选题: web代理服务器

姓名:陈实

学号: 2019141230153

完成时间: 2022/6/15

开发工具: VScode, 控制台

编程语言:Python

程序框架:

第一步: client与proxy 建立TCP连接

第二部: client 向proxy发送http请求(带有请求的时间戳和请求的资源名)

第三步: proxy从请求中解析出请求的文件与发起请求的时间戳

第四步: 检查cache上有无请求的文件

1.如果有: 计算diff = B-A

1)如果diff大于设定的时间值(用户输入的), 进行conditional get(检查B是否大于C), 如果文件未被修改, 则直接从cache中返回请求的文件(也要更新B), 如果文件被修改过, 则proxy与server建立TCP连接,从server端返回请求的文件资源,并更新在proxy上的文件.

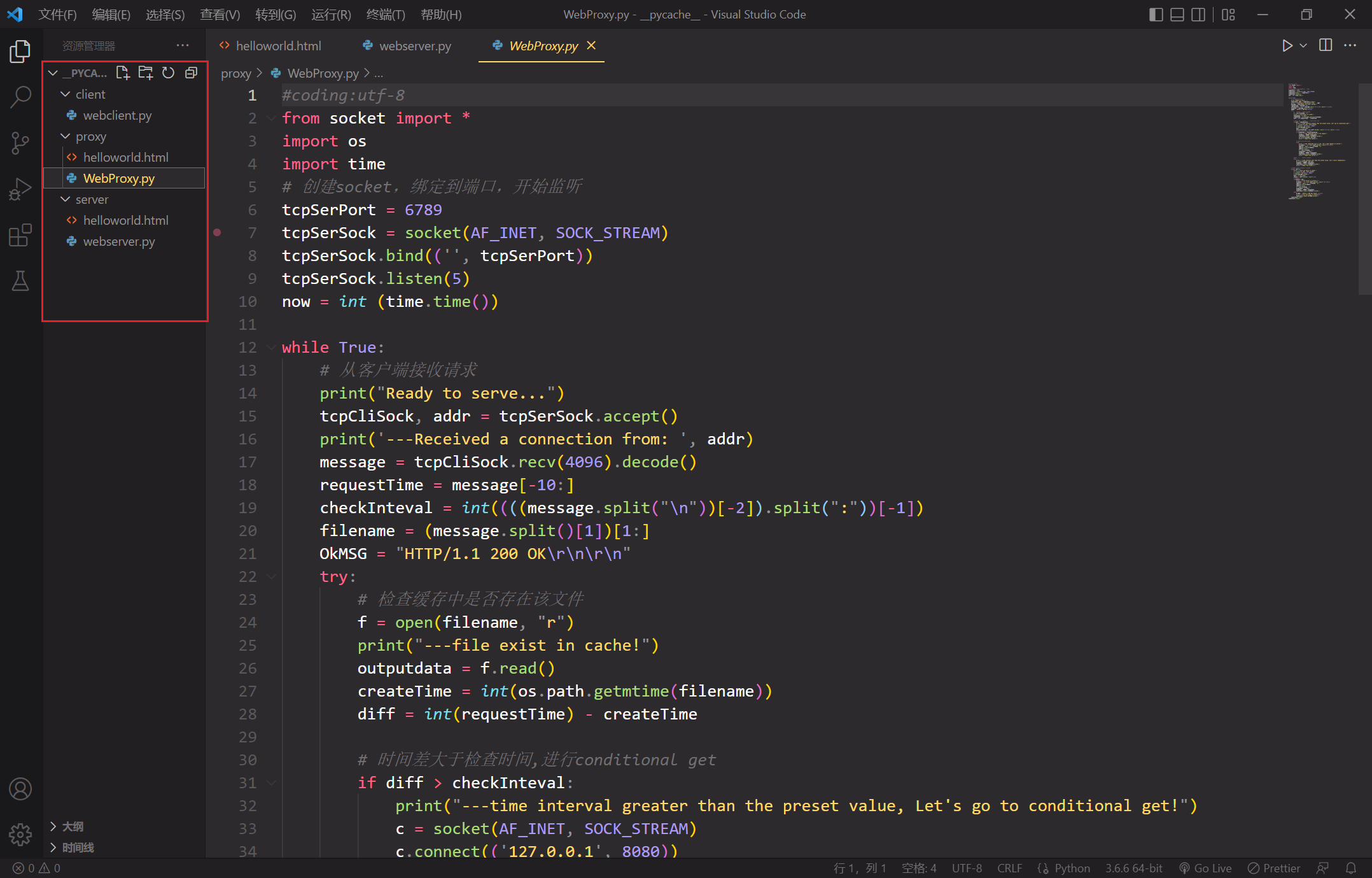
2) 如果diff小于设定的时间戳, 直接从cache返回请求文件并修改B.

2.如果没有: proxy与server建立TCP连接,请求资源

1)server端有该请求资源: 返回给proxy, proxy保存该资源并返回给client

2)server端没有该请求资源: 返回404 NOT FOUND.

二. 代码结构



如图, 项目目录下一共分为了3个文件夹: client, server, proxy,各自内部都有对应的python代码, 结构清晰明了, proxy是该项目的主要要求, 在要求之上,我又额外实现了多线程, 而client和server的实现是为了验证proxy的功能, 总体来说,整个项目结构和功能都非常完整, 这里仅对client稍作解释: 之所以没有用浏览器来实现client的原因在于, 最初用浏览器来作为client的时候发现及其容易被网络上的其他包数据所干扰, 对代码的测试难度较大.

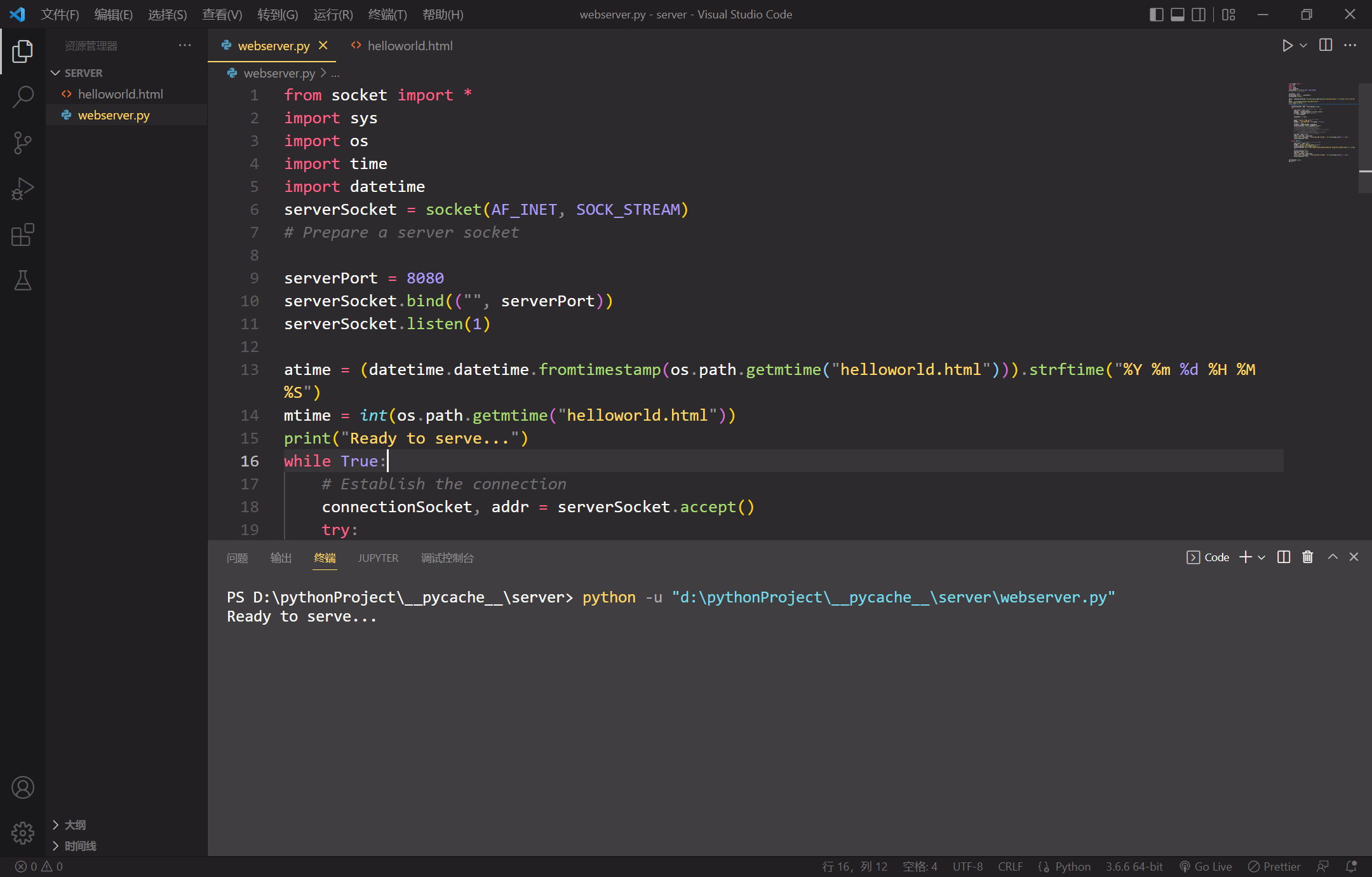
三: 运行示例

这里本人选择在VScode上运行Server和Proxy, 在cmd运行client(方便输入命令行参数), 如果没有Vscode, 也可以在IDE或者其他装有python代码解释器的编辑器运行,也

