Programming Tip 5.1

Function Comments

Whenever you write a function, you should *comment* its behavior. Comments are for human readers, not compilers. Various individuals prefer different layouts for function comments. In this book, we will use the following layout:

```
## Computes the volume of a cube.
#  @param sideLength the length of a side of the cube
#  @return the volume of the cube
#
def cubeVolume(sideLength) :
   volume = sideLength ** 3
   return volume
```

Function comments explain the purpose of the function, the meaning of the parameter variables and return value, as well as any special requirements.

This particular documentation style is borrowed from the Java programming language. It is supported by a wide variety of documentation tools such as Doxygen (`www.doxygen.org`), which extracts the documentation in HTML format from the Python source.

Each line of the function comment begins with a hash symbol (#) in the first column. The first line, which is indicated by two hash symbols, describes the purpose of the function. Each `@param` clause describes a parameter variable and the `@return` clause describes the return value.

There is an alternative (but, in our opinion, somewhat less descriptive) way of documenting the purpose of a Python function. Add a string, called a "docstring", as the first statement of the function body, like this:

```
def cubeVolume(sideLength) :
   "Computes the volume of a cube."
   volume = sideLength ** 3
   return volume
```

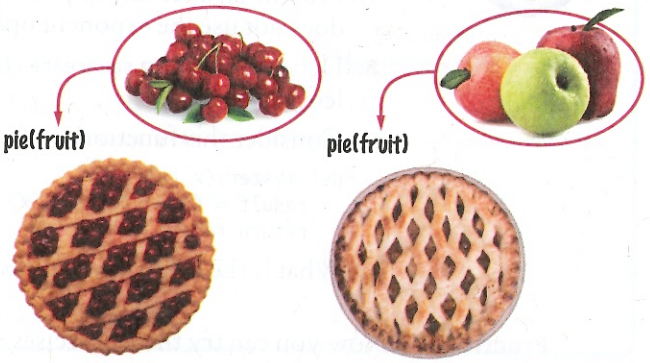
We don't use this style, but many Python programmers do.

Note that the function comment does not document the implementation (*how* the function does what it does) but rather the design (*what* the function does, its inputs, and its results). The comment allows other programmers to use the function as a "black box".

5.3 Parameter Passing

Parameter variables hold the arguments supplied in the function call.

In this section, we examine the mechanism of parameter passing more closely. When a function is called, variables are created for receiving the function's arguments. These variables are called **parameter variables**. (Another commonly used term is **formal parameters**.) The values that are supplied to the function when it is called are the **arguments** of the call. (These values are also commonly called the **actual parameters**.) Each parameter variable is initialized with the corresponding argument.



A recipe for a fruit pie may say to use any kind of fruit. Here, "fruit" is an example of a parameter variable. Apples and cherries are examples of arguments.