**五邑大学智能制造学部**

**专业综合实习报告**

**题 目：** 基于Python的聊天室

**班 级**： 160808

**学 号**： 3116003980

**姓 名**： 游 大 满

**指导教师**： 曾 爱 国

**起止日期**： 2019年9月2日 至2019年10月12日

递交的材料清单：

1. 源代码一份
2. 设计报告一份

**指导教师审阅意见及成绩评定：**

指导老师签章：

日 期： 年 月 日

目录

[一 设计内容简介 1](#_Toc21735668)

[1.1界面采用python的tkinter 1](#_Toc21735669)

[1.2采用TCP协议 1](#_Toc21735670)

[1.3使用套接字(socket)设计 1](#_Toc21735671)

[1.4客户端连接 1](#_Toc21735672)

[1.5服务器端连接 2](#_Toc21735673)

[二 设计分析 2](#_Toc21735674)

[2.1实验系统设计 4](#_Toc21735675)

[2.2界面设计 6](#_Toc21735676)

[2.3设计代码 7](#_Toc21735677)

[2.3.1服务端代码 9](#_Toc21735678)

[2.3.2客户端代码 12](#_Toc21735679)

[三 功能介绍 17](#_Toc21735680)

[3.1基本聊天功能 17](#_Toc21735681)

[3.2表情功能 17](#_Toc21735682)

[3.3图片发送功能 17](#_Toc21735683)

[3.4文件上传功能 17](#_Toc21735684)

[3.5视频通话功能 17](#_Toc21735685)

[3.6爬虫功能 18](#_Toc21735686)

[3.7AIML机器人功能 18](#_Toc21735687)

[四 系统测试 19](#_Toc21735688)

[4.1启动服务端 19](#_Toc21735689)

[4.2启动客户端 20](#_Toc21735690)

[4.3发送聊天信息测试 23](#_Toc21735691)

[4.4增加新的客户端，测试多用户聊天 23](#_Toc21735692)

[4.5表情功能测试 26](#_Toc21735693)

[4.6图片发送功能测试 28](#_Toc21735694)

[4.7文件上传功能测试 30](#_Toc21735695)

[4.8视频通话功能测试 32](#_Toc21735696)

[4.9爬虫功能测试 32](#_Toc21735697)

[4.10 AIML机器人功能测试 33](#_Toc21735698)

[五 心得体会 34](#_Toc21735699)

[参考文献 35](#_Toc21735700)

# 一 设计内容简介

## 1.1界面采用python的tkinter

聊天室界面采用python的tkinter来实现。通讯采用TCP来实现。服务端和客户端分开运行。支持多个客户端，也就是支持多个用户同时在线实时聊天。使用的开发工具：IDLE 3.7 (64-bit)与JetBrains PyCharm Community Edition 2019.1.1 x64。

## 1.2采用TCP协议

本作业采用TCP协议。TCP是面向连接的，通信双方保持一条通路，好比目前的电话线，使用telnet登陆BBS，用的就是TCP协议；UDP是无连接的，通信双方都不保持对方的状态，浏览器访问Internet时使用的HTTP协议就是基于UDP协议的。

## 1.3使用套接字(socket)设计

本实验使用套接字(socket)来设计。socket之间的连接可以分为三种类型：客户端连接，监听连接以及服务器端连接。

## 1.4客户端连接

客户端连接是指由客户端的socket提出连接请求，要连接的目标是服务器端的socket。为此，客户端的socket必须首先描述它要连接的服务器端socket(主要是指服务器 端socket的地址和端口号)，然后再定位所要连接的服务器端socket，找到以后，就向服务器端 socket请求连接。当然，服务器端的socket此时未必正好处于准备好状态，不过，服务器端的 socket会自动维护客户请求连接的队列，然后在它认为合适的时候向客户端socket发出"允许连接" (accept)的信号，这时客户端socket与服务器端socket的连接就建立了。

监听连接，服务器端 socket并不定位具体的客户端socket，而是处于等待连接的状态。当服务器端socket监听到或者说 接收到客户端socket的连接请求，它就响应客户端socket的请求建立一个新的socket句柄并与客户 端连接，而服务器端socket继续处于监听状态，还可以接收其它客户端socket的连接请求。

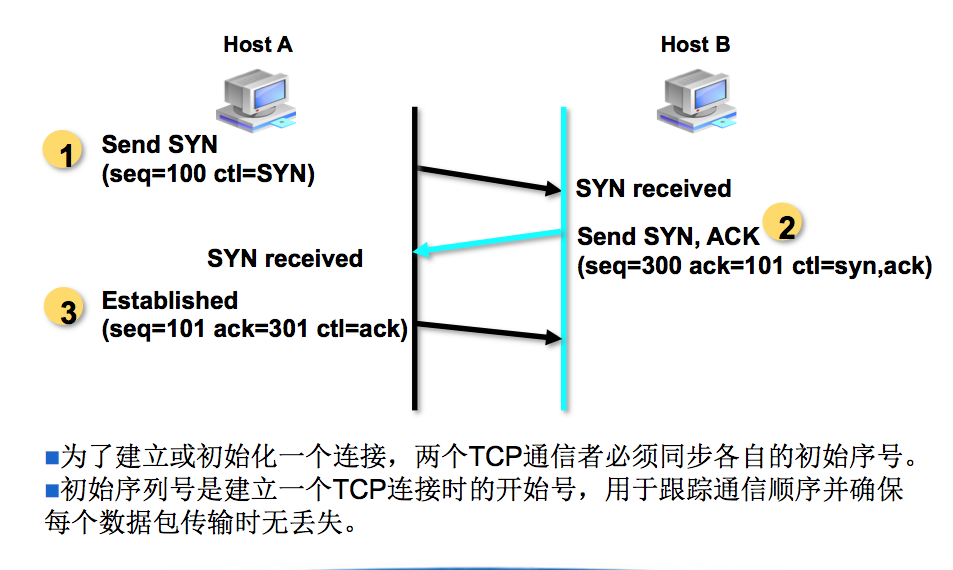
## 1.5服务器端连接

服务器端连接，是指当服务器端socket接收到客户端socket的连接请求后，就把服务器端socket的描述发给客户端，一旦客户端确认了此描述，连接就建立了。在本文中的聊天程序用的就是监听连接，即服务器设置连接个数后进行监听，客户端进行对服务器端的连接，这样就可以进行相互通信了。本实验的服务器端默认设置为本地IP 127.0.01，端口号采用50601。