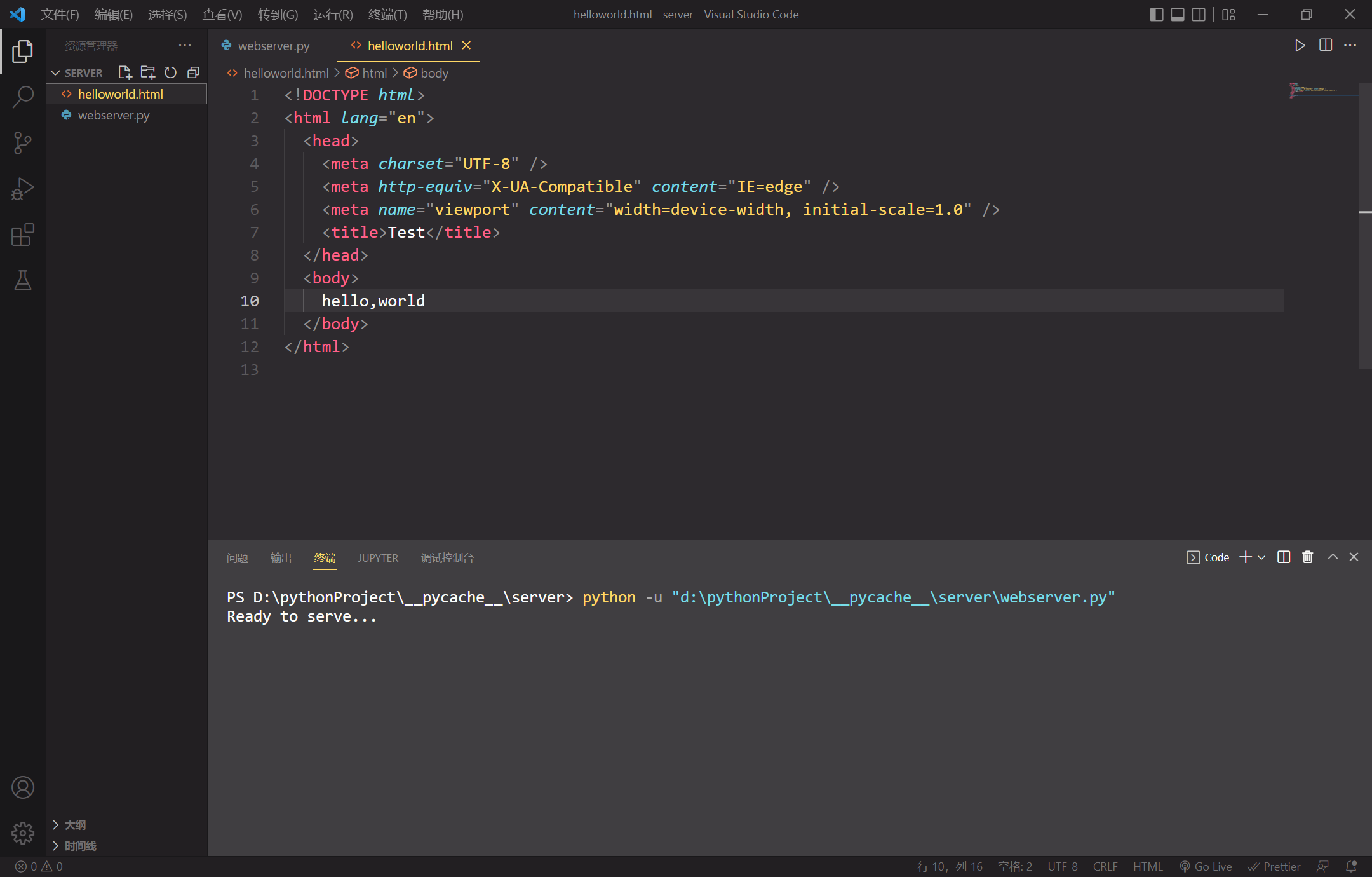
可以开3个命令行分别运行server,proxy,client.

流程:

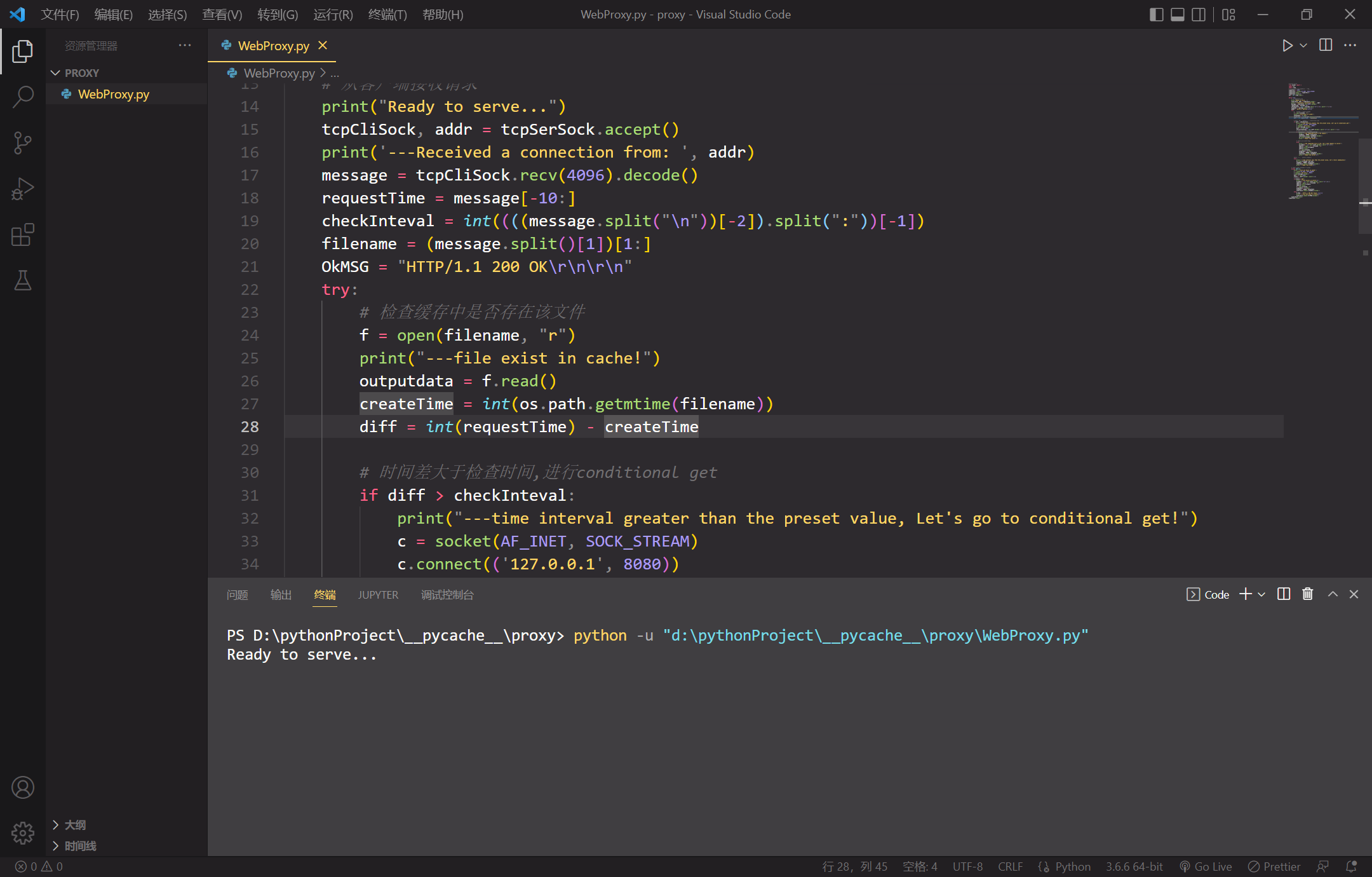
1.首先,运行server:



此时server端的文件helloworld.html如下:



2. 再运行proxy(如下图):

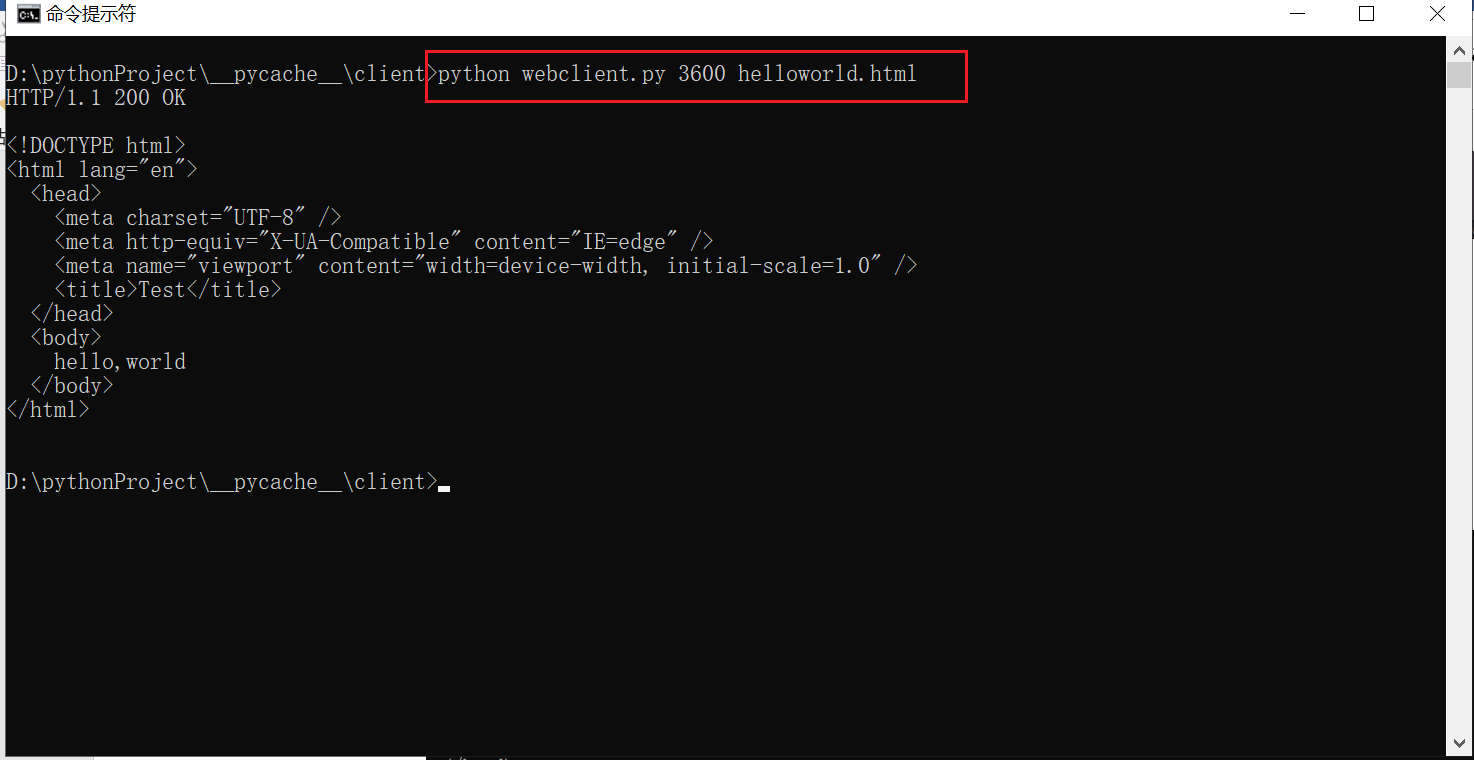


注意,此时proxy上并没有存放hellowolrd.html文件

3. 运行client

这里以proxy的几个功能为划分点,依次展示:

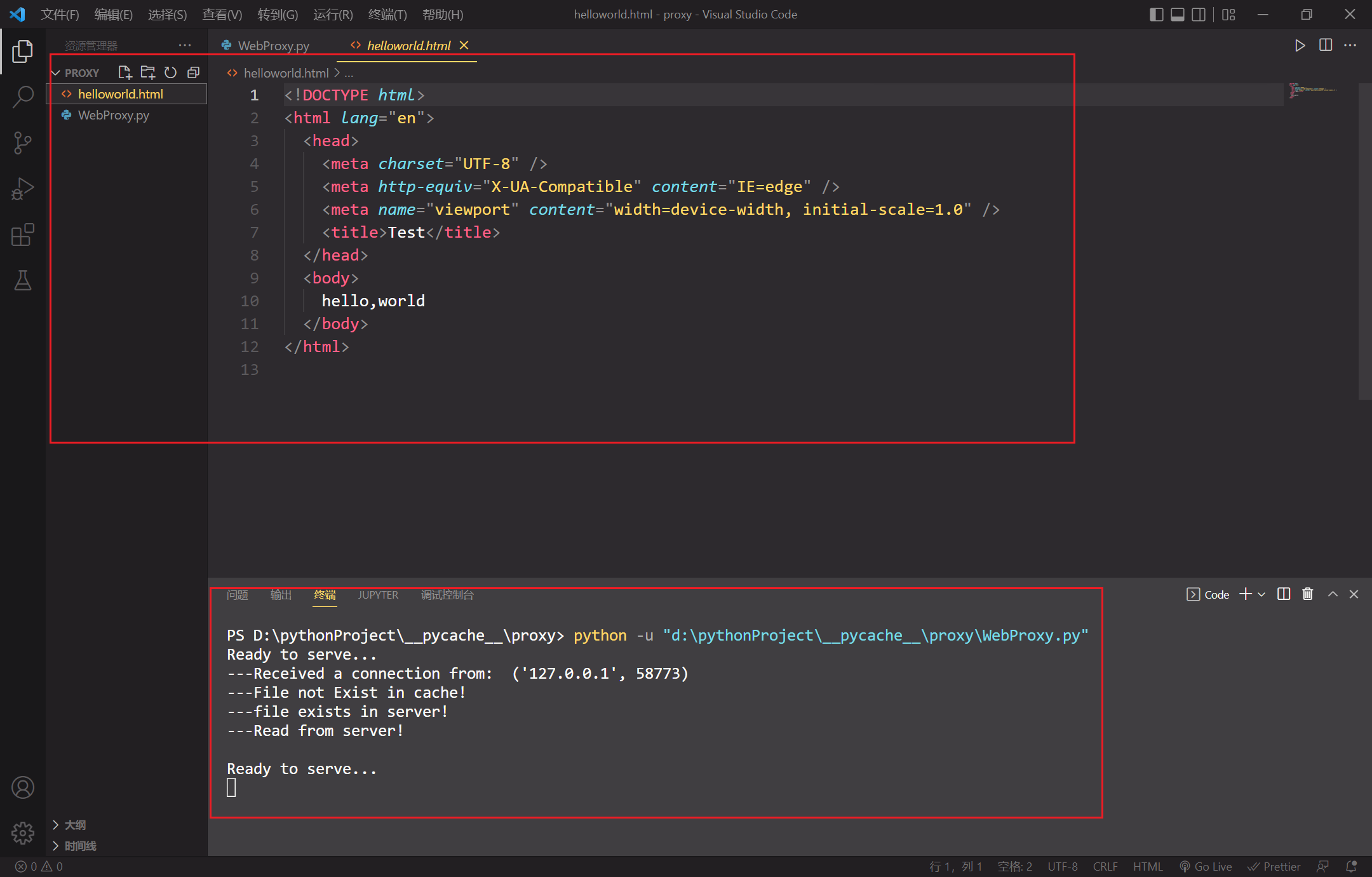
1. Proxy上并没有对应的文件(如上图所示),



这里稍作解释, 3600是用户输入的时间(单位:s, 即在这个时间之后proxy要进行conditional get), helloworld.html是请求的文件名.

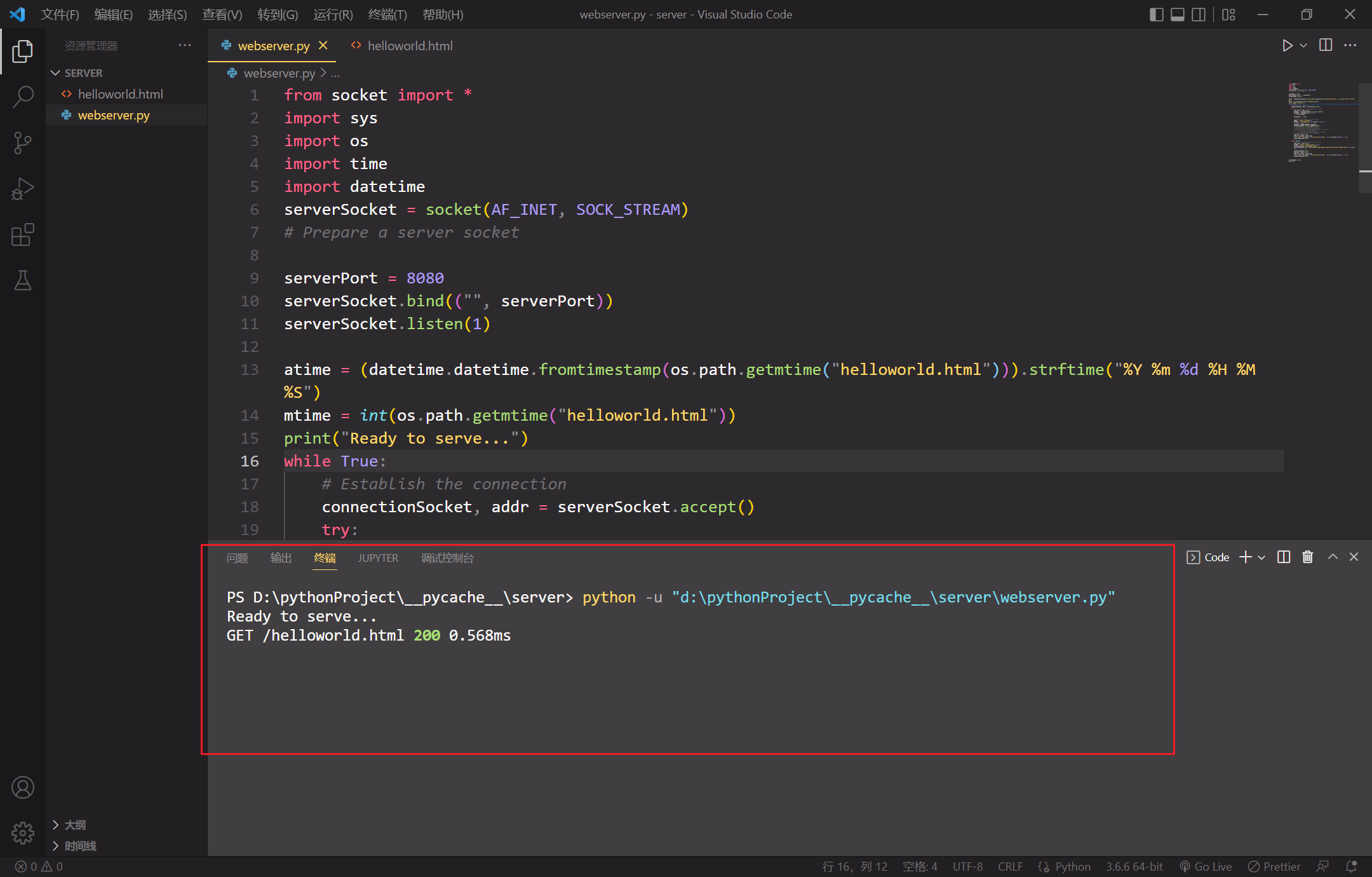
注意到, 在cmd打印出了请求的文件(模拟了浏览器收到的response), 即client成功收到了请求文件.

Proxy端的响应:



