In Exercises 11 through 14, use the file `StatesANC.txt` containing the name, abbreviation, nickname, and capital of each state in the United States. The states are listed in alphabetical order. The first three lines of the file are

```
Alabama,AL,Cotton State,Montgomery
Alaska,AK,The Last Frontier,Juneau
Arizona,AZ,Grand Canyon State,Phoenix
```

11. U.S. States Write a program that places the names of the states into a list box. See Fig. 8.13. ***Hint:*** Use list comprehension.

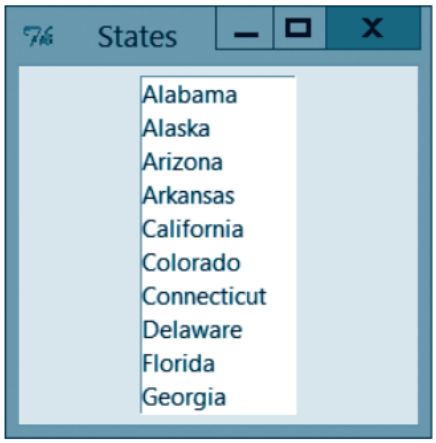


FIGURE 8.13 Outcome of Exercise 11.

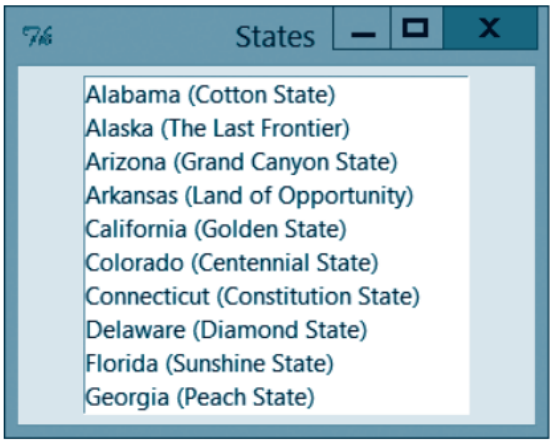


FIGURE 8.14 Outcome of Exercise 12.

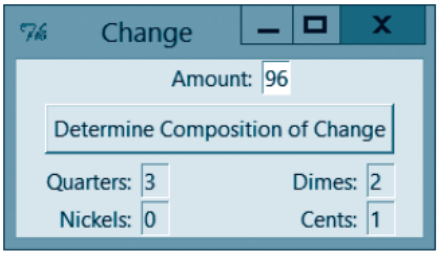
12. U.S. States Modify the program in Exercise 11 so that both the name of the state and its nickname are displayed in the list box. See Fig. 8.14.

```
11. from tkinter import *
    window = Tk()
    window.title("States")
    infile = open("StatesANC.txt", 'r')
    listOfStates = [line.split(',')[0] for line in infile]
    infile.close()
    conOFlstStates = StringVar()
    lstStates = Listbox(window, height=10, width=12,
                        listvariable=conOFlstStates)
    lstStates.grid(padx=75, pady=5)
    conOFlstStates.set(tuple(listOfStates))
    window.mainloop()
```

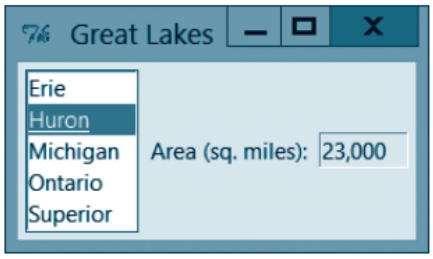
```
12. from tkinter import *
    window = Tk()
    window.title("States")
    infile = open("StatesANC.txt", 'r')
    listOfStates = [line.split(',')[0] + " (" + line.split(',')[2] + ")"\
                    for line in infile]
    infile.close()
    conOFlstStates = StringVar()
    lstStates = Listbox(window, height=10,
                        width=30, listvariable=conOFlstStates)
    lstStates.grid(row=1, column=0, padx=40, pady=5)
    conOFlstStates.set(tuple(listOfStates))
    window.mainloop()
```

Each screen capture below shows the output of a complete program. In Exercises 9 through 22, write just the part of the program that displays the interface. When the half-finished program is run, all widgets should appear as shown, but no text should appear inside the Entry widgets or list boxes.

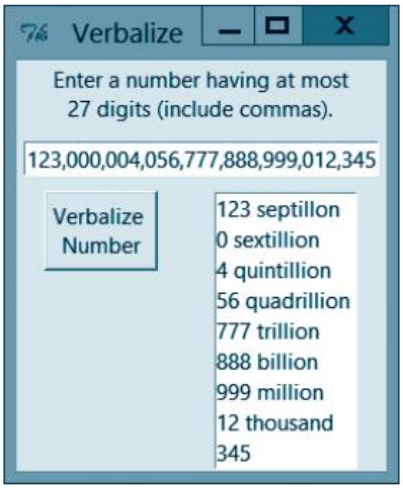
12.



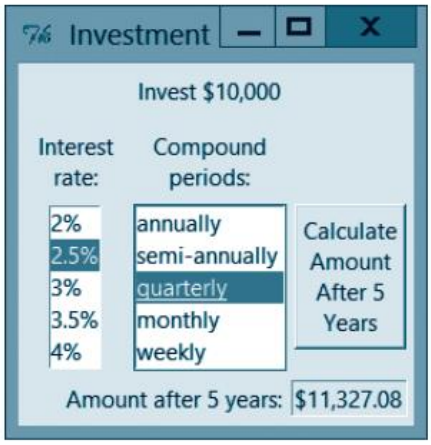
16.



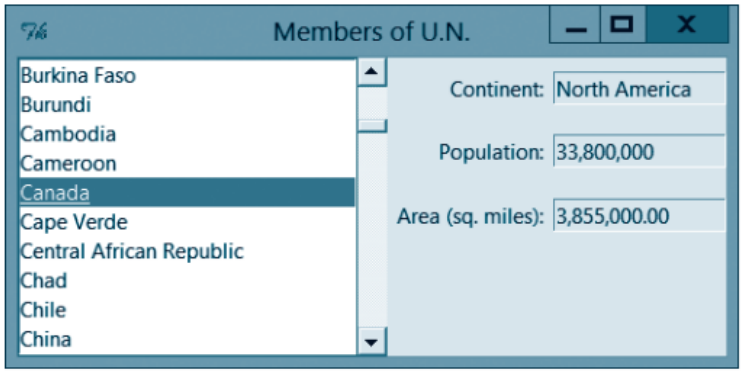
17.



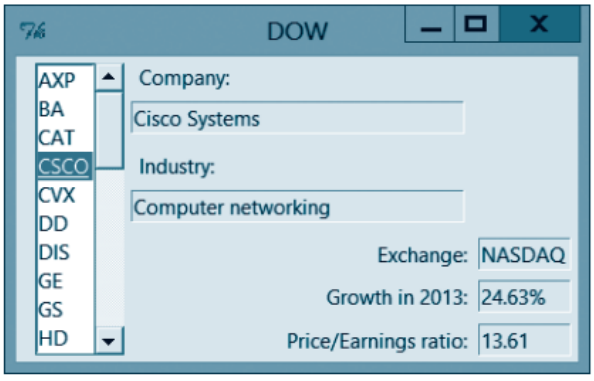
18.



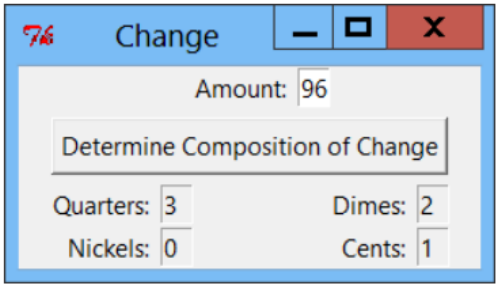
21.