# 二 设计分析

本作业是基于TCP的聊天室设计。所有的聊天信息的传输都是基于TCP来进行的。

TCP是面向连接的传输层协议，所谓面向连接就是在真正的数据传输开始前要完成连接建立的过程，否则不会进入真正的数据传输阶段。  
TCP的连接建立过程通常被称为三次握手（three-way handshake），过程如下：请求端（通常称为客户）发送一个SYN段指明客户打算连接的服务器的端口，以及初始序号（ISN）。这个SYN段为报文段1。 服务器发回包含服务器的初始序号的SYN报文段（报文段2）作为应答。同时，将确认序号设置为客户的ISN加1以对客户的SYN报文段进行确认。一个SYN将占用一个序号。 客户必须将确认序号设置为服务器的ISN加1以对服务器的SYN报文段进行确认（报文段3）。 这三个报文段完成连接的建立。  
发送第一个SYN的一端将执行主动打开（active open）。接收这个SYN并发回下一个SYN的另一端执行被动打开（passive open）。



为了实现TCP通信，本作业采用客户机/服务器模式。

C/S结构（Client/Server结构）是大家熟知的客户机和服务器结构。它是软件系统体系结构，通过它可以充分利用两端硬件环境的优势，将任务合理分配到Client端和Server端来实现，降低了系统的通讯开销。

首先服务器方要启动，并根据请求提供相应服务：  
 （1）打开一通信通道并告知本地主机，它在某一公认地址端口上(本作业测试采用50601)接受客户请求。  
 （2）等待客户请求到达该端口。  
 （3）接收到重复服务请求，处理该请求并发送应答信号。  
 （4）返回第二步，等待另外的客户请求  
 （5）关闭服务器。  
 客户方：  
 （1）打开一通信通道，并连接到服务器所在主机的特定端口。  
 （2）向服务器发出服务请求报文，等待并接收应答；继续提出请求。  
 从上面的描述过程可知：  
 （1）客户与服务器进程的作用是非对称的。因此编码不同。  
 （2）服务进程是先于客户请求启动的。直到正常终止或者强迫终止。在TCP/IP网络中两个进程间的相互作用的主机模式是客户机/服务器模式(Client/Server model)。

2.1实验系统设计

基本架构设计如下：

聊天室服务器

Listen for incoming client connections

(Listen)监听端口

Accept client connections

(Accept)接受客户端连接

Send and receive data

(send receive)发送信息

Connect to sever

(Connect)连接到服务器

Close connections

(Close)关闭连接

Open a Socket

(Socket)设置连接

Send and receive data

(send receive)发送信息

Close connections

(Close)关闭连接

客户端

客户端

Open the Server

开启服务器

程序流程如下：

运行服务器端

收到链接请求，建立SOCKET链接

退出

提示连接成功

进行聊天并处于Listen状态，等待其它用户链接

开启监听，等待链接

运行客户端

进行聊天

退出

断开连接

退出

输入服务器IP、端口号和用户名，请求链接

链接成功，进入聊天室

关闭服务器

失去服务器

2.2界面设计

本程序服务端和客户端分开运行，服务器没有界面，客户端界面使用tkinter模块完成。

客户端界面设计



界面最上方的标题处展示聊天状态，群聊天或是私聊。

左上方最大的窗口是消息显示框。下面紧挨着的是“表情”、“发送图片”、“文件上传”和“视频通话”四个按钮。再下面是文本输入框，在文本输入框的右下角是发送按钮。

右边三个窗口从上到下依次是群文件显示框、聊天室成员显示框和爬虫抓取的热点信息显示框。

2.3设计代码

本作业采用socket来完成。

什么是socket？网络上的两个程序通过一个双向的通信连接实现数据的交换，这个连接的一端称为一个socket。

socket通常也称作“套接字”，用于描述IP地址和端口，是一个通信链的句柄，应用程序通常通过“套接字”向网络发出请求或应答网络请求。

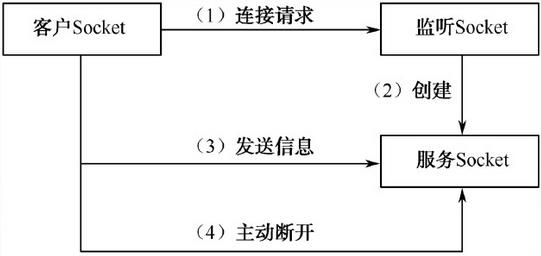
socket起源于Uinx。Unix/Linux的基本哲学之一就是“一切皆文件”，都可以用“打开open–>读写write/read–>关闭close”模式来操作，socket就是该模式的一个实现，socket即是一种特殊的文件，一些socket函数就是对其进行的操作(读/写 IO，打开，关闭)。

Socket的英文原义是“孔”或“插座”。作为BSD UNIX的进程通信机制，取后一种意思。通常也称作”套接字”，用于描述IP地址和端口，是一个通信链的句柄，可以用来实现不同虚拟机或不同计算机之间的通信。在Internet上的主机一般运行了多个服务软件，同时提供几种服务。每种服务都打开一个Socket，并绑定到一个端口上，不同的端口对应于不同的服务。Socket正如其英文原义那样，像一个多孔插座。一台主机犹如布满各种插座的房间，每个插座有一个编号，有的插座提供220伏交流电,有的提供110伏交流电，有的则提供有线电视节目。客户软件将插头插到不同编号的插座，就可以得到不同的服务。

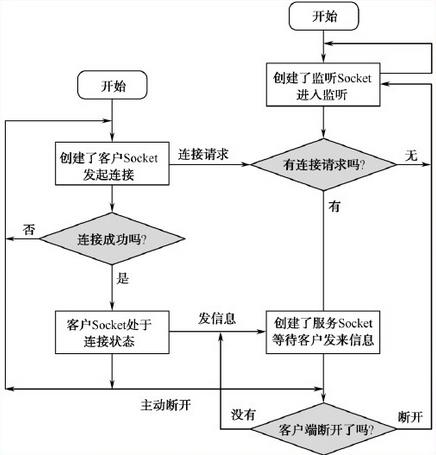
又如：中国移动客服。对于移动来说:一直监听一个号码10086，当有电话进来后，就分配一个客服和客户去沟通并处理请求。对于用户来说：需要知道10086这个号码，并拨打打电话。

