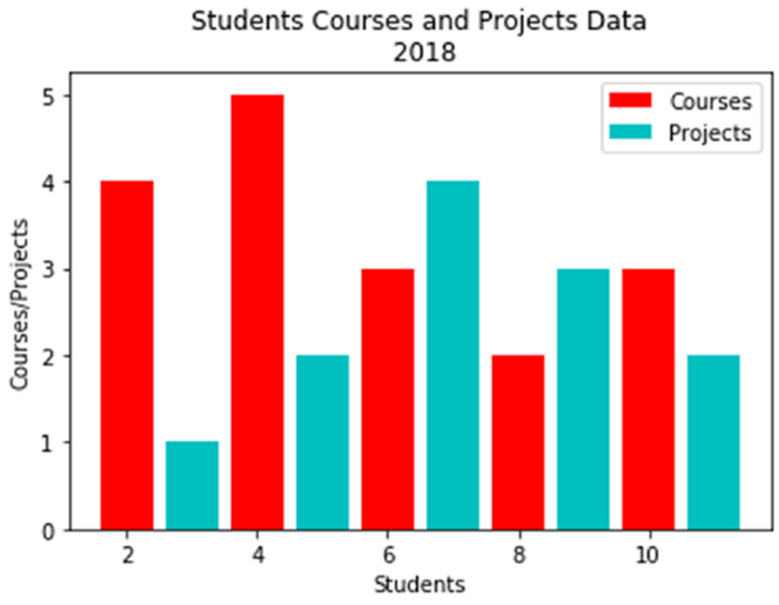
Listing 7-17 shows how to create a bar chart to present students registered for courses; there are two students who are registered for four courses.

Listing 7-17. Matplotlib Bar Chart Plotting

```
In [3]: Students = [2,4,6,8,10]
        Courses = [4,5,3,2,1]
        plt.bar(Students,Courses, label="Students/Courses")
        plt.xlabel('Students ')
        plt.ylabel('Courses')
        plt.title('Students Courses Data\n 2018')
        plt.legend()
        plt.show()
```



```
In [4]: Students = [2,4,6,8,10]
        Courses = [4,5,3,2,3]
        stds = [3,5,7,9,11]
        Projects = [1,2,4,3,2]
        plt.bar(Students, Courses, label="Courses", color='r')
        plt.bar(stds, Projects, label="Projects", color='c')
        plt.xlabel('Students')
        plt.ylabel('Courses/Projects')
        plt.title('Students Courses and Projects Data\n 2018')
        plt.legend()
        plt.show()
```

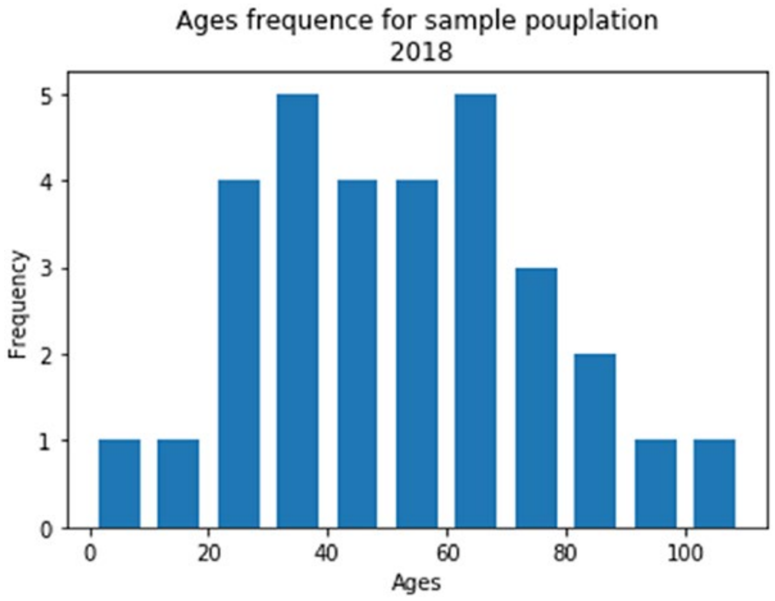


Histogram Plot

Listing 7-18 shows how to create a histogram showing age frequencies; most people in the data set are between 30 and 40. In addition, you can create a histogram of the years of service and the number of PhDs.