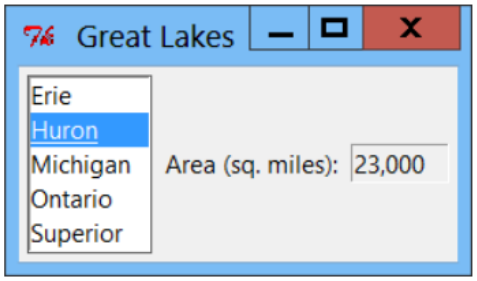
22.



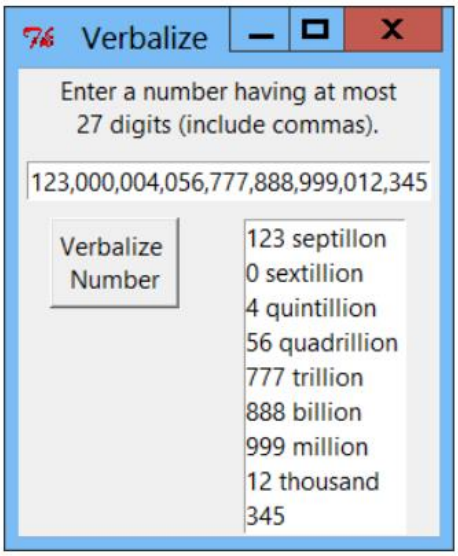
12.
```
from tkinter import *
window = Tk()
window.title("Change")
caption = "Amount: "
Label(window, text=caption).grid(row=0, column=1, sticky=E)
entAmount = Entry(window, width=2)
entAmount.grid(row=0, column=2, sticky=W)
caption = "Determine Composition of Change"
btnDetermine = Button(window, text=caption)
btnDetermine.grid(row=1, column=0, columnspan=4, padx=20, pady=5)
Label(window, text="Quarters: ").grid(row=2, column=0, sticky=E)
Label(window, text="Nickels: ").grid(row=3, column=0, sticky=E)
Label(window, text="Dimes: ").grid(row=2, column=2, sticky=E)
Label(window, text="Cents: ").grid(row=3, column=2, sticky=E)
entQuarters = Entry(window, width=2, state="readonly")
entQuarters.grid(row=2, column=1, sticky=W)
entNickels = Entry(window, width=2, state="readonly")
entNickels.grid(row=3, column=1, sticky=W)
entDimes = Entry(window, width=2, state="readonly")
entDimes.grid(row=2, column=3, sticky=W)
entCents = Entry(window, width=2, state="readonly")
entCents.grid(row=3, column=3, sticky=W)
window.mainloop()
```



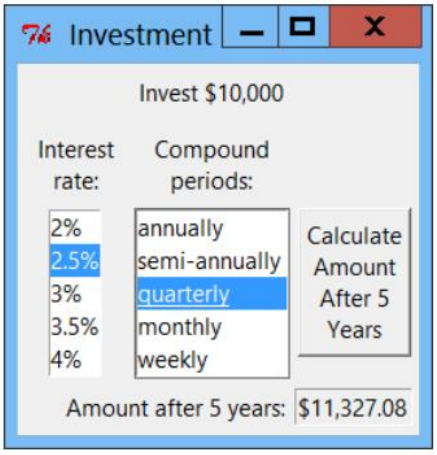
16.
```
from tkinter import *
import pickle
window = Tk()
window.title("Great Lakes")
global lakesDict
lstLakes = Listbox(window, height=5, width=9)
lstLakes.grid(row=0, column=0, padx=5, pady=5, rowspan=5, sticky=NSEW)
lstLakes.bind("<<ListboxSelect>>")
Label(window, text="Area (sq. miles):").grid(row=2, column=1, sticky=E)
entArea = Entry(window, width=7, state="readonly")
entArea.grid(row=2, column=2, padx=5)
window.mainloop()
```



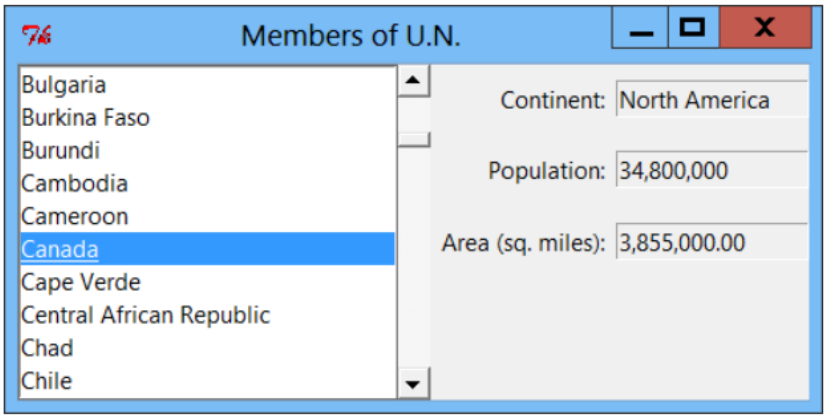
17.
```
from tkinter import *
window = Tk()
window.title("Verbalize")
instruction = "Enter a number having at most\n" + \
              "27 digits (include commas)."
Label(window, text=instruction).grid(row=0, column=0,
      columnspan=2, padx=15)
entNum = Entry(window, width=27)
entNum.grid(row=1, column=0, columnspan=2, pady=5)
btnVerbalize = Button(window, text="Verbalize\nNumber")
btnVerbalize.grid(row=2, column=0, sticky=N)
lstEnglish = Listbox(window, height=9, width=14)
lstEnglish.grid(row=2, column=1)
window.mainloop()
```



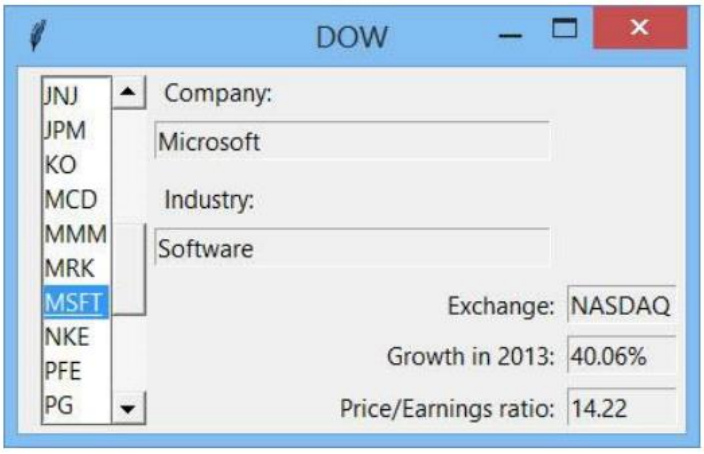
18.
```
from tkinter import *
window = Tk()
window.title("Investment")
Label(window, text="Invest $10,000").grid(row=0, column=1, pady=5)
Label(window, text="Interest\nrate:").grid(row=1, column=0, padx=10, pady=5)
Label(window, text="Compound\nperiods:").grid(row=1, column=1,
                                              padx=10, pady=5)
btnCalculate = Button(window, text="Calculate\nAmount\nAfter 5\nYears")
btnCalculate.grid(row=3, column=2, padx=5, sticky=N)
lstRates = Listbox(window, height=5, width=4)
lstRates.grid(row=3, column=0)
lstPeriods = Listbox(window, height=5, width=12)
lstPeriods.grid(row=3, column=1)
Label(window, text="Amount after 5 years:").grid(row=4, column=0,
                                        pady=5, columnspan=2, sticky=E)
entAmount = Entry(window,  width=9, state="readonly")
entAmount.grid(row=4, column=2, padx = 3, pady=5, sticky=W)
window.mainloop()
```



