2.1会话对象Session()

会话对象让你能够跨请求保持某些参数。它也会在同一个 Session 实例发出的所有请求之间保持 cookie， 期间使用 urllib3 的 connection pooling 功能。所以如果你向同一主机发送多个请求，底层的 TCP 连接将会被重用，从而带来显著的性能提升。 (参见 HTTP persistent connection).

会话对象具有主要的 Requests API 的所有方法。

我们来跨请求保持一些 cookie:

```
s = requests.Session()

s.get('http://httpbin.org/cookies/set/sessioncookie/123456789')
r = s.get("http://httpbin.org/cookies")

print(r.text)
# '{"cookies": {"sessioncookie": "123456789"}}'
```

会话也可用来为请求方法提供缺省数据。这是通过为会话对象的属性提供数据来实现的：

```
s = requests.Session()
s.auth = ('user', 'pass')
s.headers.update({'x-test': 'true'})

# both 'x-test' and 'x-test2' are sent
s.get('http://httpbin.org/headers', headers={'x-test2': 'true'})
```

任何你传递给请求方法的字典都会与已设置会话层数据合并。方法层的参数覆盖会话的参数。

不过需要注意，就算使用了会话，方法级别的参数也不会被跨请求保持。下面的例子只会和第一个请求发送 cookie ，而非第二个：

```
s = requests.Session()

r = s.get('http://httpbin.org/cookies', cookies={'from-my': 'browser'})
print(r.text)
# '{"cookies": {"from-my": "browser"}}'

r = s.get('http://httpbin.org/cookies')
print(r.text)
# '{"cookies": {}}'
```

如果你要手动为会话添加 cookie，就是用 Cookie utility 函数 来操纵 **Session.cookies**。

会话还可以用作前后文管理器：

```
with requests.Session() as s:
    s.get('http://httpbin.org/cookies/set/sessioncookie/123456789')
```

这样就能确保 with 区块退出后会话能被关闭，即使发生了异常也一样。

从字典参数中移除一个值

有时你会想省略字典参数中一些会话层的键。要做到这一点，你只需简单地在方法层参数中将那个键的值设置为 None ，那个键就会被自动省略掉。