Function Exercises

4

Functions allow a programmer to break a problem into pieces that can be reused. They can also help a programmer focus on one part a larger problem at a time. As a result, writing functions is often an important part of developing larger pieces of software. The exercises in this chapter will help you practice these skills:

- Define a function for later use
- Pass one or more values into a function
- Perform a complex calculation within a function
- Return one or more results from a function
- Call a function that you have defined previously

Exercise 81: Compute the Hypotenuse

(*23 Lines*)

Write a function that takes the lengths of the two shorter sides of a right triangle as its parameters. Return the hypotenuse of the triangle, computed using Pythagorean theorem, as the function's result. Include a main program that reads the lengths of the shorter sides of a right triangle from the user, uses your function to compute the length of the hypotenuse, and displays the result.

Exercise 82: Taxi Fare

(*22 Lines*)

In a particular jurisdiction, taxi fares consist of a base fare of $4.00, plus $0.25 for every 140 meters traveled. Write a function that takes the distance traveled (in kilometers) as its only parameter and returns the total fare as its only result. Write a main program that demonstrates the function.

B. Stephenson, *The Python Workbook*, DOI 10.1007/978-3-319-14240-1_4

Hint: Taxi fares change over time. Use constants to represent the base fare and the variable portion of the fare so that the program can be updated easily when the rates increase.

Exercise 83: Shipping Calculator

(*23 Lines*)

An online retailer provides express shipping for many of its items at a rate of $10.95 for the first item, and $2.95 for each subsequent item. Write a function that takes the number of items in the order as its only parameter. Return the shipping charge for the order as the function's result. Include a main program that reads the number of items purchased from the user and displays the shipping charge.

Exercise 84: Median of Three Values

(*Solved—42 Lines*)

Write a function that takes three numbers as parameters, and returns the median value of those parameters as its result. Include a main program that reads three values from the user and displays their median.

Hint: The median value is the middle of the three values when they are sorted into ascending order. It can be found using if statements, or with a little bit of mathematical creativity.

Exercise 85: Convert an Integer to its Ordinal Number

(*47 Lines*)

Words like *first*, *second* and *third* are referred to as ordinal numbers. In this exercise, you will write a function that takes an integer as its only parameter and returns a string containing the appropriate English ordinal number as its only result. Your function must handle the integers between 1 and 12 (inclusive). It should return an empty string if a value outside of this range is provided as a parameter. Include a main program that demonstrates your function by displaying each integer from 1 to 12 and its ordinal number. Your main program should only run when your file has not been imported into another program.

Exercise 86: The Twelve Days of Christmas

(*Solved—48 Lines*)

The Twelve Days of Christmas is a repetitive song that describes an increasingly long list of gifts sent to one's true love on each of 12 days. A single gift is sent on the first day. A new gift is added to the collection on each additional day, and then the complete collection is sent. The first three verses of the song are shown below. The complete lyrics are available on the internet.

```
On the first day of Christmas
my true love sent to me:
A partridge in a pear tree.

On the second day of Christmas
my true love sent to me:
Two turtle doves,
And a partridge in a pear tree.

On the third day of Christmas
my true love sent to me:
Three French hens,
Two turtle doves,
And a partridge in a pear tree.
```

Your task is to write a program that displays the complete lyrics for The Twelve Days of Christmas. Write a function that takes the verse number as its only parameter and displays the specified verse of the song. Then call that function 12 times with integers that increase from 1 to 12.

Each item that is sent to the recipient in the song should only appear once in your program, with the possible exception of the partridge. It may appear twice if that helps you handle the difference between "A partridge in a pear tree" in the first verse and "And a partridge in a pear tree" in the subsequent verses. Import your solution to Exercise 85 to help you complete this exercise.

Exercise 87: Center a String in the Terminal

(*Solved—31 Lines*)

Write a function that takes a string of characters as its first parameter, and the width of the terminal in characters as its second parameter. Your function should return a new string that consists of the original string and the correct number of leading spaces so that the original string will appear centered within the provided width when it is printed. Do not add any characters to the end of the string. Include a main program that demonstrates your function.

Exercise 88: Is it a Valid Triangle?

