**day18**

网络编程：

什么是网络编程？

网络：它是一种隐形的媒介，可以将多台计算机连接到一起(多台计算机共同使用它)

网络编程：让多台计算机之间可以进行通信(数据交互)

涉及到了网络编程，划分为两个方向层面：1).我们称**为客户端(cilent)** 2).我们称为**服务端(server)**

冲浪概念(上网)：打开浏览器，访问百度页面，

我们可以认为是本机浏览器的进程和百度服务器之间进行数据交互

IP：

IP地址的作用：

如果我们需要上网，每台计算机都会有**一个唯一标识(符号)，就是我们所谓的ip地址**

【注意事项】：

ip地址是每台计算机在网络中的唯一标识(记住)

在最初的时候人们是考虑使用主机名(计算机名称)来查询在网络环境中的对应计算机，

但是可能多态计算机之间存在主机名重复的现象，会造成安全隐患，

所以我们就舍弃了此操作，改用了**ip作用计算机的唯一标识；**

**如何查看ip地址(有两种方式)：**

方式一：启动cmd，在窗口中输入ipconfig指令

方式二：打开网络和共享中心，选择更改适配器设置，选择属性，点击查看ipv4的信息

对于ip地址的获取有两种方式：

方式一：自动获取ip

方式二：手动获取ip

ip地址由4个网组成的：

它的本质为：4个字节的二进制数据组成

二进制数据：10110101 11000101 11101010 10011111

十进制数据：数据1 数据2 数据3 数据4

【注意】：对于每个网段的数据设置有相应的要求，必须在0~255之间

普及常识性问题：

1).ping指令：

2).127.0.0.1：本机回环地址(解析为本机的ip)，理解：localhost

3).xxx.xxx.xxx.255：广播地址

端口：一台计算机可能被装载很多的应用程序，如果我们需要去实现网络编程的话，

在两个计算机之间程序数据交互需要通过端口号来找到相应的应用程序；

理解：端口号是计算机中应用程序的唯一标识

范围：0~65535之间：

注意：0~1023之间的端口我们不能直接设置使用，因为它们在安装完毕操作系统之后就被系统占用了；

常见的一些端口号如下：

0~1023之间：

http：80

ftp： 22

smtp：25

...

1024~65535之间：

mysql：3306 （端口号）不成文规定 默认了

redis：6379

...

网络模型：

分类学习：两个层面

1).osi参考模型：(七层)

2).tcp/ip参考模型：(四层)

只需要理解传输层的主要作用，确定到底使用tcp还是udp协议(规则)，进行数据交互

socket(套接字)：

python将网络编程的函数封装到了socket模块中，我们需要导入模块，使用函数即可;

核心思想：**不管是客户端还是服务端，都需要得到/获取socket对象，**然后调用一些核心函数完成需求

协议：tcp & udp

**tcp协议：**

特点：面向连接、可靠(安全)、效率慢

类似于我们生活中的：打电话

tcp的连接过程：3次握手

cilent(客) sever(服)

a：哥们，能听见我说话吗？

b: 我能，你能听见我说话吗？

a: 我也能，我们开始说话吧

此处正在进行数据交互

断开连接的过程：第4次挥手 --> 程序中的体现：close()

梳理：

核心对象：**socket对象**

发送和接受数据相关的函数**：send()、recv()**

**实现tcp程序的客户端编写：  
核心：socket对象  
在创建对象过程中，需要传入两个参数：  
参数1：满足了ipv4规则(socket.AF\_INET)  
参数2：判断/确定到底使用的是tcp还是udp协议来编写程序(socket.SOCK\_STREAM)** socket.SOCK\_STREAM表示tcp协议 **面向连接的协议：  
  
import** socket*#得到客户端的socket对象，模块名socket调用socket()函数*tcp\_cilent\_socket = socket.socket(socket.AF\_INET,socket.SOCK\_STREAM)  
  
*#定义服务端的ip和端口(封装到一个元祖对象中)*tcp\_server = (**'10.11.62.88'**,8888) *#10.11.62.88本机的ip地址，和自己连接，*

*8888是取的端口名，只要大于1023都可以*  
*#连接服务器*tcp\_cilent\_socket.connect(tcp\_server) *#connect链接服务的地址address*  
  
*#循环发送和接受数据：（多次聊下去）***while** 1:  
 *#客户端从键盘输入数据内容(字符串数据)（你要说给服务端的话）* data = input(**'>>>：'**)  
 *#将客户端的数据发送给服务端过程中，把字符串进行编码操作(二进制数据，认为加密行为)* tcp\_cilent\_socket.send(data.encode(**'utf-8'**)) *#send发送data信息*

