mysocks

Release 1.1.0

Aug 06, 2020

Contents

CHAPTER 1

mysocks

1.1 mysocks package

1.1.1 Submodules

1.1.2 mysocks.Test_Server module

1.1.3 mysocks.file_send module

class `mysocks.file_send.`**`file_send`**(*host='127.0.0.1'*, *port=5560*, *n_listen=1*)

Bases: `mysocks.Model`

This class is parent class to file transfer module.

`accept_connections`(*filename*)

1.1.4 mysocks.file_transfer module

class `mysocks.file_transfer.`**`file_transfer`**(*host='127.0.0.1'*, *port=5560*, *n_listen=1*)

Bases: `mysocks.Model`

This class is parent class to file transfer module.

Parameters

- **`host`** (*`string, optional`*) – IP Address of the socket that the server will bind to. Defaults to localhost
- **`port`** (*`int`*) – port of the socket server. Defaults to 5560
- **`n_listen`** (*`int`*) – Max. number of clients to connect to at one time

Return s socket object

Rtype s socket object

accept_connections()

class mysocks.file_transfer.**receive_files**(*save_path: str = ”*, *host: str = '127.0.0.1'*, *port: int = 5560*)

Bases: `mysocks.Model`

This class is a parent class for all the procedures related to receiving a file.

Parameters

- **save_path** (*str*) – path to save the received files.
- **host** (*str*) – IP Address of the socket server
- **port** (*int*) – port of the socket server. Defaults to 5560

receive_files()

Method to receive files from the socket server

class mysocks.file_transfer.**send_files**(*filenames*, *host='127.0.0.1'*, *port=5560*, *n_listen=1*)

Bases: `mysocks.Model`

This class is a parent class for all file sending methods. Call this class to send file_send

Parameters

- **filenames** (*list*) – List of filenames to send to the client
- **host** (*string*) – IP Address of the socket that the server will bind to. Defaults to localhost
- **port** (*int*) – port of the socket server. Defaults to 5560
- **n_listen** (*int*) – Max. number of clients to connect to at one time

accept_connections()

Method to accept client connections

Returns tuple of conn and addr of the connected client

Return type tuple

lineno()

Returns the current line number in our program. Used to find n=line number where the program is running during error handling

Returns Integer referring the current line of program execution

Return type int

send_files(*filenames*)

Method to send files to client.

Parameters **filenames** – list of path of all the filenames to be sent over to the client. Paths can be absolute or relative.

1.1.5 mysocks.chat module

class mysocks.chat.**chat**(*host='127.0.0.1'*, *port=5660*, *n_listen=5*)

Bases: `mysocks.Model`

This class is parent class to chat module

Parameters

- **host** (*string, optional*) – IP Address of the socket that the server will bind to. Defaults to 127.0.01
- **port** (*int*) – port of the socket server. Defaults to 5660
- **n_listen** (*int*) – Max. number of clients to connect to at one time. Defaults to 5

accept_connections()

class mysocks.chat.**client**(*host='127.0.0.1'*, *port=5660*, ***kwargs*)

Bases: mysocks.Model

This class is a parent class for all text message client methods. Call this class to start a chat client. This method connects to the server at the provided host and port

Parameters

- **host** (*string, optional*) – IP Address of the socket that the client will connect to. Defaults to 127.0.01
- **port** (*int*) – port of the socket server. Defaults to 5660

gui_thread()

Method that will start a new thread to launch the gui when the function is called from terminal

receive_data()

Method to receive data from the server

send_data(***kwargs*)

Method to send text messages to the server that will be relayed to other clients

set_username(***kwargs*)

Method to set the username of the client

start_client(*host*, *port*)

Method to connect the client to the socket server and start sending and receiving messages.

class mysocks.chat.**server**(*host='127.0.0.1'*, *port=5660*, *n_listen=5*, ***kwargs*)

Bases: mysocks.Model

This class is a parent class for all text message sending methods. Call this class to start a chat server. CLients will be able to connect to this server and all the clients will be able to chat with each other. Server will not participate in the server.

Parameters

- **host** (*string, optional*) – IP Address of the socket that the server will bind to. Defaults to 127.0.01
- **port** (*int*) – port of the socket server. Defaults to 5660
- **n_listen** (*int*) – Max. number of clients to connect to at one time. Defaults to 5

Attr clients_connected Total number of clients connected till now including the ones which may have left in between the chat.

Attr all_connections socket object of all client connections. Deleted connections will be replaced by the keyword *deleted*.

Attr type all_names User names of all clients connected.

