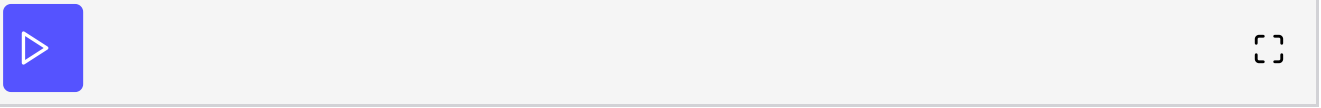
Instantiating Classes

Instantiating classes in Python is straightforward. To instantiate a class, simply call the class as if it were a function, passing the arguments that the `__init__()` method requires. The return value will be the newly created object.

```
import fibonacci2
fib = fibonacci2.Fib(100)          #①
print (fib)                        #②
#<fibonacci2.Fib object at 0x7f7594e75ba8>

print (fib.__class__)              #③
#<class 'fibonacci2.Fib'>

print (fib.__doc__ )               #④
#iterator that yields numbers in the Fibonacci sequence
```



① You are creating an instance of the `Fib` class (defined in the `fibonacci2` module) and assigning the newly created instance to the variable fib. You are passing one parameter, 100, which will end up as the max argument in Fib's `__init__()` method.

② `fib` is now an instance of the `Fib` class.

③ Every class instance has a built-in attribute, `__class__`, which is the object's class. Java programmers may be familiar with the `Class` class, which contains methods like `getName()` and `getSuperclass()` to get metadata information about an object. In Python, this kind of metadata is available through attributes, but the idea is the same.

④ You can access the instance's `docstring` just as with a function or a module. All instances of a class share the same docstring.

> *In Python, simply call a class as if it were a function to create a **new** instance of the class. There is no explicit new operator like there is in C++ or Java.*