**Python Remote Module**

Une image contenant texte

Description générée automatiquement

Table des matières

[1. Create\_server\_socket 2](#_Toc97823064)

[2. Create\_client\_socket 3](#_Toc97823065)

[3. Wait\_for\_connection 4](#_Toc97823066)

[4. Create\_connection 5](#_Toc97823067)

[5. Bind\_referee 6](#_Toc97823068)

[6. Close\_connection 7](#_Toc97823069)

[7. Notify\_remote\_orders 8](#_Toc97823070)

[8. Get\_remote\_orders 9](#_Toc97823071)

# Create\_server\_socket

*def* create\_server\_socket(*local\_port*, *verbose*):

    """Creates a server socket.

    Parameters

    ----------

    local\_port: port to listen to (int)

    verbose: True if verbose (bool)

    Returns

    -------

    socket\_in: server socket (socket. socket)

    """

# Create\_client\_socket

*def* create\_client\_socket(*remote\_IP*, *remote\_port*, *verbose*):

    """Creates a client socket.

    Parameters

    ----------

    remote\_IP: IP address to send to (int)

    remote\_port: port to send to (int)

    verbose: True if verbose (bool)

    Returns

    -------

    socket\_out: client socket (socket.socket)

    """

# Wait\_for\_connection

*def* wait\_for\_connection(*socket\_in*, *verbose*):

    """Waits for a connection on a server socket.

    Parameters

    ----------

    socket\_in: server socket (socket.socket)

    verbose: True if verbose (bool)

    Returns

    -------

    socket\_in: accepted connection (socket.socket)

    """

# Create\_connection

*def* create\_connection(*your\_group*, *other\_group*=0, *other\_IP*='127.0.0.1', *verbose*=False):

    """Creates a connection with a referee or another group.

    Parameters

    ----------

    your\_group: id of your group (int)

    other\_group: id of the other group, if there is no referee (int, optional)

    other\_IP: IP address where the referee or the other group is (str, optional)

    verbose: True only if connection progress must be displayed (bool, optional)

    Returns

    -------

    connection: socket(s) to receive/send orders (dict of socket.socket)

    Raises

    ------

    IOError: if your group fails to create a connection

    Notes

    -----

    Creating a connection can take a few seconds (it must be initialized on both sides).

    If there is a referee, leave other\_group=0, otherwise other\_IP is the id of the other group.

    If the referee or the other group is on the same computer than you, leave other\_IP='127.0.0.1',

    otherwise other\_IP is the IP address of the computer where the referee or the other group is.

    The returned connection can be used directly with other functions in this module.

    """

# Bind\_referee

*def* bind\_referee(*group\_1*, *group\_2*, *verbose*=False):

    """Put a referee between two groups.

    Parameters

    ----------

    group\_1: id of the first group (int)

    group\_2: id of the second group (int)

    verbose: True only if connection progress must be displayed (bool, optional)

    Returns

    -------

    connections: sockets to receive/send orders from both players (dict)

    Raises

    ------

    IOError: if the referee fails to create a connection

    Notes

    -----

    Putting the referee in place can take a few seconds (it must be connect to both groups).

    connections contains two connections (dict of socket.socket) which can be used directly

    with other functions in this module.  connection of first (second) player has key 1 (2).

    """

# Close\_connection

*def* close\_connection(*connection*):

    """Closes a connection with a referee or another group.

    Parameters

    ----------

    connection: socket(s) to receive/send orders (dict of socket.socket)

    """

# Notify\_remote\_orders

*def* notify\_remote\_orders(*connection*, *orders*):

    """

    Notifies orders to a remote player.

    Parameters

    ----------

    connection: sockets to receive/send orders (dict of socket.socket)

    orders: orders to notify (str)

    Raises

    ------

    IOError: if remote player cannot be reached

    """

# Get\_remote\_orders

*def* get\_remote\_orders(*connection*):

    """Returns orders from a remote player.

    Parameters

    ----------

    connection: sockets to receive/send orders (dict of socket.socket)

    Returns

    ----------

    player\_orders: orders given by remote player (str)

    Raises

    ------

    IOError: if remote player cannot be reached

    """