```
    def main():
        print("This program plots the growth of a 10 year investment.")

        principal = float(input("Enter the initial principal: "))
        apr = float(input("Enter the annualized interest rate: "))

        win = createLabeledWindow()
        drawBar(win, 0, principal)
        for year in range(1, 11):
            principal = principal * (1 + apr)
            drawBar(win, year, principal)

        input("Press <Enter> to quit.")
        win.close()

main()
```

Although this version is longer than the previous version, experienced programmers would find it much easier to understand. As you get used to reading and writing functions, you too will learn to appreciate the elegance of more modular code.

6.8 Chapter Summary

- A function is a kind of subprogram. Programmers use functions to reduce code duplication and to help structure or modularize programs. Once a function is defined, it may be called multiple times from many different places in a program. Parameters allow functions to have changeable parts. The parameters appearing in the function definition are called formal parameters, and the expressions appearing in a function call are known as actual parameters.
- A call to a function initiates a four-step process:
    1. The calling program is suspended.
    2. The values of actual parameters are assigned to the formal parameters.
    3. The body of the function is executed.

4. Control returns immediately following the function call in the calling program. The value returned by the function is used as the expression result.

- The scope of a variable is the area of the program where it may be referenced. Formal parameters and other variables inside function definitions are local to the function. Local variables are distinct from variables of the same name that may be used elsewhere in the program.

- Functions can communicate information back to the caller through return values. In Python, functions may return multiple values. Value-returning functions should generally be called from inside an expression. Functions that don't explicitly return a value return the special object `None`.

- Python passes parameters by value. If the value being passed is a mutable object, then changes made to the object may be visible to the caller.

6.9 Exercises

Review Questions

True/False

1. Programmers rarely define their own functions.
2. A function may only be called at one place in a program.
3. Information can be passed into a function through parameters.
4. Every Python function returns some value.
5. In Python, some parameters are passed by reference.
6. In Python, a function can return only one value.
7. Python functions can never modify a parameter.
8. One reason to use functions is to reduce code duplication.
9. Variables defined in a function are local to that function.
10. It's a bad idea to define new functions if it makes a program longer.

Multiple Choice

1. The part of a program that uses a function is called the
   a) user b) caller c) callee d) statement

2. A Python function definition begins with
   a) `def` b) `define` c) `function` d) `defun`

3. A function can send output back to the program with a(n)
   a) `return` b) `print` c) assignment d) SASE

4. Formal and actual parameters are matched up by
   a) name b) position c) ID d) interests

5. Which of the following is *not* a step in the function-calling process?
   a) The calling program suspends.
   b) The formal parameters are assigned the value of the actual parameters.
   c) The body of the function executes.
   d) Control returns to the point just before the function was called.

6. In Python, actual parameters are passed to functions
   a) by value b) by reference c) at random d) by networking

7. Which of the following is *not* a reason to use functions?
   a) to reduce code duplication
   b) to make a program more modular
   c) to make a program more self-documenting
   d) to demonstrate intellectual superiority

8. If a function returns a value, it should generally be called from
   a) an expression b) a different program
   c) `main` d) a cell phone

9. A function with no `return` statement returns
   a) nothing b) its parameters c) its variables d) `None`

10. A function can modify the value of an actual parameter only if it's
    a) mutable b) a list c) passed by reference d) a variable

Discussion

1. In your own words, describe the two motivations for defining functions in your programs.

2. We have been thinking about computer programs as sequences of instructions where the computer methodically executes one instruction and then moves on to the next one. Do programs that contain functions fit this model? Explain your answer.

3. Parameters are an important concept in defining functions.

   a) What is the purpose of parameters?

   b) What is the difference between a formal parameter and an actual parameter?

   c) In what ways are parameters similar to and different from ordinary variables?

4. Functions can be thought of as miniature (sub)programs inside other programs. Like any other program, we can think of functions as having input and output to communicate with the main program.

   a) How does a program provide "input" to one of its functions?

   b) How does a function provide "output" to the program?

5. Consider this very simple function:

```
def cube(x):
    answer = x * x * x
    return answer
```

   a) What does this function do?

   b) Show how a program could use this function to print the value of y^3, assuming y is a variable.

   c) Here is a fragment of a program that uses this function:

```
answer = 4
result = cube(3)
print(answer, result)
```

   The output from this fragment is `4 27`. Explain why the output is not `27 27`, even though `cube` seems to change the value of `answer` to 27.

Programming Exercises

1. Write a program to print the lyrics of the song "Old MacDonald." Your program should print the lyrics for five different animals, similar to the example verse below.