Attr type active_connections Total number of active clients connected to the chat server.

Attr s server socket object.

`accept_connections`()
Method to accept client connections. Creates a separate thread for each client to receive_data from clients

Returns tuple of conn and addr of the connected client

Return type tuple

`gui_thread`()
Method that will start a new thread to launch the gui when the function is called from terminal

`receive_data`(*conn*, *addr*, *client_no*)
Method to receive data from clients.

Parameters

- **`conn`** (*`socket object`*) – socket object of the client
- **`addr`** (*`socket object`*) – Address of the socket object
- **`client_no`** – Integer identifier of a client

`start_server`(*host*, *port*, *n_listen*)
Method to start socket server

Parameters

- **`host`** (*`string, optional`*) – IP Address of the socket that the server will bind to. Defaults to 127.0.01
- **`port`** (*`int`*) – port of the socket server. Defaults to 5660
- **`n_listen`** (*`int`*) – Max. number of clients to connect to at one time. Defaults to 5

1.1.6 mysocks.gui module

`class` `mysocks.gui.`**`CustomText`**(**args*, ***kwargs*)
Bases: `tkinter.Text`

A text widget with a new method, HighlightPattern

example:

text = CustomText() text.tag_configure("red",foreground="#ff0000") text.HighlightPattern("this should be red", "red")

The HighlightPattern method is a simplified python version of the tcl code at http://wiki.tcl.tk/3246

`HighlightPattern`(*pattern*, *tag*, *start='1.0'*, *end='end'*, *regexp=True*)
Apply the given tag to all text that matches the given pattern

`class` `mysocks.gui.`**`launch`**(***kwargs*)
Bases: `object`

This class is a parent class for all tkinter gui related tasks Call this class to get an object of launch class to start a tkinter gui. Filemenu options can be used to start a chatroom server or a client

Parameters **`**kwargs`** (*`kwargs`*) – ***kwargs.*

Attr type launch_gui Function module to start gui

Text_insert()
Method to show messages in the group2 window. It looks for messages in a message queue. If any message is found, message is shown and deleted from the queue. Message queue is populated by receive methods of chat.server and chat.client

get_id()
Method to get thread id of the chat thread.

launch_gui(***kwargs*)
Method to design the GUI widgets and launch the window The gui window is divided in three parts. 1. Menubar 2. group1 for textbox to type messages to be sent 3. group2 for textbox to show show received messages and internal processings

new_window()
Method to create a new window of the gui. Not implemented yet in this version of the package. It will be implemented in future version

raise_exception()
Method that raises an exception when GUI window is forcefully closed. This forces the gui to close directly without causing any daemon or non daemon threads to cause exception while closing the gui

start_client()
Method to start a chat.client thread that will connect to the socket server. This method is called when connect to chatroom option in filemenu is clicked.

start_server()
Method to start a chat.server thread that will start the socket server. This method is called when Start Server option in filemenu is clicked.

submit(*event=None*)
Method to handle sending of message events. events are mouse click on send button or pressing of Enter key (<return>)

Parameters **event** – Button click or <return> event

1.1.7 mysocks.utilities module

mysocks.utilities.**mapcount**(*filename*)

1.1.8 Module contents

sockets module. Provides a library to create socket server and clients to chat or transfer files

class mysocks.**Model**
Bases: object

Base Model Class for mysocks package

host [string] IP Address of the socket that the server will bind to. Defaults to localhost

port [int] port of the socket server. Defaults to 5560

accept_connections()

create_client_socket(*host, port*)
Function to create a socket for a server that can handle *n_listen* number of clients at a time

Args: host (string): IP address of the socket to be created on in decimal notation or in hexadecimal notation port (integer): Port number of the server that will be open for the clients

host [string] IP Address of the socket that the server will bind to. Defaults to localhost

port [int] port of the socket server. Defaults to 5560

s [socket object] socket object

`create_server_socket`(*host*, *port*, *n_listen*)
Function to create a socket for a server that can handle *n_listen* number of clients at a time

Args: host (string): IP address of the socket to be created on in decimal notation or in hexadecimal notation port (integer): Port number of the server that will be open for the clients n_listen (integer): Maximum number of clients that the server can handle at a time

host [string] IP Address of the socket that the server will bind to. Defaults to localhost

port [int] port of the socket server. Defaults to 5560

n_listen [int] Max. number of clients to connect to at one time

s [socket object] socket object

`file_transfer`(*src*, *dest*)

`host = ''`

`port = 5560`

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Python Module Index

Index