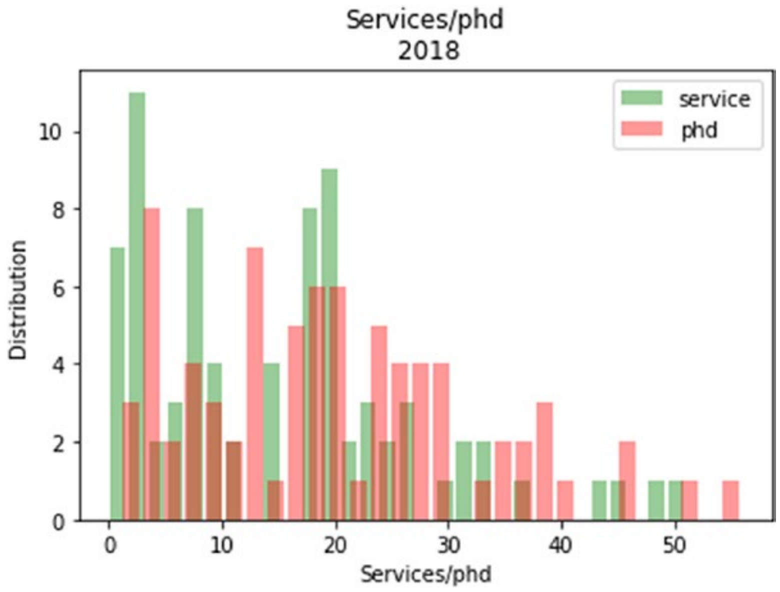
Listing 7-18. Matplotlib Histogram Plotting

```
In [5]: Ages = [22.5, 10, 55, 8, 62, 45, 21, 34, 42, 45, 99,
               75, 82,
                77, 55, 43, 66, 66, 78, 89, 101, 34, 65, 56,
                25, 34,
                52, 25, 63, 37, 32]
        binsx = [0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110]
        plt.hist(Ages, bins=binsx, histtype='bar', rwidth=0.7)
        plt.xlabel('Ages')
        plt.ylabel('Frequency')
        plt.title('Ages frequency for sample pouplation\n 2018')
        plt.show()
```

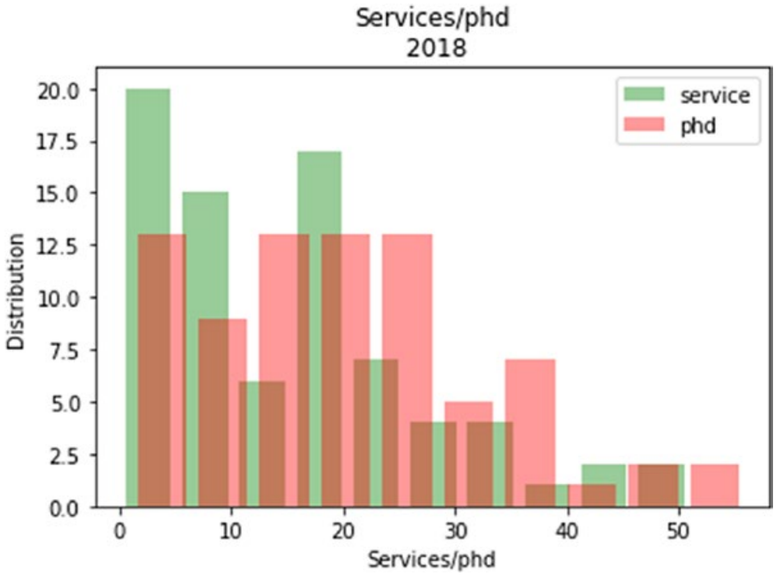


```
In [18]: plt.hist(service, bins=30, alpha=0.4, rwidth=0.8,
         color='green', label='service')
         plt.hist(phd, bins=30, alpha=0.4, rwidth=0.8,
         color='red', label='phd')
         plt.xlabel('Services/phd')
         plt.ylabel('Distribution')
         plt.title('Services/phd\n 2018')
         plt.legend(loc='upper right')
         plt.show()
```

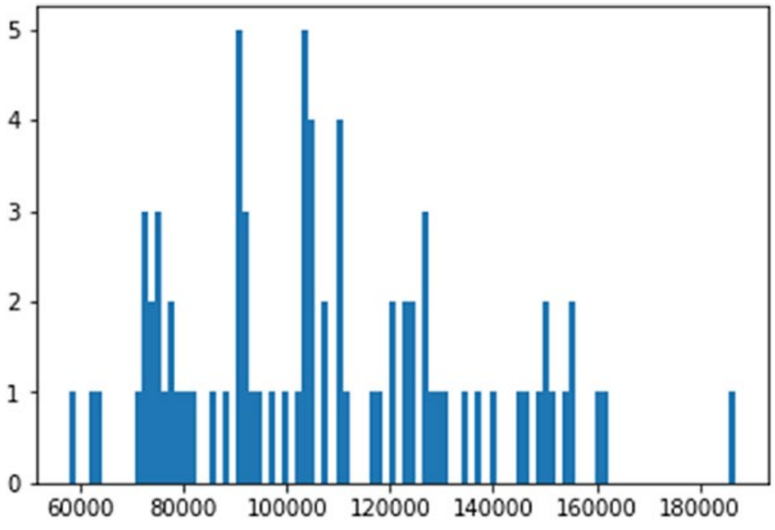
Visualize service years since Ph.D. had attained.



```
In [19]: plt.hist(service, bins=10, alpha=0.4, rwidth=0.8,
         color='green', label='service')
         plt.hist(phd, bins=10, alpha=0.4, rwidth=0.8,
         color='red', label='phd')
         plt.xlabel('Services/phd')
         plt.ylabel('Distribution')
         plt.title('Services/phd\n 2018')
         plt.legend(loc='upper right')
         plt.show()
```



```
In [21]: plt.hist(salary, bins=100)
         plt.show()
```

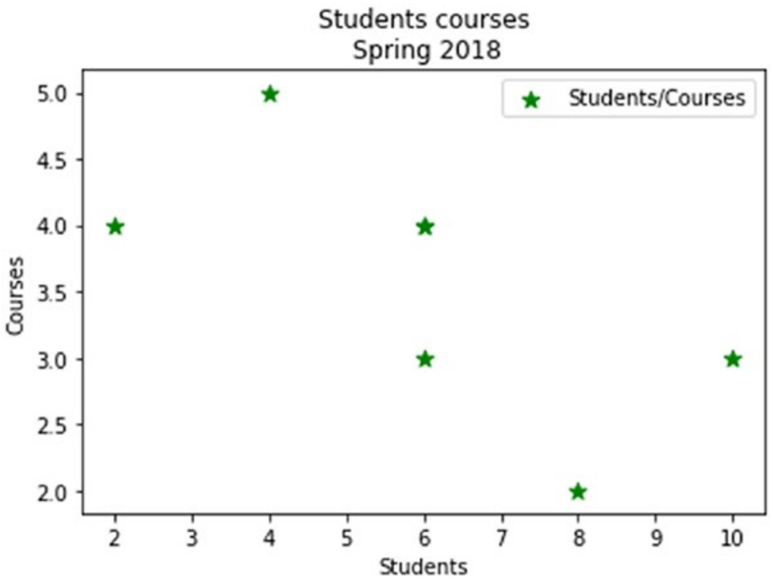


Scatter Plot

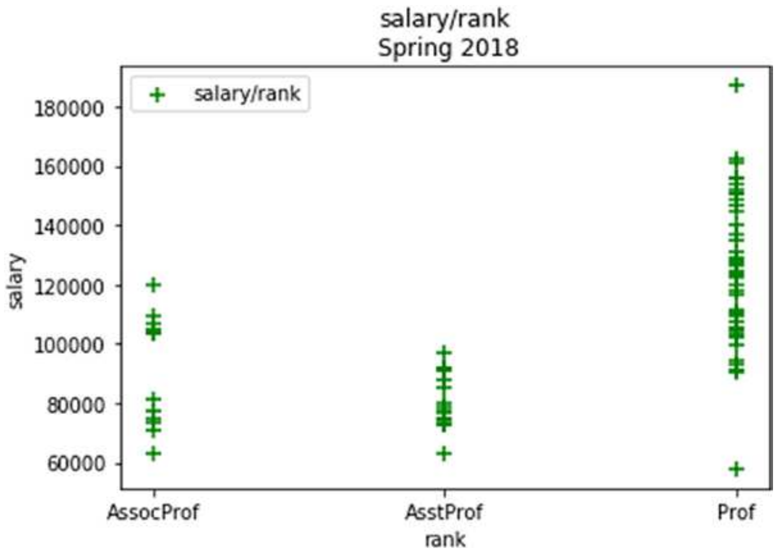
Listing 7-19 shows how to create a scatter plot to present students registered for courses, where four students are registered for five courses.

