Lesson Plan 13 | Form 2 | Chatrooms with Python

Objective

Students will analyze and experiment with a chatroom program in Python to understand how it functions. They will also assess it for shortcomings, with suggestions on how to fix them.

Warm-up (requires central file server)

Prior to class, place the two Python files below in each student's central file directory. Explain to students that they will not be told what the program does; rather, they are to open it in PyCharm, run it, and experiment to understand its purpose. Encourage them to work together to figure out how the program works.

Presentation

After a half hour or more of independent experimentation in the lab, ask the students what the program does and what can be done with it. Walk through the main sections of the *chat.py* program with the class, asking for explanation as to what various parts do. Point out, with emphasis, the sections where sockets and threads are used. Finally, compare the program to chat applications like Facebook, IM, etc. Ask students what improvements we could make to this program, write them down on the whiteboard, and ask them to keep one or two in mind for implementation in future lessons.

Guided Practice

None.

Independent Practice

For the remainder of class, allow students to continue chatting with their friends, reviewing the program, etc.

Closing

None.

In Hindsight

02/20/2017: Most students did well with independently and collaboratively figuring out how the program worked, although a few had to be prodded to interact with the program as it waited for user input. Some had trouble with the prompt for a port number, while others breezed through it with no issues. The collaborative, investigative nature of this lesson was an asset, and I could see many students enjoyed the independence given to them, even if some of them weren't interested in how the chatrooms were actually programmed to work.

chat.py

```
import chatroomlib
import socket, threading
choice = raw_input("Type 'h' to create a chatroom or 'j' to join one: ")
# If the user pressed 'h', create a new chatroom.
if choice == 'h':
    socket = chatroomlib.create_chatroom()
    try:
        while True:
            msg = raw_input()
            chatroomlib.send(msg)
    except KeyboardInterrupt:
        print "Closing your chatroom..."
        socket.close()
        print "Chatroom closed. Goodbye."
# If the user pressed 'j', join the chatroom of their choice.
elif choice == 'j':
    ip = raw_input("What is the IP address of the chatroom you want to join: ")
    port = raw_input("What is the port of the chatroom you want to join: ")
    s = socket.socket()
    s.connect((ip, int(port)))
    def show_new_messages():
        while True:
            messages = s.recv(1024)
            if messages:
                print messages
            else:
                print "The chatroom was closed, goodbye."
                s.close()
                return
    new_msg_thread = threading.Thread(target=show_new_messages)
    new_msg_thread.daemon = True
    new_msg_thread.start()
    try:
        while True:
            msg = raw_input()
            s.send(msg)
    except KeyboardInterrupt:
        s.close()
        print "You are leaving the chatroom. Goodbye."
# If the user didn't press 'j' or 'h',
# tell them that they need to try again.
else:
    print "You didn't type 'h' or 'j', try running this program again."
```

chatroomlib.py

```
import socket, threading


people_in_chatroom = {}
unsent_messages = []


def create_chatroom():
    PORT = 4000
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
    s.bind(('', PORT))
    s.listen(100)
    print ("Chatroom created on port %d successfully, " + \
            "now listening for people to join.") % PORT
    bcast_thread = threading.Thread(target=broadcast_received_messages)
    bcast_thread.daemon = True
    bcast_thread.start()
    def listen():
        while True:
            connection, ip_and_port = s.accept()
            people_in_chatroom[ip_and_port[0]] = connection
            t = threading.Thread(target=new_person_in_chatroom, \
                    args=(ip_and_port[0],))
            t.start()
    listen_thread = threading.Thread(target=listen)
    listen_thread.daemon = True
    listen_thread.start()
    return s


def new_person_in_chatroom(ip_address):
    unsent_messages.insert(0, '%s has just joined the chatroom.' % ip_address)
    while True:
        message = people_in_chatroom[ip_address].recv(1024)
        if not message:
            break
        unsent_messages.insert(0, message)
    people_in_chatroom[ip_address].close()
    people_in_chatroom.pop(ip_address, None)
    unsent_messages.insert(0, '%s has left the chatroom.' % ip_address)


def broadcast_received_messages():
    while True:
        if len(unsent_messages) > 0:
            msg = unsent_messages.pop()
            print msg # Ensures that we see messages ourselves.
            for (ip_address, connection) in people_in_chatroom.items():
                connection.sendall(msg)


def send(msg):
    unsent_messages.insert(0, msg)
```