**Python3 网络编程实例**

## 简单实例：服务器端

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。

#!/usr/bin/python3

# 文件名：server.py

# 导入 socket、sys 模块

import socket

import sys

# 创建 socket 对象

serversocket = socket.socket(

            socket.AF\_INET, socket.SOCK\_STREAM)

# 获取本地主机名

host = socket.gethostname()

port = 9999

# 绑定端口

serversocket.bind((host, port))

# 设置最大连接数，超过后排队

serversocket.listen(5)

while True:

    # 建立客户端连接

    clientsocket,addr = serversocket.accept()

    print("连接地址: %s" % str(addr))

    msg='欢迎访问菜鸟教程！'+ "\r\n"

    clientsocket.send(msg.encode('utf-8'))

    clientsocket.close()

## 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后期数据，记住，操作完成后需要关闭连接。

#!/usr/bin/python3

# 文件名：client.py

# 导入 socket、sys 模块

import socket

import sys

# 创建 socket 对象

s = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

# 获取本地主机名

host = socket.gethostname()

# 设置端口好

port = 9999

# 连接服务，指定主机和端口

s.connect((host, port))

# 接收小于 1024 字节的数据

msg = s.recv(1024)

s.close()

print (msg.decode('utf-8'))

现在我们打开两个终端，第一个终端执行 server.py 文件：

|  |  |
| --- | --- |
| 1 | $ python3 server.py |

第二个终端执行 client.py 文件：

|  |  |
| --- | --- |
| 1  2 | $ python3 client.py  欢迎访问菜鸟教程！ |

这是我们再打开第一个终端，就会看到有以下信息输出：

|  |  |
| --- | --- |
| 1 | 连接地址： ('192.168.0.118', 33397) |

打开两个终端是指在spider的console里面打开两个。打开顺序，先打开服务器—客户端

## 套路代码：

#服务器端

import socket

s = socket.socket()

host = socket.gethostname()

port = 9000

s.bind((host, port))

s.listen(5)

c, addr = s.accept()

#your code here

c.close()

#客户端

import socket

s = socket.socket()

host = "服务器host"

port = "服务器port"

s.connect((host, port))

#your code here

s.close()

# 每秒产生一个随机数

import random

import time

while True:

num = 1

time.sleep(num)

print(random.randint(0,20))

## sleep() 方法暂停给定秒数后执行程序。该参数可以是一个浮点数来表示一个更精确的睡眠时间。

以下是sleep()方法的语法：

time.sleep(t)  
参数 t – 这是要暂停执行的秒数。 返回值  
此方法不返回任何值。  
示例

# IndentationError: unindent does not match any outer indentation level解决方法

出现该错误时将tab 改为空格，在notepad ++中可以找到。

# Python中plot实现动态显示的方法

需要保存历史数据：

## **示例代码1-1**

import matplotlib.pyplot as plt

import numpy as np

import time

from math import \*

plt.ion() #开启interactive mode 成功的关键函数

plt.figure(1)

t = [0]

t\_now = 0

m = [sin(t\_now)]

for i in range(200):

plt.clf() #清空画布上的所有内容

t\_now = i\*0.1

t.append(t\_now)#模拟数据增量流入，保存历史数据（）

m.append(sin(t\_now))#模拟数据增量流入，保存历史数据

plt.plot(t,m,'.r')#画出红色的点

#plt.draw()#注意此函数需要调用

#time.sleep(0.01)

plt.pause(1)

## **示例代码1-2**

import matplotlib.pyplot as plt

import numpy as np

import time

from math import \*

plt.ion() #开启interactive mode 成功的关键函数

plt.figure(1)

t = [0]

t\_now = 0

m = [sin(t\_now)]

for i in range(2000):

# plt.clf() # 清空画布上的所有内容。此处不能调用此函数，不然之前画出的点，将会被清空。

t\_now = i\*0.1

"""

由于第次只画一个点，所以此处有两种方式，第一种plot函数中的样式选

为点'.'、'o'、'\*'都可以，就是不能为线段'-'。因为一条线段需要两

个点才能确定。第二种方法是scatter函数，也即画点。

"""

plt.plot(t\_now,sin(t\_now),'.') # 第次对画布添加一个点，覆盖式的。

# plt.scatter(t\_now, sin(t\_now))

plt.draw()#注意此函数需要调用

time.sleep(0.01)

以上是动态的显示一个函数，也即直观上一条轨迹不断的延伸。这是一种应用，另一种应用是在一张画布上增量式的画多条轨迹（函数）。

## **示例代码1-3**

import matplotlib.pyplot as plt

import numpy as np

import time

from math import \*

plt.ion() #开启interactive mode 成功的关键函数

plt.figure(1)

t = np.linspace(0, 20, 100)

for i in range(20):

# plt.clf() # 清空画布上的所有内容。此处不能调用此函数，不然之前画出的轨迹，将会被清空。

y = np.sin(t\*i/10.0)

plt.plot(t, y) # 一条轨迹

plt.draw()#注意此函数需要调用

time.sleep(1)

# pause函数概述

pause函数功能运行GUI事件循环若干秒。

pause函数的签名为matplotlib.pyplot.pause(interval)。参数interval表示秒数，类型为整数。

如果当前有活动的图形，在pause函数运行前，图形将会更新并显示，在等待期间事件循环会一直运行，直到暂停时间interval秒后结束。

如果没有当前有活动的图形，将会调用time.sleep函数，休眠interval秒。

原文链接：<https://blog.csdn.net/mighty13/article/details/116671083>

### 使用time.sleep函数

import matplotlib.pyplot as plt

import numpy as np

import time

np.random.seed(19680801)

data = np.random.random((50, 50, 50))

fig, ax = plt.subplots()

for i in range(len(data)):

ax.cla()

ax.imshow(data[i])

ax.set\_title("frame {}".format(i))

# Note that using time.sleep does \*not\* work here!

plt.draw()

plt.show(block=False)

time.sleep(0.01)

# Socket中send()函数和recv()函数详解

## send函数

1. int send( SOCKET s, const char FAR \*buf, int len, int flags );
2. 不论是客户还是服务器应用程序都用send函数来向TCP连接的另一端发送数据。
3. 客户程序一般用send函数向服务器发送请求，而服务器则通常用send函数来向客户程序发送应答。
4. （1）第一个参数指定发送端套接字描述符；
5. （2）第二个参数指明一个存放应用程序要发送数据的缓冲区；
6. （3）第三个参数指明实际要发送的数据的字节数；
7. （4）第四个参数一般置0。

## 2、recv函数

int recv( SOCKET s, char FAR \*buf, int len, int flags );

不论是客户还是服务器应用程序都用recv函数从TCP连接的另一端接收数据。

（1）第一个参数指定接收端套接字描述符；

（2）第二个参数指明一个缓冲区，该缓冲区用来存放recv函数接收到的数据；

（3）第三个参数指明buf的长度；

（4）第四个参数一般置0。