Функции имеют собственные локальные переменные, как и классы – объекты, существующие внутри модулей, которые начнут рассматриваться в следующей главе. Как было показано в части IV, функции тоже допускают вложение, но конечном итоге все содержится в модулях на верхнем уровне.

Сокрытие данных в модулях

Как мы уже видели, модуль Python экспортирует все имена, которым производится присваивание на верхнем уровне его файла. Понятие объявления, какие имена должны быть видны за пределами модуля, а какие нет, не существует. Фактически отсутствует способ запретить клиенту изменять имена внутри модуля, если у него возникнет такое желание.

В языке Python сокрытие данных в модулях является соглашением, а не синтаксическим ограничением. Если вы хотите нарушить работу модуля, уничтожив его имена, то вполне можете поступить так, но к счастью мне не приходилось встречать программиста, для которого это было бы жизненно важной целью. Некоторые сторонники пуризма возражают против такого свободного отношения к сокрытию данных, заявляя о том, что подобное положение вещей означает отсутствие возможности реализовать инкапсуляцию в Python. Тем не менее, инкапсуляция в Python больше связана с организацией пакетов, чем с установлением ограничений. В следующей части мы расширим эту идею в отношении классов, которые также не имеют синтаксиса закрытости, но часто способны эмулировать его в коде.

Сведение к минимуму вреда от `from *`: `_X` и `__all__`

В качестве особого случая вы можете снабжать имена префиксом в виде одиночного подчеркивания (например, `_X`), чтобы предотвратить их копирование, когда клиент импортирует имена модуля с помощью оператора `from *`. В действительности намерение такого действия заключается в том, чтобы свести к минимуму засорение пространства имен; поскольку `from *` копирует все имена, импортер может получить больше, чем он ожидал (в том числе имена, которые перезапишут имена в импортере). Подчеркивания не являются объявлениями "закрытых" имен: вы по-прежнему можете видеть и изменять такие имена посредством других форм импортирования, таких как оператор `import`:

```
# unders.py
a, _b, c, _d = 1, 2, 3, 4
```

```
>>> from unders import *    # Загружает только имена без подчеркиваний
>>> a, c
(1, 3)
>>> _b
NameError: name '_b' is not defined
Ошибка в имени: имя _b не определено
```

```
>>> import unders           # Но другие импортеры получают все имена
>>> unders._b
2
```

По-другому достичь эффекта сокрытия, похожего на соглашение по именованию `_X`, можно за счет присваивания переменной `__all__` списка со строками имен переменных на верхнем уровне модуля. В случае применения такой возможности оператор `from *` будет копировать только имена, перечисленные в списке `__all__`.