Manual

FlexiVision

UNIVERSAL ROBOTS PLUG-IN

ars | Feeding Industrial Robotics

INDEX

This Plugin was developed with the idea of communicating **quickly and safely with FlexiVision 2.0** through **UR robots** by using an intermediate server written in **Python**.

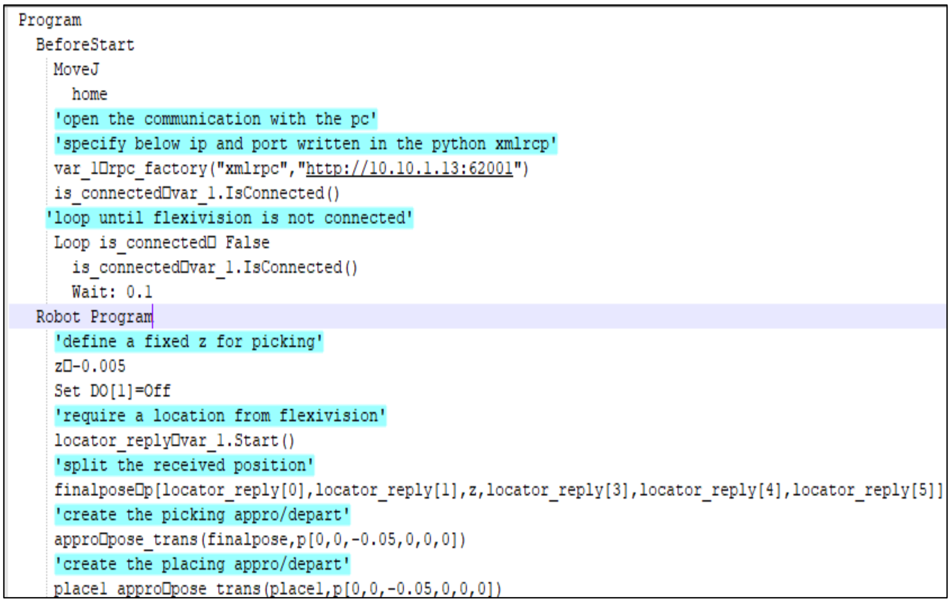
The Plugin does not require any additional license.

FlexiBowl® Plug-In

UNIVERSAL ROBOTS

Plug-In Installation

Together with this guide, a basic example developed directly with URScript is provided in order to understand the steps to implement the application.

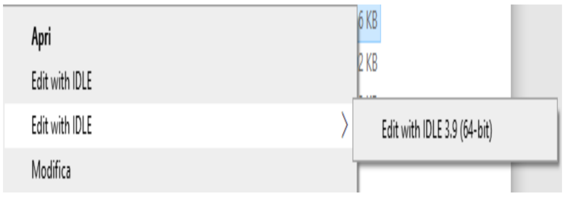


```
Program
  BeforeStart
    MoveJ
      home
    'open the communication with the pc'
    'specify below ip and port written in the python xmlrcp'
    var_1□rpc_factory("xmlrpc","http://10.10.1.13:62001")
    is_connected□var_1.IsConnected()
   'loop until flexivision is not connected'
    Loop is_connected□ False
      is_connected□var_1.IsConnected()
      Wait: 0.1
  Robot Program
    'define a fixed z for picking'
    z□-0.005
    Set DO[1]=Off
    'require a location from flexivision'
    locator_reply□var_1.Start()
    'split the received position'
    finalpose□p[locator_reply[0],locator_reply[1],z,locator_reply[3],locator_reply[4],locator_reply[5]]
    'create the picking appro/depart'
    appro□pose_trans(finalpose,p[0,0,-0.05,0,0,0])
    'create the placing appro/depart'
    placel_appro□pose_trans(placel,p[0,0,-0.05,0,0,0])
```

Step 1.

Copy the Plug-in provided by ARS to your PC desktop, right click and then "***edit with idle***".

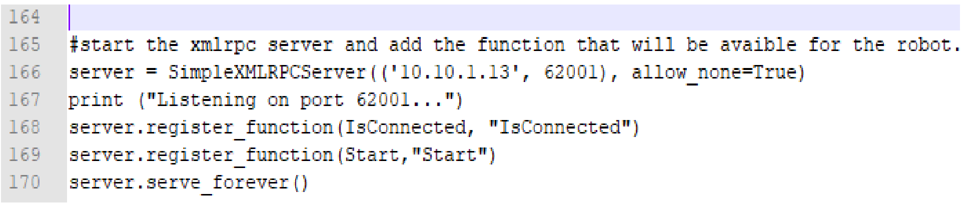
Note: If it is not available on your PC, install Python version 3.9



Plug-In Installation

Step 2.