*encode编码* *#接受服务端的数据信息并且进行解码操作* recv\_data = tcp\_cilent\_socket.recv(1024) *#接收到的数据大小，1024*  
  
 print(**'服务器：'** + recv\_data.decode(**'gbk'**)) *#可以用gbk解码，接收的*

*数据*  
  
*#关闭套接字对象*tcp\_cilent\_socket.close()

**实现tcp程序的服务端编写：  
核心：socket对象  
在创建对象过程中，需要传入两个参数：  
参数1：满足了ipv4规则(socket.AF\_INET)  
参数2：判断/确定到底使用的是tcp还是udp协议来编写程序(socket.SOCK\_STREAM)**

**面向连接：**

**import** socket*#得到服务端的socket对象*tcp\_server\_socket = socket.socket(socket.AF\_INET,socket.SOCK\_STREAM)  
  
*#绑定****服务端****自己的ip和端口(元祖)*tcp\_server = (**'10.11.62.88'**,8888)  
tcp\_server\_socket.bind(tcp\_server) *#绑定端口，等着客户端来链接*  
*#开启监听*tcp\_server\_socket.listen()  
  
*#接受客户端的访问 （接收的是两个对象，元祖对象）,一个是新的socket，一个是带过来的客户端信息*new\_socket,client\_info = tcp\_server\_socket.accept()  
  
print(new\_socket) *#后面都是new\_socket说了算 tcp\_server\_socket已经功成身退*print(client\_info)  
  
*#循环接受和发送数据***while** 1:  
 *#接受客户端发送过来的数据* recv\_data = new\_socket.recv(1024)  
  
 print(**'客户端：'** + recv\_data.decode(**'utf-8'**))  
  
 *#给客户端发送数据* data = input(**'>>>：'**)  
 new\_socket.send(data.encode(**'gbk'**))  
  
*#关闭两个套接字对象*new\_socket.close() *#后开的先关，相当于孩子*tcp\_server\_socket.close() *#相当于妈妈*

**tcp程序的客户端 和 tcp程序的服务端可以互发信息**

**服务端先开**

udp协议：

特点：面向无连接、不可靠、效率快

类似于我们生活中的：写信

梳理：

核心对象：socket对象

发送和接受数据相关的函数：**sendto()、recvfrom()**

**使用udp协议编写客户端程序：  
核心：socket对象  
在创建对象过程中，需要传入两个参数：  
参数1：满足了ipv4规则(socket.AF\_INET)  
参数2：判断/确定到底使用的是tcp还是udp协议来编写程序(socket.SOCK\_DGRAM)** socket.SOCK\_DGRAM表示udp协议来编程的  
 **面向无连接：**

**import** socket*#得到客户端的socket对象*udp\_client\_socket = socket.socket(socket.AF\_INET,socket.SOCK\_DGRAM)  
  
*#服务器的ip和端口(元祖的形式)*udp\_server = (**'10.11.62.88'**,8888) *#自己跟自己玩*  
  
*#循环接受和发送数据：***while** 1:  
 data = input(**'>>>：'**)  
 *#客户端发送数据 #因为无连接connect，所以多了参数udp\_server* udp\_client\_socket.sendto(data.encode(**'gbk'**),udp\_server)   
  
 *#接受服务端的数据* recv\_data = udp\_client\_socket.recvfrom(1024) *#大小*  
 print(recv\_data)  *#recv\_data是容器对象，拿出的内容是在0索引位置* print(**'服务端：'** + recv\_data[0].decode(**'gbk'**))   
  
*#关闭套接字对象*udp\_client\_socket.close()

**使用udp协议编写服务端程序：  
核心：socket对象  
在创建对象过程中，需要传入两个参数：  
参数1：满足了ipv4规则(socket.AF\_INET)  
参数2：判断/确定到底使用的是tcp还是udp协议来编写程序(socket.SOCK\_DGRAM)  
  
面向无连接：**

**import** socket  
*#得到服务端的socket对象*udp\_server\_socket = socket.socket(socket.AF\_INET,socket.SOCK\_DGRAM)  
  
*#绑定服务端的ip和端口*udp\_server = (**'10.11.62.88'**,8888)  
udp\_server\_socket.bind(udp\_server)  
  
*#循环接受和发送数据：***while** 1:  
 *#接受客户端的数据* recv\_data,cilent\_info = udp\_server\_socket.recvfrom(1024)  
 print(recv\_data)  
 print(cilent\_info)  
 print(**'客户端说：'** + recv\_data.decode(**'gbk'**))  
  
 *#发送数据给客户端* data = input(**'>>>：'**)  
 udp\_server\_socket.sendto(data.encode(**'gbk'**),cilent\_info)  
  
*#关闭套接字对象*udp\_server\_socket.close()