(*33 Lines*)

If you have 3 straws, possibly of differing lengths, it may or may not be possible to lay them down so that they form a triangle when their ends are touching. For example, if all of the straws have a length of 6 inches. then one can easily construct an equilateral triangle using them. However, if one straw is 6 inches. long, while the other two are each only 2 inches. long, then a triangle cannot be formed. In general, if any one length is greater than or equal to the sum of the other two then the lengths cannot be used to form a triangle. Otherwise they can form a triangle.

Write a function that determines whether or not three lengths can form a triangle. The function will take 3 parameters and return a Boolean result. In addition, write a program that reads 3 lengths from the user and demonstrates the behaviour of this function.

Exercise 89: Capitalize It

(*Solved—48 Lines*)

Many people do not use capital letters correctly, especially when typing on small devices like smart phones. In this exercise, you will write a function that capitalizes the appropriate characters in a string. A lowercase "i" should be replaced with an uppercase "I" if it is both preceded and followed by a space. The first character in the string should also be capitalized, as well as the first non-space character after a ".", "!" or "?". For example, if the function is provided with the string "what time do i have to be there? what's the address?" then it should return the string "What time do I have to be there? What's the address?". Include a main program that reads a string from the user, capitalizes it using your function, and displays the result.

Exercise 90: Does a String Represent an Integer?

(*Solved—30 Lines*)

In this exercise you will write a function named `isInteger` that determines whether or not the characters in a string represent a valid integer. When determining if a string represents an integer you should ignore any leading or trailing white space. Once this white space is ignored, a string represents an integer if its length is at least 1 and it only contains digits, or if its first character is either + or – and the first character is followed by one or more characters, all of which are digits.

Write a main program that reads a string from the user and reports whether or not it represents an integer. Ensure that the main program will not run if the file containing your solution is imported into another program.

Hint: You may find the `lstrip`, `rstrip` and/or `strip` methods for strings helpful when completing this exercise. Documentation for these methods is available online.

Exercise 91: Operator Precedence

(*30 Lines*)

Write a function named `precedence` that returns an integer representing the precedence of a mathematical operator. A string containing the operator will be passed to the function as its only parameter. Your function should return 1 for + and −, 2 for * and /, and 3 for ^. If the string passed to the function is not one of these operators then the function should return −1. Include a main program that reads an operator from the user and either displays the operator's precedence or an error message indicating that the input was not an operator. Your main program should only run when the file containing your solution has not been imported into another program.

In this exercise, along with others that appear later in the book, we will use ^ to represent exponentiation. Using ^ instead of Python's choice of ** will make these exercises easier because an operator will always be a single character.

Exercise 92: Is a Number Prime?

(*Solved—28 Lines*)

A prime number is an integer greater than 1 that is only divisible by one and itself. Write a function that determines whether or not its parameter is prime, returning `True` if it is, and `False` otherwise. Write a main program that reads an integer from the user and displays a message indicating whether or not it is prime. Ensure that the main program will not run if the file containing your solution is imported into another program.

Exercise 93: Next Prime

(*27 Lines*)

In this exercise you will create a function named `nextPrime` that finds and returns the first prime number larger than some integer, *n*. The value of *n* will be passed to

the function as its only parameter. Include a main program that reads an integer from the user and displays the first prime number larger than the entered value. Import and use your solution to Exercise 92 while completing this exercise.

Exercise 94: Random Password

(*Solved—33 Lines*)

Write a function that generates a random password. The password should have a random length of between 7 and 10 characters. Each character should be randomly selected from positions 33 to 126 in the ASCII table. Your function will not take any parameters. It will return the randomly generated password as its only result. Display the randomly generated password in your file's main program. Your main program should only run when your solution has not been imported into another file.

> Hint: You will probably find the `chr` function helpful when completing this exercise. Detailed information about this function is available online.

Exercise 95: Random License Plate

(*45 Lines*)

In a particular jurisdiction, older license plates consist of three letters followed by three numbers. When all of the license plates following that pattern had been used, the format was changed to four numbers followed by three letters.

Write a function that generates a random license plate. Your function should have approximately equal odds of generating a sequence of characters for an old license plate or a new license plate. Write a main program that calls your function and displays the randomly generated license plate.