Python中的\_socket模块是在链接库里的。也就是说这不是用python实现的,是socket操作的C实现，这个是非常底层的操作。socket.py是用py代码把C实现的模块的封装起来之后的模块，供人使用。

最简单的Socket通信是一个只有客户端向服务器端发信息的单向通信。器双方Socket之间的关系如下图所示：



由上面这个十分简单的过程很容易得出最简单的Socket通信流程，如下图所示：



要实现这样一个完整流程需要三个套接字对象：客户端一个，服务器端两个（一个用于监听，另一个用于接收客户端发来的信息）。

### 2.3.1服务端代码

代码1：实现与机器人聊天的函数

def speak\_with\_robot(msg):

def get\_module\_dir(name):

path = getattr(sys.modules[name], '\_\_file\_\_', None)

if not path:

raise AttributeError('module %s has not attribute \_\_file\_\_' % name)

return os.path.dirname(os.path.abspath(path))

alice\_path = get\_module\_dir('aiml') + '\\botdata\\alice'

# 切换到语料库所在工作目录

os.chdir(alice\_path)

alice = aiml.Kernel()

alice.learn("std-startup.xml")

alice.respond('load aiml b')

return alice.respond(msg)

代码2：聊天服务类

class ChatServer(threading.Thread):

global users, que, lock

def \_\_init\_\_(self, port):

threading.Thread.\_\_init\_\_(self)

self.ADDR = ('', port)

os.chdir(sys.path[0])

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

# 用于接收所有客户端发送信息的函数

def tcp\_connect(self, conn, addr):

# 判断断开用户在users中是第几位并移出列表, 刷新客户端的在线用户显示

def delUsers(self, conn, addr):

# 将接收到的信息(ip,端口以及发送的信息)存入que队列

def recv(self, data, addr):

# 将队列que中的消息发送给所有连接到的用户

def sendData(self):

# 运行

def run(self):

self.s.bind(self.ADDR)

self.s.listen(5)

print('聊天服务开始运行......')

q = threading.Thread(target=self.sendData)

q.start()

while True:

conn, addr = self.s.accept()

t = threading.Thread(target=self.tcp\_connect, args=(conn, addr))

t.start()

self.s.close()

代码3：文件传输服务类

class FileServer(threading.Thread):

def \_\_init\_\_(self, port):

threading.Thread.\_\_init\_\_(self)

self.ADDR = ('', port)

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

self.first = r'.\resources'

os.chdir(self.first) # 把first设为当前工作路径

# 用于接收所有客户端发送信息的函数

def tcp\_connect(self, conn, addr):

# 传输当前目录列表

def sendList(self, conn):

# 发送文件函数

def sendFile(self, message, conn):

# 保存上传的文件到当前工作目录

def recvFile(self, message, conn):

# 切换工作目录

def cd(self, message, conn):

# 判断输入的命令并执行对应的函数

def recv\_func(self, order, message, conn):

# 运行

def run(self):

print('文件服务开始运行......')

self.s.bind(self.ADDR)

self.s.listen(3)

while True:

conn, addr = self.s.accept()

t = threading.Thread(target=self.tcp\_connect, args=(conn, addr))

t.start()

self.s.close()

代码4：图片传输服务类

class PictureServer(threading.Thread):

def \_\_init\_\_(self, port):

threading.Thread.\_\_init\_\_(self)

self.ADDR = ('', port)

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

os.chdir(sys.path[0])

self.folder = r'.\Server\_image\_cache' # 图片的保存文件夹

# 用于接收所有客户端发送信息的函数

def tcp\_connect(self, conn, addr):

# 发送图片函数

def sendFile(self, message, conn):

# 保存上传的图片到当前工作目录

def recvFile(self, message, conn):

# 判断输入的命令并执行对应的函数

def recv\_func(self, order, message, conn):

# 运行

def run(self):

self.s.bind(self.ADDR)

self.s.listen(5)

print('图片服务开始运行......')

while True:

conn, addr = self.s.accept()

t = threading.Thread(target=self.tcp\_connect, args=(conn, addr))

t.start()

self.s.close()

### 2.3.2客户端代码

代码1：爬虫功能代码

def get\_page(url):

try:

response = requests.get(url)

response.encoding = response.apparent\_encoding

if response.status\_code == 200:

return response.text

return None

except RequestException:

return None

def parse\_page(html):

pattern = re.compile('<span.\*?num-top.\*?>(.\*?)</span>.\*?<a.\*?list-title.\*?>(.\*?)</a>', re.S)

#pattern = re.compile('<li><a target="\_blank" href="(.\*?)".\*?>(.\*?)</a>', re.S)

items = re.findall(pattern, html)

for item in items:

yield{

'rank': item[0],

'title': item[1],

#'image': item[2],

}

# 创建列表框

list3 = tkinter.Listbox(root)

list3.place(x=670, y=330, width=175, height=240)

def show\_pachong():

list3.delete(0, tkinter.END) # 清空列表框

list3.insert(tkinter.END, '今日热点（爬虫实时抓取）')

list3.itemconfig(tkinter.END, fg='red')

pachong1()

list3.insert(tkinter.END, '实时热点（爬虫实时抓取）')

list3.itemconfig(tkinter.END, fg='red')

pachong2()

list3.insert(tkinter.END, '体育热点（爬虫实时抓取）')

list3.itemconfig(tkinter.END, fg='red')

pachong3()

# 今日热点

def pachong1():

url = "http://top.baidu.com/buzz?b=341&c=513&fr=topbuzz\_b1\_c513"

html = get\_page(url)

for item in parse\_page(html):

list3.insert(tkinter.END, ('' + item['rank'] + ':' + item['title']))

list3.itemconfig(tkinter.END, fg='orange')

# 实时热点

def pachong2():

url = "http://top.baidu.com/buzz?b=1&c=513&fr=topbuzz\_b1\_c513"

html = get\_page(url)

for item in parse\_page(html):

list3.insert(tkinter.END, ('' + item['rank'] + ':' + item['title']))

list3.itemconfig(tkinter.END, fg='green')

# 体育热点

def pachong3():

url = "http://top.baidu.com/buzz?b=11&c=513&fr=topbuzz\_b344\_c513"

html = get\_page(url)

for item in parse\_page(html):

list3.insert(tkinter.END, ('' + item['rank'] + ':' + item['title']))

list3.itemconfig(tkinter.END, fg='blue')

