Шаг 1: создание экземпляров

Итак, закончим стадию проектирования и приступим к реализации. Наша первая задача – начать написание кода главного класса, `Person`. Откроем текстовый редактор и создадим новый файл для кода, который будет написан. В Python принято довольно строгое соглашение начинать имена модулей с буквы нижнего регистра, а имена классов – с буквы верхнего регистра. Как и имя аргументов `self` в методах, язык этого не требует, но соглашение получило настолько широкое распространение, что отклонение от него может сбить с толку тех, кто впоследствии будет читать ваш код. Для соответствия соглашению мы назовем новый файл модуля `person.py` и назначим классу внутри него имя `Person`:

```
# Файл person.py (начало)
class Person:    # Начало класса
```

Вся работа внутри файла будет делаться далее в главе. В одном файле модуля Python можно создавать любое количество функций и классов, поэтому имя файла `person.py` может утратить смысл, если позже мы добавим в данный файл несвязанные компоненты. Пока что мы предположим, что абсолютно все в этом файле будет иметь отношение к `Person`. Вероятно, так и должно быть в любом случае – как выяснится, модули работают лучше всего, когда они преследуют единую цель *сцепления*.

Написание кода конструкторов

Первое, что мы хотим делать с помощью класса `Person`, связано с регистрацией основных сведений о людях – заполнением полей записей, если так понятнее. Разумеется, в терминологии Python они известны как *атрибуты* объекта экземпляра и обычно создаются путем присваивания значений атрибутам `self` в функциях методов класса. Нормальный способ предоставления атрибутам экземпляра первоначальных значений предусматривает их присваивание через `self` в *методе конструктора* `__init__`, который содержит код, автоматически выполняемый Python каждый раз, когда создается экземпляр. Давайте добавим к классу метод конструктора:

```
# Добавление инициализации полей записи
class Person:
    def __init__(self, name, job, pay): # Конструктор принимает три аргумента
        self.name = name                # Заполнить поля при создании
        self.job = job                  # self - новый объект экземпляра
        self.pay = pay
```

Такая схема написания кода весьма распространена: мы передаем данные, подлежащие присоединению к экземпляру, в виде аргументов методу конструктора и присваиваем их атрибутам `self`, чтобы сохранить их на постоянной основе. В терминах ООП аргумент `self` является вновь созданным объектом экземпляра, а `name`, `job` и `pay` становятся *информацией о состоянии* – описательными данными, сохраняемыми в объекте для использования в будущем. Хотя другие методики (такие как замыкания вложенных областей видимости) тоже способны сохранять детали, атрибуты экземпляра делают это очень явным и легким для понимания.

Обратите внимание, что имена аргументов здесь встречаются *дважды*. Поначалу код может даже показаться несколько избыточным, но это не так. Например, аргумент `job` представляет собой локальную переменную в области видимости функции `__init__`, но `self.job` – атрибут экземпляра, в котором передается подразумевае-