Listing 7-19. Matplotlib Scatter Plot

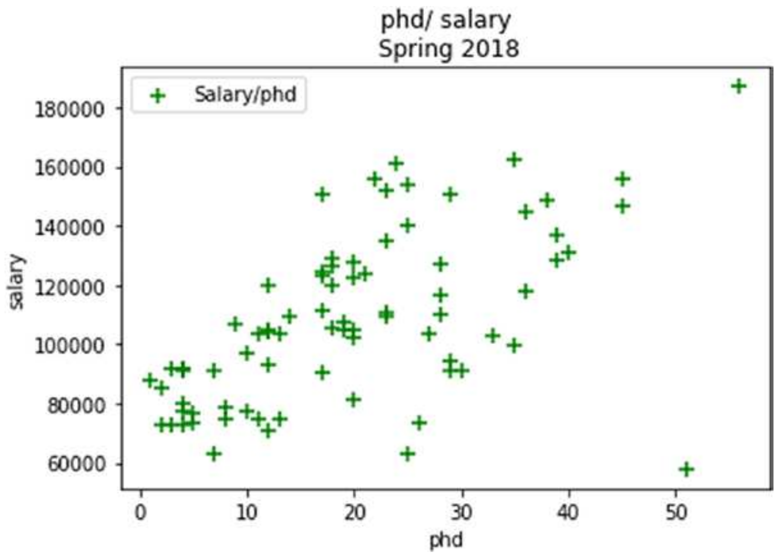
```
In [7]: Students = [2,4,6,8,6,10, 6] Courses = [4,5,3,2,4, 3, 4]
        plt.scatter(Students,Courses, label='Students/Courses',
        color='green', marker='*', s=75 )
        plt.xlabel('Students')
        plt.ylabel('Courses')
        plt.title('Students courses\n Spring 2018')
        plt.legend()
        plt.show()
```



```
In [16]: plt.scatter(rank,salary, label='salary/rank',
         color='g', marker='+', s=50 )
         plt.xlabel('rank') plt.ylabel('salary')
         plt.title('salary/rank\n Spring 2018')
         plt.legend() plt.show()
```



```
In [20]: plt.scatter(phd,salary, label='Salary/phd', color='g',
         marker='+', s=80 )
         plt.xlabel('phd') plt.ylabel('salary')
         plt.title('phd/ salary\n Spring 2018')
         plt.legend() plt.show()
```

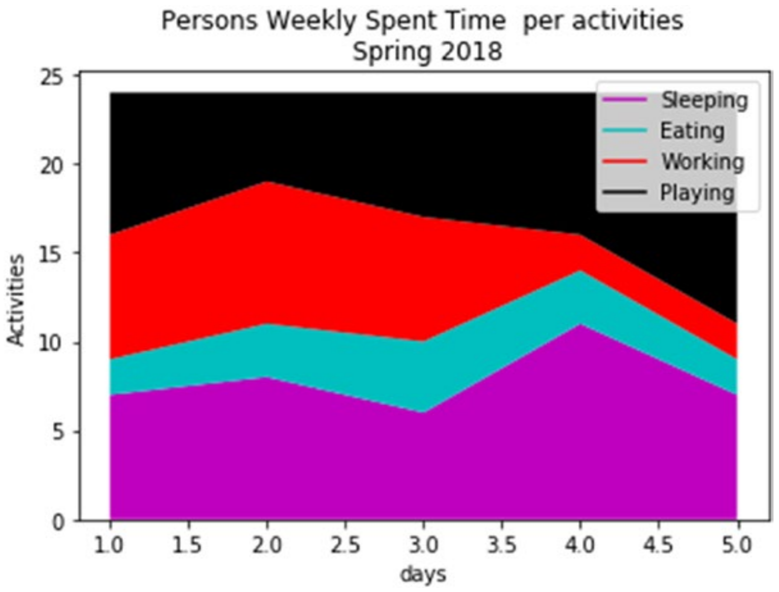


Stack Plot

Stack plots present the frequency of every activity, such as the frequency of sleeping, eating, working, and playing per day (see Listing 7-20). In this data set, on day 2, a person spent eight hours sleeping, three hours in eating, eight hours working, and five hours playing.