代码2：实现与机器人聊天的函数

def speak\_with\_robot(msg):

def get\_module\_dir(name):

path = getattr(sys.modules[name], '\_\_file\_\_', None)

if not path:

raise AttributeError('module %s has not attribute \_\_file\_\_' % name)

return os.path.dirname(os.path.abspath(path))

alice\_path = get\_module\_dir('aiml') + '\\botdata\\alice'

# 切换到语料库所在工作目录

os.chdir(alice\_path)

alice = aiml.Kernel()

alice.learn("std-startup.xml")

alice.respond('load aiml b')

return alice.respond(msg)

算法3：简单的界面设计

# 聊天窗口

# 创建客户端图形界面

root = tkinter.Tk()

root.title(user + ' 群聊天') # 窗口命名

root['height'] = 580

root['width'] = 850

root.resizable(0, 0) # 限制窗口大小

# 创建多行文本框输出聊天内容

listbox = ScrolledText(root)

listbox.place(x=5, y=0, width=660, height=450)

# 文本框使用的字体颜色

listbox.tag\_config('red', foreground='red')

listbox.tag\_config('blue', foreground='blue')

listbox.tag\_config('green', foreground='green')

listbox.tag\_config('pink', foreground='pink')

# 创建多行文本框, 显示在线用户

listbox1 = tkinter.Listbox(root)

listbox1.place(x=670, y=155, width=175, height=170)

# 创建输入文本框和关联变量

a = tkinter.StringVar()

a.set('')

entry = tkinter.Entry(root, width=120, textvariable=a)

entry.place(x=5, y=480, width=660, height=90)

list2 = '' # 列表框

# 创建列表框展示群文件

list2 = tkinter.Listbox(root)

list2.place(x=670, y=25, width=175, height=125)

# 创建列表框展示热点信息

list3 = tkinter.Listbox(root)

list3.place(x=670, y=330, width=175, height=240)

# 创建表情按钮

eBut = tkinter.Button(root, text='表情', command=express, bg='#d3fbfb', relief='raised')

eBut.place(x=5, y=450, width=60, height=30)

# 创建发送图片按钮, 并绑定发送图片功能

pBut = tkinter.Button(root, text='发送图片', command=picture, bg='#d3fbfb', relief='raised')

pBut.place(x=70, y=450, width=60, height=30)

# 创建上传按钮, 并绑定上传文件功能

upload = tkinter.Button(root, text='文件上传', command=put, bg='#d3fbfb', relief='raised')

upload.place(x=135, y=450, width=60, height=30)

# 创建视频通话, 并绑定视频通话功能

vbutton = tkinter.Button(root, text="视频通话", command=video\_invite, bg='#d3fbfb', relief='raised')

vbutton.place(x=200, y=450, width=60, height=30)

# 创建发送按钮

button = tkinter.Button(root, text='发送', command=send, bg='#d3fbfb', relief='raised')

button.place(x=605, y=540, width=60, height=30)

root.bind('<Return>', send) # 绑定回车发送信息

# 三 功能介绍

3.1基本聊天功能

服务器端：初始化socket,在特定的端口上等待来自客户的连接请求，并且能够记录所有成功的连接；接受用户传送来的连接判断命令，并向用户发出响应命令；支持多个客户端的连接，在服务器和多个客户端之间通过TCP进行数据传输；能接受用户传送来的聊天信息，然后所有链接的客户端转发。

客户端：建立和维护与服务器的连接，并且随时检测连接的状态；能够把用户输入的信息及时发送到服务端；有一个消息显示区。客户端右侧的中部有一个在线用户列表，显示在线人数和在线用户的用户名，点击用户名可以切换到与该用户私聊的状态，点击“群聊天”切换回群聊状态，聊天状态会显示在客户端左上角处。

客户端还有一些操作信息提示功能：例如如果客户端用户不输入用户名时，直接点击“登录”，会提示“用户名为空！”。如果用户不输入聊天信息时，直接点击“发送”，会提示输入“发送信息为空！” 客户端提供四种表情图标，分别是“抱拳”、“玫瑰”、“微笑”和“再见”。点击“表情”按钮，选择表情图标，单击图标发送。

3.2表情功能

客户端提供四种表情图标，分别是“抱拳”、“玫瑰”、“微笑”和“再见”。点击“表情”按钮，选择表情图标，单击图标发送。

3.3图片发送功能

客户端提供GIF图片发送功能。单击“发送图片”，选择要发送的给GIF图片，点击“打开”，成功发送。

3.4文件上传功能

客户端提供文件上传功能，上传的文件保存在群资源文件夹“resources”内。单击“文件上传”按钮，选择上传文件，点击“打开”，成功上传。

3.5视频通话功能

客户端提供视频通话功能，此功能仅用于用户之间私聊，不能与自己视频通话、不支持视频群聊，也不支持与AIML机器人视频通话。在私聊状态下单击“视频通话”，对方的客户端会弹出询问窗口，若对方点击“接受”，立即弹出视频窗口，可以开始视频通话。

3.6爬虫功能

在客户端右下角有一个信息窗口，显示今日热点、实时热点和体育热点的前三条头条信息。这些信息都是通过爬虫在百度热点的网页上实时抓取的，与最新信息保持同步。

3.7AIML机器人功能

在客户端处，用户能与AIML机器人聊天。当用户发送的信息能与对话库中的信息匹配上时，机器人就会回答事先设计好的句子。

AIML机器人使用AIML——人工智能标记语言(Artificial Intelligence Markup Language)，它是一种XML兼容的语言，是一种定义了匹配模式和决定响应的规则的一种XML，很容易学习。AIML最重要的单元有：

开始和结束一个AIML文档的标签（tag）

标志着一个艾尔博特知识库（Alicebot's knowledge base）的“知识单元”标签<</font>类别>

用于包含匹配用户给机器人可以说或键入字符的一个简单模式（simple pattern）<<font face="宋体">模式>

包含对用户输入（user input）的响应<<font face="宋体">模板>

AIML文件中经常发现有20个左右或附加多个标签（tags），创建自己所谓“自定义谓词（custom predicates）”是可能的。

机器人要变活，需要用一些语言来实现AIML规范，但一些好心人已经用Python做了。先用pip装aiml包，但aiml包只能在Python 2下用。 Python 3可以用GitHub上的Py3kAiml替代。