21.
```
from tkinter import *
import pickle
window = Tk()
window.title("Members of U.N.")
yscroll = Scrollbar(window, orient=VERTICAL)
yscroll.grid(row=0, column=1, rowspan=7, sticky=NS)
lstNations = Listbox(window, height=10, width=30, yscrollcommand=yscroll.set)
lstNations.grid(row=0, column=0, rowspan=7, sticky=NSEW)
yscroll["command"] = lstNations.yview
Label(window, text="Continent:").grid(row=0, column=3, padx=4, sticky=E)
Label(window, text="Population:").grid(row=1, column=3, padx=4, sticky=E)
Label(window, text="Area (sq. miles):").grid(row=2, column=3,
                                             padx=4,sticky=E)
entContinent = Entry(window, width=15, state="readonly")
entContinent.grid(row=0, column=4, sticky=W)
entPopulation = Entry(window, width=15, state="readonly")
entPopulation.grid(row=1, column=4,)
entArea = Entry(window, width=15, state="readonly")
entArea.grid(row=2, column=4)
window.mainloop()
```



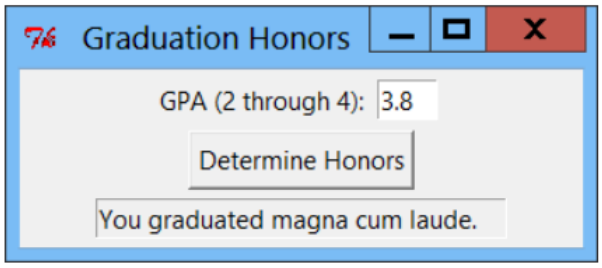
```
22. from tkinter import *
    window = Tk()
    window.title("DOW")
    Label(window, text="", width=1).grid(row=0, column=0)
    Label(window, text="  Company:").grid(row=0, column=3,  sticky=W)
    Label(window, text="  Industry:").grid(row=3, column=3, sticky=W)
    Label(window, text="Exchange:").grid(row=6, column=4, sticky=E)
    Label(window, text="Growth in 2013:").grid(row=7, column=4, sticky=E)
    Label(window, text="Price/Earnings ratio:").grid(row=8, column=4, sticky=E)
    yscroll = Scrollbar(window, orient=VERTICAL)
    yscroll.grid(row=0, column=2, rowspan=9,pady=5, sticky=NS)
    lstSymbols = Listbox(window, width=5, yscrollcommand=yscroll.set)
    lstSymbols.grid(row=0, column=1, rowspan=9, pady=5, sticky=E)
    lstSymbols.bind("<<ListboxSelect>>")
    entCompany = Entry(window, state="readonly", width=30)
    entCompany.grid(row=1, column=3, columnspan=2, padx=5, sticky=W)
    entIndustry = Entry(window, state="readonly", width=30)
    entIndustry.grid(row=4, column=3, columnspan=2, padx=5, sticky=W)
    entExchange = Entry(window, width=8, state="readonly")
    entExchange.grid(row=6, column=5, padx=5, sticky=W)
    entGrowth = Entry(window, width=8, state="readonly")
    entGrowth.grid(row=7, column=5, padx=5, sticky=W)
    entPE = Entry(window, width=8, state="readonly")
    entPE.grid(row=8, column=5, padx=5, sticky=W)
    yscroll["command"] = lstSymbols.yview
    window.mainloop()
```



2. Graduation Honors Write a program that assumes that the user will graduate (that is, has a GPA of 2 or more) and determines if the user will graduate with honors. (*Summa cum laude* requires a GPA of 3.9, *magna cum laude* requires a GPA of 3.6, and *cum laude* requires a GPA of 3.3.) See Fig. 8.33.

FIGURE 8.33 Possible outcome of Exercise 2.

2.
```
from tkinter import *

def honors():
    gpa = float(conOFentGPA.get())
    if gpa >= 3.9:
        honor = " summa cum laude."
    elif gpa >= 3.6:
        honor = " magna cum laude."
    elif gpa >= 3.3:
        honor = " cum laude."
    else:
        honor = "."
    # Display conclusion.
    conOFentHonors.set("You graduated" + honor)

window = Tk()
window.title("Graduation Honors")
caption = "GPA (2 through 4):"
Label(window, text=caption).grid(row=0, column=0, pady=5, sticky=E)
conOFentGPA = StringVar()
entGPA = Entry(window, width=4, textvariable=conOFentGPA)
entGPA.grid(row=0, column=1, padx=5, sticky=W)
btnDisplay = Button(text="Determine Honors", command=honors)
btnDisplay.grid(row=1, column=0, columnspan=2, padx=100)
conOFentHonors = StringVar()
entHonors = Entry(window, state="readonly", width=30,
                          textvariable=conOFentHonors)
entHonors.grid(row=2, column=0, columnspan=2, padx=5, pady=5)
window.mainloop()
```



8. Powerball Powerball numbers are obtained by drawing 5 balls out of a drum containing 59 white balls (numbered 1 through 59) and then drawing 1 ball (the Powerball) out of a drum containing 35 red balls (numbered 1 through 35). Write a program to produce a Powerball drawing. See Fig. 8.39. ***Note:*** The attribute bg has no effect on ReadOnly Entry widgets. Therefore, the first Entry widget should not be designated as ReadOnly.

9. Calculator Write a program that allows the user to specify two numbers and then adds, subtracts, or multiplies them when the user clicks on the appropriate button. See Fig. 8.40 on the next page.

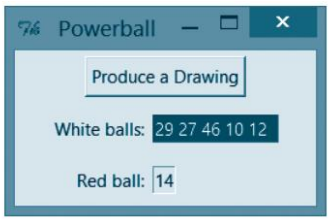


FIGURE 8.39 Possible outcome of Exercise 8.

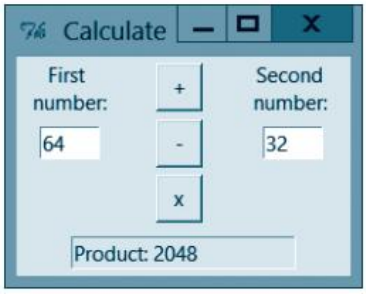
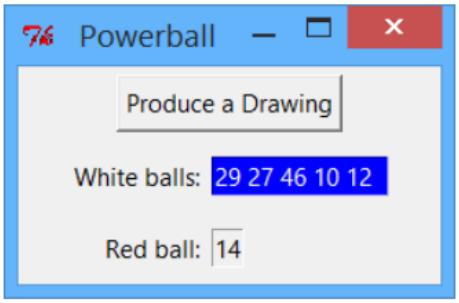


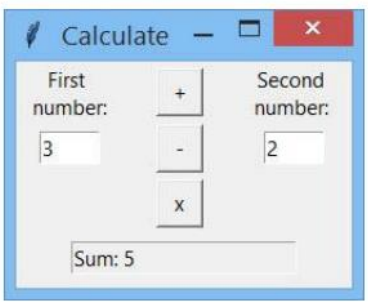
FIGURE 8.40 Possible outcome of Exercise 9.

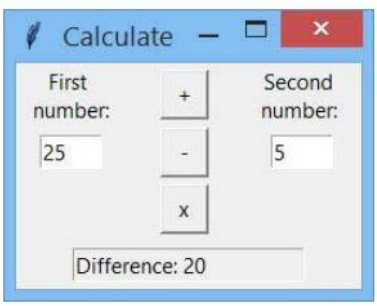
FIGURE 8.41 Possible outcome of Exercise 10.