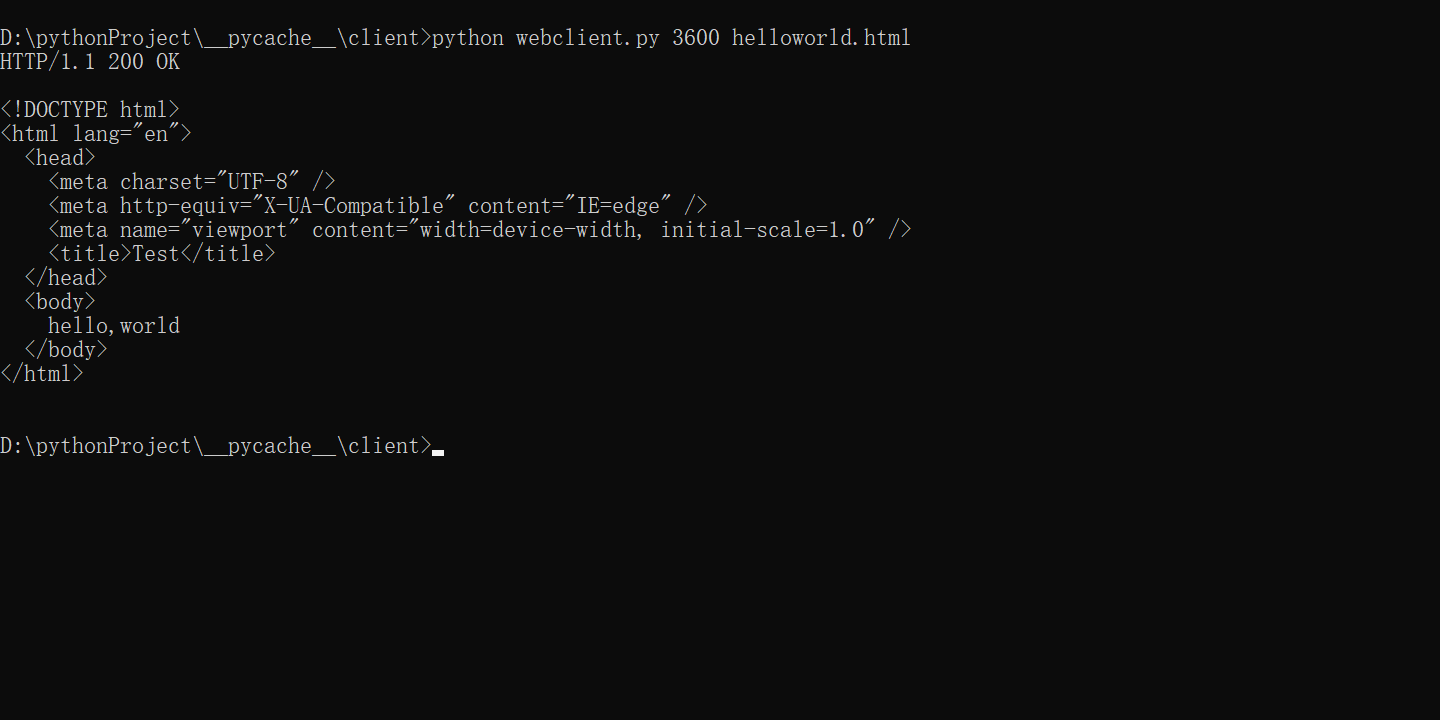
如上图, 在终端中给出了友好的代码提示, 首先与client建立了连接, 然后发现cache中没有文件, 于是向server发出请求,发现server中有文件便从server中返回了文件. 同时也创立了helloworld.html文件,与server中的helloworld.html完全一致.

Server端的响应:



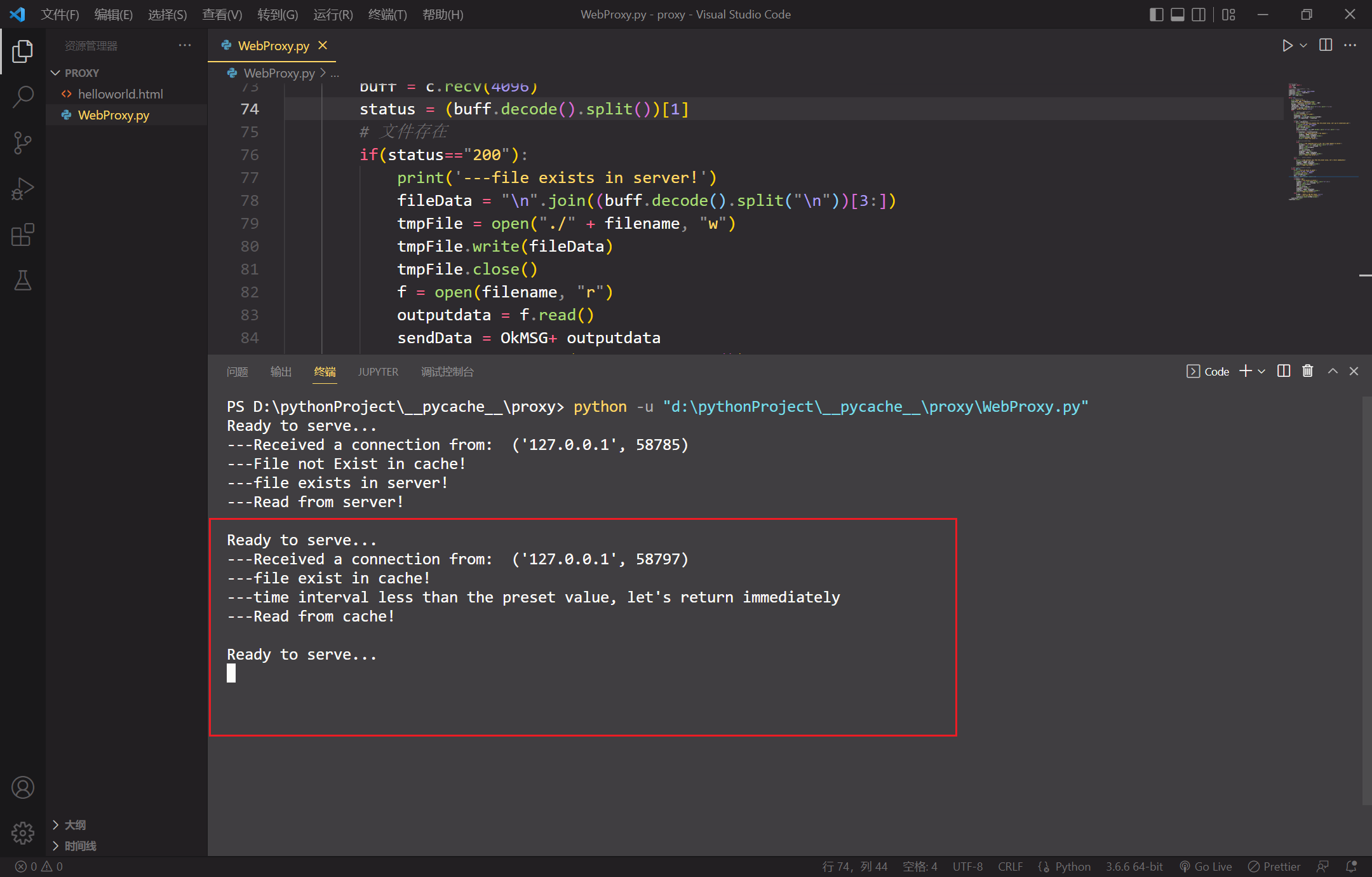
打印出了请求相关的信息,

1. 第二次请求, proxy已经有了文件,设置checkTime的时间为3600,即不进行conditional get Client端进行请求:



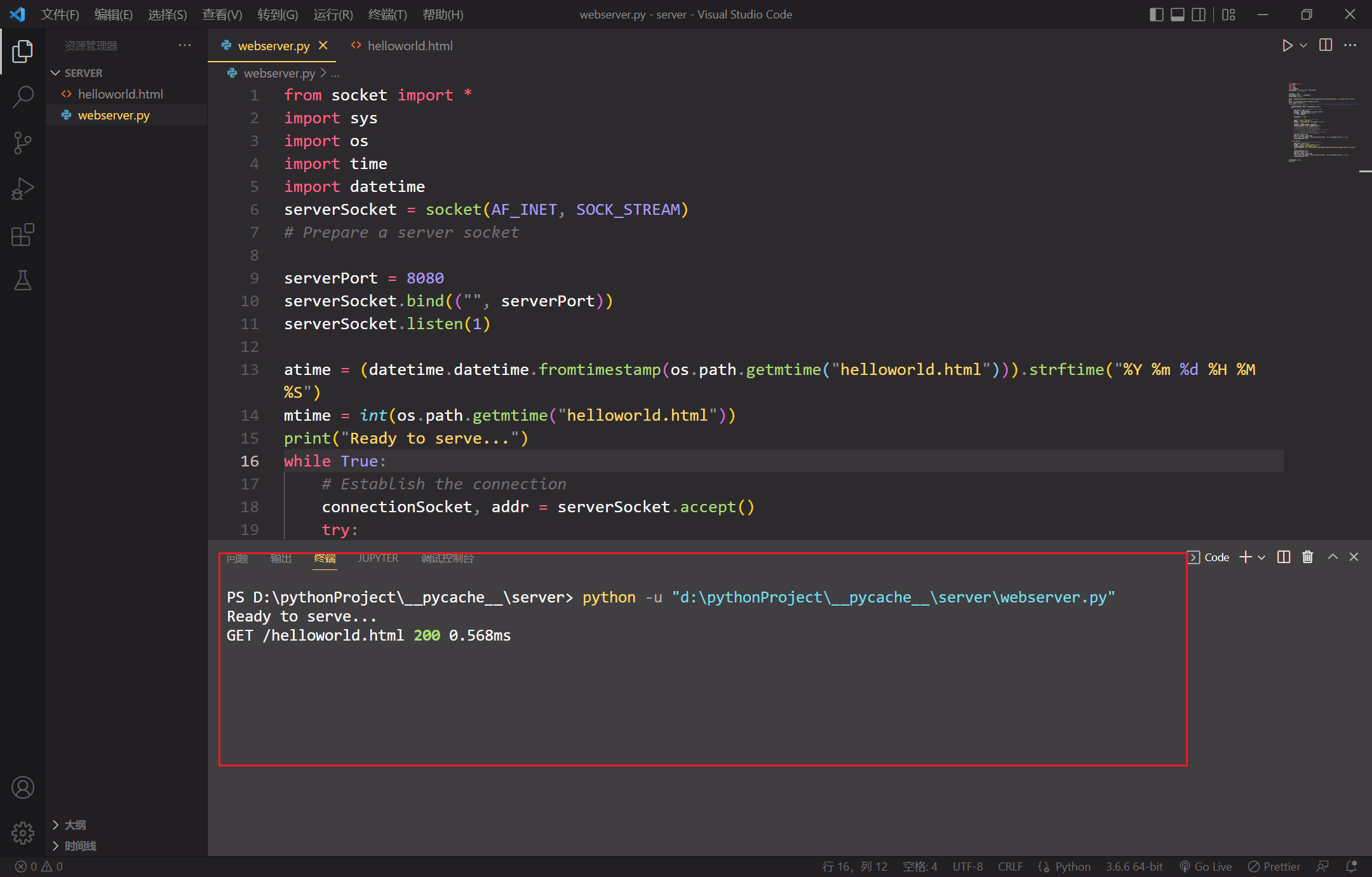
如上图,同样收到了完整的response.

Proxy端的响应:



打印出了请求的信息, 包括建立连接, 发现文件存在, 不需要进行conditional get, 直接返回.