Listing 7-20. Persons Weekly Spent Time per activities using Matplotlib Stack Plot

```
In [9]: days = [1,2,3,4,5]
        sleeping = [7,8,6,11,7]
        eating = [2,3,4,3,2]
        working = [7,8,7,2,2]
        playing = [8,5,7,8,13]
        plt.plot([],[], color='m', label='Sleeping')
        plt.plot([],[], color='c', label='Eating')
        plt.plot([],[], color='r', label='Working')
plt.plot([],[], color='k', label='Playing')
plt.stackplot(days, sleeping, eating, working ,
playing, colors=['m','c', 'r', 'k'])
plt.xlabel('days')
plt.ylabel('Activities')
plt.title('Persons Weekly Spent Time per activities\n
Spring 2018')
plt.legend()
plt.show()
```

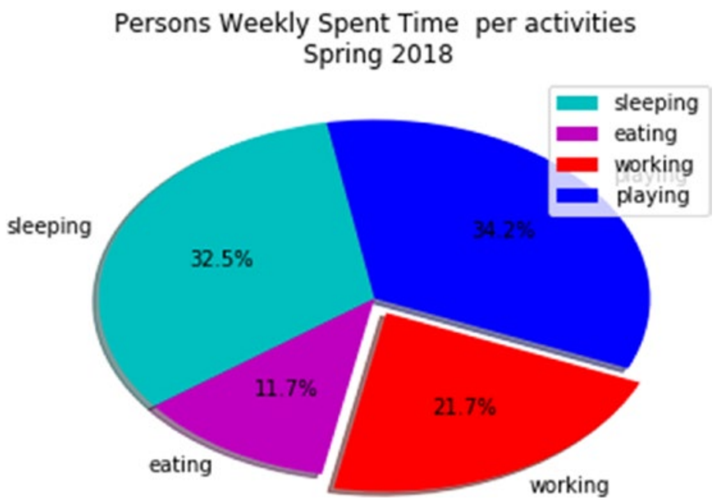


Pie Chart

In Listing 7-21, you are using the explode attribute to slice out a specific activity. After that, you can add the gender and title to the pie chart.

Listing 7-21. Persons Weekly Spent Time per activities using Matplotlib Pie Chart

```
In [10]: days = [1,2,3,4,5]
         sleeping = [7,8,6,11,7]
         eating = [2,3,4,3,2]
         working = [7,8,7,2,2]
         playing = [8,5,7,8,13]
         slices = [39,14,26,41]
         activities = ['sleeping', 'eating', 'working',
         'playing']
         cols = ['c','m','r', 'b','g']

         plt.pie(slices,
             labels= activities,
             colors= cols,
             startangle=100,
                 shadow=True,
         explode = (0.0,0.0,0.09,0),
         autopct = '%1.1f%%')
         plt.title('Persons Weekly Spent Time per activities\n
         Spring 2018')
         plt.legend()
         plt.show()
```



Summary

This chapter covered how to plot data from different collection structures. You learned the following:

– How to directly plot data from a series, data frame, or panel using Python plotting tools such as line plots, bar plots, pie charts, box plots, histogram plots, and scatter plots

– How to implement the Seaborn plotting system using strip plotting, box plotting, swarm plotting, and joint plotting

– How to implement Matplotlib plotting using line plots, bar charts, histogram plots, scatter plots, stack plots, and pie charts

The next chapter will cover the techniques you've studied in this book via two different case studies; it will make recommendations, and much more.

