socket(套接字)

```
它提供了标准的Sockets API
目的是能够实现TCP和UDP的通信
```

TCP实现

服务器端

1.

```
1.创建套接字对象
socket.socket(socket.AF_INET,socket.SOCK_STREAM)
socket.AF_INET:服务器之间通信使用ipv4
socket.SOCK_STREAM:流失socket TCP
2.绑定本地地址，地址是用元组表示，一般里面包含ip和port
socket.bind(("127.0.0.1",9000))
3.开始监听
socket.listen()
4.接受请求
socket.accept()
5.接收信息或发送信息
socket.recv()#接收
socket.sendall()#发送
6.信息传输完毕，要关闭连接
socket.close()


#!/usr/bin/env python
# -*- coding:utf-8 -*-
#====#====#====#====
#Author:
#CreatDate:
#Version:
#====#====#====#====
import socket
#1.创建套接字对象
sk=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
#2.绑定本地地址
sk.bind(("127.0.0.1",9000))
#3.开始监听
sk.listen()
#4.不断的接收客户端的请求
while True:
    #接受请求,返回一个socket对象和一个客户端地址
    obj,clientAddress=sk.accept()
    print("%s:%d客户端连接成功"%clientAddress)
    while True:
```

```
        #使用accept返回的套接字对象来接收和发送数据
        msg=obj.recv(1024)
        print(msg.decode("utf8"))
        #判断客户端是否要退出
        if msg.decode("utf8")=="exit":
            obj.sendall("serverexitok".encode('utf8'))
            obj.close()
            break

        inmsg=input(">>:").strip()
        # 判断用户有没有输入信息
        if len(inmsg) == 0:
            continue
        #发送数据给客户端
        obj.sendall(inmsg.encode('utf8'))
```

客户端

1.

```
1.创建套接字对象
socket.socket(socket.AF_INET,socket.SOCK_STREAM)
socket.AF_INET:服务器之间通信使用ipv4
socket.SOCK_STREAM:流失socket TCP

2.连接服务器
socket.connect((ip,port))

3.接收数据或发送数据
socket.recv()
socket.send()

4.关闭连接
socket.close()


#!/usr/bin/env python
# -*- coding:utf-8 -*-
#====#====#====#====
#Author:
#CreatDate:
#Version:
#====#====#====#====
import socket

#1.创建套接字对象
sk=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
#2.连接服务器
sk.connect(("127.0.0.1",9000))
#3.发送数据给服务端
```

```
while True:
    msg=input("请输入信息>>:").strip()
    #判断用户有没有输入信息
    if len(msg)==0:
        continue

    sk.sendall(msg.encode('utf8'))

    #接收服务端发送的信息
    ret=sk.recv(1024)
    print(ret.decode("utf8"))
    if ret.decode("utf8")=="serverexitok":
        break

sk.close()
```

UDP的实现

服务端

1.
```
1.创建套件字
socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
socket.AF_INET:服务器之间通信使用ipv4
socket.SOCK_DGRAM:流失socket UDP

2.绑定本地地址
socket.bind(ip和port)

3.收数据和发数据
socket.recvfrom()
socket.sendto(信息，对方的地址)

4.关闭套接字
socket.close()


```

客户端

1.
```
1.创建套接字
socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
socket.AF_INET:服务器之间通信使用ipv4
socket.SOCK_DGRAM:流失socket UDP

2.收数据和发数据
socket.recvfrom()
socket.sendto(信息，对方的地址)

3.关闭套接字
socket.close()

```

```
13
14
15
16
```

总结

```
我们可以使用socket套接字去实现TCP和UDP连接

TCP实现:
1.服务器和客户端都需要创建套接字，并写明使用的协议（IPV4）和传输方式（TCP）
2.服务器需要绑定本地地址和监听，客户端不需要，但客户端需要连接服务端
3.服务端需要接受连接请求
4.服务器和客户端都可以进行接收和发送数据
5.最后双方都需要关闭套接字

UDP实现:
1.服务器和客户端都需要创建套接字，并写明使用的协议（IPV4）和传输方式（TCP）
2.服务器需要绑定本地地址
3.发送数据都需要信息和对方的地址
4.接收数据都会接收到对方的地址
5.最后双方都需要关闭套接字
```

WWW:万维网（world wide web）

三项基本技术：

1. HTML: HyperText Markup Language 超文本标记语言-如何去构建超文本
2. URL: Uniform Resource Locator 统一资源定位符-资源存放的位置
3. HTTP： HyperText Transfer Protoco 超文本传输协议-如何在网络当中去传输超文本

超文本：

1. 超出普通文本文档范畴的文档（包含：图片，音频，视频，动画....）
2. 包含超链接的文本文档

HTTP协议

```
HTTP协议用于客户端和服务器端进行通信
通过请求和响应的交换来达成信息
请求必须由客户端发起
响应是由服务器端返回
HTTP的数据传输是基于传输层的TCP协议
```

HTTP请求

```
HTTP报文是面向文本的，报文中的每一个字段都是一些ASCII码串，每个字段的长度是不确定的。HTTP报文传过来的都是一堆的0x ASCII码，例如" 41 63 63 65 70 74"这段十六进制ASCII码串对应的是“accept” 单词。

这些十六进制的数字经过浏览器或者专用工具比如wireshark或fiddler的翻译，可以得到HTTP的报文结构。

HTTP有两种报文：请求报文和响应报文。
```

报文格式

```
POST http://101.91.150.147:8008/login/ HTTP/1.1
Host: 101.91.150.147:8008
Connection: keep-alive
Content-Length: 39
Accept: application/json, text/plain, */*
language: zh-hans
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/110.0.0.0 Safari/537.36
Content-Type: application/json
Origin: http://101.91.150.147:8008
Referer: http://101.91.150.147:8008/
Accept-Encoding: gzip, deflate
Accept-Language: zh-CN,zh;q=0.9
Cookie: lang=zh-cn; device=desktop; theme=default; tab=my

{"name":"test0107","password":"123456"}
```

格式：

1. 请求行

   POST http://101.91.150.147:8008/login/ HTTP/1.1 请求方法： POST 要做什么操作
   URI: http://101.91.150.147:8008/login/ 要请求的资源的位置
   HTTP协议版本: HTTP/1.1

2. 请求头域 headers

   Host: 101.91.150.147:8008#接受请求的服务器地址 Connection: keep-alive#短连接 Content-Length: 39#数据长度 Accept: application/json, text/plain, / #指定客户端接收的数据类型 language: zh-hans#支持IE11 User-Agent: {Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/110.0.0.0 Safari/537.36 }#客户端的相关信息 Content-Type: application/json#请求的内容类型 Origin: http://101.91.150.147:8008 #请求来自哪个站点,只有服务器的名字 Referer: http://101.91.150.147:8008/ #请求来自哪个页面,包含服务器名字和路径 Accept-Encoding: gzip, deflate#可接受的内容编码 Accept-Language: zh-CN,zh;q=0.9#客户端可以接受的语言 Cookie: lang=zh-cn; device=desktop; theme=default; tab=my#Cookie

3. 空行

4. 内容实体

   {"name":"test0107","password":"123456"}