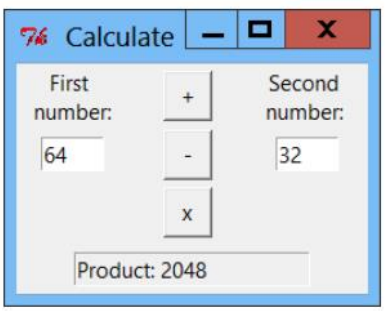
8.
```
from tkinter import *
import random

def drawing():
    conOFentWhiteBalls.set("")
    nums = [x for x in range(1, 60)]
    five = random.sample(nums, 5)
    fiveString = [str(x) for x in five]
    conOFentWhiteBalls.set(" ".join(fiveString))
    num = random.choice(range(1, 36))
    conOFentRedBalls.set(str(num))

window = Tk()
window.title("Powerball")
btnProduce = Button(window, text="Produce a Drawing", command=drawing)
btnProduce.grid(row=0, column=0, columnspan=2, padx=60, pady=5)
Label(window, text="White balls: ").grid(row=1, column=0, sticky=E)
conOFentWhiteBalls = StringVar()
entWhiteBalls = Entry(window, width=13, fg="white", bg="blue",
                      textvariable=conOFentWhiteBalls)
entWhiteBalls.grid(row=1, column=1, pady=10, sticky=W)
Label(window, text="Red ball: ").grid(row=2, column=0, sticky=E)
conOFentRedBalls = StringVar()
entRedBalls = Entry(window, width=2, fg="black", bg="white",
                    textvariable=conOFentRedBalls)
entRedBalls.grid(row=2, column=1, pady=10, sticky=W)
window.mainloop()
```



9.
```
from tkinter import *

def add():
    num1 = eval(conOFentFirst.get())
    num2 = eval(conOFentSecond.get())
    sum = num1 + num2
    conOFentResult.set("Sum: " + str(sum))

def subtract():
    num1 = eval(conOFentFirst.get())
    num2 = eval(conOFentSecond.get())
    difference = num1 - num2
    conOFentResult.set("Difference: " + str(difference))

def multiply():
    num1 = eval(conOFentFirst.get())
    num2 = eval(conOFentSecond.get())
    product = num1 * num2
    conOFentResult.set("Product: " + str(product))

window = Tk()
window.title("Calculate")
Label(window, text="First \nnumber:").grid(row=0, column=0)
Label(window, text="Second \nnumber: ").grid(row=0, column=2)
conOFentFirst = StringVar()
entFirst = Entry(window, width=5, textvariable=conOFentFirst)
entFirst.grid(row=1, column=0)
conOFentSecond = StringVar()
entSecond = Entry(window, width=5, textvariable=conOFentSecond)
entSecond.grid(row=1, column=2)
btnAdd = Button(window, text='+', width=3, command=add)
btnAdd.grid(row=0, column=1, padx=15)
btnSubtract = Button(window, text='-', width=3, command=subtract)
btnSubtract.grid(row=1, column=1, padx=15)
btnMultiply = Button(window, text='x', width=3, command=multiply)
btnMultiply.grid(row=2, column=1, padx=15, pady=5)
conOFentResult = StringVar()
entResult = Entry(window, state="readonly", width=20,
                  textvariable=conOFentResult)
entResult.grid(row=3, column=0, columnspan=3, padx=40, pady=5)
window.mainloop()
```



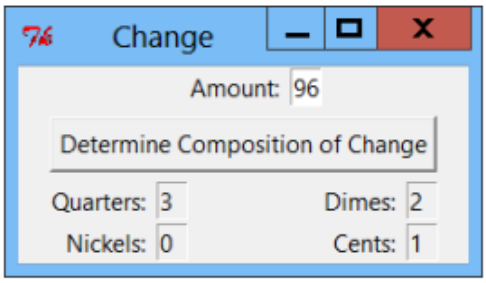




9. (Object-oriented style)

```
from tkinter import *

class Calculate:
    def   init  (self):
        window = Tk()
        window.title("Calculate")
        Label(window, text="First \nnumber:").grid(row=0, column=0)
        Label(window, text="Second \nnumber: ").grid(row=0, column=2)
        self._conOFentFirst = StringVar()
        self.entFirst = Entry(window, width=5,
                              textvariable=self._conOFentFirst)
        self.entFirst.grid(row=1, column=0)
        self._conOFentSecond = StringVar()
        self.entSecond = Entry(window, width=5,
                               textvariable=self._conOFentSecond)
        self.entSecond.grid(row=1, column=2)
        btnAdd = Button(window, text='+', width=3, command=self.add)
        btnAdd.grid(row=0, column=1, padx=15)
        btnSubtract = Button(window, text='-', width=3,
                             command=self.subtract)
        btnSubtract.grid(row=1, column=1, padx=15)
        btnMultiply = Button(window, text='x', width=3,
                             command=self.multiply)
        btnMultiply.grid(row=2, column=1, padx=15, pady=5)
        self.conOFentResult = StringVar()
        self.entResult = Entry(window, state="readonly", width=20,
                               textvariable=self.conOFentResult)
        self.entResult.grid(row=3, column=0, columnspan=3, padx=40,
                            pady=5)
        window.mainloop()

    def add(self):
        num1 = eval(self._conOFentFirst.get())
        num2 = eval(self._conOFentSecond.get())
        sum = num1 + num2
        self.conOFentResult.set("Sum: " + str(sum))

    def subtract(self):
        num1 = eval(self._conOFentFirst.get())
        num2 = eval(self._conOFentSecond.get())
        difference = num1 - num2
        self.conOFentResult.set("Difference: " + str(difference))

    def multiply(self):
        num1 = eval(self._conOFentFirst.get())
        num2 = eval(self._conOFentSecond.get())
        product = num1 * num2
        self.conOFentResult.set("Product: " + str(product))

Calculate()
```

10.
```
from tkinter import *

def makeChange():
    amount = int(conOFentAmount.get())
    remainder = amount
    quarters = remainder // 25
    remainder %= 25
    dimes = remainder // 10
    remainder %= 10
    nickels = remainder // 5
    remainder %= 5
    cents = remainder
    conOFentQuarters.set(str(quarters))
    conOFentDimes.set(str(dimes))
    conOFentNickels.set(str(nickels))
    conOFentCents.set(str(cents))

window = Tk()
window.title("Change")
caption = "Amount: "
Label(window, text=caption).grid(row=0, column=1, sticky=E)
conOFentAmount = StringVar()
entAmount = Entry(window, width=2, textvariable=conOFentAmount)
entAmount.grid(row=0, column=2, sticky=W)
caption = "Determine Composition of Change"
btnDetermine = Button(window, text=caption, command=makeChange)
btnDetermine.grid(row=1, column=0, columnspan=4, padx=20, pady=5)
Label(window, text="Quarters: ").grid(row=2, column=0, sticky=E)
Label(window, text="Nickels: ").grid(row=3, column=0, sticky=E)
Label(window, text="Dimes: ").grid(row=2, column=2, sticky=E)
Label(window, text="Cents: ").grid(row=3, column=2, sticky=E)
conOFentQuarters = StringVar()
entQuarters = Entry(window, width=2, state="readonly",
                    textvariable=conOFentQuarters)
entQuarters.grid(row=2, column=1, sticky=W)
conOFentNickels = StringVar()
entNickels = Entry(window, width=2, state="readonly",
                    textvariable=conOFentNickels)
entNickels.grid(row=3, column=1, sticky=W)
conOFentDimes = StringVar()
entDimes = Entry(window, width=2, state="readonly",
                    textvariable=conOFentDimes)
entDimes.grid(row=2, column=3, sticky=W)
conOFentCents = StringVar()
entCents = Entry(window, width=2, state="readonly", textvariable=conOFentCents)
entCents.grid(row=3, column=3, sticky=W)
window.mainloop()
```