Exercises and Answers

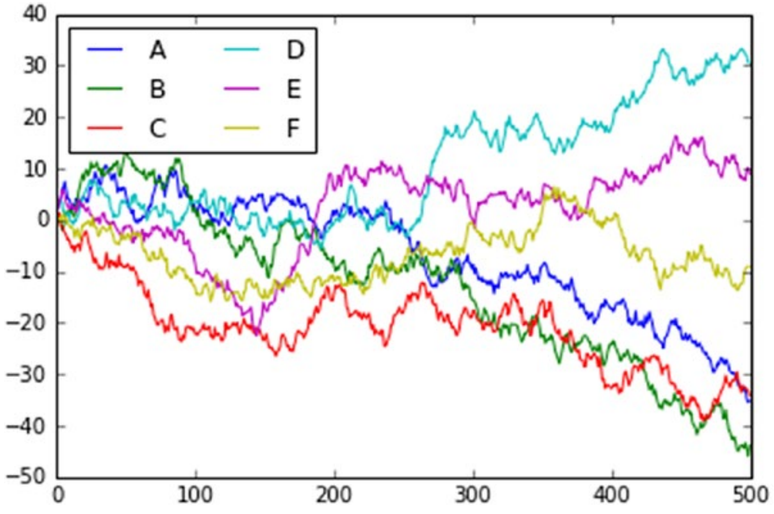
1. Create 500 random temperature readings for six cities over a season and then plot the generated data using Matplotlib.

Answer:

See Listing 7-22.

Listing 7-22. Plotting the Temperature Data of Six Cities

```
In [4]: import matplotlib.pyplot as plt
        plt.style.use('classic')
        %matplotlib inline
        import numpy as np
        import pandas as pd

In [30]: # Create temperature data
        rng = np.random.RandomState(0)
        season1 = np.cumsum(rng.randn(500, 6), 0)

In [32]: # Plot the data with Matplotlib defaults
        plt.plot(season1)
        plt.legend('ABCDEF', ncol=2, loc='upper left');
```



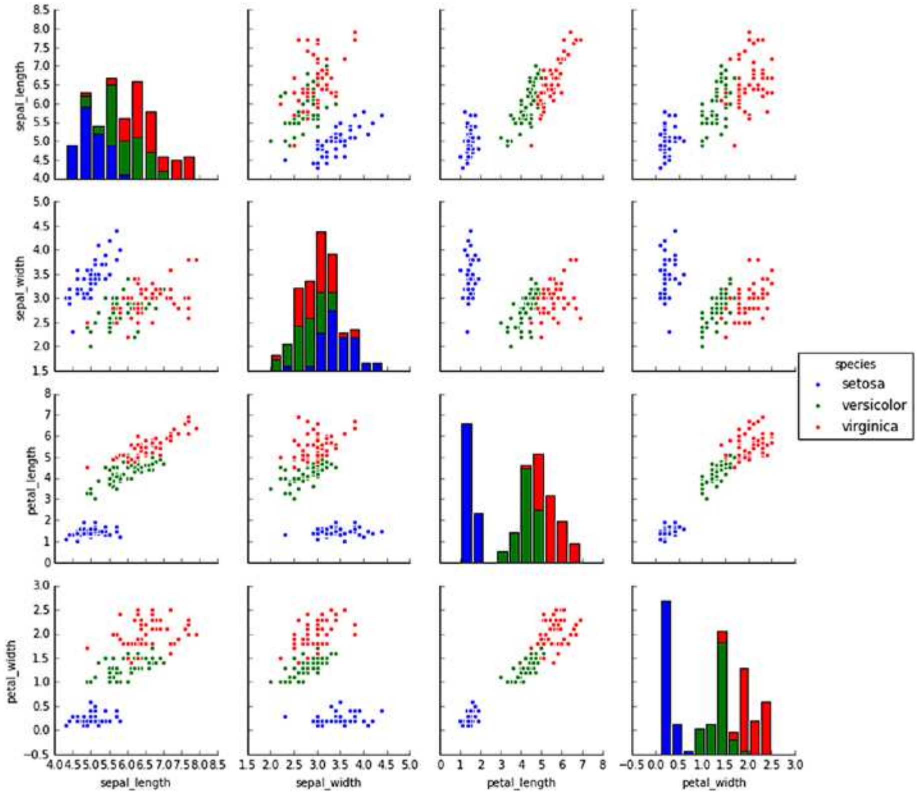
2. Load the well-known Iris data set, which lists measurements of petals and sepals of three iris species. Then plot the correlations between each pair using the `.pairplot()` method.

Answer:

See Listing 7-23.