# 四 系统测试

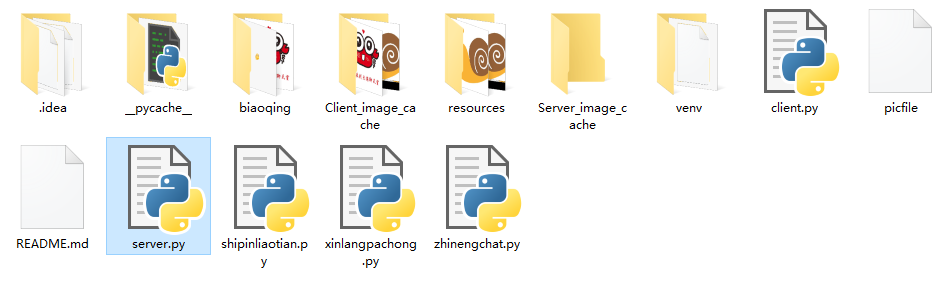
本实验服务端和客户端是分开运行。

1. 启动服务端，打开监听
2. 启动客户端，连接到服务端。
3. 测试发送和接受聊天信息。
4. 增加新的客户端，测试多用户聊天
5. 表情功能测试
6. 图片发送功能测试
7. 文件上传功能测试
8. 视频通话功能测试
9. 爬虫功能测试

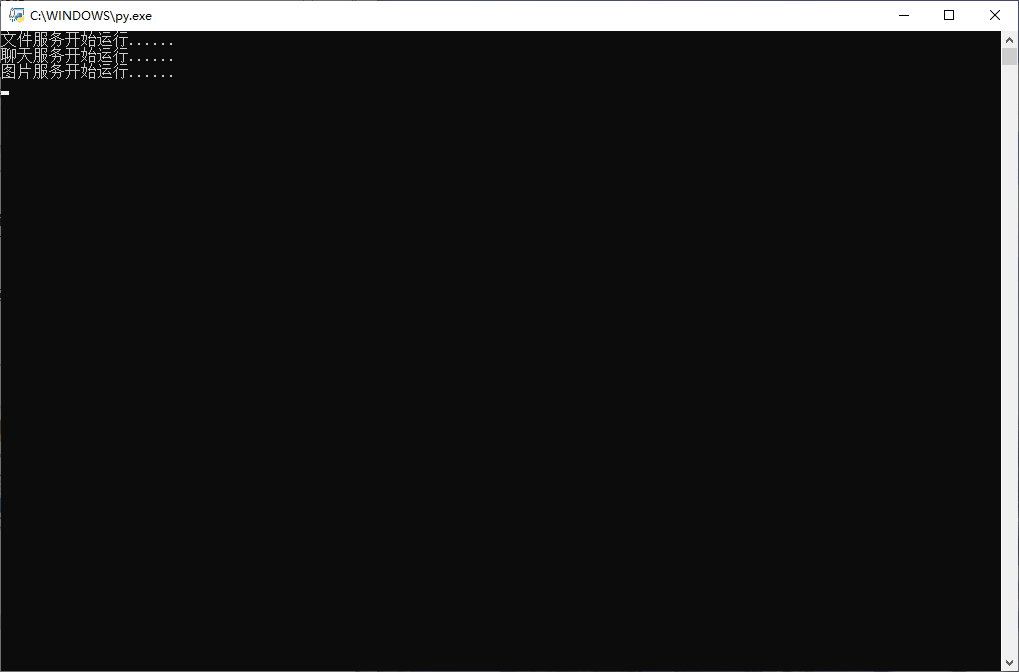
10、AIML机器人功能测试

4.1启动服务端

直接运行server.py文件



弹出启动如下窗口

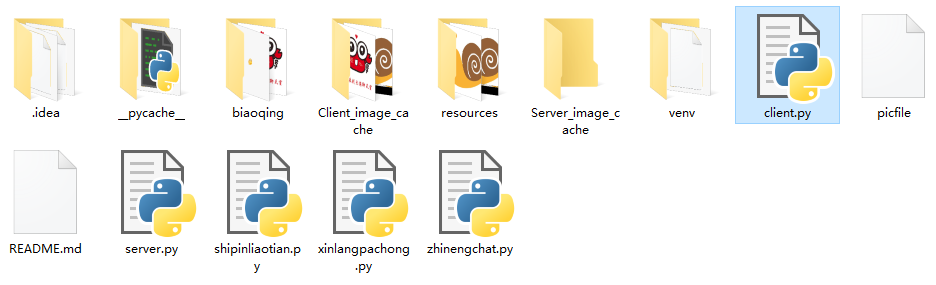


出现“聊天服务开始运行......”、“文件服务开始运行......”和“图片服务开始运行......”这三句话，说明服务器正常开启成功。

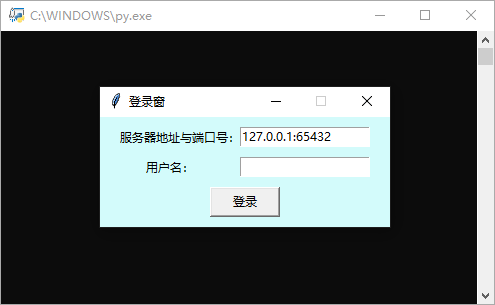
本作业为测试方便，都是在本机上进行测试，所以默认IP为 127.0.0.1，聊天端口为65432，文件传输端口为65433，图片传输端口为65434。