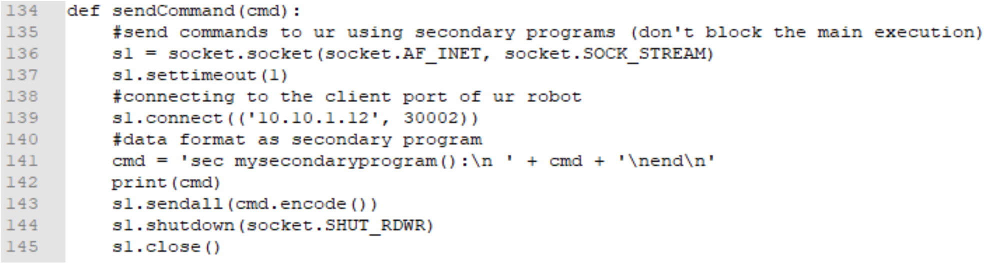
We will have to use an ***XMLRCP server*** as a link between FlexiVision and our UnivarsalRobot as well as specifying the PC IP and the desired port.



```

#start the xmlrpc server and add the function that will be avaible for the robot.
server = SimpleXMLRPCServer(('10.10.1.13', 62001), allow_none=True)
print ("Listening on port 62001...")
server.register_function(IsConnected, "IsConnected")
server.register_function(Start,"Start")
server.serve_forever()
```

Step 3.

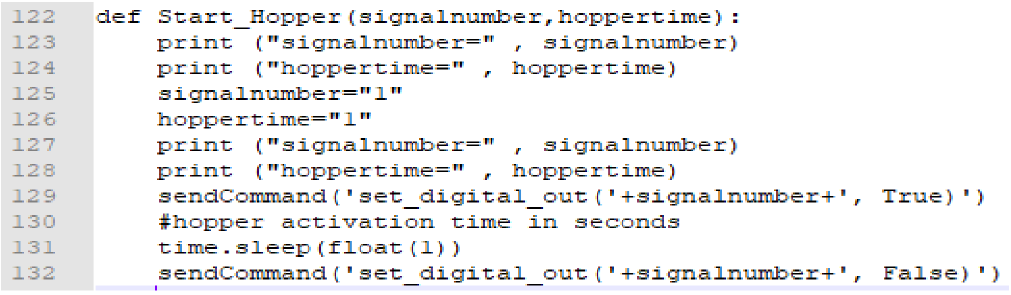
Specify the robot's IP address within the "***sendCommand***" function.



```
def sendCommand(cmd):
    #send commands to ur using secondary programs (don't block the main execution)
    s1 = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s1.settimeout(1)
    #connecting to the client port of ur robot
    s1.connect(('10.10.1.12', 30002))
    #data format as secondary program
    cmd = 'sec mysecondaryprogram():\n ' + cmd + '\nend\n'
    print(cmd)
    s1.sendall(cmd.encode())
    s1.shutdown(socket.SHUT_RDWR)
    s1.close()
```

Step 4.

Specify the number of the signal associated with the hopper and the activation time (in seconds) within the "***Start_Hopper***" function.

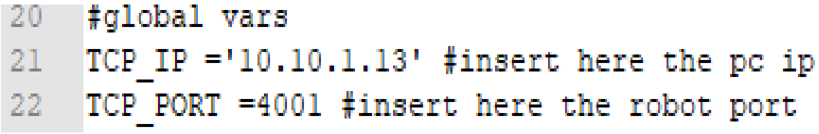


```
def Start_Hopper(signalnumber,hoppertime):
    print ("signalnumber=" , signalnumber)
    print ("hoppertime=" , hoppertime)
    signalnumber="1"
    hoppertime="1"
    print ("signalnumber=" , signalnumber)
    print ("hoppertime=" , hoppertime)
    sendCommand('set_digital_out('+signalnumber+', True)')
    #hopper activation time in seconds
    time.sleep(float(1))
    sendCommand('set_digital_out('+signalnumber+', False)')
```

Plug-In Installation

Step 5.

Specify the PC IP address and the port through which we want to listen. They must match those specified on FlexiVision on the “***Robot***” page.