Listing 7-23. Pair Correlations

```
In [33]: import seaborn as sns
         iris = sns.load_dataset("iris")
         iris.head()
         sns.pairplot(iris, hue='species', size=2.5);
```



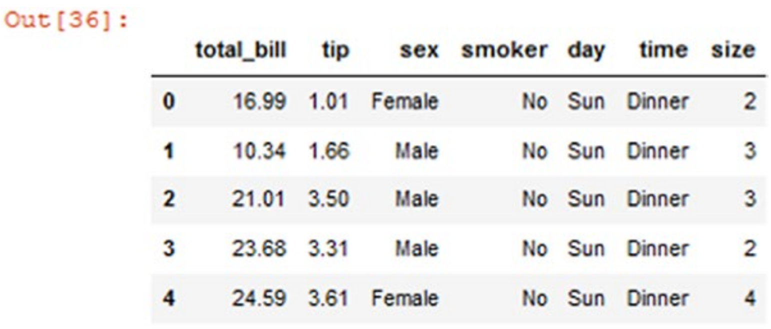
3. Load the well-known Tips data set, which shows the number of tips received by restaurant staff based on various indicator data; then plot the percentage of tips per bill according to staff gender.

Answer:

See Listing 7-24.

Listing 7-24. First five records in the Tips dataset

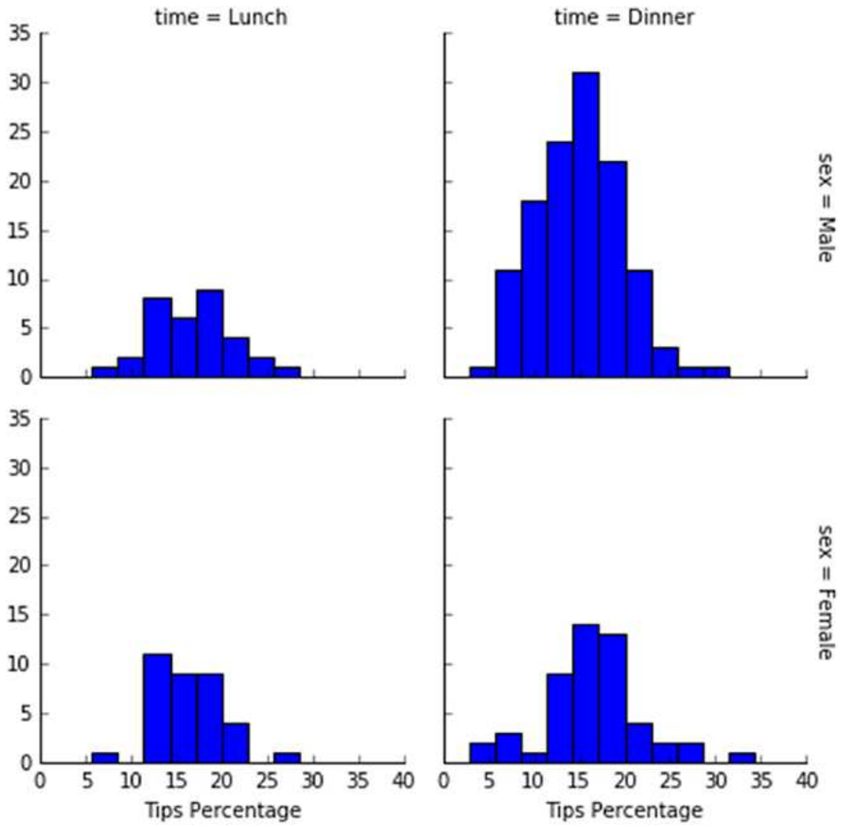
```
In [36]: import seaborn as sns
         tips = sns.load_dataset('tips')
         tips.head()
```



```
Out[36]:
```

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4

```
In [37]: tips['Tips Percentage'] = 100 * tips['tip'] /
tips['total_bill']
         grid = sns.FacetGrid(tips, row="sex", col="time",
         margin_titles=True)
         grid.map(plt.hist, "Tips Percentage", bins=np.
         linspace(0, 40, 15));
```



