Sofia Saavedra

John Hernandez

ICOM 4036

PyConnect

PyConnect is a language aimed at simplifying communication with devices by allowing the creation of servers and clients in very few steps. It makes use of Python libraries to achieve this such as socket, threading, pickle and PLY. The socket library was used to create the sockets responsible for allowing the communication in the clients and servers created—it establishes the socket in the respective connections and either listens or connects, according to its client or server nature. It also is the main medium from which both processes will communicate themselves, as it uses the socket's buffer. PyConnect allows sending messages to all the users connected in a server or to specific clients in the server.

It is defined in the following way:

Tokens:

Character ::= a-z | A-Z

Digit ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |9

Operators ::= : | ,

Ip ::= \b((([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])(\.)){3}([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5]))\b

Grammar:

run_command ::= create_server
| create_client
| allow_conn
| show_info
| close_connection
| send_message

create_server ::= initiate server : {address Ip ,} port Int , name string

create_client ::= connect to server : address Ip , port Int

allow_conn	::=	allow connection in String : Int
show_info	::=	show clients info in String | show String info
close_connection	::=	close connection : String { in String }
send_message	::=	send message { to String } : String
Int	::=	Int ::= Digit+
String	::=	" Character {Character | Digit | Operators}* "

How to run it:

Run the python file called pyCon_parser.py. It will initiate a loop taking input from the console. Typing in "exit" will close the program.

Functions:

initiate server : {address Ip ,} port Int , name <String>
allows the user to begin the socket to be able to create a server and connect two users; receives the address in which to create the server or if not added, opens the server for any user; the port number as an int; name of server as a String

connect to server : address Ip , port Int
allows user to access previously established server(created with "initiate server:") so that the user can begin communications with each other; address of server of which user desires to communicate in, the port in which the server was initialized in.
allow connection in <String>: Int
is able to limit the amount of users within server; receives name of server as a String, int amount of users allowed

show clients info in <String>
shows users currently connected to server; receives name of server as a String

show <String> info
shows information about server; receives name of server as a String

Close connection <String> {in <String>}
closes server for specific user or current user; receives name of server as a String; can also receive name of user and close the connection of that certain user in the server

Send message {to <String>} <String>
allows user to send message to other user; receives message as a String and can send message towards a specific person

Example of the language:

server:
Initiate server : address 127.0.0.1 , port 12345, name server1
Allow connection in server1 : 2
Show clients info in server1
Close connection : client1 in server1

client:
Connect to server : address 127.0.0.1 , port 12345
Send message : “hello”
Send message to user2 : “how ya doing?”
Close connection : user1