10. (Object-oriented style)

```
from tkinter import *

class Change:
    def   init  (self):
        window = Tk()
        window.title("Change")
        caption = "Amount: "
        Label(window, text=caption).grid(row=0, column=1,
                                         sticky=E)
        self._conOFentAmount = StringVar()
        self.entAmount = Entry(window, width=2,
                               textvariable=self._conOFentAmount)
        self.entAmount.grid(row=0, column=2, sticky=W)
        caption = "Determine Composition of Change"
        btnDetermine = Button(window, text=caption,
                              command=self.makeChange)
        btnDetermine.grid(row=1, column=0, columnspan=4, padx=20, pady=5)
        Label(window, text="Quarters: ").grid(row=2, column=0, sticky=E)
        Label(window, text="Nickels: ").grid(row=3, column=0, sticky=E)
        Label(window, text="Dimes: ").grid(row=2, column=2, sticky=E)
        Label(window, text="Cents: ").grid(row=3, column=2, sticky=E)
        self._conOFentQuarters = StringVar()
        self.entQuarters = Entry(window, width=2, state="readonly",
                                 textvariable=self._conOFentQuarters)
        self.entQuarters.grid(row=2, column=1, sticky=W)
        self._conOFentNickels = StringVar()
        self.entNickels = Entry(window, width=2, state="readonly",
                                textvariable=self._conOFentNickels)
        self.entNickels.grid(row=3, column=1, sticky=W)
        self._conOFentDimes = StringVar()
        self.entDimes = Entry(window, width=2, state="readonly",
                              textvariable=self._conOFentDimes)
        self.entDimes.grid(row=2, column=3, sticky=W)
        self._conOFentCents = StringVar()
        self.entCents = Entry(window, width=2, state="readonly",
                              textvariable=self._conOFentCents)
        self.entCents.grid(row=3, column=3, sticky=W)
        window.mainloop()

    def makeChange(self):
        amount = int(self._conOFentAmount.get())
        remainder = amount
        quarters = remainder // 25
        remainder %= 25
        dimes = remainder // 10
        remainder %= 10
        nickels = remainder // 5
        remainder %= 5
        cents = remainder
        self._conOFentQuarters.set(str(quarters))
        self._conOFentDimes.set(str(dimes))
        self._conOFentNickels.set(str(nickels))
        self._conOFentCents.set(str(cents))

Change()
```

13. Academy Awards Write a program using the file **`Oscars.txt`** that fills a list box with genres and then displays the Oscar-winning films of a specific genre when the user clicks on the genre in the list box. See Fig. 8.44.

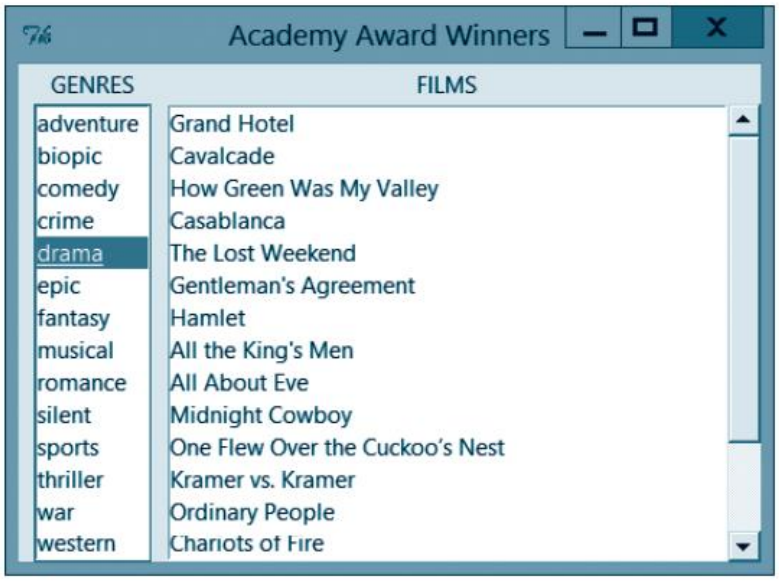


FIGURE 8.44 Possible outcome of Exercise 13.

```
13. from tkinter import *

    def films(e):
        genre = lstGenres.get(lstGenres.curselection())
        F = [line.split(',')[0] for line in open("Oscars.txt", 'r') if
             line.split(',')[1].rstrip() == genre]
        conOFlstFilms.set(tuple(F))

    window = Tk()
    window.title("Academy Award Winners")
    Label(window, text="GENRES").grid(row=0, column=0)
    Label(window, text="FILMS").grid(row=0, column=1)
    infile = open("Oscars.txt", 'r')
    genreSet = {line.split(',')[1].rstrip() for line in infile}
    infile.close()
    L = list(genreSet)
    L.sort()
    conOFlstGenres = StringVar()
    lstGenres = Listbox(window, width=9, height=len(L), listvariable=conOFlstGenres)
    lstGenres.grid(row=1, column=0, padx=10, sticky=N)
    conOFlstGenres.set(tuple(L))
    lstGenres.bind("<<ListboxSelect>>", films)
    yscroll = Scrollbar(window, orient=VERTICAL)
    yscroll.grid(row=1, column=2, sticky=NS)
    conOFlstFilms = StringVar()
    lstFilms = Listbox(window, width=45, height=len(L),
                       listvariable=conOFlstFilms, yscrollcommand=yscroll.set)
    lstFilms.grid(row=1, column=1, sticky=NSEW)
    yscroll["command"] = lstFilms.yview
    window.mainloop()
```

13. (Object-oriented style)

```
from tkinter import *
class Oscars:
    def   init  (self):
        window = Tk()
        window.title("Academy Award Winners")
        Label(window, text="GENRES").grid(row=0, column=0)
        Label(window, text="FILMS").grid(row=0, column=1)
        infile = open("Oscars.txt", 'r')
        self._genreSet = {line.split(',')[1].rstrip() \
                          for line in infile}
        infile.close()
        self._L = list(self._genreSet)
        self._L.sort()
        self._conOFlstGenres = StringVar()
        self._lstGenres = Listbox(window, width=9, height=len(self._L),
                                  listvariable=self._conOFlstGenres)
        self._lstGenres.grid(row=1, column=0, padx=10, sticky=N)
        self._conOFlstGenres.set(tuple(self._L))
        self._lstGenres.bind("<<ListboxSelect>>", self.films)
        yscroll = Scrollbar(window, orient=VERTICAL)
        yscroll.grid(row=1, column=2, sticky=NS)
        self._conOFlstFilms = StringVar()
        lstFilms = Listbox(window, width=45, height=len(self._L),
                           listvariable=self._conOFlstFilms,
                           yscrollcommand=yscroll.set)
        lstFilms.grid(row=1, column=1, sticky=NSEW)
        yscroll["command"] = lstFilms.yview
        window.mainloop()

    def films(self, e):
        genre = self._lstGenres.get(self._lstGenres.curselection())
        F = [line.split(',')[0] for line in open("Oscars.txt", 'r') \
             if line.split(',')[1].rstrip() == genre]
        self._conOFlstFilms.set(tuple(F))

Oscars()
```

14. Academy Awards Write a program using the file `Oscars.txt` that requests a year and then displays the name and genre of that year's best picture winner. See Fig. 8.45.

FIGURE 8.45 Possible outcome of Exercise 14.

14.
```
from tkinter import *
class Oscars:
    def   init  (self):
        window = Tk()
        window.title("Academy Awards")
        caption = "Year (1928-2013): "
        Label(window, text=caption).grid(row=0, column=0)
        self._conOFentYear = StringVar()
        self.entYear = Entry(window, width=4,
                             textvariable=self._conOFentYear)
        self.entYear.grid(row=0, column=1, sticky=W)
        caption = "Find Best Picture"
        btnFind = Button(window, text=caption, command=self.displayFilm)
        btnFind.grid(row=1, column=1, pady=2)
        Label(window, text="Film:").grid(row=2, column=0, sticky=E)
        Label(window, text="Genre:").grid(row=3, column=0, pady=5,
                                          sticky=E)

        self._conOFentFilm = StringVar()
        self.entFilm = Entry(window, width=30, state="readonly",
                             textvariable=self._conOFentFilm)
        self.entFilm.grid(row=2, column=1, padx=5,sticky=W)
        self._conOFentGenre = StringVar()
        self.entGenre = Entry(window, width=30, state="readonly",
                              textvariable=self._conOFentGenre)
        self.entGenre.grid(row=3, column=1, padx=5,pady=5, sticky=W)
        window.mainloop()

    def displayFilm(self):
        infile = open("Oscars.txt", 'r')
        for i in range(int(self._conOFentYear.get()) - 1928):
            infile.readline()
        line = infile.readline().rstrip()
        infile.close()
        data = line.split(',')
        self._conOFentFilm.set(data[0])
        self._conOFentGenre.set(data[1])

Oscars()
```

18. Presidential Colleges This exercise requires the file **`PresColl.txt`** that contains the names of U.S. presidents and the undergraduate college attended by each of them. The presidents are listed in the order they served. The first three lines of the file are

```
George Washington,No college
John Adams,Harvard
Thomas Jefferson,William and Mary
```

Write a program that fills a list box with the colleges (in alphabetical order) attended by U.S. presidents and then displays the presidents who attended that college when the user clicks on a college in the list box. See Fig. 8.49.

