Socket是网络编程的一个抽象概念。通常我们用一个Socket表示"打开了一个网络链接"，而打开一个Socket需要知道目标计算机的ip地址和端口号，再指定协议类型即可。

客户端

大多数连接都是可靠的TCP连接。创建TCP连接时，主动发起连接的叫客户端，被动响应连接的叫服务器。

举个例子，当我们在浏览器中访问新浪时，我们自己的计算机就是客户端，浏览器主动向新浪的服务器发起连接。如果一切顺利，新浪的服务器接受了我们的连接，一个tcp连接就建立起来了，后面的通信就是发送网页内容了。

```
# 导入socket库
import socket

# 创建socket
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
# 建立连接
s.connect(('www.sina.com.cn', 80))
```

创建Socket时，AF_INET指定使用IPV4协议，如果要使用更先进的IPV6，就指定为AF_INET。SOCK_STREAM指定使用面向流的TCP协议，这样一个Socket对象就创建成功，但是还没有建立连接。

客户端要主动发起TCP连接，必须知道服务器的IP地址和端口号。新浪网站的IP地址可以用域名www.sina.com.cn自动转换到ip地址，但是怎么知道新浪服务器的端口号呢？

答案是作为服务器，提供什么样的服务端口号就必须固定下来。由于我们想要访问网页，因此新浪提供网页服务的服务器必须把端口号固定在80端口，因为80端口是web服务的标准端口。其他服务对应的标准端口号，例如SMTP服务是25端口，FTP服务是21端口，等等。端口号小于1024的是Internet标准服务的端口，端口号大于1024的，可以任意使用。

因此我们连接新浪服务器的端口号如下：

```
s.connect(('www.sina.com.cn', 80))
```

建立TCP连接后，我们就可以向新浪服务器发送请求，要求返回首页的内容：

```
# 发送数据:
s.send(b'GET / HTTP/1.1\r\nHost: www.sina.com.cn\r\nConnection: close\r\n\r\n')
```

TCP创建的连接是双向通道，双方都可以同时给对方发数据。但是谁先发谁后发，怎么协调，要根据具体的协议来决定。例如:HTTP协议规定客户端必须线发送请求给服务器，服务器收到后才发数据给客户端。

发送的文本格式必须符合HTTP标准，如果格式没有问题，接下来，就可以接受新浪服务器返回的数据了：

```
# 接收数据:
buffer = []
while True:
    # 每次最多接收1k字节:
    d = s.recv(1024)
    if d:
        buffer.append(d)
    else:
        break
data = b''.join(buffer)
```

接收数据时，调用recv(max)方法，一次最多接收指定的字节数，因此在一个while循环中反复接收，直到recv()返回空数据，表示接收完毕，退出循环。

当我们就收完数据后，调用close()方法关闭Socket,这样一次完整的网络通信就结束了：

```
# 关闭连接:
s.close()
```

接收到的数据包括HTTP头和网页本身，我们只需要把HTTP头和网页分离一下，把HTTP头打印出来，网页内容保存到文件:

```
header, html = data.split(b'\r\n\r\n', 1)
print(header.decode('utf-8'))
# 把接收的数据写入文件:
with open('sina.html', 'wb') as f:
    f.write(html)
```

str.split(str="", num=string.count(str))

- str--分隔符，默认为所有字符，包括空格、换行等。
- num--分割次数。 返回值--返回分割后的字符串列表。

现在，只需要在留言其中打开这个sina.html文件，就可以看到新浪首页了。

服务器

```
import socket
import threading
import time


def tcplink(sock, addr):
    print('Accept new connection from %s:%s...' % addr)
    sock.send(b'Welcome')
    while True:
        data = sock.recv(1024)
        time.sleep(1)
        if not data or data.decode('utf-8') == 'exit':
            break
        sock.send(('Hello,%s!' % data.decode('utf-8')).encode('utf-8'))
    sock.close()
    print('Connection from %s:%s closed' % addr)


s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind(('127.0.0.1', 10101))
s.listen(5)
print("waiting for connection...")

while True:
    sock, addr = s.accept()
    t = threading.Thread(target=tcplink, args=(sock, addr))
    t.start()
```

```
import socket

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

s.connect(('127.0.0.1', 10101))

print(s.recv(1024).decode('utf-8'))
for data in [b'qxh', b'Tracy', b'Sarah']:
    s.send(data)
    print(s.recv(1024).decode('utf-8'))

s.send(b'exit')
s.close()
```