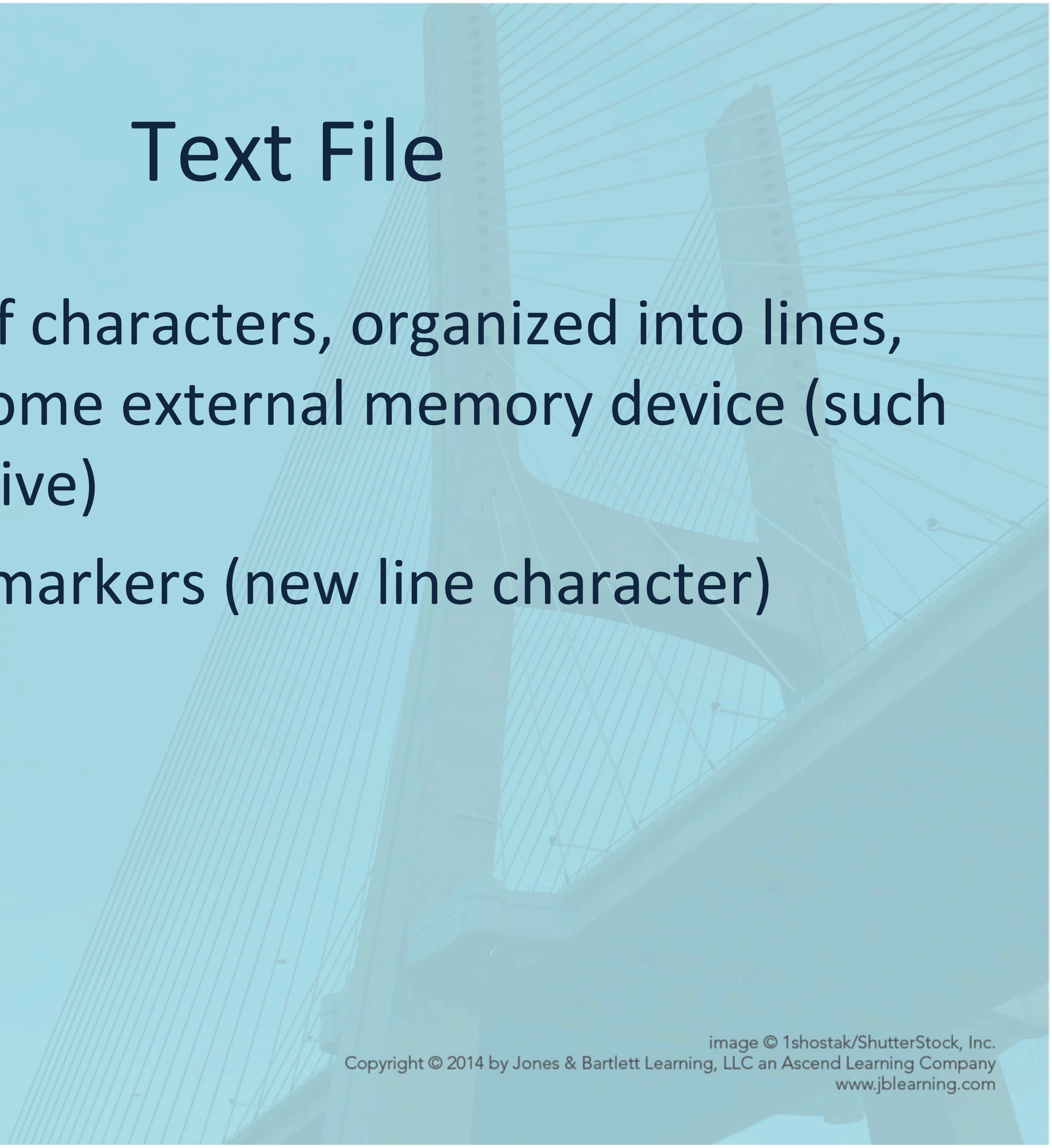
Objectives

- To use text files to store large data sets
- To access online data sources
- To introduce the while loop
- To introduce Boolean expressions



Text File

- Sequence of characters, organized into lines, stored on some external memory device (such as a hard drive)
- End of line markers (new line character)

