Introduction to super built-ins

The **super** built-in function was introduced way back in Python 2.2. The super function will return a proxy object that will delegate method calls to a parent or sibling class of type. If that was a little unclear, what it allows you to do is access inherited methods that have been overridden in a class. The super function has two use cases. The first is in single inheritance where super can be used to refer to the parent class or classes without actually naming them explicitly. This can make your code more maintainable in the future. This is similar to the behavior that you will find in other programming languages, like Dylan's *next-method*.

The second use case is in a dynamic execution environment where super supports cooperative multiple inheritance. This is actually a pretty unique use case that may only apply to Python as it is not found in languages that only support single inheritance nor in statically compiled languages.

super has had its fair share of controversy even among core developers. The original documentation was confusing and using super was tricky. There were some who even labeled super as harmful, although that article seems to apply more to the Python 2 implementation of super then the Python 3 version. We will start out this chapter by looking at how to call super in both Python 2 and 3. Then we will learn about **Method Resolution Order**.