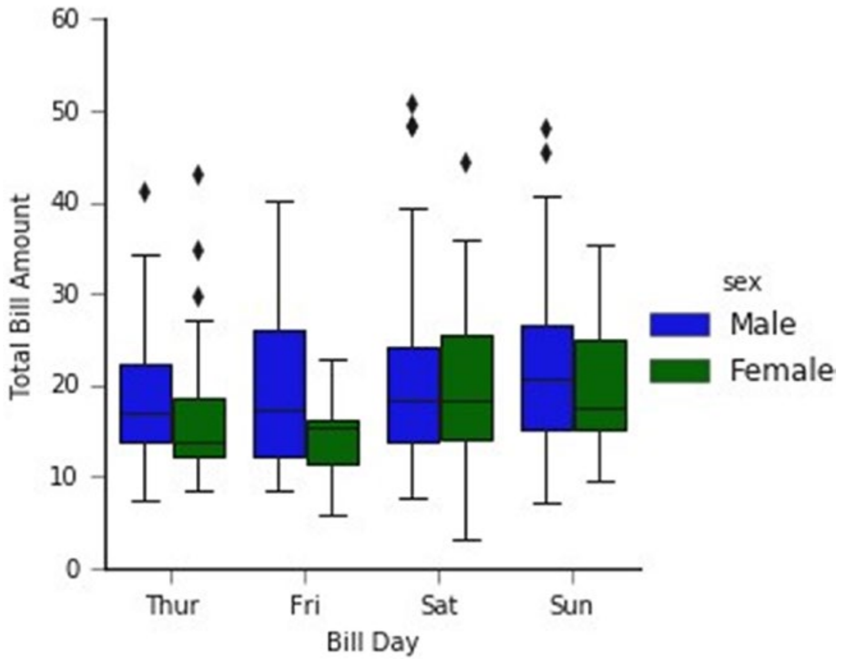
4. Load the well-known Tips data set, which shows the number of tips received by restaurant staff based on various indicator data; then implement the factor plots to visualize the total bill per day according to staff gender.

Answer:

See Listing 7-25.

Listing 7-25. Implementing Factor Plotting

```
In [39]: import seaborn as sns
         tips = sns.load_dataset('tips')
         with sns.axes_style(style='ticks'):
         g = sns.factorplot("day", "total_bill",
         "sex", data=tips, kind="box")
         g.set_axis_labels("Bill Day", "Total Bill Amount")
```



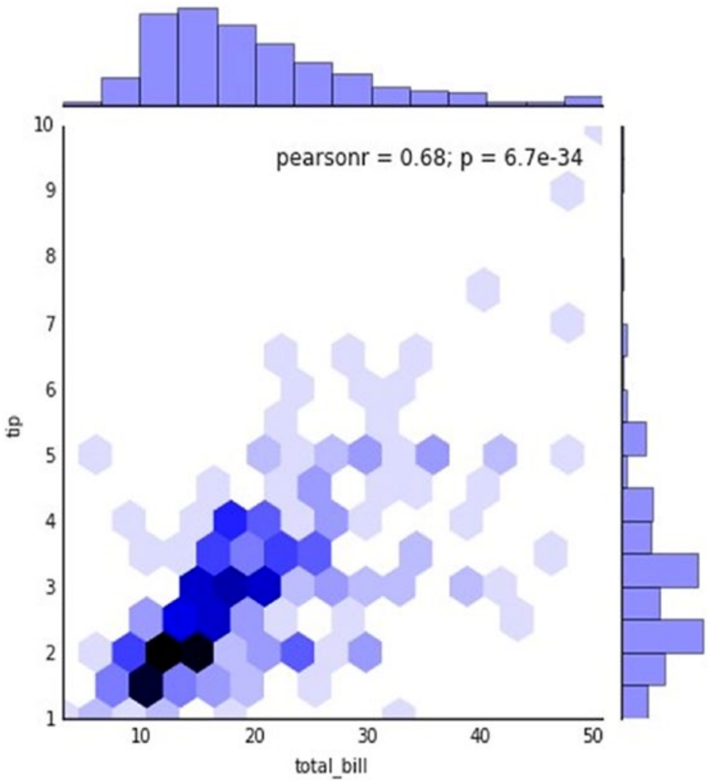
5. Reimplement the previous exercise using the Seaborn joint plot distributions.

Answer:

See Listing 7-26.

Listing 7-26. Implementing Joint Plot Distributions

```
In [43]: import seaborn as sns
         tips = sns.load_dataset('tips')
         with sns.axes_style('white'):
         sns.jointplot( "total_bill", "tip",
         data=tips, kind='hex')
```


pearsonr = 0.68; p = 6.7e-34
tip
total_bill