FIGURE 8.49 Possible outcome of Exercise 18.

```
18. from tkinter import *

    class PresColleges:

        def   init  (self):
            window = Tk()
            window.title("Presidential Colleges")
            instruction = "Click on a college."
            Label(window, text=instruction).grid(row=0, column=0,
                                                 columnspan=3, pady=5)
            Label(window, text="COLLEGE", width=14).grid(row=1, column=0)
            Label(window, text="PRESIDENTS").grid(row=1, column=2)
            yscroll = Scrollbar(window, orient=VERTICAL)
            yscroll.grid(row=2, column=1, pady=5, sticky=NS)
            infile = open("PresColl.txt", 'r')
            collegeSet = {line.split(',')[1].rstrip() for line in infile}
            infile.close()
            collegeList = list(collegeSet)
            collegeList.sort()
            conOFlstColleges = StringVar()
            self._lstColleges = Listbox(window, width=20, height=8,
                  listvariable=conOFlstColleges, yscrollcommand=yscroll.set)
            self._lstColleges.grid(row=2, column=0, padx=(5,0), pady=5, sticky=E)
            self._lstColleges.bind("<<ListboxSelect>>", self.presidents)
            conOFlstColleges.set(tuple(collegeList))
            self._conOFlstPresidents = StringVar()
            self._lstPresidents = Listbox(window, width=18, height=8,
                                  listvariable=self._conOFlstPresidents)
            self._lstPresidents.grid(row=2, column=2, padx=8, pady=5, sticky=N)
            yscroll["command"] = self._lstColleges.yview
            window.mainloop()
```

```
    def presidents(self, e):
        self.L = []
        college = self._lstColleges.get(self._lstColleges.curselection())
        for line in open("PresColl.txt", 'r'):
            temp = line.split(',')
        if temp[1].rstrip() == college:
            self.L.append(temp[0])
        self._conOFlstPresidents.set(tuple(self.L))

PresColleges()
```

19. Workplaces Table 8.3 holds the names of five people and their places of employment. Write a program that displays the people in one list box and the workplaces in another list box, with the items in each list box in alphabetical order. The user should try to match a person with their workplace by selecting an item from each list. When they click on the button, they should be told whether or not they made a correct match. See Fig. 8.50. ***Note:*** Normally, if there are two list boxes in the window, when you select a value in one, it deselects whatever you selected in the other. However, this behavior will not occur if you insert the argument **`exportselection=0`** into each list box's constructor.

TABLE 8.3 Place of employment.

Person	Workplace
Bruce Wayne	Wayne Enterprises
Clark Kent	Daily Planet
Peter Parker	Daily Bugle
Rick Blaine	Rick's American Cafe
Willie Wonka	Chocolate Factory

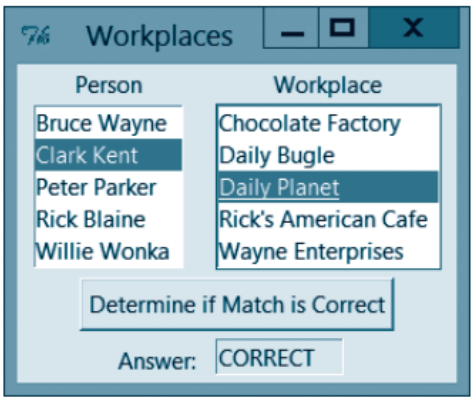
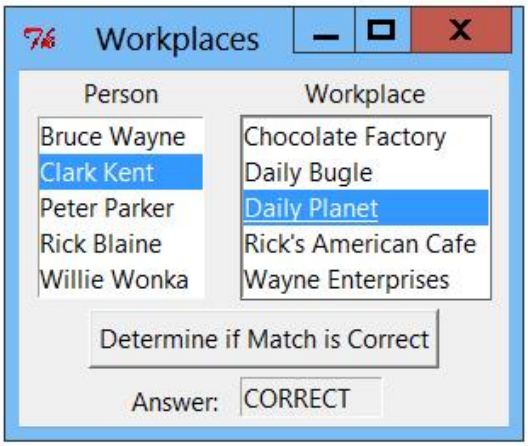
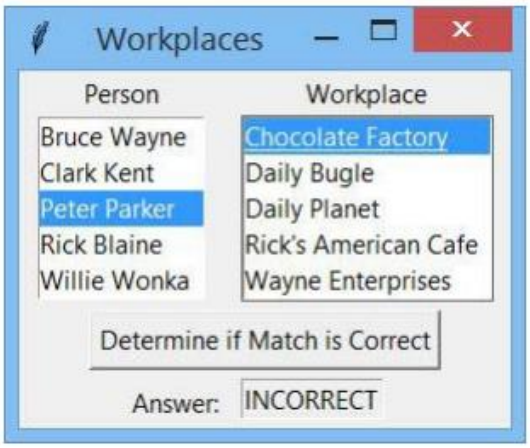


FIGURE 8.50 Possible outcome of Exercise 19.

```
19. from tkinter import *

    def checkAnswer():
        m = people.index(lstPeople.get(lstPeople.curselection()))
        n = places.index(lstPlaces.get(lstPlaces.curselection()))
        if m == n:
            conOFentAnswer.set("CORRECT")
        else:
            conOFentAnswer.set("INCORRECT")
```

```
window = Tk()
window.title("Workplaces")
Label(window, text="Person").grid(row=0, column=0)
Label(window, text="Workplace").grid(row=0, column=1)
people = ["Bruce Wayne", "Clark Kent", "Peter Parker",
          "Rick Blaine", "Willie Wonka"]
places = ["Wayne Enterprises", "Daily Planet", "Daily Bugle",
          "Rick's American Cafe", "Chocolate Factory"]
placesSorted = list(places)
placesSorted.sort()
conOFlstPeople = StringVar()
lstPeople = Listbox(window, width=12, height=5, exportselection=0,
            listvariable=conOFlstPeople)
lstPeople.grid(row=1, column=0, padx=10)
conOFlstPeople.set(tuple(people))

conOFlstPlaces = StringVar()
lstPlaces = Listbox(window, width=18, height=5, exportselection=0,
                    listvariable=conOFlstPlaces)

lstPlaces.grid(row=1, column=1, padx=10)
conOFlstPlaces.set(tuple(placesSorted))
btnDetermine = Button(window, text="Determine if Match is Correct",
                      command=checkAnswer)
btnDetermine.grid(row=2, column=0, columnspan=2, pady=5)
Label(window, text="Answer:").grid(row=3, column=0, sticky=E)
conOFentAnswer = StringVar()
entAnswer = Entry(window, width=10, textvariable=conOFentAnswer,
                  state="readonly")
entAnswer.grid(row=3, column=1, padx=10, pady=(0,5), sticky=W)
window.mainloop()
```