4.2启动客户端

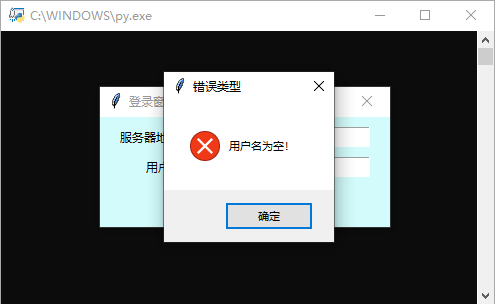
直接运行client.py文件



弹出客户端登录窗口如下



若不输入用户名直接点击登录，会弹出错误提示，如下图所示

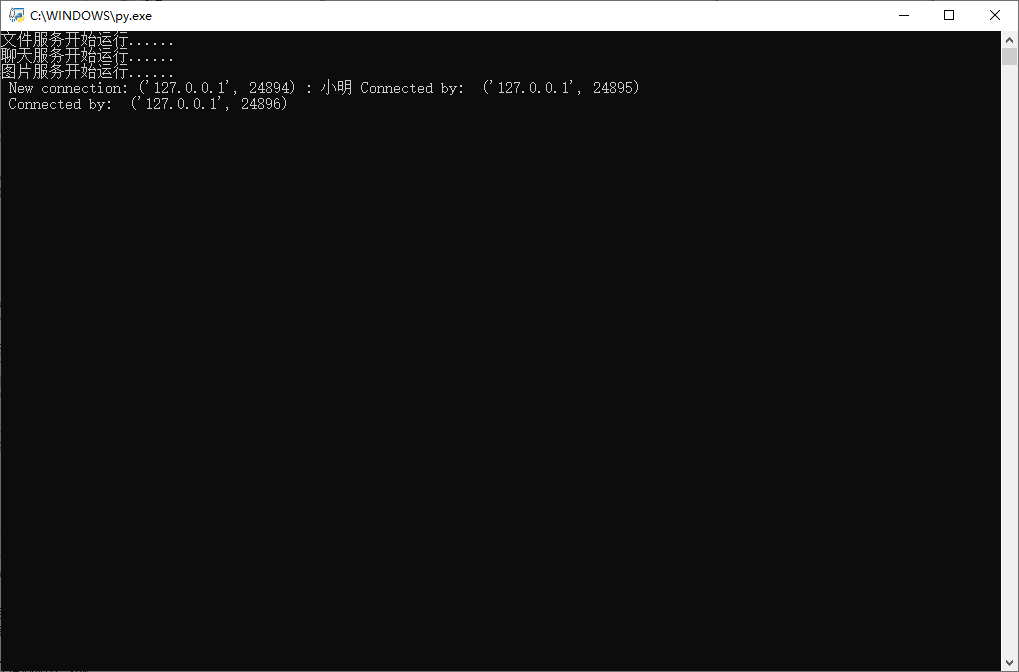


输入用户名“小明”，点击登录，弹出如下窗口



用户默认为群聊天状态。

如下图所示服务器与“小明”建立连接



4.3发送聊天信息测试

客户端用户发送“大家好！我是小明。。。”的信息



4.4增加新的客户端，测试多用户聊天

开启多个客户端

用户“aaa”发送“你们好”，用户“bbb”回复“你好”，群聊功能良好。



用户不能与自己私聊。



用户不能与AIML机器人视频通话。



群聊天不支持视频通话。



用户不输入聊天信息时，直接点击“发送”，会提示“发送信息为空！”



