实验2爬虫基础实训

Python网络编程

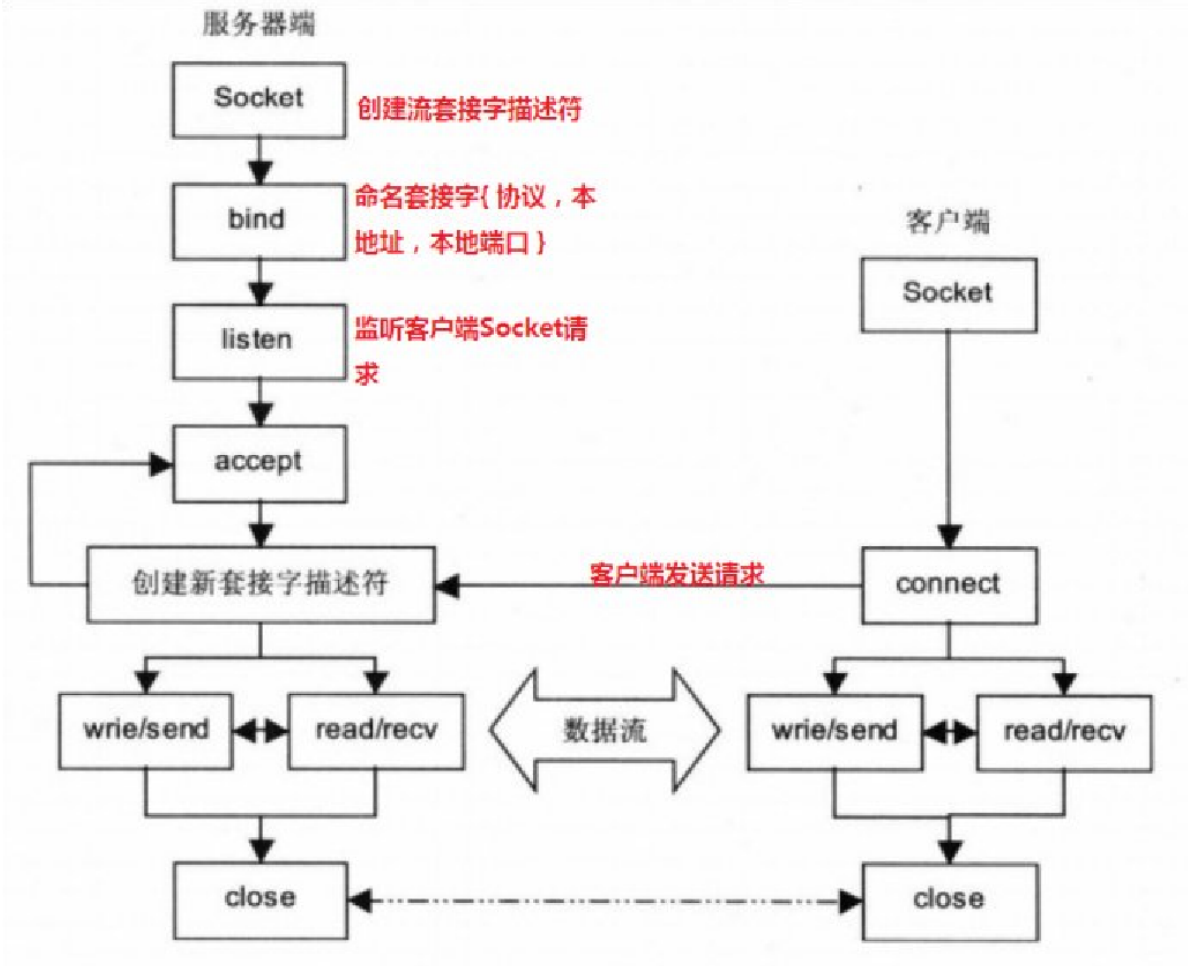
网络编程Socket库

套接字（socket）

网络上的两个程序通过一个双向的通信连接实现数据的交换，这个连接的一端称为一个socket。

套接字是socket的通常叫法，用于描述IP地址和端口，是一个通信链的句柄，可以用来实现不同虚拟机或不同计算机之间的通信。

Python中Socket库为操作系统的socket实现提供了一个Python接口。



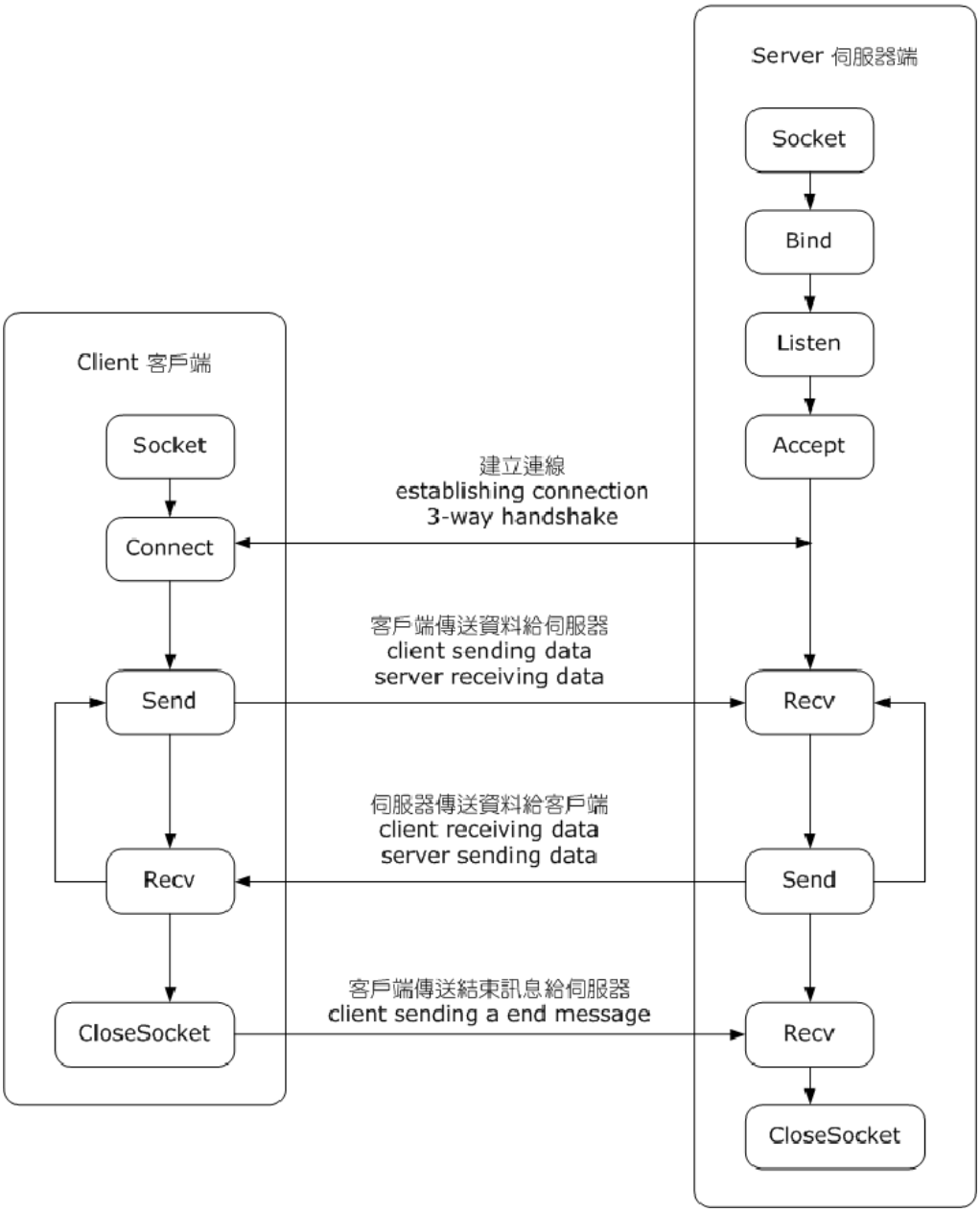
使用Socket进行TCP编程

建立一个服务器，服务器进程需要绑定一个端口并监听来自其他客户端的连接。

若有客户端发起连接请求，服务器就与该客户端建立Socket连接，随后的通信就通过此Socket连接进行。

服务器依赖服务器地址，服务器端口，客户端地址，客户端端口这4项来唯一确定一个Socket连接

TCP Socket 基本流程圖
TCP Socket flow diagram



服务端TCP

```
# 代码 2-1

# 导入socket库及依赖库
import socket
import threading
import time

# 建立TCP连接
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# 代码 2-2

# 绑定地址及监听端口
```

```
s.bind(('127.0.0.1', 6666))

# 代码 2-3

# 调用listen方法监听端口
s.listen(5)
print('Wait for connection...')


# 代码 2-4

# 服务器端应答函数：
def tcp(sock, addr):
    print('Accept new connection from %s:%s...' % addr)
    sock.send(b'Success!')
    while True:
        data = sock.recv(1024)
        time.sleep(1)
        if not data or data.decode('utf-8') == 'exit':
            break
        #sock.send(('Welcom! %s!' % data.decode('utf-8')).encode('utf-8'))
        #decode后拼字符串，拼接好后再encode
        print('Welcome! %s!' % data.decode('utf-8'))
        sock.send(('Welcome! %s!' % data.decode('utf-8')).encode('utf-8'))
        # sock.send(b'Welcom! %s!' % data)

    sock.close()
    print('Connection from %s:%s closed.' % addr)


# 代码 2-5

# 循环处理客户端连接
while True:
    # 接受来自客户端的新连接:
    sock, addr = s.accept()
    # 创建新线程来处理TCP连接:
    t = threading.Thread(target=tcp, args=(sock, addr))
    t.start()
```