Python

- Open a file
  - open
- Read from a file
  - read
  - readline
  - readlines
- Write to a file
  - Write

Python

- Iterate through the file line by line
- Use split method to process the line

Listing 5.1

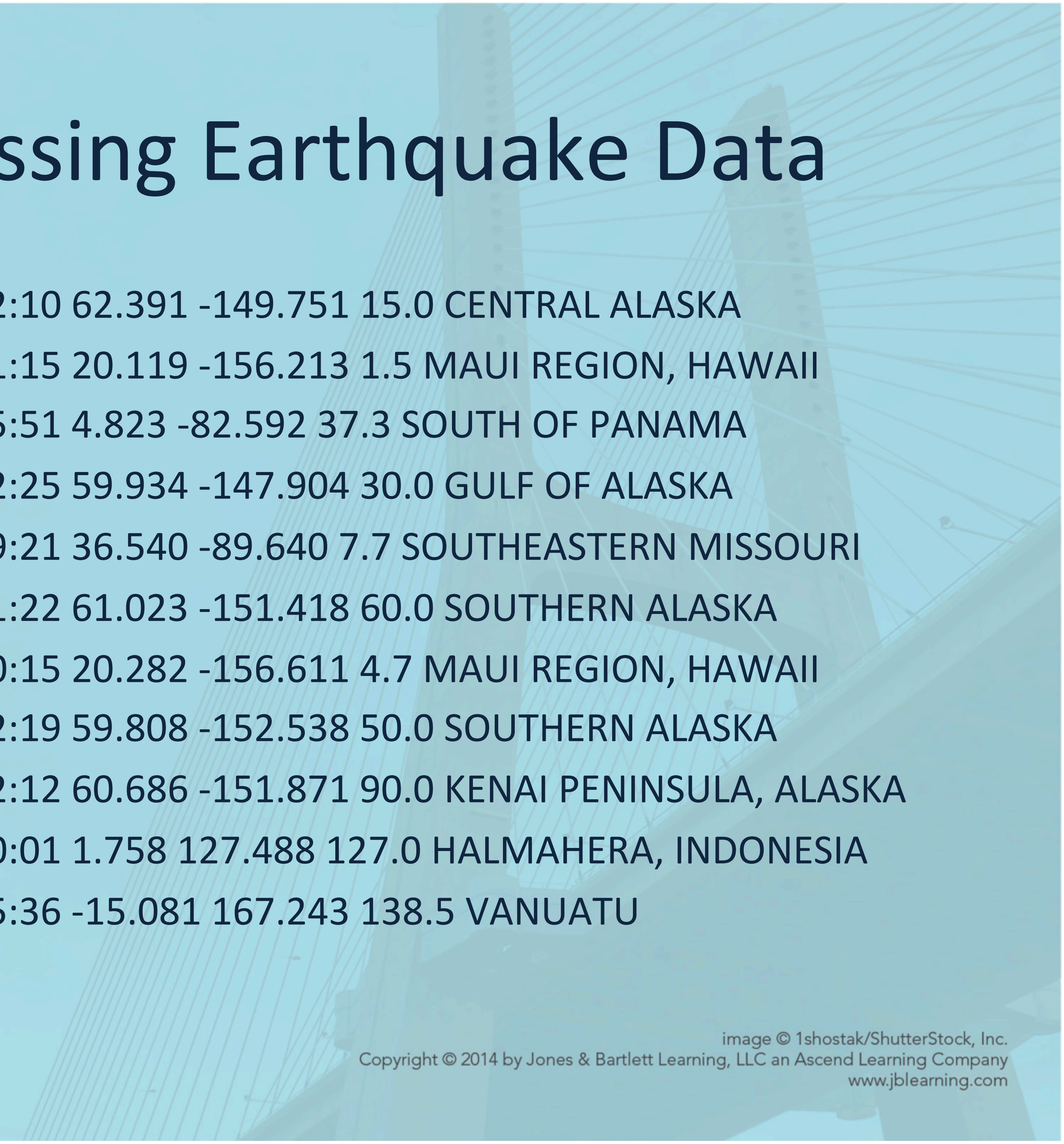
```
rainfile = open("rainfall.txt","r")

for aline in rainfile:
    values = aline.split()
    print(values[0], "had",values[1],"inches of rain.")

rainfile.close()
```

