Lab 5

Sockets: Rock-paper-scissors

[Compulsory]

Authors: Ragnar Nohre, moved to English Johannes Schmidt

This lab's goal is to exercise socket programming in python. Recommended reading is the lecture on sockets and the references *https://docs.python.org/3.13/library/socket.html* and *docs.python.org/3.13/howto/sockets.html*

5.1 Rock-paper-scissors

Rock-paper-scissors is a classic. We represent the rock by 'R', paper by 'P', and scissors by 'S'. The game's rules are simple. Two players. Each enters a symbol (R, P, S), then gets the opponent's symbol and then one (or no) player gets points according to the following rules:

1. rock wins over scissors (rock destroys scissors)
2. scissors win over paper (scissors cut paper)
3. paper wins over rock (paper wraps rock)

It shall work like this:

1. Both players use the same program, each player launches the program on her own computer.
2. The program asks if the player wants to be client (C) or server (S). One player must be server and the other must be client. The player who chooses to be server must answer first!

   (a) If the player answers S, the program creates a server that listens on port 60003 on the current computer[1]

   (b) If the player answers C, the program shall ask for the server's IP-address and then connect to the port 60003 under the given IP-address.

3. Now the program shall print the game's state and ask the player to enter a move. It shall look like this (*this* player's points first)

   (0,0) Your move:

4. The program shall then receive the player's move. Verify that it is a valid move (only R, P or S is allowed). Ask to reenter the move if invalid.

5. When the player has entered a valid move, the program shall send it to the opponent over the socket connection.

6. Then the program shall receive the opponent's move over the socket connection. This can imply that the program is frozen while waiting for the opponent's move, but it can also be that the opponent has already sent her move and thus the opponent had been waiting for a while.

7. When we have read the opponent's move we shall print it to the screen like this:

   (opponents move: X)

8. Then we determine who won the round (if one won) and update the game's state. If no player got 10 points total we go on, that is we continue at 3 above.

9. If one player has got 10 points the program shall tell each player if she won or not and with how many points. Like this:

   You won 10 against 7

10. After this the game shall stop and the socket connection close.

Below you find an example run:

[1] If that port number does not work, try another one. Know also this: if one has recently run a server under a certain port number, one usually has to wait some seconds after shutting down the server until the port number can be used again.

```
schjoh$ python rps.py
Do you want to be server (S) or client (S): C
Enter the server's name or IP: 127.0.0.1
(0,0) Your move: S
(opponent's move: P)
(1,0) Your move: R
(opponent's move: S)
(2,0) Your move: S
(opponent's move: S)
(2,0) Your move: P
(opponent's move: S)
(2,1) Your move: R
...

(8,9) Your move: R
(opponent's move: P)
You lost with 8 against 10
```

5.1.1 Hints

1. Use python version 3.x (not 2.x)!
2. Test run in interactive mode before you write in your file.
3. Avoid Swedish characters (ÄÖÅäöå), they can cause problems.
4. In python 3 you can read from the keyboard with `input("...")`.
   Example:

   ```
   ans =  input("Do you want to be server (S) or client (S): ")
   ```

5. The first lines in my program look about like this:

   ```
   def serversideGetPlaySocket():
      bla bla bla....

   def clientsideGetPlaySocket(host):
      bla bla..

   ans = "?"
   while ans not in {"C", "S"}:
   ```

```
      ans =  input("Do you want to be server (S) or client (S):: ")

if ans=="S":
      sock = serversideGetPlaySocket()
else:
      host = input("Enter the server's name or IP: ")
      sock = clientsideGetPlaySocket(host)
```

The code lines that follow are the same for client and server!

6. A string should be transformed to a *byte*-array before it can be sent over a socket connection. The following code shows how this can work:

```
barr = bytearray(str, "ASCII")
```

7. If you receive a *byte*-array and want to transform it to a string you can do the following:

```
str = barr.decode("ASCII")
```

5.1.2 Examination

Submit your python file on Canvas. Present your solution to your lab assistant.