4.5表情功能测试

有四种表情可以选择

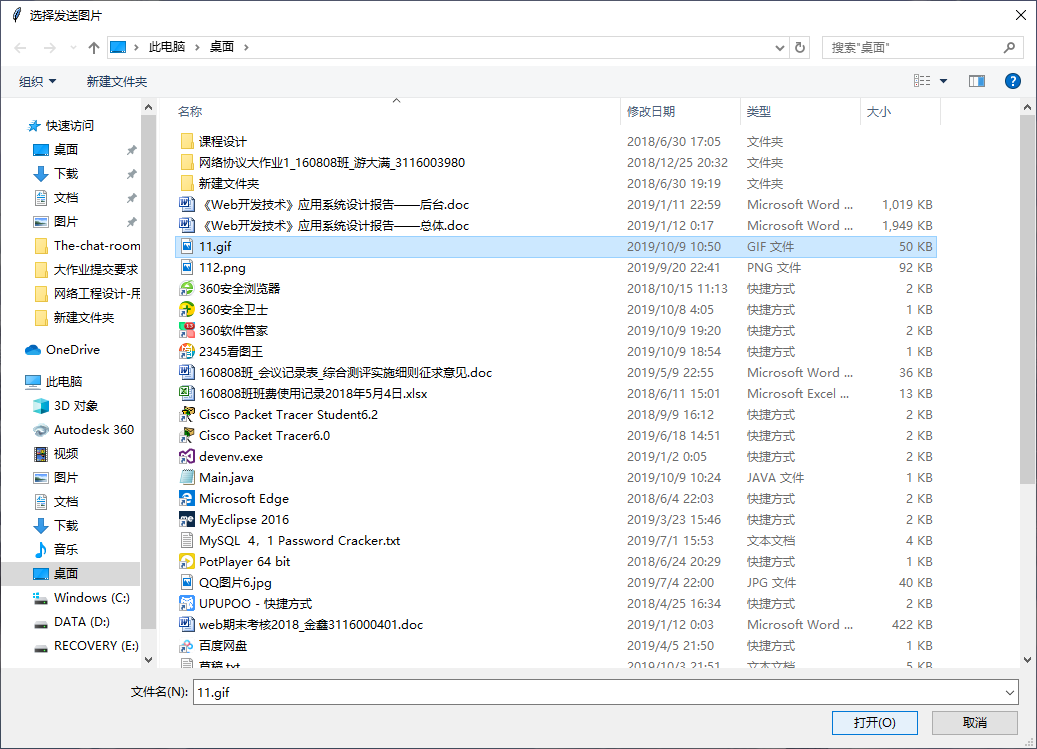


发送效果如下图所示。



4.6图片发送功能测试

单击“发送图片”，选择11.gif图片文件，单击“打开”。

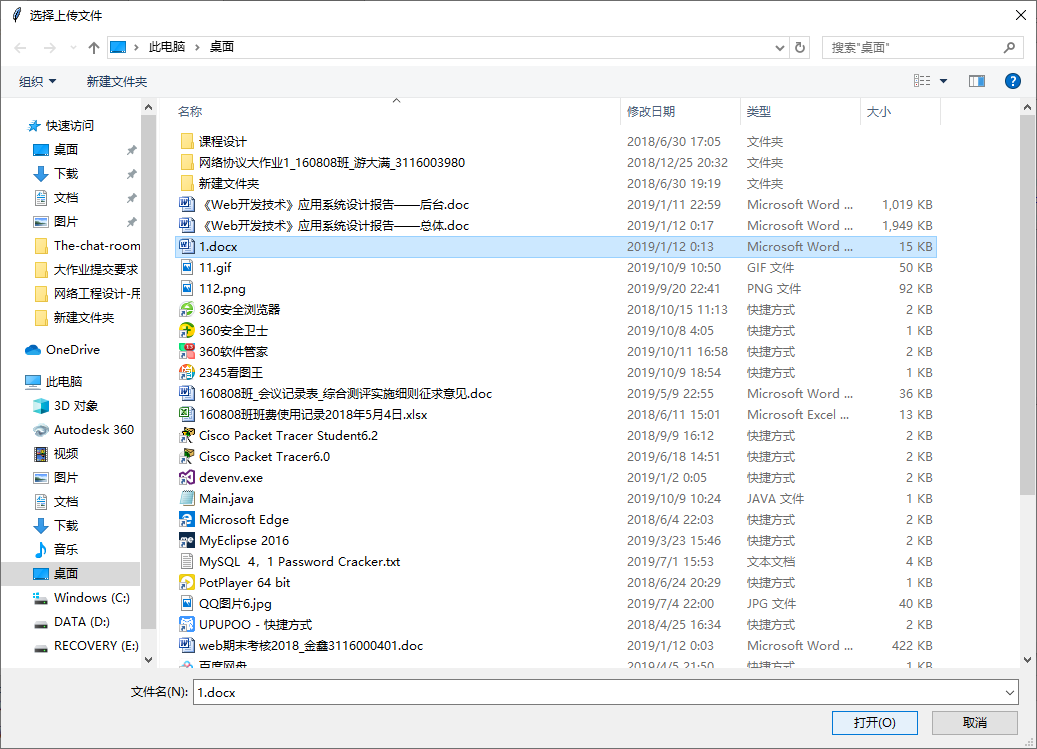


发送成功。



4.7文件上传功能测试

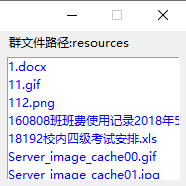
单击“文件上传”，弹出选择窗口，选择1.docx文件，单击打开



上传成功，弹出提示框



单击“确定”文件列表更新。

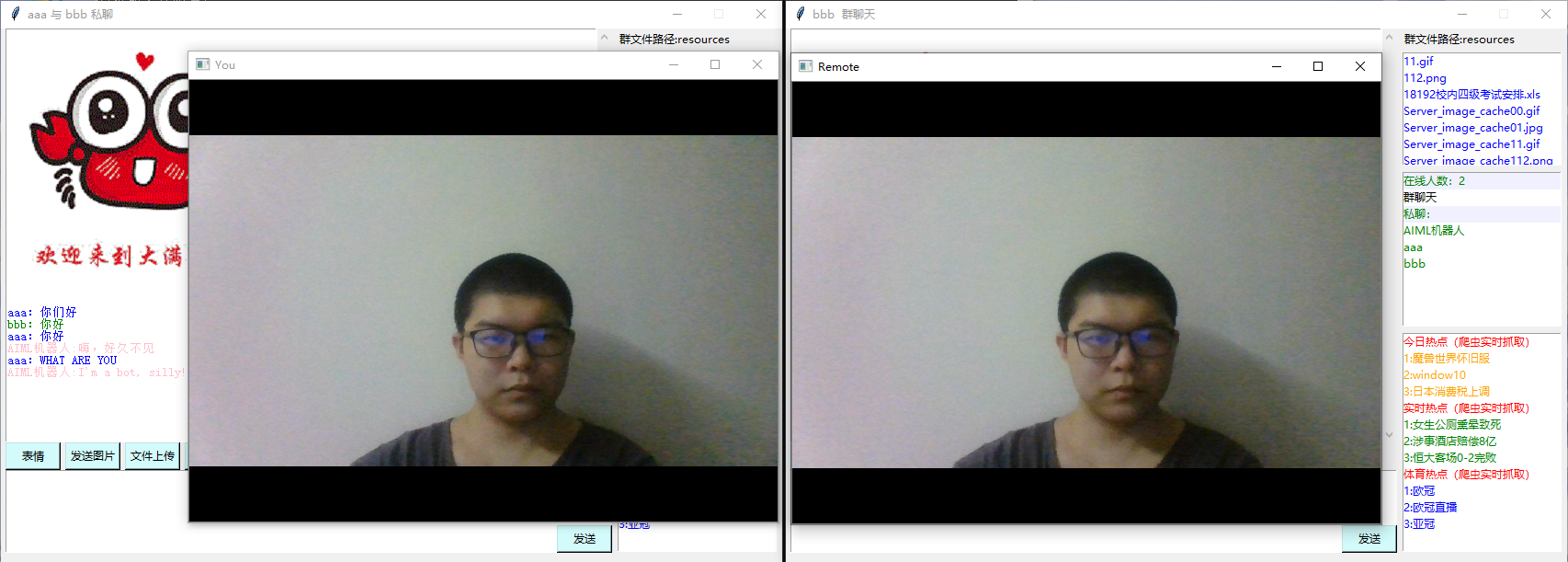


4.8视频通话功能测试

用户“aaa”向用户“bbb”发起视频通话邀请，用户“bbb”的客户端弹出提示框询问。

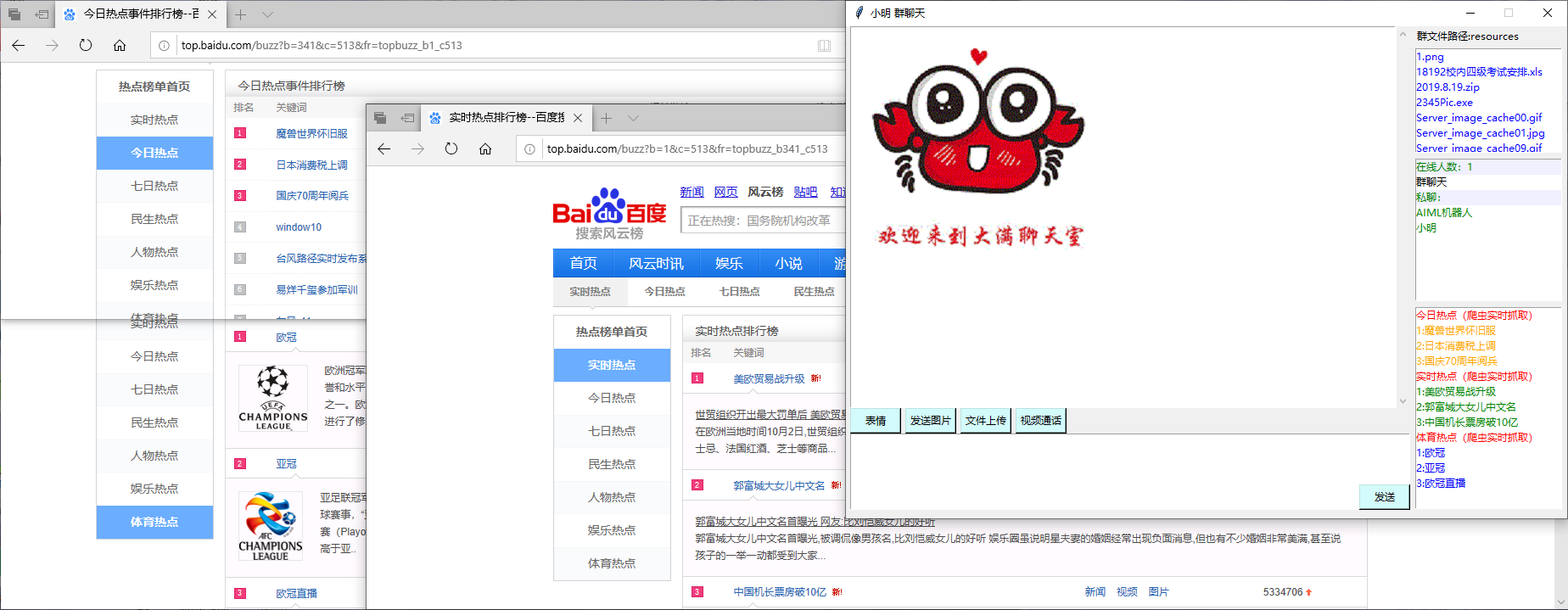


用户“bbb”单击“接受”弹出视频聊天框。



4.9爬虫功能测试

右侧热点信息是爬虫从百度热点实时抓取而来，保持同步更新



4.10 AIML机器人功能测试

用户“aaa”机器人与机器人私聊，发送“你好”，机器人回复“嗨，好久不见”。

用户“aaa”机器人与机器人私聊，发送“WHAT ARE YOU”，机器人回复“I'm a bot, silly!”。

私聊信息仅双方可见，其他用户不会收到。



用户“小明”与AIML机器人聊天



# 五 心得体会

感谢曾老师的悉心指导！此次的作业的时间比较长，完成的功能也比较多，特别是多用户同时聊天，需要同时用到tkinter和Soket的大量知识，让我学会了许多东西，也算能进一步学习了一下python了，以及进一步了解如何应用Socket编程的一些基础知识，学习到了用Socket编写服务器应用程序。学习到了用CSocket编写客户端应用程序，以及服务器如何接受来自客户端的请求。学习到了Socket程序是如何让接收来自客户端的请求的，以及Socket是如何在服务器与客户端之间发送和接受数据的。然后提示用户输入向服务器发送的字符串，更加深刻的理解Socket服务器与客户端的特点。在课程设计的初期，从开始熟悉这些知识到对整体设计有了解，再设计到开始编码，以及最后的测试，整个过程感觉很充实，虽然遇到了不少困难，但设计出解决方案并成功实现测试时，那种成就感和满足感足以鼓励自己加班加点的辛苦。同时，在实践过程中，回顾书本上的理论知识，巩固了我的知识。特别是与tkinter有关的编程以及Scoket服务端和客户端的编程能力有所提高，加深理解Windows程序设计的基本概念。此外还学习了如何实现一个爬虫功能，和使用AIML定制一个对话机器人。

通过这个作业，确实学到了不少的东西，同时也深感自己知识的欠缺。使在以后的学习工作中，一定要继续坚持不断地学习新兴的专业知识及相关的非专业知识，只有这样才能紧跟时代的潮流。

# 参考文献

[1]赵宏,包广斌,马栋林.Python网络编程(Linux)【M】.北京：清华大学出版社.2018.