Listing 5.2

```
rainfile = open("rainfall.txt","r")
outfile = open("rainfallInCM.txt","w")

for aline in rainfile:
    values = aline.split()

    inches = float(values[1])
    cm = 2.54 * inches

    outfile.write(values[0]+" "+str(cm)+"\n")

rainfile.close()
outfile.close()
```

Processing Earthquake Data

```
2.8 2006/10/19 02:02:10 62.391 -149.751 15.0 CENTRAL ALASKA
2.5 2006/10/19 00:31:15 20.119 -156.213 1.5 MAUI REGION, HAWAII
5.0 2006/10/18 21:15:51 4.823 -82.592 37.3 SOUTH OF PANAMA
2.6 2006/10/18 21:12:25 59.934 -147.904 30.0 GULF OF ALASKA
3.4 2006/10/18 20:59:21 36.540 -89.640 7.7 SOUTHEASTERN MISSOURI
2.7 2006/10/18 20:11:22 61.023 -151.418 60.0 SOUTHERN ALASKA
3.1 2006/10/18 16:40:15 20.282 -156.611 4.7 MAUI REGION, HAWAII
2.7 2006/10/18 14:12:19 59.808 -152.538 50.0 SOUTHERN ALASKA
2.8 2006/10/18 14:02:12 60.686 -151.871 90.0 KENAI PENINSULA, ALASKA
4.9 2006/10/18 12:10:01 1.758 127.488 127.0 HALMAHERA, INDONESIA
6.2 2006/10/18 10:45:36 -15.081 167.243 138.5 VANUATU
```



Listing 5.3

```
def makeMagnitudeList():
    quakefile = open("earthquakes.txt","r")

    maglist = [ ]
    for aline in quakefile:
        vlist = aline.split()
        maglist.append(float(vlist[0]))
    return maglist
```

Processing Data from the Internet

- urllib.request module
- Internet behaves just like a text file

Listing 5.4

```
import urllib.request

def countHead(url):
    page = urllib.request.urlopen(url)
    numHeadLines = 0

    line = page.readline().decode('utf-8')
    while '<head>' not in line:
        line = page.readline().decode('utf-8')

    line = page.readline().decode('utf-8')
    while '</head>' not in line:
        numHeadLines = numHeadLines + 1
        line = page.readline().decode('utf-8')

    line = page.readline().decode('utf-8')
    while "<body>" not in line:
        line = page.readline().decode('utf-8')

    line = page.readline().decode('utf-8')
    while line != "" and "</body>" not in line:
        print (line[:-1])
        line = page.readline().decode('utf-8')

    print ("number of lines in header = ", numHeadLines)

    page.close()
```

Correlating Data from Internet

- Using real data streams
- Stock market data

Listing 5.5

```
def correlation(xlist, ylist):
    xbar = mean(xlist)
    ybar = mean(ylist)
    xstd = standardDev(xlist)
    ystd = standardDev(ylist)
    num = 0.0
    for i in range(len(xlist)):
        num = num + (xlist[i]-xbar) * (ylist[i]-ybar)
    corr = num / ((len(xlist)-1) * xstd * ystd)
    return corr
```

Listing 5.6

```
def stockCorrelate(ticker1, ticker2):
    url1 = urllib.request.urlopen('http://ichart.yahoo.com/table.csv?s=%s'%ticker1)
    url2 = urllib.request.urlopen('http://ichart.yahoo.com/table.csv?s=%s'%ticker2)
    t1Data = url1.readlines()
    t2Data = url2.readlines()
    t1Data = [line[0:-1].decode("utf-8").split(',') for line in t1Data[1:] ]
    t2Data = [line[0:-1].decode("utf-8").split(',') for line in t2Data[1:] ]
    t1Close = []
    t2Close = []
    for i in range(min(len(t1Data), len(t2Data))):
        if t1Data[i][0] == t2Data[i][0]:
            t1Close.append(float(t1Data[i][4]))
            t2Close.append(float(t2Data[i][4]))

    return correlation(t1Close, t2Close)
```