концепция, рассматриваемая в той же главе 32, *метод класса*, предусматривает передачу класса взамен экземпляра во время вызова метода; она может использоваться для управления данными каждого класса и ее наличие подразумевается в метаклассах.

Тем не менее, указанные расширения являются более сложными и обычно необязательными. В нормальной ситуации методу всегда должен передаваться экземпляр – то ли автоматически при его вызове через экземпляр, то ли вручную, когда метод вызывается через класс.

Как было указано во врезке "Как насчет `super`?" в главе 28, язык Python также располагает встроенной функцией `super`, которая позволяет вызывать методы суперкласса более обобщенно, но из-за ее недостатков и сложностей мы отложим представление функции `super` до главы 32. Дополнительные сведения ищите в упомянутой врезке; с этим вызовом связаны известные компромиссы в случае базового применения, а также экзотический расширенный сценарий использования, который для наибольшей эффективности требует универсального развертывания. По причине проблем такого рода предпочтение в книге отдается политике обращения к суперклассам по явному имени, в не посредством встроенной функции `super`; если вы – новичок в Python, тогда я рекомендую пока применять такой же подход, особенно во время первоначального изучения ООП. Освойте сейчас простой способ, а позже вы сможете сравнить его с другими.

Наследование

Разумеется, весь смысл пространства имен, созданного оператором `class`, заключается в поддержке наследования имен. В этом разделе мы расширим ряд механизмов и ролей наследования атрибутов в Python.

Как было показано, наследование в Python происходит, когда объект уточняется, и оно инициирует поиск в дереве определения атрибутов – в одном и большем числе пространств имен. Каждый раз, когда вы используете выражение вида *`объект.атрибут`*, где *`объект`* представляет собой объект экземпляра или класса, Python осуществляет поиск в дереве пространств имен снизу вверх, начиная с *`объекта`*, с целью нахождения первого *`атрибута`*, который удастся обнаружить. Сюда входят ссылки на атрибуты `self` в ваших методах. Поскольку определения, расположенные ниже в дереве, переопределяют те, что находятся выше, наследование формирует основу специализации.

Построение дерева атрибутов

На рис. 29.1 иллюстрируется способ построения деревьев атрибутов и наполнение их именами. В целом:

- атрибуты экземпляров генерируются операторами присваивания значений атрибутам `self` в методах;
- атрибуты классов создаются операторами (присваивания) в операторах `class`;
- ссылки на суперклассы образуются на основе списков классов внутри круглых скобок в заголовке оператора `class`.