3 Working with Functions

In This Chapter

3.1 INTRODUCTION

Large programs are generally avoided because it is difficult to manage a single list of instructions. Thus, a large program is broken down into smaller units known as functions. A function is a named unit of a group of program statements. This unit can be invoked from other parts of the program.

The most important reason to use functions is to make program handling easier as only a small part of the program is dealt with at a time, thereby avoiding ambiguity. Another reason to use functions is to reduce program size. Functions make a program more readable and understandable to a programmer thereby making program management much easier.

In this chapter, we shall talk about functions, especially, how a function works ; how you can create your own functions in Python ; and how you can use the functions created by you.

FUNCTION

A *Function* is a subprogram that acts on data and often returns a value.

3.2 UNDERSTANDING FUNCTIONS

In order to understand what a function is, in terms of a programming language, read the following lines carefully.

You have worked with polynomials in Mathematics. Say we have following polynomial :

$$2x^2$$

For $x = 1$, it will give result as $2 \times 1^2 = 2$
For $x = 2$, it will give result as $2 \times 2^2 = 8$
For $x = 3$, it will give result as $2 \times 3^2 = 18$

and so on.

Now, if we represent above polynomial as somewhat like

$$f(x) = 2x^2$$

Then we can say (from above calculations) that

$$f(1) = 2 \quad ...(1)$$

$$f(2) = 8 \quad ...(2)$$

$$f(3) = 18 \quad ...(3)$$

The notation $f(x) = 2x^2$ can be termed as a **function**, where for function namely f, x is its argument *i.e.*, value given to it, and $2x^2$ is its functionality, *i.e.*, the functioning it performs. For different values of argument x, function $f(x)$ will return different results (refer to equations (1), (2) and (3) given above).

On the similar lines, programming languages also support functions. You can create functions in a program, that :

- can have arguments (*values given to it*), if needed
- can perform certain functionality (*some set of statements*)
- can return a result

For instance, above mentioned mathematical function $f(x)$ can be written in Python like this :

```
def calcSomething ( x ) :
    r = 2 * x ** 2
    return r
```

where

- **def** means a function definition is starting
- identifier following 'def' is the name of the function, *i.e.*, here the function name is ***calcSomething***
- the variables/identifiers inside the parentheses are the *arguments* or *parameters* (values given to function), *i.e.*, here x is the argument to function ***calcSomething***.
- there is a colon at the end of *def* line, meaning it requires a block

- the statements indented below the function, (*i.e.*, block below *def* line) define the functionality (working) of the function. This block is also called ***body-of-the-function***. Here, there are ***two statements in the body of function calcSomething.***
- The **return** statement returns the computed result.

The non-indented statements that are below the function definition are not part of the function **calcSomething's** definition. For instance, consider the example-function given in Fig. 3.1 below :

To see Function anatomy in action

Scan QR Code

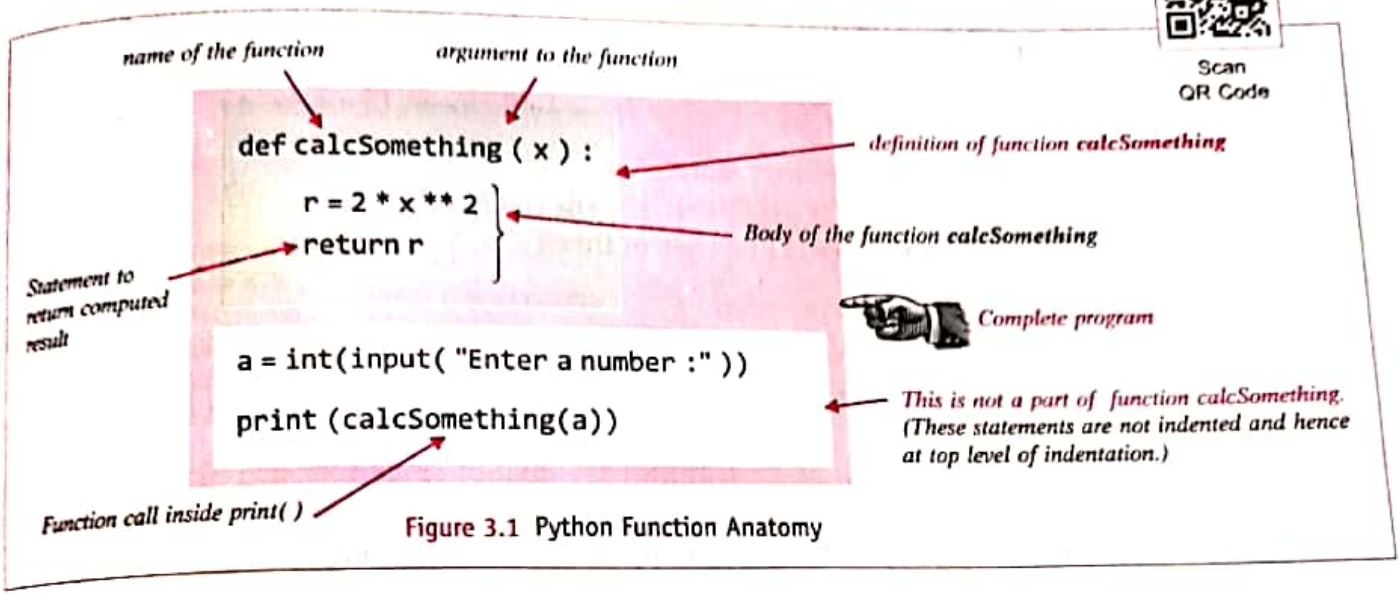

Figure 3.1 Python Function Anatomy

3.2.1 Calling/Invoking/Using a Function

To use a function that has been defined earlier, you need to write a *function call* statement in Python. A *function call* statement takes the following form :

```
<function-name>(<value-to-be-passed-to-argument>)
```

For example, if we want to call the function **calcSomething( )** defined above, our function call statement will be like :

```
calcSomething(5)          # value 5 is being sent as argument
```

Another function call for the same function, could be like :

```
a = 7
calcSomething(a)          # this time variable a is being sent as argument
```

Carefully notice that number of values being passed is same as number of parameters.

Also notice, in Fig. 3.1, the last line of the program uses a function call statement. (*print* ( ) is using the function call statement.)

To see Part of a function in action

Scan QR Code

Consider one more function definition given below :

```
def cube(x) :
    res = x ** 3          # cube of value in x
    return res            # return the computed value
```