Everything Is An Object

WE'LL COVER THE FOLLOWING

- What's An Object?

In case you missed it, I just said that Python functions have attributes, and that those attributes are available at runtime. A function, like everything else in Python, is an object.

Run the interactive Python shell and follow along:

```
import humansize                                #①

print(humansize.approximate_size(4096, True))  #②
#4.0 KiB

print(humansize.approximate_size.__doc__)      #③
#Convert a file size to human-readable form.

#    Keyword arguments:
#    size -- file size in bytes
#    a_kilobyte_is_1024_bytes -- if True (default), use multiples of 1024
#                                if False, use multiples of 1000

#    Returns: string
```

① The first line imports the `humansize` program as a module — a chunk of code that you can use interactively, or from a larger Python program. Once you import a module, you can reference any of its public functions, classes, or attributes. Modules can do this to access functionality in other modules, and you can do it in the Python interactive shell too. This is an important concept, and you'll see a lot more of it throughout this book.

② When you want to use functions defined in imported modules, you need to

include the module name. So you can't just say `approximate_size`; it must be

`humansize.approximate_size`. If you've used classes in Java, this should feel vaguely familiar.

③ Instead of calling the function as you would expect to, you asked for one of the function's attributes, `__doc__`.

> **import** in Python is like **require** in Perl. Once you **import** a Python module, you access its functions with **module.function**; once you **require** a Perl module, you access its functions with **module::function**.

What's An Object?

Everything in Python is an object, and everything can have attributes and methods. All functions have a built-in attribute `__doc__`, which returns the `docstring` defined in the function's source code. The `sys` module is an object which has (among other things) an attribute called `path`. And so forth.

Still, this doesn't answer the more fundamental question: what is an object? Different programming languages define "object" in different ways. In some, it means that `all` objects `must` have attributes and methods; in others, it means that all objects are subclassable. In Python, the definition is looser. Some objects have neither attributes nor methods, `but they could`. Not all objects are subclassable. But everything is an object in the sense that it can be assigned to a variable or passed as an argument to a function.

You may have heard the term "first-class object" in other programming contexts. In Python, functions are `first-class objects`. You can pass a function as an argument to another function. Modules are `first-class objects`. You can pass an entire module as an argument to a function. Classes are `first-class objects`, and individual instances of a class are also `first-class objects`.

This is important, so I'm going to repeat it in case you missed it the first few times: `everything in Python is an object`. Strings are objects. Lists are objects. Functions are objects. Classes are objects. Class instances are objects. Even modules are objects.