源码链接：https://biglab.site//b62505scrapy/file/SY_2_1_TCP_SRV.py

客户端TCP

```
# 导入socket库
import socket

# 建立TCP连接
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
# 与服务器建立连接
s.connect(('127.0.0.1', 6666))
# 接受服务器的连接成功提示信息
print(s.recv(1024).decode('utf-8'))
# 发送数据并接受服务器返回结果
for data in [b'myname', "王某某".encode('utf-8')]:
```

```
    s.send(data)
    print(s.recv(1024).decode('utf-8'))
# 发送退出信息断开连接
s.send(b'exit')
s.close()

```

源码链接：https://biglab.site//b62505scrapy/file/SY_2_6_TCP_CLI.py

课堂互动

练习1：使用pycharm，调试运行服务端TCP脚本和客户端TCP脚本,要求：

1. 调试运行服务端TCP脚本，接收他人的请求，截图服务端TCP脚本源码和运行过程，需观察到他人的请求信息，上传到随堂练习；
2. 调试运行客户端TCP脚本，将myname和王某某换成本人姓名，然后将服务器地址127.0.0.1 换成教师机的IP，然后运行并截图客户端TCP脚本源码和运行过程，需观察到服务端返回的信息，上传到随堂练习

注意事项

1，充当服务端的电脑需要关闭防火墙，方法：打开：控制面板\系统和安全\Windows Defender 防火墙\自定义设置，选择关闭：

Http网络编程

HTTP请求方式与过程

爬虫在爬取数据时将会作为客户端模拟整个HTTP通信过程，该过程也需要通过HTTP协议实现。HTTP请求过程如下。由HTTP客户端向服务器发起一个请求，创建一个到服务器指定端口（默认是80端口）的TCP连接。

HTTP服务器从该端口监听客户端的请求。

一旦收到请求，服务器会向客户端返回一个状态，比如“HTTP/1.1 200 OK”，以及返回的响应内容，如请求的文件、错误消息、或其它信息。

基于TCP的HTTP请求

```
import socket

# 建立TCP连接
# 实例化一个套接字，2个参数分别是： IPV4、TCP 协议
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
# 与服务器建立连接
# 建立连接，2个参数是： 网址、端口
s.connect(('www.baidu.com', 80))

# 向服务器发送请求,传递的参数是：1.请求方式 2.地址 3.链接方式（open or close）
# 注：‘GET / HTTP’这里的‘/’是跟目录的意思
s.send(b'GET / HTTP/1.1\r\nHost:baidu.com\r\nConnection: close\r\n\r\n')

# 接受服务器的连接成功提示信息
# print(s.recv(1024).decode('utf-8'))

# 开始接受服务器传来的数据
buffer = []  # 新建一个空列表，buffer即缓存的意思
while True:  # 【降一级，防止出现不可控错误？】
    d = s.recv(5120)  # 每次最多接收5k字节
    if d:  # 如果能正常接收到d（即d不为空）
        buffer.append(d)
    else:
        break
data = b''.join(buffer)  # 组合传来的（列表格式的）数据为字符串(b)格式

s.close()

# 开始处理数据
# 分离网页头部与html,注：头部信息是网络传输时的标识信息，通常不需要展示出这部分
header, html = data.split(b'\r\n\r\n', 1)
# 以utf-8解码为正常文本
print("HEADER IS:")
print(header.decode('utf-8'))
print("HTML IS:")
print(html.decode('utf-8'))
# 新建文件，将接收到的数据接入文件内
with open('baidu_myname.html', 'wb')as f:
    f.write(html)
```

```
40
```

源码链接: https://biglab.site//b62505scrapy/file/SY_2_9_BAIDU.py

课堂互动

练习1：使用pycharm，调试运行基于TCP的HTTP请求的脚本,要求：

1. 请38行的myname换成本人，然后调试脚本，截图源码和运行过程；
2. 打开baidu_myname.html，使用菜单->code->reformate code格式化html脚本，截图，上传到随堂练习