

北京林业大学

2022 学年—2023 学年第 2 学期 Python 应用 实验报告书

专 业： 大数据 班 级： 大数据 212

姓 名： 余睿捷 学 号： 211002328

实验地点： 机房 N07 任课教师： 王春玲

实验题目： 实验 4 使用 TCP 实现智能聊天机器人

实验环境： Python、PyCharm 等

一、实验目的

1. 熟悉标准库 socket 的用法。
2. 熟悉 TCP 的工作原理。
3. 理解端口号的概念与作用。
4. 熟悉 socket 编程。
5. 熟练掌握字典的使用。
6. 熟悉集合的常用运算。
7. 了解 os.path 中 commonprefix() 函数的用法。
8. 熟练掌握字符串的常用方法。

二、实验内容

编写聊天程序的服务端代码和客户端代码。完成后,先启动服务端代码,然后启动客户端程序输入问题,服务端可以返回相应的答案。要求服务端代码具有一定的智能,能够根据不完整的问题识别客户端真正要问的问题。

程序运行后界面如下图所示。

```
Listening on port: 50007
Connected by ('127.0.0.1', 49404)
Received message: how are you
Received message: what's your name
Received message: how old are you
Received message: bye
>>> |
```

服务端

```
Input the content you want to send:how are you
Received: Fine,thank you.
Input the content you want to send:what's your name
Received: xiaoming
Input the content you want to send:how old are you
Received: 18
Input the content you want to send:bye
Received: Bye
>>> |
```

客户端

程序运行界面

三、实验步骤及结果

Server:

```
import socket
from os.path import commonprefix

# 建立聊天回复字典
words = {'how are you?': 'Fine, thank you!',
         'what's your name?': 'xiaoming',
         'how old are you?': '18',
         'bye': 'Bye'}

# 服务端主机 IP 地址和端口号，空字符串表示本机任何可用 IP 地址
HOST = ""
PORT = 2328

# 使用 IPV4 协议，使用 tcp 协议传输数据
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# 绑定端口和端口号
s.bind((HOST, PORT))

# 开始监听，规定最多支持 1 个客户端连接
s.listen(1)
print('目前监听的端口号是: ', PORT)
conn, addr = s.accept()
print('目前连接的 IP 地址是: ', addr)

# 开始聊天
while True:
    # 最多可以接收 1024 比特大小的内容，并解码
    data = conn.recv(1024).decode()
    # 如果是空，退出
    if not data:
        break
    print('接收到的内容: ', data)
    # 尽可能猜测对方的意思
    m = 0
    key = ""
    for k in words.keys():
        # 删除多余的空白字符
        data = ' '.join(data.split())
        # 与某个键非常接近，就直接返回
        if len(commonprefix([k, data])) > len(k) * 0.7:
            key = k
            break
```

```

        #使用选择法，选择一个重合度较高的键
        length=len(set(data.split())&set(k.split()))
        if length>m:
            m=length
            key=k
    #选择合适的信息进行回复
    conn.sendall(words.get(key,'Sorry,can\'t find your problem! ').encode())
conn.close()

```

Client:

```

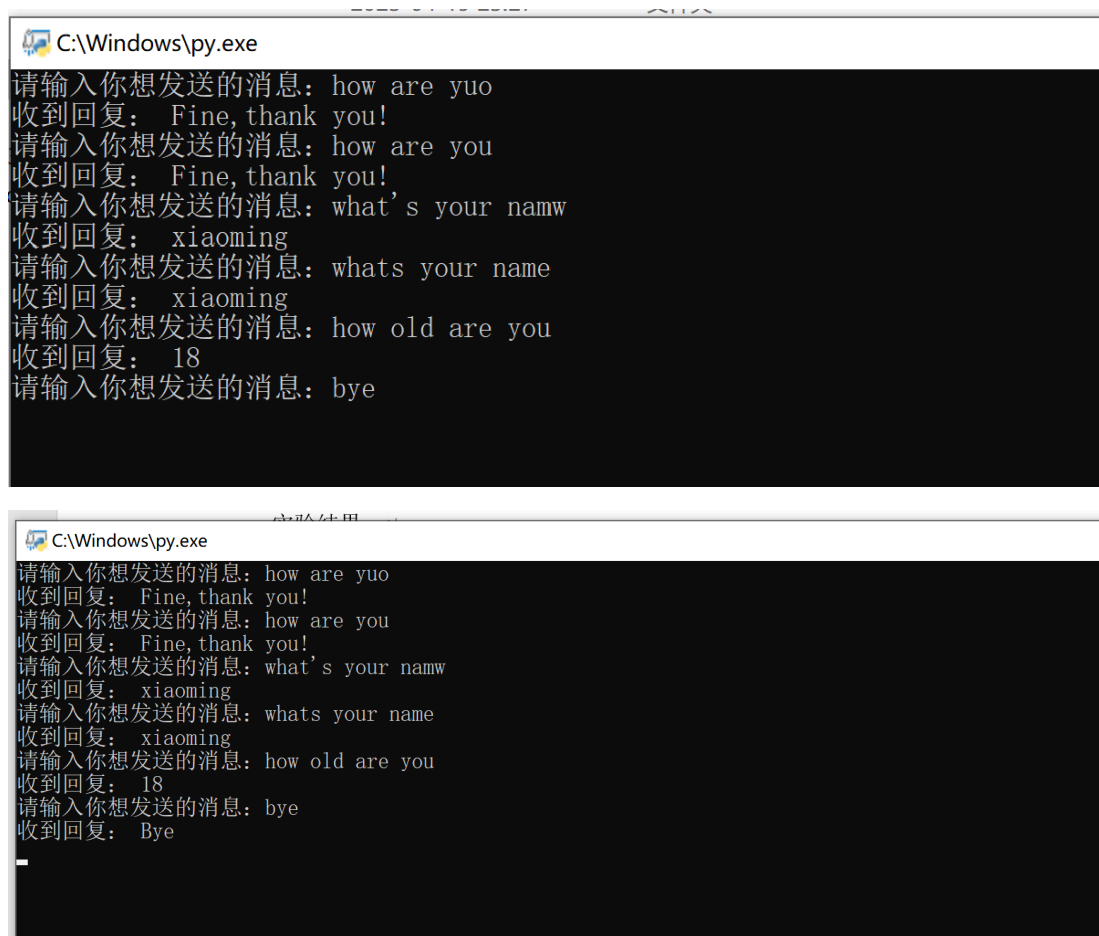
import socket
import sys

# 服务端主机 IP 地址和端口号
HOST = "127.0.0.1"
PORT = 2328

# 使用 IPV4 协议，使用 tcp 协议传输数据
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
try:
    # 连接服务器
    s.connect((HOST, PORT))
except Exception as e:
    print('找不到服务器，请稍后重试！')
    sys.exit()
while True:
    c = input('请输入你想发送的消息：')
    # 发送数据，使用 UTF-8 编码成字节码
    s.sendall(c.encode())
    # 从服务端接收数据，大小最多为 1024 比特
    data = s.recv(1024)
    # 解码
    data = data.decode()
    print('收到回复：', data)
    if c.lower() == 'bye':
        break
# 关闭连接
s.close()