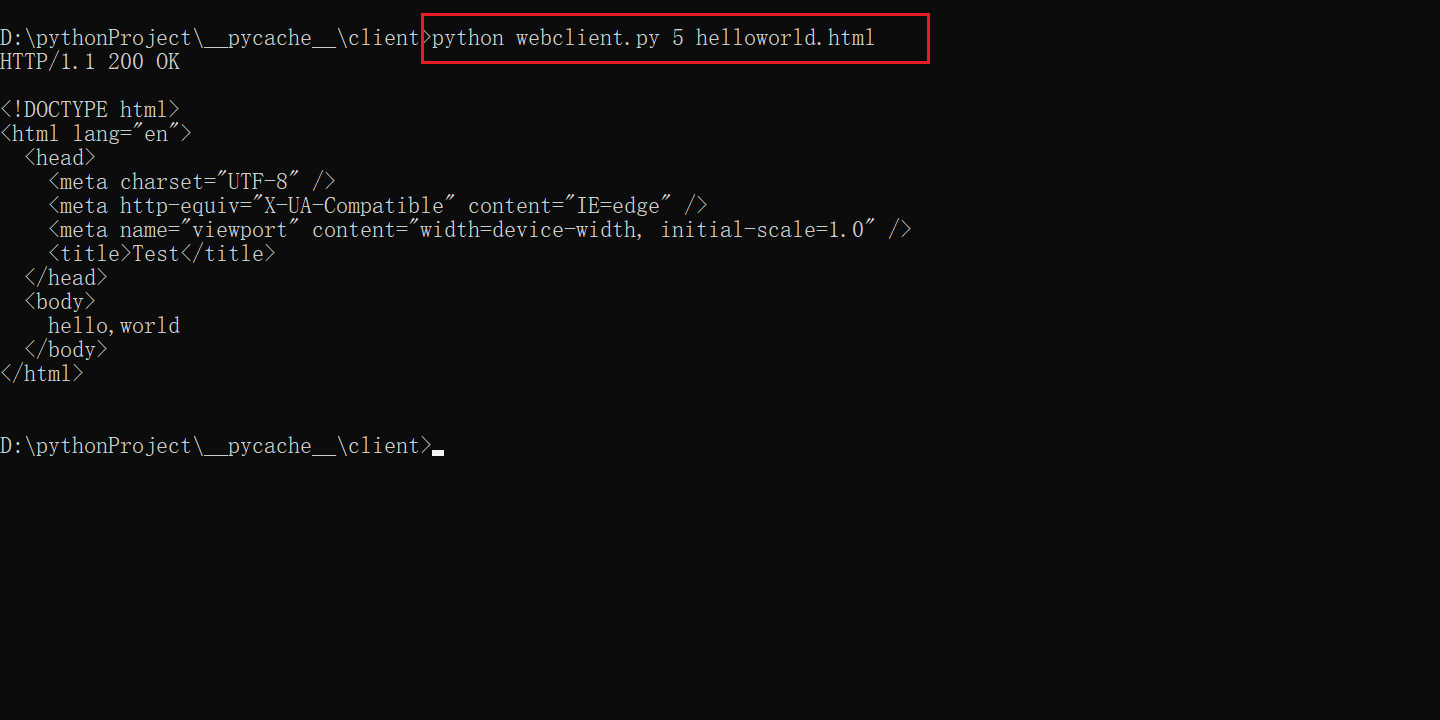
Server端的响应:



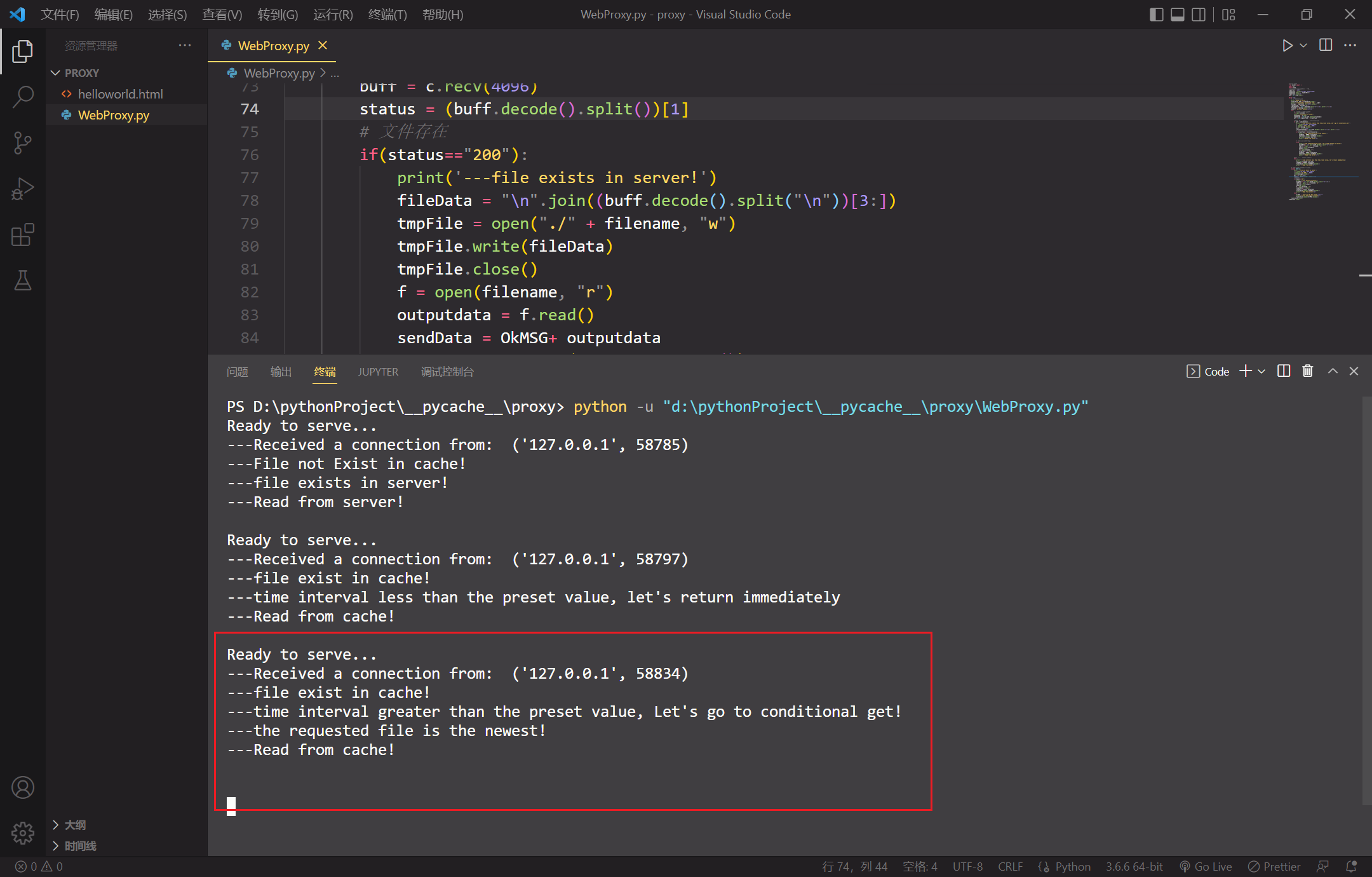
如图,并没有新的请求进来.

1. 第三次请求, 这次设置conditional get的时间为5s, 进行condition get:

Client: 再次受到了请求

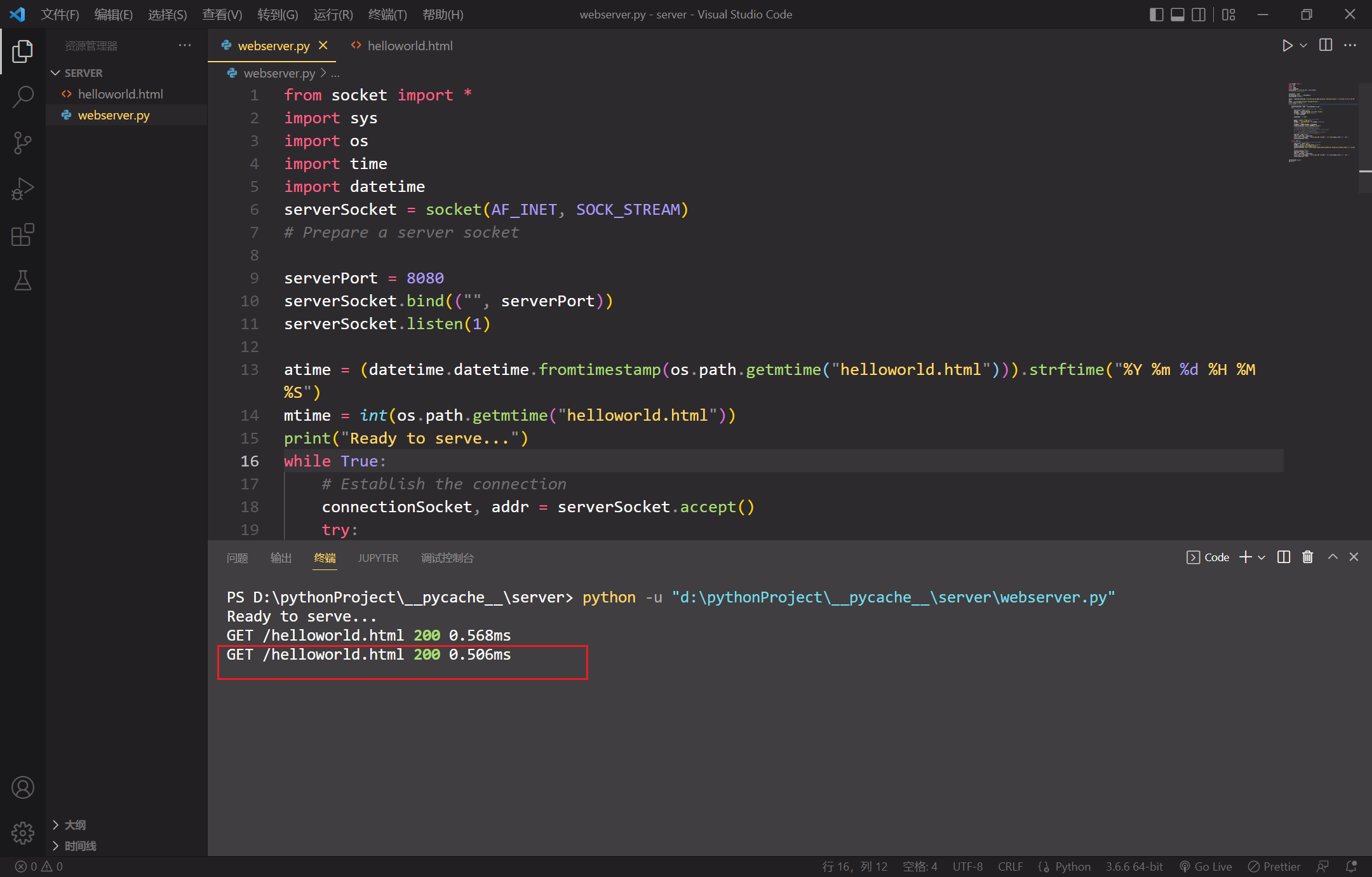


Proxy端响应:



如图, proxy进行了conditional get, 但发现文件没有被修改过, 直接从cache中返回结果.