[2]张若愚.Python 科学计算【M】.北京：清华大学出版社.2012.

[3][挪] Magnus Lie Hetland.Python基础教程【M】.北京：人民邮电出版社.2014.

[4]凌俊峰.TCP/IP协议浅释【J】.韶关学院学报.2001年09期.

[5]电脑虎.网络大厦的互联基础——TCP/IP协议【N】.中国电脑教育报.2004:12-23.

[6]高传善.数据通信与计算机网络【M】.北京：高等教育出版社.2002.

[7]梁振军，梁波.计算机互联网技术与TCP/IP协议【M】.北京：海洋出版社.2006.

[8]夏阳,陆余良,孙乐昌.基于网络攻击的安全防范策略【J】.安徽大学学报(自然科学版).2002年02期.

[9]沈建苗.排除TCP/IP的基本故障【N】.计算机世界.2005年.

[10]唐松，陈智铨.Python网络爬虫从入门到实践【M】.北京：机械工业出版社.2017.

[11]崔庆才.Python 3网络爬虫开发实战 【M】.北京：人民邮电出版社.2018.

[12]mishidemudong.10分钟构建人人都能学会的个性化聊天机器人-使用AIML【EB/OL】.https://blog.csdn.net/u010159842/article/details/81975036, 2018-08-23/2019-9-15.

[13]HadoopDev.自己动手开发智能聊天机器人完全指南【EB/OL】.

https://blog.csdn.net/hadoopdevelop/article/details/79282901, 2018-02-07/2019-9-15.

[14]四季信风.智能客服 对话实现--python aiml包【EB/OL】.

https://www.cnblogs.com/brucekun/p/7667780.html, 2017-10-14/2019-9-15.

[15]shuifu1988.Python 的 AIML【EB/OL】.

https://blog.csdn.net/shuifu1988/article/details/77965320, 2017-09-13/2019-9-15.

[16]许恕.AIML应答机器人(一)aiml简介【EB/OL】.

https://blog.csdn.net/xvshu/article/details/80973649, 2018-07-09/2019-9-15.

[17]ahilll.Python GUI之tkinter窗口视窗教程大集合【EB/OL】.

https://blog.csdn.net/ahilll/article/details/81531587, 2018-08-09/2019-9-15.

[18]小喜.python-入门的第一个爬虫例子【EB/OL】.

https://www.cnblogs.com/xiaoxi-3-/p/9029065.html, 2018-05-12/2019-9-15.