присваивания self.X относятся к тому же самому одиночному экземпляру, существует только один атрибут X — I.X — вне зависимости от того, сколько классов применяют такое имя атрибута.

Это не проблема, если она ожидается, и так в действительности взаимодействуют классы — экземпляр является разделяемой памятью. Тем не менее, чтобы гарантировать принадлежность атрибута классу, который его использует, необходимо снабдить имя префиксом в виде двух символов подчеркивания везде, где оно применяется в классе, как показано в следующем файле pseudoprivate.py, работающем в Python 2.X/3.X:

```
class C1:
    def meth1(self): self.__X = 88      # Теперь я имею свой атрибут X
    def meth2(self): print(self.__X)    # Становится _C1__X в I
class C2:
    def metha(self): self.__X = 99      # Я тоже
    def methb(self): print(self.__X)    # Становится _C2__X в I

class C3(C1, C2): pass
I = C3()                                # В I есть два имени X

I.meth1(); I.metha()
print(I.__dict__)
I.meth2(); I.methb()
```

При наличии таких префиксов перед добавлением к экземпляру атрибуты X будут расширены для включения имен своих классов. Если вы выполните вызов dir на I либо проинспектируете его словарь пространства имен после присваивания значений атрибутам, то увидите расширенные имена, _C1__X и _C2__X, а не X. Поскольку расширение делает имена более уникальными внутри экземпляра, разработчики классов могут вполне безопасно предполагать, что они по-настоящему владеют любыми именами, которые снабжают префиксами в форме двух символов подчеркивания:

```
% python pseudoprivate.py
{'_C2__X': 99, '_C1__X': 88}
88
99
```

Показанный трюк позволяет избежать потенциальных конфликтов имен в экземпляре, но учтите, что он не означает подлинную защиту. Зная имя включающего класса, вы по-прежнему можете получать доступ к любому из двух атрибутов везде, где имеется ссылка на экземпляр, с использованием полностью развернутого имени (например, I._C1__X = 77). Кроме того, имена все еще могут конфликтовать, когда неосведомленные программисты явно применяют расширенную схему именования (маловероятно, но возможно). С другой стороны, описанное средство делает менее вероятными *случайные* конфликты с именами какого-то класса.

Псевдозакрытые атрибуты также полезны в более крупных фреймворках либо инструментах, помогая избежать введения новых имен методов, которые могут неумышленно скрыть определения где-то в другом месте дерева классов, и снизить вероятность замещения внутренних методов именами, определяемыми ниже в дереве. Если метод предназначен для использования только внутри класса, который может смешиваться с другими классами, тогда префикс в виде двух символов подчеркивания фактически гарантирует, что снабженный им метод не будет служить препятствием другим именам в дереве, особенно в сценариях с множественным наследованием: