Chapter 4

The name of the function is type. The expression in parentheses is called the

*argument* of the function. The argument is a value or variable that we are passing

into the function as input to the function. The result, for the type function, is the

type of the argument.

It is common to say that a function “takes” an argument and “returns” a result.

The result is called the *return value*.

The max function tells us the “largest character” in the string (which turns out to

be the letter “w”) and the min function shows us the smallest character (which

turns out to be a space).

Another very common built-in function is the len function which tells us how many

items are in its argument. If the argument to len is a string, it returns the number

of characters in the string.

These functions are not limited to looking at strings. They can operate on any set

of values, as we will see in later chapters.

You should treat the names of built-in functions as reserved words (i.e., avoid using

“max” as a variable name).

Python has a math module that provides most of the familiar mathematical functions.

Before we can use the module, we have to import it:

The module object contains the functions and variables defined in the module. To

access one of the functions, you have to specify the name of the module and the

name of the function, separated by a dot (also known as a period). This format is

called *dot notation*.

A *function definition* specifies the name of a new

function and the sequence of statements that execute when the function is called.

Once we define a function, we can reuse the function over and over throughout our

program.

def is a keyword that indicates that this is a function definition.

**algorithm** A general process for solving a category of problems.

**argument** A value provided to a function when the function is called. This value

is assigned to the corresponding parameter in the function.

**body** The sequence of statements inside a function definition.

**composition** Using an expression as part of a larger expression, or a statement

as part of a larger statement.

**deterministic** Pertaining to a program that does the same thing each time it

runs, given the same inputs.

**dot notation** The syntax for calling a function in another module by specifying

the module name followed by a dot (period) and the function name.

**flow of execution** The order in which statements are executed during a program

run.

**fruitful function** A function that returns a value.

**function** A named sequence of statements that performs some useful operation.

Functions may or may not take arguments and may or may not produce a

result.

**function call** A statement that executes a function. It consists of the function

name followed by an argument list.

**function definition** A statement that creates a new function, specifying its name,

parameters, and the statements it executes.

**function object** A value created by a function definition. The name of the function

is a variable that refers to a function object.

**header** The first line of a function definition.

**import statement** A statement that reads a module file and creates a module

object.

**module object** A value created by an import statement that provides access to

the data and code defined in a module.

**parameter** A name used inside a function to refer to the value passed as an

argument.

**pseudorandom** Pertaining to a sequence of numbers that appear to be random,

but are generated by a deterministic program.

**return value** The result of a function. If a function call is used as an expression,

the return value is the value of the expression.

**void function** A function that does not return a value.