3.15 字符串转换为日期¶

问题¶

你的应用程序接受字符串格式的输入，但是你想将它们转换为 `datetime` 对象以便在上面执行非字符串操作。

解决方案¶

使用Python的标准模块 `datetime` 可以很容易的解决这个问题。比如：

```
>>> from datetime import datetime
>>> text = '2012-09-20'
>>> y = datetime.strptime(text, '%Y-%m-%d')
>>> z = datetime.now()
>>> diff = z - y
>>> diff
datetime.timedelta(3, 77824, 177393)
>>>
```

讨论¶

`datetime.strptime()` 方法支持很多的格式化代码， 比如 `%Y` 代表4位数年份， `%m` 代表两位数月份。 还有一点值得注意的是这些格式化占位符也可以反过来使用，将日期输出为指定的格式字符串形式。

比如，假设你的代码中生成了一个 `datetime` 对象， 你想将它格式化为漂亮易读形式后放在自动生成的信件或者报告的顶部：

```
>>> z
datetime.datetime(2012, 9, 23, 21, 37, 4, 177393)
>>> nice_z = datetime.strftime(z, '%A %B %d, %Y')
>>> nice_z
'Sunday September 23, 2012'
>>>
```

还有一点需要注意的是， `strptime()` 的性能要比你想象中的差很多， 因为它是使用纯Python实现，并且必须处理所有的系统本地设置。 如果你要在代码中需要解析大量的日期并且已经知道了日期字符串的确切格式，可以自己实现一套解析方案来获取更好的性能。 比如，如果你已经知道所以日期格式是 `YYYY-MM-DD` ，你可以像下面这样实现一个解析函数：

```
from datetime import datetime
def parse_ymd(s):
    year_s, mon_s, day_s = s.split('-')
    return datetime(int(year_s), int(mon_s), int(day_s))
```

实际测试中，这个函数比 `datetime.strptime()` 快7倍多。 如果你要处理大量的涉及到日期的数据的话，那么最好考虑下这个方案！