Exercise2EscapeRoomSockets

Jump to bottom

sethnielson edited this page 11 days ago · 3 revisions

Exercise 2: Escape Room with Sockets

Assigned	9/4/2019
Due	9/9/2019
Points	25

Overview

The goal of this exercise is three fold:

1. To learn some Python programming
2. To learn how to use Sockets
3. To learn how to work in Groups

Part 1: Learn some Python

The first thing you need to do is clone this repository. If you don't know how to use git, google it! I'll give you a hint:

```
git clone https://github.com/CrimsonVista/20194NetworkSecurity
```

Once you have the repo, you will find a file called `src/samples/escape_room_001.py` . Before you look at the code, go ahead and run it.

You'll find it's a very simplistic text adventure game. It mimics an escape room. This one is super easy. See if you can figure it out yourself. When you've take a minute or two to explore, you can skip ahead to read the steps:

SPACER

SPACER

SPACER

Just giving you a chance to try it out before I give you the answer.

The steps are these:

1. look mirror
2. get hairpin
3. unlock door with hairpin
4. open door

You have to look at the mirror *first*. If you try to get the hairpin first, it won't work. It only becomes "visible" after you noticed it in the mirror.

Now take a look at the code. There are various Python (v3) elements to learn about. These include (in no particular order):

1. `__setitem__` and `__getitem__`
2. list comprehensions, such as `[item for item in list]`
3. anonymous functions (e.g., `lambda`)
4. classes and functions
5. default arguments
6. using "and" and "or" instead of if/else
7. for loop
8. dynamic function dispatch using `hasattr`
9. string processing
10. dictionaries and lists

Go ahead and look through the code for these elements. See if you can figure them out from the code itself. If you can't, try Google for some of the topics above. The head over to the slack for additional questions and discussions.

For this exercise, you won't need to modify the code. But you *will* need to modify this code for the next exercises.

What you will need for this exercise is to create a networked version that can pass the auto-grader.

Part 2: Using sockets

The escape room code provided is only used for local operations. What if you wanted to connect to an escape room over the internet?

The first thing you need to do is read the assignment about sockets. You need to understand the following:

1. `socket.connect`
2. `socket.bind`
3. `socket.listen`
4. `socket.accept`
5. `socket.recv`
6. `socket.send`

You will *not* need to use `accept` and `listen` to use the auto-grader, but you will if you (wisely) want to test your own code. So I *highly* recommend that you learn how they work.

Once you have an idea of how these functions work, let's move on to the next section.

First, let's talk about a server. This is the part that uses `accept` and `listen`. I'm not going to show you how to use that. That's *your* job. But here's how you interface with EscapeRoom once you have `accept` working.

In the current format, the code looks like this:

```
game = EscapeRoomGame()
game.create_game(cheat=("--cheat" in args))
game.start()
while game.status == "playing":
    command = input(">> ")
    game.command(command)
```

What's going on here is that we create an EscapeRoomGame object, called `game`, start the game, and then run the game in a while loop until it's done.

Notice the `input` command. It prints out `>>` to the screen, reads input from the keyboard, and returns it (stored, in this example, in `command`). The command is then passed to the game's command processing function, also called `command()`.

You need to create a server. The `accept` command will provide you with a socket with inputs and outputs. Then, instead of using `input` to get user input, you will receive input from `socket.recv`. (There's no need to send the `>>` to the client. The client can get input from the screen however it chooses)..

If you experiment with socket a bit, this should make a lot of sense.

There is one other place you need to deal with output. The game, itself, has an "output" engine for outputting data. The default output is "print", which just prints to the screen. You can change the output to any function during the construction of the EscapeRoomGame object.

```
EscapeRoomGame(output=my_write_function)
```

Of course, you can change `my_write_function` to whatever you call your function. Whatever function you create, it has to take a string as input. This is important because `socket.send` takes *bytes* as input. You should read up on how Python 3 distinguishes between bytes and strings. This is different from Python 2.

Once you pass in the network-enabled function, the game will automatically write the output over the network.

Finally, there is one other issue with sockets, and we will cover this in class, is that there is no way to distinguish between two separate "messages" sent via `socket.send`. That is, you could call the function twice, thinking you are sending two separate messages, but on the other side, the `recv` methods receives the two "messages" combined into a single bytes string. How would you recognize the difference?

We will learn about different kinds of messages in Internet, but for this exercise, you need to end every message with `<EOL>\n`. When receiving data, you need to split the data:

```
data = s.recv(1024) # this could be multiple messages
data_as_string = data.encode() # convert from bytes to string
lines = data_as_string.split("\n")
for line in lines:
    # process each line.
```

For your information, EOL just stands for End Of Line. This is made-up. Nobody does this, although SMTP has something similar.

On the client side, things are conceptually easier. All you need to do is use a command like `input` to get a user's input, and then use `socket.send` to send the input over the wire, and `socket.recv` to get the response. All the previous comments apply, and the data needs to be sent with `<EOL>\n` appended to each transmission.

Now, for auto grading, you do NOT need the server, per-se. Instead, you will create a client that will operate as both a client and a server. Here is how your client must work in order to get full credit

1. Your client must connect from a class VM to 192.168.200.52 port 19002
2. The server will ask you for your name.
3. Your client must submit a name. This name must be recognizable, as it will be used to assign grades.
4. The server will start transmitting data and a request for a command.
5. Your client must send commands that eventually result in escaping the room.
6. The server will report any output and a success message. The client *must* pause at least 0.25 seconds after sending the last command before moving onto the next section. I recommend `time.sleep(0.25)`.
7. The server will then begin to issue commands to your client. You need to pass these commands to an EscapeRoomGame, get the results, and send them back to the server
8. The server will perform a series of commands, eventually exiting the room, and then sending a success message

Although the server will send a success message, from the client side, you only need to stay connected as long as the game is running. Once the server escapes, you have completed the assignment

As before, make sure that ALL output of any kind has "\n" appended to it. The server will try to send back error messages, but it is unlikely that you will completely figure out what goes wrong without thinking through it carefully.

Part 3: Learn to work as a team.

EVERY STUDENT MUST SUBMIT THEIR OWN ENTRY. YOU MUST NOT SUBMIT AS A TEAM. EVERY PERSON MUST WRITE THEIR OWN CODE

HOWEVER, I do expect you to work as a team. As a team, you should

1. Teach each other what you know or are learning about Python
2. Test each other's code against each other
3. Observe submissions to the auto-grader together to figure out what is going right and/or wrong
4. Review each person's submission to talk about what you like better about each person's code style, approach, etc.

The teaching part of this is crucial. This class requires learning to work together and a big part of that is teaching.

I have no way of grading this yet. But later assignments may include a written element describing how you learned to work together.

Clone this wiki locally

```
https://github.com/CrimsonVista/20194NetworkSecurity.wiki.git
```