Generators are really just a special case of *iterators*. A function that yields values is a nice, compact way of building an iterator without building an iterator. Let me show you what I mean by that.

Remember the Fibonacci generator from Chapter 6? Listing 7-1 shows it as a built-from-scratch iterator:

*Listing 7-1. A Fibonacci Iterator*

class Fib:
    '''iterator that yields numbers in the Fibonacci sequence'''
    
    def __init__(self, max):
        self.max = max
    
    def __iter__(self):
        self.a = 0
        self.b = 1
        return self
    
    def __next__(self):
        fib = self.a
        if fib > self.max:
            raise StopIteration
        self.a, self.b = self.b, self.a + self.b
        return fib

Let’s take that one line at a time (see Listing 7-2).

*Listing 7-2. Class?*

class Fib:
    
    class? What’s a class?
Defining Classes

Python is fully object-oriented: you can define your own classes, inherit from your own or built-in classes, and instantiate the classes you define.

Defining a class in Python is simple. As with functions, there is no separate interface definition. Just define the class and then start coding. A Python class starts with the reserved word `class`, followed by the class name. Technically, that’s all that is required because a class doesn’t need to inherit from any other class, as shown in Listing 7-3.

Listing 7-3. The Simplest Class

class PapayaWhip:  
    pass           

The following notes refer to the numbered lines in Listing 7-3:

1. The name of this class is `PapayaWhip`, and it doesn’t inherit from any other class. Class names are usually capitalized, `EachWordLikeThis`, but this is only a convention, not a requirement.

2. You probably guessed this, but everything in a class is indented, just like the code within a function, `if` statement, `for` loop, or any other block of code. The first line not indented is outside the class.

This `PapayaWhip` class doesn’t define any methods or attributes, but syntactically there needs to be something in the definition thus the `pass` statement. This is a Python reserved word that just means “move along; nothing to see here”. It’s a statement that does nothing, and it’s a good placeholder when you’re stubbing out functions or classes.

---

The `pass` statement in Python is like an empty set of curly braces ( `{}` ) in Java or C.

---

Many classes are inherited from other classes, but this one is not. Many classes define methods, but this one does not. There is nothing that a Python class absolutely must have, other than a name. In particular, C++ programmers might find it odd that Python classes don’t have explicit constructors and destructors. Although it’s not required, Python classes can have something similar to a constructor: the `__init__` method.

The `__init__` Method

This example shows the initialization of the `Fib` class using the `__init__` method (see Listing 7-4).