19. (Object-oriented style)

```
from tkinter import *


class Workplaces:
    def   init  (self):
        window = Tk()
        window.title("Workplaces")
        Label(window, text="Person").grid(row=0, column=0)
        Label(window, text="Workplace").grid(row=0, column=1)
        self._people = ["Bruce Wayne", "Clark Kent", "Peter Parker",
                        "Rick Blaine", "Willie Wonka"]
        self._places = ["Wayne Enterprises", "Daily Planet",
              "Daily Bugle", "Rick's American Cafe", "Chocolate Factory"]
        self._placesSorted = list(self._places)
        self._placesSorted.sort()
        self._conOFlstPeople = StringVar()
        self._lstPeople = Listbox(window, width=12, height=5,
                    exportselection=0, listvariable=self._conOFlstPeople)
        self._lstPeople.grid(row=1, column=0, padx=10)
        self._conOFlstPeople.set(tuple(self._people))
        self._conOFlstPlaces = StringVar()
        self._lstPlaces = Listbox(window, width=18, height=5,
                    exportselection=0, listvariable=self._conOFlstPlaces)
        self._lstPlaces.grid(row=1, column=1, padx=10)
        self._conOFlstPlaces.set(tuple(self._placesSorted))
        self._btnDetermine = Button(window,
                                    text="Determine if Match is Correct",
                                    command=self.checkAnswer)
        self._btnDetermine.grid(row=2, column=0, columnspan=2, pady=5)
        Label(window, text="Answer:").grid(row=3, column=0, sticky=E)
        self._conOFentAnswer = StringVar()
        self._entAnswer = Entry(window, width=10,
                                textvariable=self._conOFentAnswer,
                                state="readonly")
        self._entAnswer.grid(row=3, column=1, padx=10, pady=(0,5),
                             sticky=W)
        window.mainloop()

    def checkAnswer(self):
        m = self._people.index(
                    self._lstPeople.get(self._lstPeople.curselection()))
        n = self._places.index(
                    self._lstPlaces.get(self._lstPlaces.curselection()))
        if m == n:
            self._conOFentAnswer.set("CORRECT")
        else:
            self._conOFentAnswer.set("INCORRECT")

Workplaces()
```

1. Investment If $10,000 is invested at an annual interest rate r compounded n times per year, then the amount of the investment after five years will be $10{,}000(1 + \frac{r}{n})^{5 \cdot n}$ Some possible values for r are .02, .025, .03, .035, and .04. Some possible values for n are 1, 2, 4, 12, and 52. Write a program that allows the user to select interest rates and compounding periods from list boxes and calculate the amount after five years. See Fig. 8.51. ***Note:*** Normally, if there are two list boxes in the window, when you select a value in one, it deselects whatever you selected in the other. However, this behavior will not occur if the argument **`exportselection=0`** is inserted into each list box's constructor.

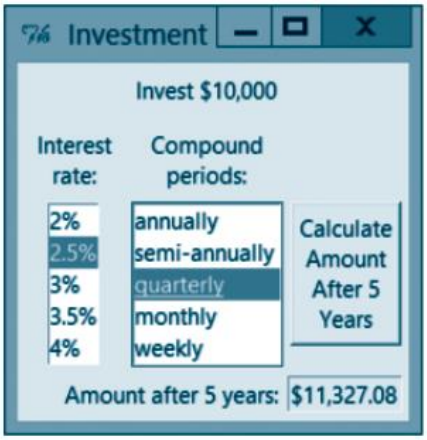


FIGURE 8.51 Possible outcome of Programming Project 1.

```
from tkinter import *
def calculate():
    rate = lstRates.get(lstRates.curselection())
    if rate == "2%":
        intRate = .02
    elif rate == "2.5%":
        intRate = .025
    elif rate == "3%":
        intRate = .03
    elif rate == "3.5%":
        intRate = .035
    elif rate == "4%":
        intRate = .04
    periods = lstPeriods.get(lstPeriods.curselection())
    if periods == "annually":
        n = 1
    elif periods == "semi-annually":
        n = 2
    elif periods == "quarterly":
        n = 4
    elif periods == "monthly":
        n = 12
    elif periods == "weekly":
        n = 52
    amount = 10000 * (1 +  intRate/n) ** (5*n)
    conOFentAmount.set("${0:,.2f}".format(amount))
```

```
window = Tk()
window.title("Investment")
Label(window, text="Invest $10,000").grid(row=0, column=1, pady=5)
Label(window, text="Interest\nrate:").grid(row=1, column=0, padx=10, pady=5)
Label(window, text="Compound\nperiods:").grid(row=1, column=1,
                                                padx=10, pady=5)
btnCalculate = Button(window, text="Calculate\nAmount\nAfter 5\nYears",
                      command=calculate)
btnCalculate.grid(row=3, column=2, padx=5, sticky=N)
conOFlstRates = StringVar()
lstRates = Listbox(window, height=5, width=4, exportselection=0,
                   listvariable=conOFlstRates)
lstRates.grid(row=3, column=0)
conOFlstPeriods = StringVar()
lstPeriods = Listbox(window, height=5, width=12, exportselection=0,
                     listvariable=conOFlstPeriods)
lstPeriods.grid(row=3, column=1)
Label(window, text="Amount after 5 years:").grid(row=4, column=0,
                                    pady=5, columnspan=2, sticky=E)
conOFentAmount = StringVar()
entAmount = Entry(window, textvariable=conOFentAmount,
                  width=9, state="readonly")
entAmount.grid(row=4, column=2, padx = 3, pady=5, sticky=W)
conOFlstRates.set(("2%", "2.5%","3%","3.5%","4%"))
conOFlstPeriods.set(("annually", "semi-annually",
                     "quarterly", "monthly", "weekly"))
window.mainloop()
```

2. United Nations Each line of the file **UN.txt** gives the name, continent, population (in millions), and area (in square miles) of a member of the United Nations. Some lines of the file are

```
Canada,North America,34.8,3855000
France,Europe,66.3,211209
New Zealand,Australia/Oceania,4.4,103738
Nigeria,Africa,177.2,356669
Pakistan,Asia,192.2,310403
Peru,South America,30.0,496226
```

This data has been placed into the following dictionary-valued dictionary, and then pickled into the binary file **UNdict.dat**.

```
nations = {"Canada":{"cont":"North America", "popl":34.8, "area":3855000},
        "France":{"cont":"Europe", "popl":66.3}, "area":211209} ...}
```

Write a program using the file **UNdict.dat** that allows the user to display the continent, population, and area of a country by clicking on the name of the country in a list box. See Fig. 8.52.

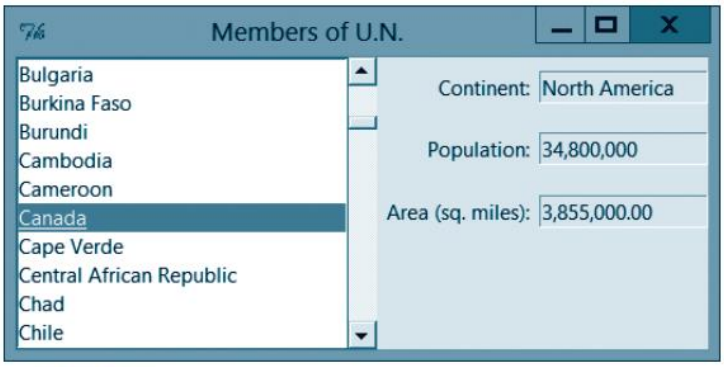


FIGURE 8.52 Possible outcome of Programming Project 2.

