Diving In

Iterators are the “secret sauce” of Python 3. They’re everywhere, underlying everything, always just out of sight. Comprehensions are just a simple form of *iterators*. Generators are just a simple form 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? Here it is as a built-from-scratch 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.

```
class Fib:
```

`class`? What’s a class?