8.17 创建不调用init方法的实例¶

问题¶

你想创建一个实例，但是希望绕过执行 `__init__()` 方法。

解决方案¶

可以通过 `__new__()` 方法创建一个未初始化的实例。例如考虑如下这个类：

```
class Date:
    def __init__(self, year, month, day):
        self.year = year
        self.month = month
        self.day = day
```

下面演示如何不调用 `__init__()` 方法来创建这个Date实例：

```
>>> d = Date.__new__(Date)
>>> d
<__main__.Date object at 0x1006716d0>
>>> d.year
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: 'Date' object has no attribute 'year'
>>>
```

结果可以看到，这个Date实例的属性year还不存在，所以你需要手动初始化：

```
>>> data = {'year':2012, 'month':8, 'day':29}
>>> for key, value in data.items():
...     setattr(d, key, value)
...
>>> d.year
2012
>>> d.month
8
>>>
```

讨论¶

当我们在反序列对象或者实现某个类方法构造函数时需要绕过 `__init__()` 方法来创建对象。 例如，对于上面的Date来讲，有时候你可能会像下面这样定义一个新的构造函数 `today()` ：

```
from time import localtime

class Date:
    def __init__(self, year, month, day):
        self.year = year
        self.month = month
        self.day = day

    @classmethod
```

```
def today(cls):
    d = cls.__new__(cls)
    t = localtime()
    d.year = t.tm_year
    d.month = t.tm_mon
    d.day = t.tm_mday
    return d
```

同样，在你反序列化JSON数据时产生一个如下的字典对象：

```
data = { 'year': 2012, 'month': 8, 'day': 29 }
```

如果你想将它转换成一个Date类型实例，可以使用上面的技术。

当你通过这种非常规方式来创建实例的时候，最好不要直接去访问底层实例字典，除非你真的清楚所有细节。 否则的话，如果这个类使用了 `__slots__` 、properties 、descriptors 或其他高级技术的时候代码就会失效。 而这时候使用 `setattr()` 方法会让你的代码变得更加通用。