2.
```
from tkinter import *
import pickle

class Nations:

    def   init  (self):
        window = Tk()
        window.title("Members of U.N.")
        infile = open("UNdict.dat", 'rb')
        self._nationDict = pickle.load(infile)
        infile.close()
        self._nationList = list((self._nationDict).keys())
        self._nationList.sort()
        self._conOFlstNations = StringVar()
        yscroll = Scrollbar(window, orient=VERTICAL)
        yscroll.grid(row=0, column=1, rowspan=7, sticky=NS)
        self._lstNations = Listbox(window, height=10, width=30,
            listvariable=self._conOFlstNations, yscrollcommand=yscroll.set)
        self._lstNations.grid(row=0, column=0, rowspan=7, sticky=NSEW)
        self._conOFlstNations.set(tuple(self._nationList))
        self._lstNations.bind("<<ListboxSelect>>", self.displayData)
        yscroll["command"] = self._lstNations.yview
        Label(window, text="Continent:").grid(row=0, column=3,
                                              padx=4, sticky=E)
        Label(window, text="Population:").grid(row=1, column=3,
                                               padx=4, sticky=E)
        Label(window, text="Area (sq. miles):").grid(row=2, column=3,
                                                     padx=4, sticky=E)
        self._conOFentContinent = StringVar()
        entContinent = Entry(window, width=15, state="readonly",
                             textvariable=self._conOFentContinent)
        entContinent.grid(row=0, column=4, sticky=W)
        self._conOFentPopulation = StringVar()
        entPopulation = Entry(window, width=15, state="readonly",
                              textvariable=self._conOFentPopulation)
        entPopulation.grid(row=1, column=4,)
        self._conOFentArea = StringVar()
        entArea = Entry(window, width=15, state="readonly",
                        textvariable=self._conOFentArea)
        entArea.grid(row=2, column=4)
        window.mainloop()


    def displayData(self, e):
        nation = self._lstNations.get(self._lstNations.curselection())
        self._conOFentContinent.set(self._nationDict[nation]["cont"])
        self._conOFentPopulation.set("{0:,.0f}". \
             format(1000000 * float(self._nationDict[nation]["popl"])))
        self._conOFentArea.set("{0:,.2f}".\
                               format(self._nationDict[nation]["area"]))

Nations()
```

3. Pensions A person in the Civil Service Retirement System can retire at age 55 with at least 20 years of service. A simplified variation for the computation of the amount of their pension is as follows:

(a) Calculate the average annual salary for the person's best three years; call it *ave*.

(b) Calculate $\left(\text{number of years} + \dfrac{\text{number of months}}{12}\right)$; call it *yrs*.

(c) Calculate percentage rate: 1.5% for first five years, 1.75% for next five years, and 2% for each additional year. Call it *perRate*.

(d) Take the minimum of *perRate* and 80%; call it *p*.

(e) The amount of the pension is *p*ave*.

Write a program that requests the input shown in Fig. 8.53 and calculates the amount of the pension.

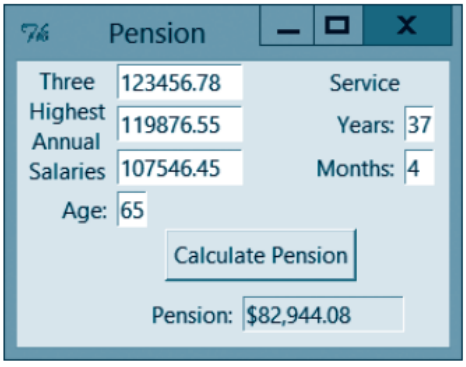


FIGURE 8.53 Possible outcome of Programming Project 3.

```
3. from tkinter import *

   def calculate():
       ave = (eval(conOFentSalary1.get()) + eval(conOFentSalary2.get()) +
              eval(conOFentSalary3.get())) / 3
       yrs = eval(conOFentYears.get())
       months = eval(conOFentMonths.get())
       yrs += months / 12
       percentage = 36.25 + (2 * (yrs - 20))
       if percentage > 80:
           percentage = 80
       pension = ave * (percentage / 100)
       conOFentPension.set("${0:,.2f}".format(pension))
   window = Tk()
   window.title("Pension")
   Label(window, text="Three\nHighest\nAnnual\nSalaries").grid(row=0,
                                                    column=0, rowspan=3, padx=5)
```

```
conOFentSalary1 = StringVar()
entSalary1 = Entry(window, width=10, textvariable=conOFentSalary1)
entSalary1.grid(row=0, column=1)
conOFentSalary2 = StringVar()
entSalary2 = Entry(window, width=10, textvariable=conOFentSalary2)
entSalary2.grid(row=1, column=1)
conOFentSalary3 = StringVar()
entSalary3 = Entry(window, width=10, textvariable=conOFentSalary3)
entSalary3.grid(row=2, column=1)
Label(window, text="Service").grid(row=0, column=2, sticky=E)
Label(window, text="Years: ").grid(row=1, column=2, sticky=E)
conOFentYears = StringVar()
entYears = Entry(window, width=2, textvariable=conOFentYears)
entYears.grid(row=1, column=3, padx=(0, 10), sticky=W)
Label(window, text="Months: ").grid(row=2, column=2, sticky=E)
conOFentMonths = StringVar()
entMonths = Entry(window, width=2, textvariable=conOFentMonths)
entMonths.grid(row=2, column=3, sticky=W)
Label(window, text="Age: ").grid(row=3, column=0, sticky=E)
conOFentAge = StringVar()
entAge = Entry(window, width=2, textvariable=conOFentAge)
entAge.grid(row=3, column=1, sticky=W)
btnCalculate = Button(window, text="Calculate Pension", command=calculate)
btnCalculate.grid(row=4, column=1, columnspan=2)
Label(window, text="Pension: ").grid(row=5, column=1, sticky=E)
conOFentPension = StringVar()
entPension = Entry(window, width=13,
                   textvariable=conOFentPension,state="readonly")
entPension.grid(row=5, column=2, pady=10)
window.mainloop()
```

4. Verbalize a Number Write a program that allows the user to enter a positive whole number having no more than 27 digits (with commas included) and then verbalizes the number. See Fig. 8.54.

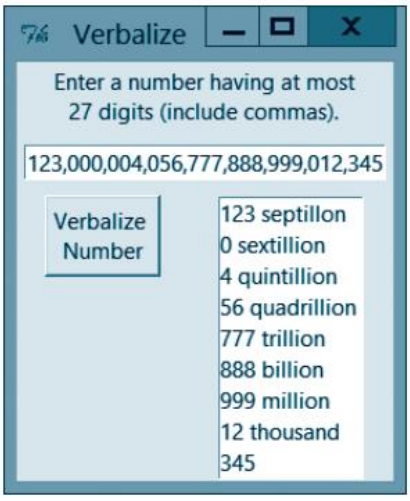


FIGURE 8.54 Possible outcome of Programming Project 4.

4.
```
from tkinter import *

def stripOutLeadingZeros(front):
    if front == "000":
        front = "0"
    elif front[:2] == "00":
        front = front[2]
    elif front[0] == "0":
        front = front[1:]
    return front

def verbalize():
    L = ["", " thousand", " million", " billion", " trillion",
         " quadrillion", " quintillion", " sextillion", " septillion"]
    N = []
    number = conOFentNumber.get()
    numberOfCommas = number.count(',')
    L = L[:numberOfCommas + 1]
    for i in range(numberOfCommas + 1, 0, -1):
        loc = number.find(',')
        if loc == -1:
            number = stripOutLeadingZeros(number)
            N.append(number)
        else:
            front = number[:loc]
            front = stripOutLeadingZeros(front)
            N.append(front + L[-1])
        conOflstBox.set(tuple(N))
        number = number[loc + 1:]
        del L[-1]

window = Tk()
window.title("Verbalize")
instructions = "Enter a number having at most\n27 digits (include commas)."
lbl = Label(window, text=instructions)
lbl.grid(row=0, column=0, columnspan=2)
conOFentNumber = StringVar()
entNumber = Entry(window, width=30, textvariable=conOFentNumber)
entNumber.grid(row=1, column=0, columnspan=2,padx=5, pady=10)
conOflstBox = StringVar()
lstBox = Listbox(window, width=12, height=9, listvariable=conOflstBox)
lstBox.grid(row=2, column=1)
b = Button(window, text="Verbalize \nNumber", command=verbalize)
b.grid(row=2, column=0, sticky=N)
window.mainloop()
```