```

实验结果：



```
C:\Windows\py.exe
请输入你想发送的消息: how are yuo
收到回复: Fine,thank you!
请输入你想发送的消息: how are you
收到回复: Fine,thank you!
请输入你想发送的消息: what's your namw
收到回复: xiaoming
请输入你想发送的消息: whats your name
收到回复: xiaoming
请输入你想发送的消息: how old are you
收到回复: 18
请输入你想发送的消息: bye

C:\Windows\py.exe
请输入你想发送的消息: how are yuo
收到回复: Fine,thank you!
请输入你想发送的消息: how are you
收到回复: Fine,thank you!
请输入你想发送的消息: what's your namw
收到回复: xiaoming
请输入你想发送的消息: whats your name
收到回复: xiaoming
请输入你想发送的消息: how old are you
收到回复: 18
请输入你想发送的消息: bye
收到回复: Bye
```

四、实验分析

问题 1: 端口号输入错误, 导致无法绑定成功或监听失败。

解决方法: 统一端口号, 用空字符串表示本机任何可用 IP 地址

问题 2: 在使用 `commonprefix()` 函数时, 没有正确理解函数的用法和返回值, 导致匹配出错

解决方法: 通过查阅资料, 了解 `commonprefix()` 函数的用法, 成功实现“尽可能猜测客户端的意思, 选择合适的信息进行回复”。

收获:

服务端代码实现了一个简单的聊天程序的服务端, 运行后会在指定的端口上监听客户端连接, 接收客户端发送的消息, 对消息进行简单的处理后返回相应的回复。具体流程如下: 定义一个聊天回复字典 `words`, 其中包含了一些常见的问题及其对应的回答。创建一个 TCP socket 对象 `s`, 指定使用 IPv4 协议和 TCP 协议进行数据传输。将 socket 对象 `s` 绑定到指定的 IP 地址和端口号上。开始监听绑定的端口, 等待客户端连接。当客户端连接成功后, 通过 `accept()` 方法接收客户端的连接请求, 得到一个新的 socket 对象 `conn` 和客户端的地址 `addr`。进入聊天循环, 使用 `recv()` 方法接收客户端发送的消息, 然后对消息进行简单的处理, 尽可能猜测客户端的意思, 选择合适的信息进行回复, 最后使用

`sendall()` 方法将回复消息发送给客户端。循环结束后，关闭连接。

客户端定义了服务端主机 IP 地址和端口号。创建一个基于 IPV4 协议、使用 TCP 协议传输数据的 `socket` 对象。连接服务器。不断循环，输入要发送的消息并发送到服务端，接收来自服务端的回复并打印出来，直到输入“bye”退出循环。

通过上面两段程序的编写，我熟悉了 Python 中 `socket` 模块和 TCP 协议的基本使用方法，加深了对于网络编程的理解和掌握。

加深了对于字典和集合的理解和掌握，加强了对于 Python 内置数据结构的掌握。

学会了如何使用 `os.path` 模块中的 `commonprefix()` 函数来解决路径匹配问题。

提高了代码设计和实现能力，学会了如何使用 Python 来实现一个简单的聊天机器人，并可以应用到实际的开发中。

了解了一些并发编程的基础知识。