← Python 命令行参数 Python format 格式化函数 →

# Python 网络编程

Python 提供了两个级别访问的网络服务。：

- 低级别的网络服务支持基本的 Socket，它提供了标准的 BSD Sockets API，可以访问底层操作系统Socket接口的全部方法。
- 高级别的网络服务模块 SocketServer， 它提供了服务器中心类，可以简化网络服务器的开发。

---

# 什么是 Socket?

Socket又称"套接字"，应用程序通常通过"套接字"向网络发出请求或者应答网络请求，使主机间或者一台计算机上的进程间可以通讯。

---

# socket()函数

Python 中，我们用 socket（ ）函数来创建套接字，语法格式如下：

```
socket.socket([family[, type[, proto]]])
```

## 参数

- family: 套接字家族可以使AF_UNIX或者AF_INET
- type: 套接字类型可以根据是面向连接的还是非连接分为SOCK_STREAM或SOCK_DGRAM
- protocol: 一般不填默认为0.

## Socket 对象(内建)方法

| 函数 | 描述 |
| --- | --- |
| 服务器端套接字 | |
| s.bind() | 绑定地址（host,port）到套接字， 在AF_INET下,以元组（host,port）的形式表示地址。 |
| s.listen() | 开始TCP监听。backlog指定在拒绝连接之前，操作系统可以挂起的最大连接数量。该值至少为1，大部分应用程序设为5就可以了。 |
| s.accept() | 被动接受TCP客户端连接,(阻塞式)等待连接的到来 |
| 客户端套接字 | |
| s.connect() | 主动初始化TCP服务器连接，。一般address的格式为元组（hostname,port），如果连接出错，返回socket.error错误。 |

| 函数 | 描述 |
| --- | --- |
| s.connect_ex() | connect()函数的扩展版本,出错时返回出错码,而不是抛出异常 |
| 公共用途的套接字函数 | |
| s.recv() | 接收TCP数据，数据以字符串形式返回，bufsize指定要接收的最大数据量。flag提供有关消息的其他信息，通常可以忽略。 |
| s.send() | 发送TCP数据，将string中的数据发送到连接的套接字。返回值是要发送的字节数量，该数量可能小于string的字节大小。 |
| s.sendall() | 完整发送TCP数据，完整发送TCP数据。将string中的数据发送到连接的套接字，但在返回之前会尝试发送所有数据。成功返回None，失败则抛出异常。 |
| s.recvfrom() | 接收UDP数据，与recv()类似，但返回值是（data,address）。其中data是包含接收数据的字符串，address是发送数据的套接字地址。 |
| s.sendto() | 发送UDP数据，将数据发送到套接字，address是形式为（ipaddr，port）的元组，指定远程地址。返回值是发送的字节数。 |
| s.close() | 关闭套接字 |
| s.getpeername() | 返回连接套接字的远程地址。返回值通常是元组（ipaddr,port）。 |
| s.getsockname() | 返回套接字自己的地址。通常是一个元组(ipaddr,port) |
| s.setsockopt(level,optname,value) | 设置给定套接字选项的值。 |
| s.getsockopt(level,optname[.buflen]) | 返回套接字选项的值。 |
| s.settimeout(timeout) | 设置套接字操作的超时期，timeout是一个浮点数，单位是秒。值为None表示没有超时期。一般，超时期应该在刚创建套接字时设置，因为它们可能用于连接的操作（如connect()） |
| s.gettimeout() | 返回当前超时期的值，单位是秒，如果没有设置超时期，则返回None。 |
| s.fileno() | 返回套接字的文件描述符。 |
| s.setblocking(flag) | 如果flag为0，则将套接字设为非阻塞模式，否则将套接字设为阻塞模式（默认值）。非阻塞模式下，如果调用recv()没有发现任何数据，或send()调用无法立即发送数据，那么将引起socket.error异常。 |
| s.makefile() | 创建一个与该套接字相关连的文件 |

# 简单实例

## 服务端

我们使用 socket 模块的 **socket** 函数来创建一个 socket 对象。socket 对象可以通过调用其他函数来设置一个 socket 服务。

现在我们可以通过调用 **bind(hostname, port)** 函数来指定服务的 *port(端口)*。

接着，我们调用 socket 对象的 *accept* 方法。该方法等待客户端的连接，并返回 *connection* 对象，表示已连接到客户端。

完整代码如下：

```
#!/usr/bin/python
# -*- coding: UTF-8 -*-
# 文件名：server.py

import socket               # 导入 socket 模块

s = socket.socket()         # 创建 socket 对象
host = socket.gethostname() # 获取本地主机名
port = 12345                # 设置端口
s.bind((host, port))        # 绑定端口

s.listen(5)                 # 等待客户端连接
while True:
    c, addr = s.accept()     # 建立客户端连接。
    print '连接地址：', addr
    c.send('欢迎访问菜鸟教程！')
    c.close()                # 关闭连接
```

## 客户端

接下来我们写一个简单的客户端实例连接到以上创建的服务。端口号为 12345。

**socket.connect(hosname, port )** 方法打开一个 TCP 连接到主机为 *hostname* 端口为 *port* 的服务商。连接后我们就可以从服务端获取数据，记住，操作完成后需要关闭连接。

完整代码如下：

```
#!/usr/bin/python
# -*- coding: UTF-8 -*-
# 文件名：client.py

import socket               # 导入 socket 模块

s = socket.socket()         # 创建 socket 对象
host = socket.gethostname() # 获取本地主机名
port = 12345                # 设置端口号

s.connect((host, port))
```

```
print s.recv(1024)
s.close()
```

现在我们打开两个终端，第一个终端执行 server.py 文件：

```
$ python server.py
```

第二个终端执行 client.py 文件：

```
$ python client.py
欢迎访问菜鸟教程!
```

这时我们再打开第一个终端，就会看到有以下信息输出：

```
连接地址:  ('192.168.0.118', 62461)
```

## Python Internet 模块

以下列出了 Python 网络编程的一些重要模块：

| 协议 | 功能用处 | 端口号 | Python 模块 |
| --- | --- | --- | --- |
| HTTP | 网页访问 | 80 | httplib, urllib, xmlrpclib |
| NNTP | 阅读和张贴新闻文章，俗称为"帖子" | 119 | nntplib |
| FTP | 文件传输 | 20 | ftplib, urllib |
| SMTP | 发送邮件 | 25 | smtplib |
| POP3 | 接收邮件 | 110 | poplib |
| IMAP4 | 获取邮件 | 143 | imaplib |
| Telnet | 命令行 | 23 | telnetlib |
| Gopher | 信息查找 | 70 | gopherlib, urllib |

更多内容可以参阅官网的 Python Socket Library and Modules。

← Python 命令行参数 Python format 格式化函数 →

点我分享笔记