4: FUNCTIONS

4.1: FUNCTION CALLS

In the context of programming, a function is a named sequence of statements that performs a computation. When you define a function, you specify the name and the sequence of statements. Later, you can "call" the function by name.

4.2: BUILT-IN FUNCTIONS

Python provides a number of important built-in functions that we can use without needing to provide the function definition. The creators of Python wrote a set of functions to solve common problems and included them in Python for us to use.

4.3: TYPE CONVERSION FUNCTIONS

Python also provides built-in functions that convert values from one type to another. The int function takes any value and converts it to an integer, if it can, or complains otherwise.

4.4: RANDOM NUMBERS

Given the same inputs, most computer programs generate the same outputs every time, so they are said to be deterministic. Determinism is usually a good thing, since we expect the same calculation to yield the same result. For some applications, though, we want the computer to be unpredictable. Games are an obvious example, but there are more.

4.5: MATH FUNCTIONS

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

4.6: ADDING NEW FUNCTIONS

So far, we have only been using the functions that come with Python, but it is also possible to add new functions. 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.

4.7: DEFINITIONS AND USES

As you might expect, you have to create a function before you can execute it. In other words, the function definition has to be executed before the first time it is called.

4.8: FLOW OF EXECUTION

In order to ensure that a function is defined before its first use, you have to know the order in which statements are executed, which is called the flow of execution. Execution always begins at the first statement of the program. Statements are executed one at a time, in order from top to bottom. Function definitions do not alter the flow of execution of the program, but remember that statements inside the function are not executed until the function is called.

4.9: PARAMETERS AND ARGUMENTS

Some of the built-in functions we have seen require arguments. For example, when you call math.sin you pass a number as an argument. Some functions take more than one argument: math.pow takes two, the base and the exponent. Inside the function, the arguments are assigned to variables called parameters.

4.10: FRUITFUL FUNCTIONS AND VOID FUNCTIONS

Some of the functions we are using, such as the math functions, yield results; for lack of a better name, I call them fruitful functions. Other functions, like print_twice, perform an action but don't return a value. They are called void functions.

4.11: WHY FUNCTIONS?

Throughout the rest of the book, often we will use a function definition to explain a concept. Part of the skill of creating and using functions is to have a function properly capture an idea such as "find the smallest value in a list of values". Later we will show you code that finds the smallest in a list of values and we will present it to you as a function namedÂ minÂ which takes a list of values as its argument and returns the smallest value in the list.

4.12: DEBUGGING

If you are using a text editor to write your scripts, you might run into problems with spaces and tabs. The best way to avoid these problems is to use spaces exclusively (no tabs). Most text editors that know about Python do this by default, but some don't.

4.E: FUNCTIONS (EXERCISES)

4.G: FUNCTIONS (GLOSSARY)