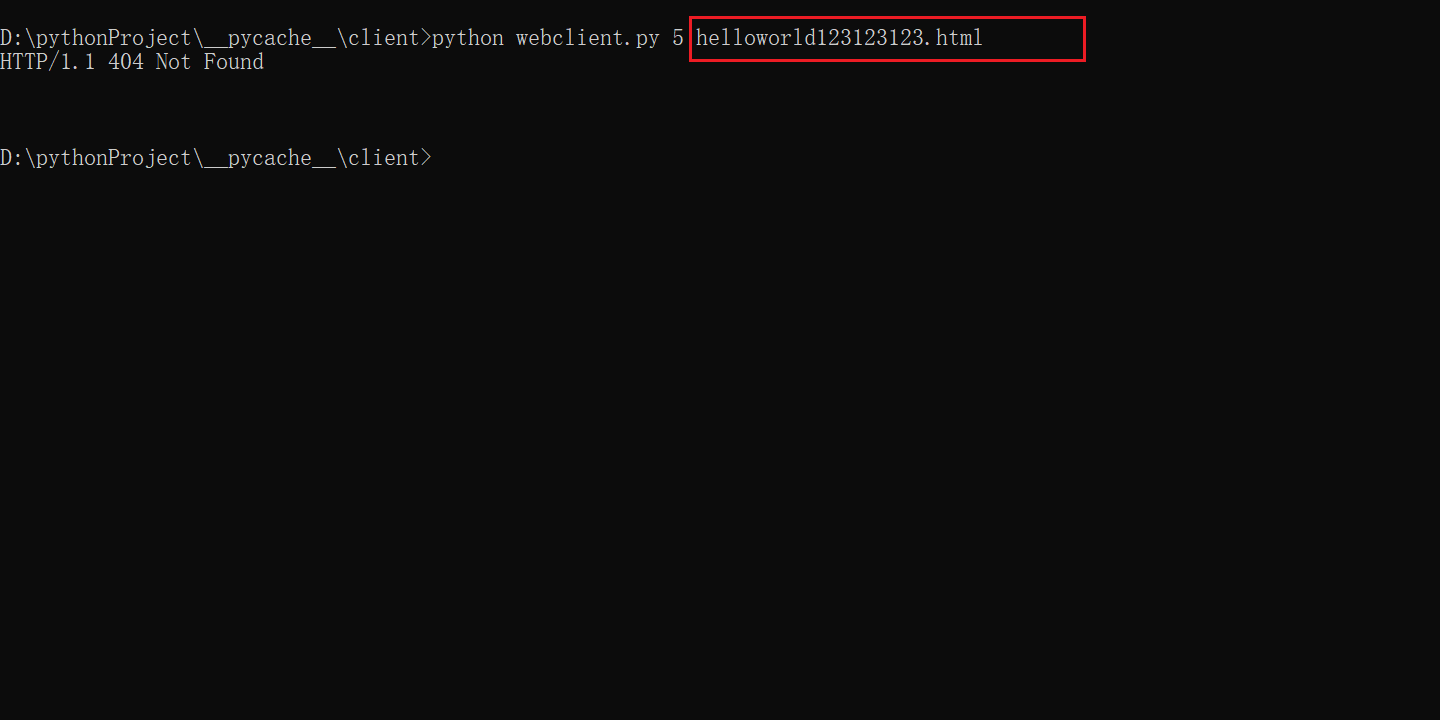
Server端响应:



如我们所期望的一致, 这一次向server进行了请求(conditional get)

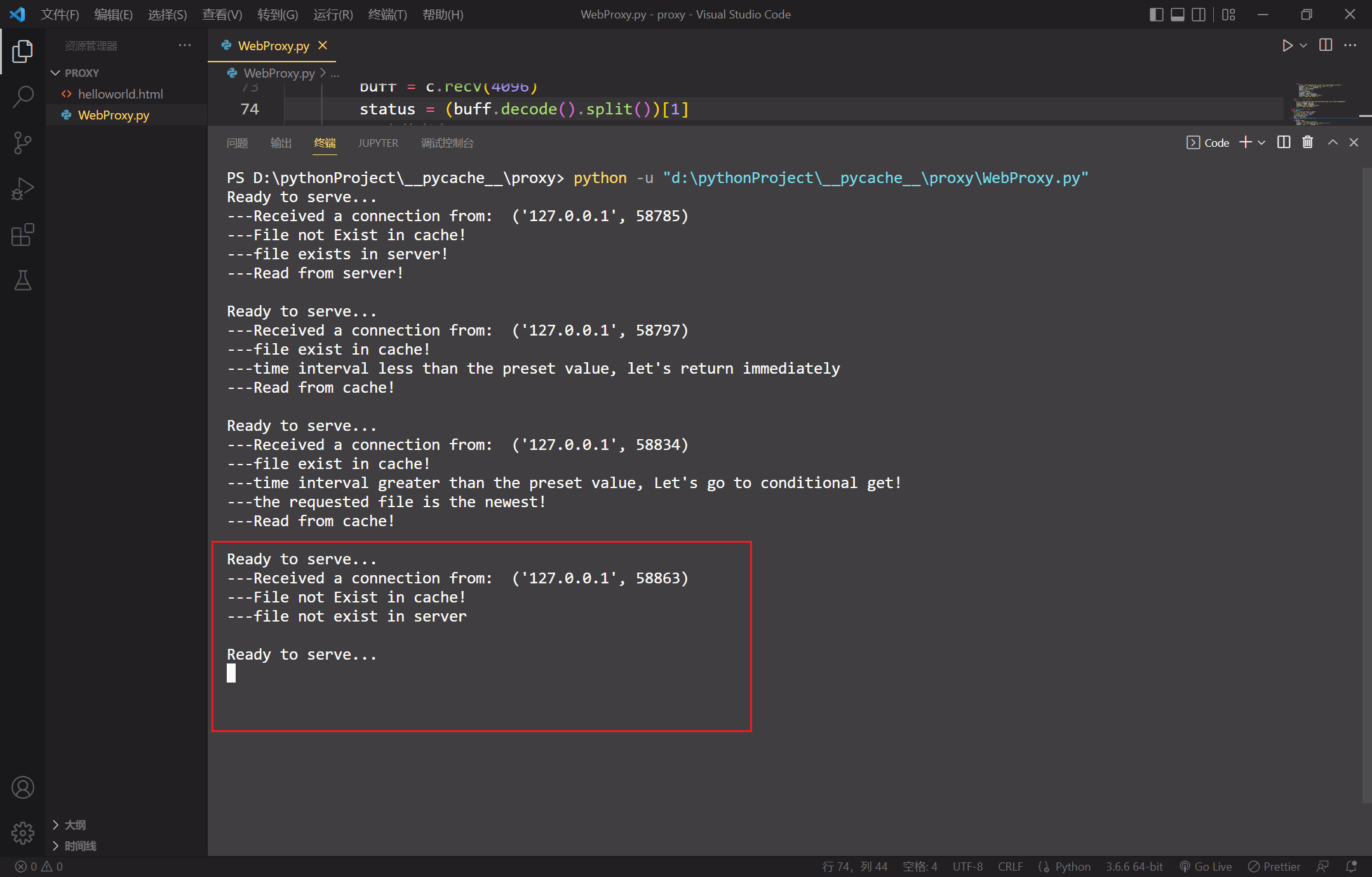
1. 文件不存在,返回404

Client端进行请求:



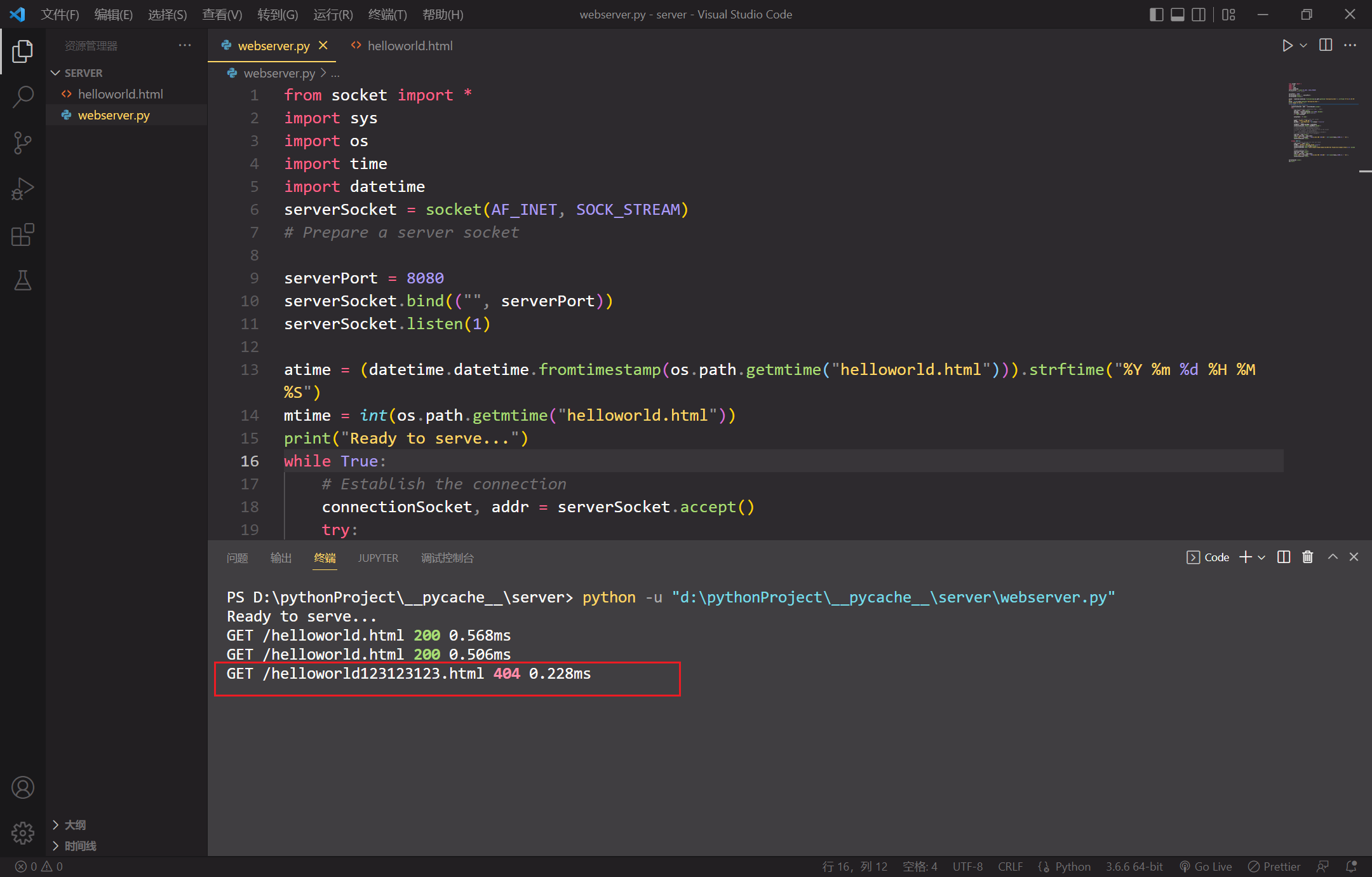
成功返回了404

Proxy端响应:



如图, 文件并不存在

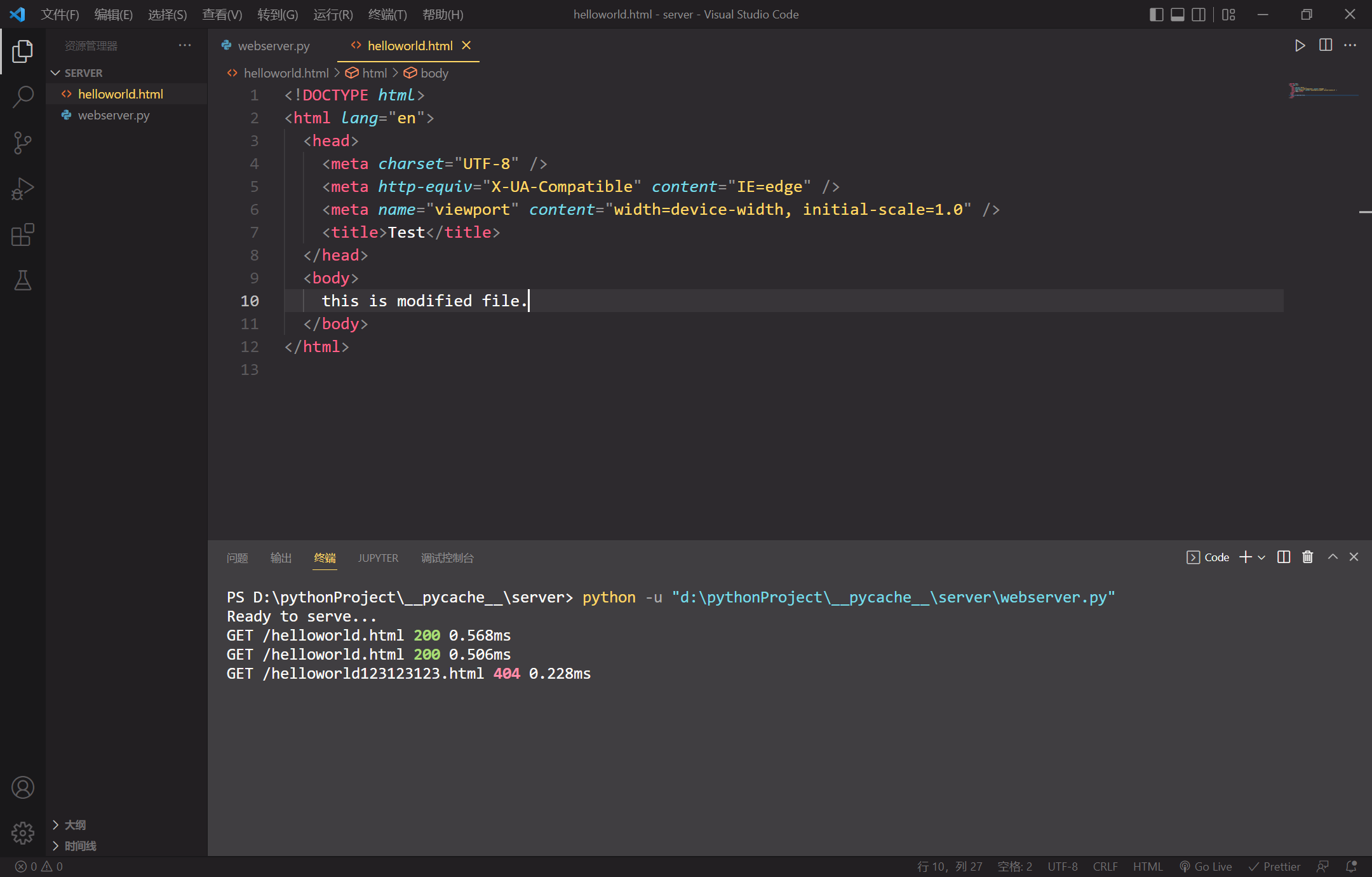
Server端的响应:



如图, 文件并不存在,返回404

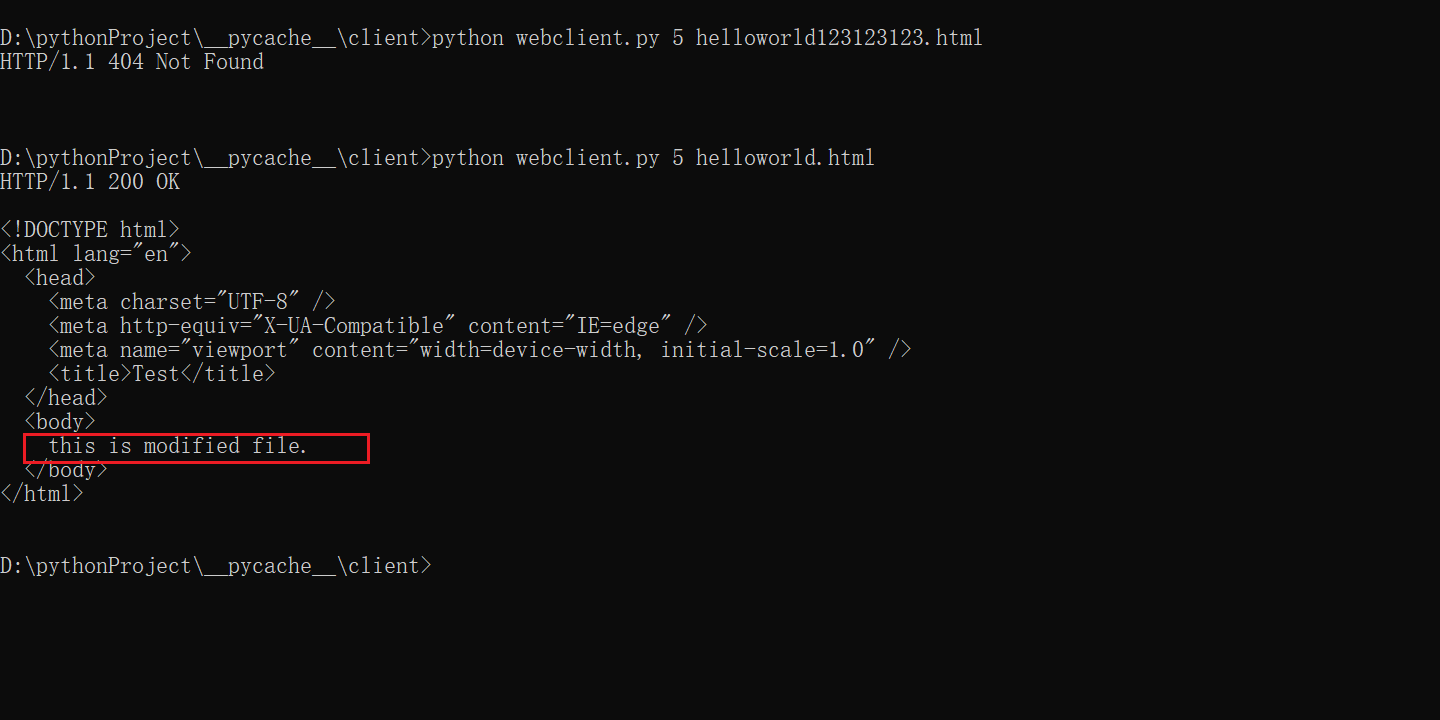
1. 进行conditional get,但文件已被修改

这里手动修改html文件的内容



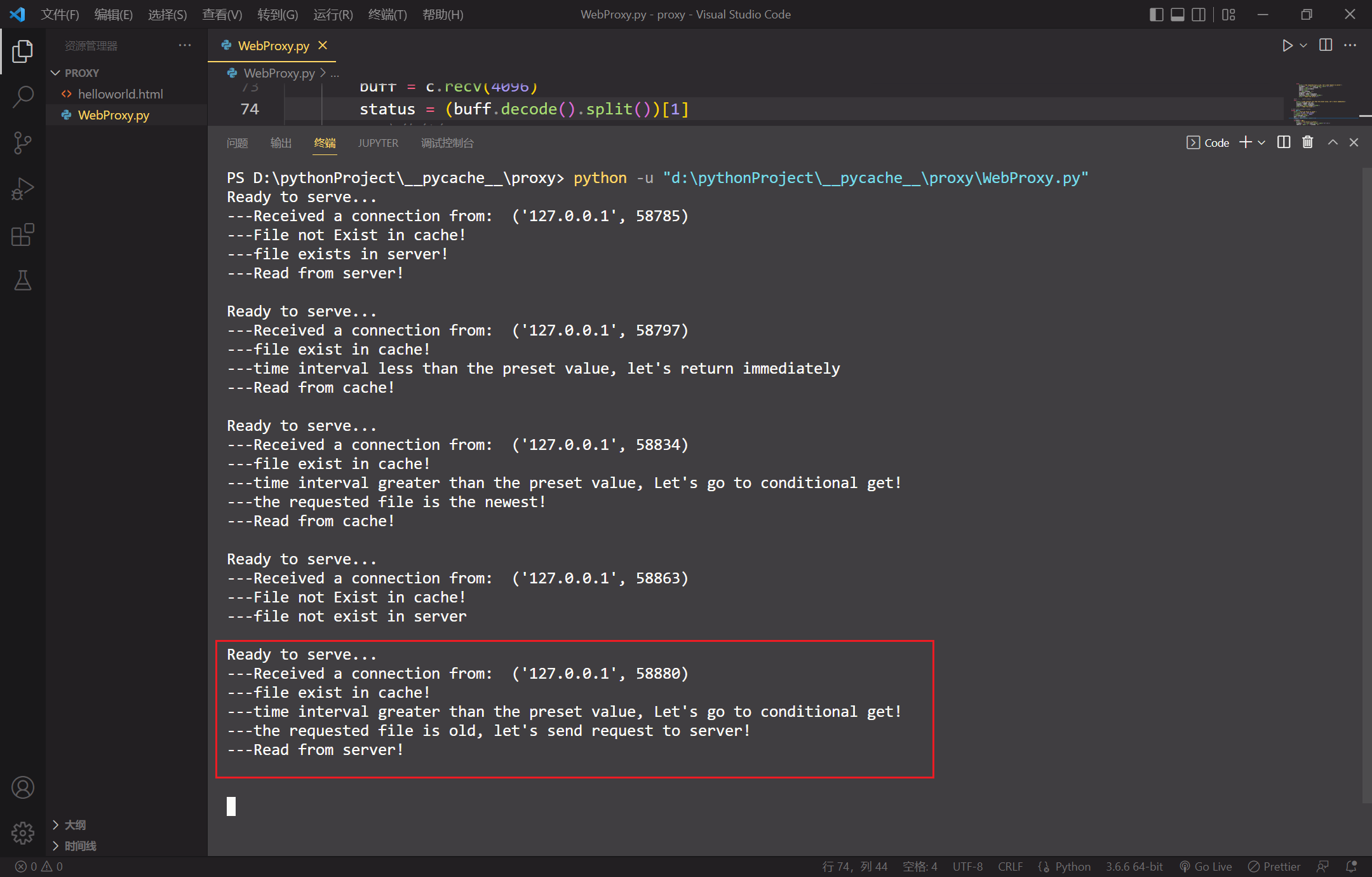
重新启动server(必须要进行重启),然后client再请求资源

Client端进行请求:

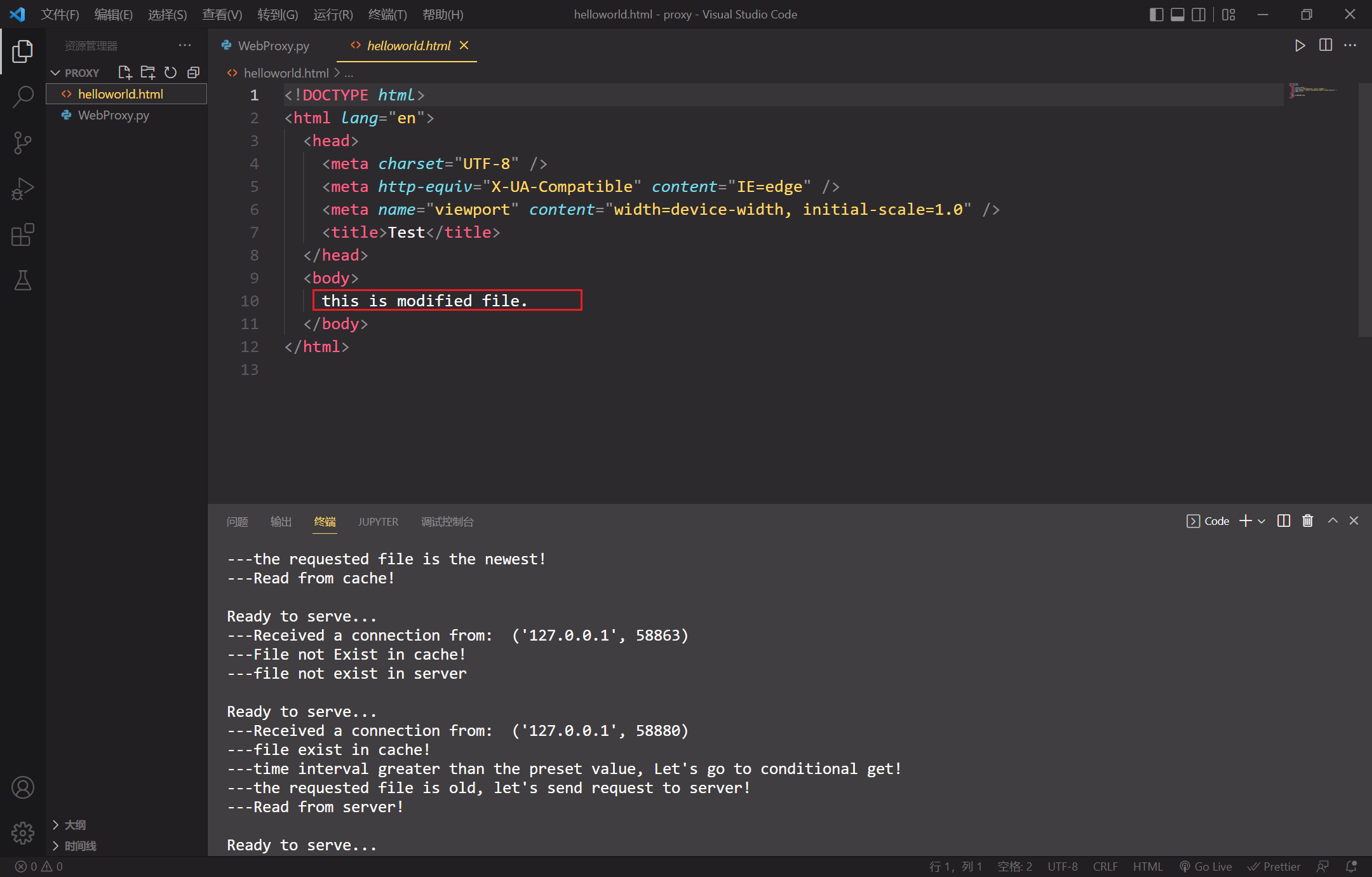


如图, 收到了最新的文件

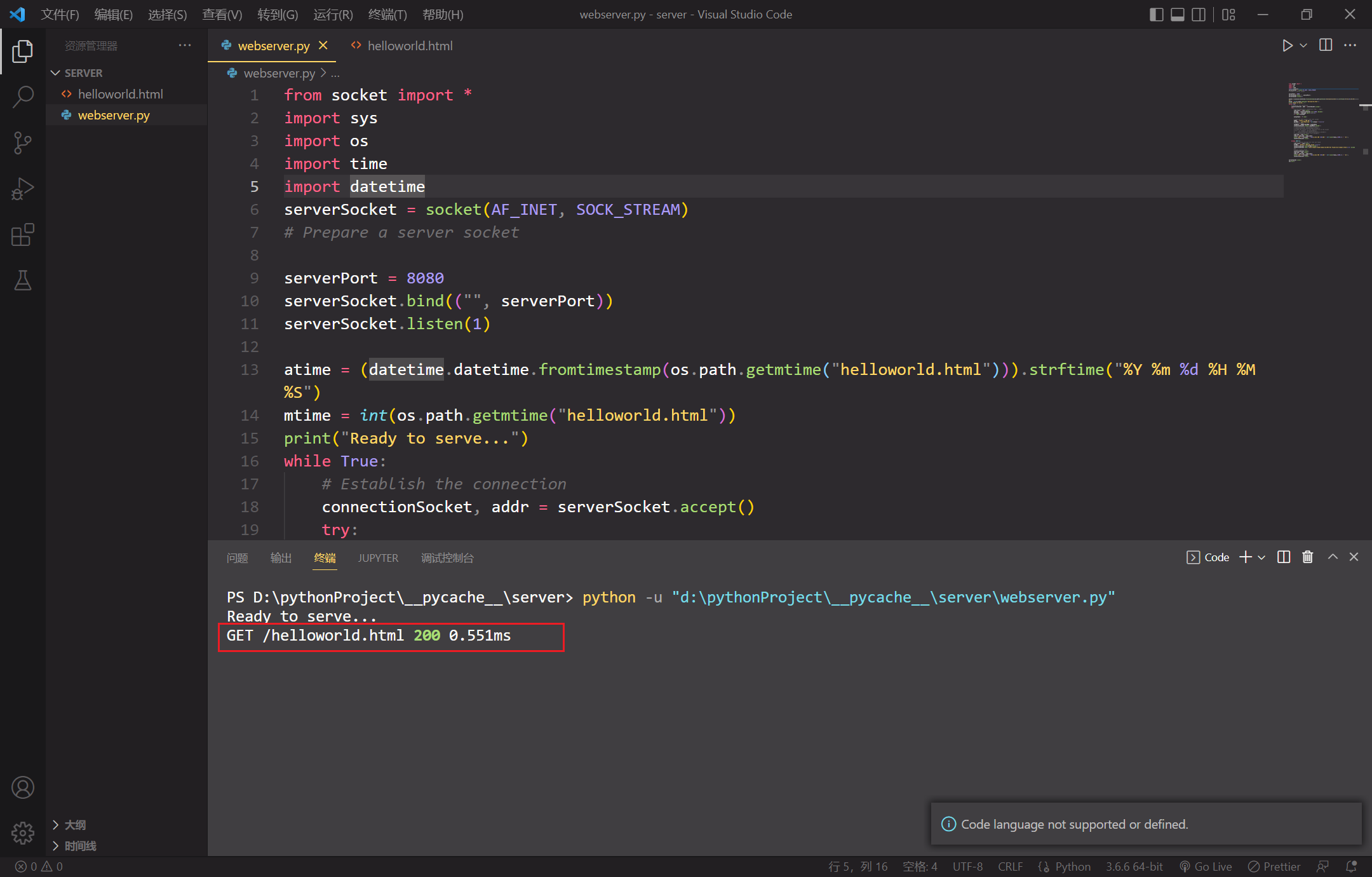
Proxy端的响应:



如图, 打印的消息符合我们的预期(检查到文件被修改,从server上请求文件并更新), 与此同时,注意到,proxy端的文件已经被更新了(如下图)



Server端的响应:



成功收到了请求.