   Old MacDonald had a farm, Ee-igh, Ee-igh, Oh!
   And on that farm he had a cow, Ee-igh, Ee-igh, Oh!
   With a moo, moo here and a moo, moo there.
   Here a moo, there a moo, everywhere a moo, moo.
   Old MacDonald had a farm, Ee-igh, Ee-igh, Oh!

2. Write a program to print the lyrics for ten verses of "The Ants Go Marching." A couple of sample verses are given below. You may choose your own activity for the "little one" in each verse, but be sure to choose something that makes the rhyme work (or almost work).

   The ants go marching one by one, hurrah! hurrah!
   The ants go marching one by one, hurrah! hurrah!
   The ants go marching one by one,
   The little one stops to suck his thumb,
   And they all go marching down...
   In the ground...
   To get out....
   Of the rain.
   Boom! Boom! Boom!

   The ants go marching two by two, hurrah! hurrah!
   The ants go marching two by two, hurrah! hurrah!
   The ants go marching two by two,
   The little one stops to tie his shoe,
   And they all go marching down...
   In the ground...
   To get out...
   Of the rain.
   Boom! Boom! Boom!

3. Write definitions for these functions:

   `sphereArea(radius)` Returns the surface area of a sphere having the given radius.

`sphereVolume(radius)` Returns the volume of a sphere having the given radius.

Use your functions to solve Programming Exercise 1 from Chapter 3.

4. Write definitions for the following two functions:

   `sumN(n)` returns the sum of the first `n` natural numbers.

   `sumNCubes(n)` returns the sum of the cubes of the first `n` natural numbers.

   Then use these functions in a program that prompts a user for an n and prints out the sum of the first n natural numbers and the sum of the cubes of the first n natural numbers.

5. Redo Programming Exercise 2 from Chapter 3. Use two functions—one to compute the area of a pizza, and one to compute cost per square inch.

6. Write a function that computes the area of a triangle given the length of its three sides as parameters (see Programming Exercise 9 from Chapter 3). Use your function to augment `triangle2.py` from this chapter so that it also displays the area of the triangle.

7. Write a function to compute the nth Fibonacci number. Use your function to solve Programming Exercise 16 from Chapter 3.

8. Solve Programming Exercise 17 from Chapter 3 using a function `nextGuess(guess, x)` that returns the next guess.

9. Do Programming Exercise 3 from Chapter 5 using a function `grade(score)` that returns the letter grade for a score.

10. Do Programming Exercise 4 from Chapter 5 using a function `acronym(phrase)` that returns an acronym for a phrase supplied as a string.

11. Write and test a function to meet this specification.

    `squareEach(nums)` `nums` is a list of numbers. Modifies the list by squaring each entry.

12. Write and test a function to meet this specification.

    `sumList(nums)` `nums` is a list of numbers. Returns the sum of the numbers in the list.

13. Write and test a function to meet this specification.

    `toNumbers(strList)` `strList` is a list of strings, each of which represents a number. Modifies each entry in the list by converting it to a number.

14. Use the functions from the previous three problems to implement a program that computes the sum of the squares of numbers read from a file. Your program should prompt for a file name and print out the sum of the squares of the values in the file. *Hint*: Use `readlines()`

15. Write and test a function to meet this specification.

    `drawFace(center, size, win)` `center` is a `Point`, `size` is an int, and `win` is a `GraphWin`. Draws a simple face of the given size in `win`.

    Your function can draw a simple smiley (or grim) face. Demonstrate the function by writing a program that draws several faces of varying size in a single window.

16. Use your `drawFace` function from the previous exercise to write a photo anonymizer. This program allows a user to load an image file (such as a PPM or GIF) and to draw cartoon faces over the top of existing faces in the photo. The user first inputs the name of the file containing the image. The image is displayed and the user is asked how many faces are to be blocked. The program then enters a loop for the user to click on two points for each face: the center and somewhere on the edge of the face (to determine the size of the face). The program should then draw a face in that location using the `drawFace` function.

    Hints: Section 4.8.4 describes the image-manipulation methods in the graphics library. Display the image centered in a `GraphWin` that is the same width and height as the image, and draw the graphics into this window. You can use a screen capture utility to save the resulting images.

17. Write a function to meet this specification.

    `moveTo(shape, newCenter)` `shape` is a graphics object that supports the `getCenter` method and `newCenter` is a `Point`. Moves `shape` so that `newCenter` is its center.

    Use your function to write a program that draws a circle and then allows the user to click the window 10 times. Each time the user clicks, the circle is moved where the user clicked.