Exercise 96: Check a Password

(*Solved—40 Lines*)

In this exercise you will write a function that determines whether or not a password is good. We will define a good password to be a one that is at least 8 characters long and contains at least one uppercase letter, at least one lowercase letter, and at least one number. Your function should return true if the password passed to it as its only parameter is good. Otherwise it should return false. Include a main program that reads a password from the user and reports whether or not it is good. Ensure

that your main program only runs when your solution has not been imported into another file.

Exercise 97: Random Good Password

(*22 Lines*)

Using your solutions to Exercises 94 and 96, write a program that generates a random good password and displays it. Count and display the number of attempts that were needed before a good password was generated. Structure your solution so that it imports the functions you wrote previously and then calls them from a function named `main` in the file that you create for this exercise.

Exercise 98: Hexadecimal and Decimal Digits

(*41 Lines*)

Write two functions, `hex2int` and `int2hex`, that convert between hexadecimal digits (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E and F) and base 10 integers. The `hex2int` function is responsible for converting a string containing a single hexadecimal digit to a base 10 integer, while the `int2hex` function is responsible for converting an integer between 0 and 15 to a single hexadecimal digit. Each function will take the value to convert as its only parameter and return the converted value as the function's only result. Ensure that the `hex2int` function works correctly for both uppercase and lowercase letters. Your functions should end the program with a meaningful error message if an invalid parameter is provided.

Exercise 99: Arbitrary Base Conversions

(*Solved—61 Lines*)

Write a program that allows the user to convert a number from one base to another. Your program should support bases between 2 and 16 for both the input number and the result number. If the user chooses a base outside of this range then an appropriate error message should be displayed and the program should exit. Divide your program into several functions, including a function that converts from an arbitrary base to base 10, a function that converts from base 10 to an arbitrary base, and a main program that reads the bases and input number from the user. You may find your solutions to Exercises 77, 78 and 98 helpful when completing this exercise.

Exercise 100: Days in a Month

(*47 Lines*)

Write a function that determines how many days there are in a particular month. Your function will take two parameters: The month as an integer between 1 and 12, and the year as a four digit integer. Ensure that your function reports the correct number of days in February for leap years. Include a main program that reads a month and year from the user and displays the number of days in that month. You may find your solution to Exercise 57 helpful when solving this problem.

Exercise 101: Reduce a Fraction to Lowest Terms

(*Solved—47 Lines*)

Write a function that takes two positive integers that represent the numerator and denominator of a fraction as its only two parameters. The body of the function should reduce the fraction to lowest terms and then return both the numerator and denominator of the reduced fraction as its result. For example, if the parameters passed to the function are 6 and 63 then the function should return 2 and 21. Include a main program that allows the user to enter a numerator and denominator. Then your program should display the reduced fraction.

Hint: In Exercise 75 you wrote a program for computing the greatest common divisor of two positive integers. You may find that code useful when completing this exercise.

Exercise 102: Reduce Measures

(*Solved—83 Lines*)

Many recipe books still use cups, tablespoons and teaspoons to describe the volumes of ingredients used when cooking or baking. While such recipes are easy enough to follow if you have the appropriate measuring cups and spoons, they can be difficult to double, triple or quadruple when cooking Christmas dinner for the entire extended family. For example, a recipe that calls for 4 tablespoons of an ingredient requires 16 tablespoons when quadrupled. However, 16 tablespoons would be better expressed (and easier to measure) as 1 cup.

Write a function that expresses an imperial volume using the largest units possible. The function will take the number of units as its first parameter, and the unit of measure (cup, tablespoon or teaspoon) as its second parameter. Return a string representing the measure using the largest possible units as the function's only result.

For example, if the function is provided with parameters representing 59 teaspoons then it should return the string "1 cup, 3 tablespoons, 2 teaspoons".

Hint: One cup is equivalent to 16 tablespoons. One tablespoon is equivalent to 3 teaspoons.

Exercise 103: Magic Dates

(*Solved—26 Lines*)

A magic date is a date where the day multiplied by the month is equal to the two digit year. For example, June 10, 1960 is a magic date because June is the sixth month, and 6 times 10 is 60, which is equal to the two digit year. Write a function that determines whether or not a date is a magic date. Use your function to create a main program that finds and displays all of the magic dates in the 20th century. You will probably find your solution to Exercise 100 helpful when completing this exercise.