11.7 在不同的Python解释器之间交互¶

问题¶

你在不同的机器上面运行着多个Python解释器实例，并希望能够在这些解释器之间通过消息来交换数据。

解决方案¶

通过使用 `multiprocessing.connection` 模块可以很容易的实现解释器之间的通信。 下面是一个简单的应答服务器例子：

```
from multiprocessing.connection import Listener
import traceback

def echo_client(conn):
    try:
        while True:
            msg = conn.recv()
            conn.send(msg)
    except EOFError:
        print('Connection closed')

def echo_server(address, authkey):
    serv = Listener(address, authkey=authkey)
    while True:
        try:
            client = serv.accept()

            echo_client(client)
        except Exception:
            traceback.print_exc()

echo_server(('', 25000), authkey=b'peekaboo')
```

然后客户端连接服务器并发送消息的简单示例：

```
>>> from multiprocessing.connection import Client
>>> c = Client(('localhost', 25000), authkey=b'peekaboo')
>>> c.send('hello')
>>> c.recv()
'hello'
>>> c.send(42)
>>> c.recv()
42
>>> c.send([1, 2, 3, 4, 5])
>>> c.recv()
[1, 2, 3, 4, 5]
>>>
```

跟底层socket不同的是，每个消息会完整保存（每一个通过send()发送的对象能通过recv()来完整接受）。 另外，所有对象会通过pickle序列化。因此，任何兼容pickle的对象都能在此连接上面被发送和接受。

讨论¶

目前有很多用来实现各种消息传输的包和函数库，比如ZeroMQ、Celery等。 你还有另外一种选择就是自己在底层

socket基础之上来实现一个消息传输层。 但是你想要简单一点的方案，那么这时候 multiprocessing.connection 就派上用场了。 仅仅使用一些简单的语句即可实现多个解释器之间的消息通信。

如果你的解释器运行在同一台机器上面，那么你可以使用另外的通信机制，比如Unix域套接字或者是Windows命名管道。 要想使用UNIX域套接字来创建一个连接，只需简单的将地址改写一个文件名即可：

```
s = Listener('/tmp/myconn', authkey=b'peekaboo')
```

要想使用Windows命名管道来创建连接，只需像下面这样使用一个文件名：

```
s = Listener(r'\\.\pipe\myconn', authkey=b'peekaboo')
```

一个通用准则是，你不要使用 multiprocessing 来实现一个对外的公共服务。 Client() 和 Listener() 中的 authkey 参数用来认证发起连接的终端用户。 如果密钥不对会产生一个异常。此外，该模块最适合用来建立长连接（而不是大量的短连接）， 例如，两个解释器之间启动后就开始建立连接并在处理某个问题过程中会一直保持连接状态。

如果你需要对底层连接做更多的控制，比如需要支持超时、非阻塞I/O或其他类似的特性， 你最好使用另外的库或者是在高层socket上来实现这些特性。