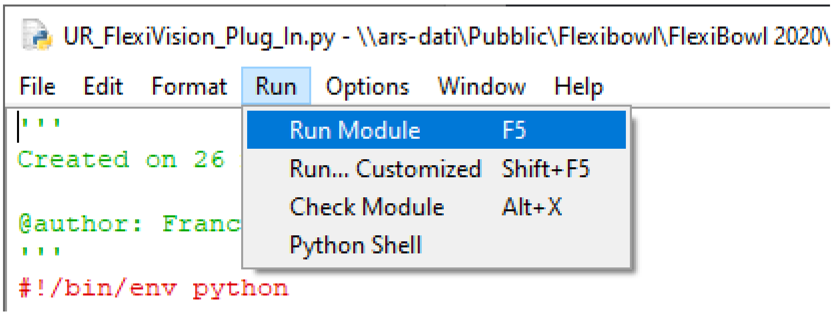


```
#global vars
TCP_IP ='10.10.1.13' #insert here the pc ip
TCP_PORT =4001 #insert here the robot port
```

Step 6.

Start the python server.

Next, connect FlexiVision and start the robot program.

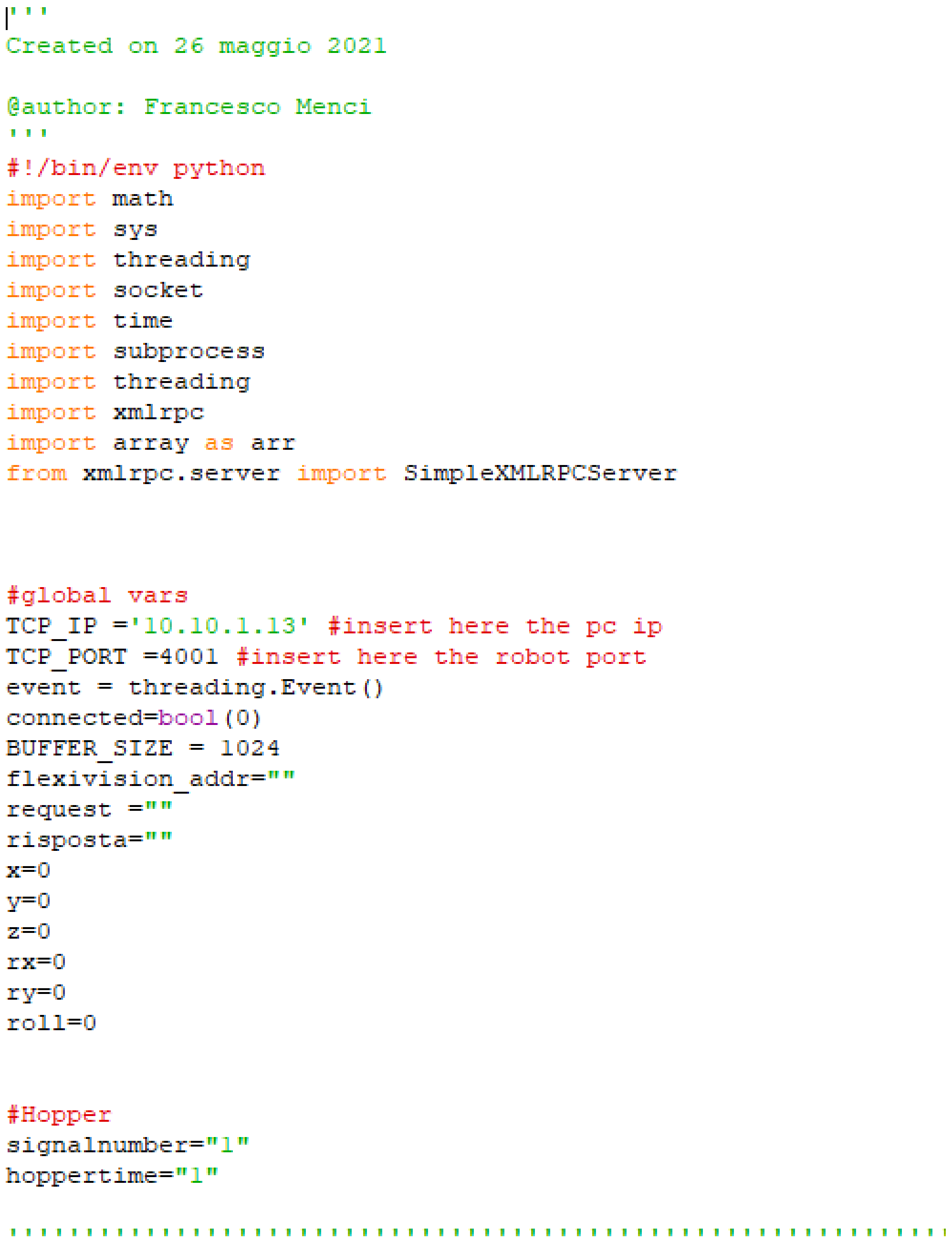


UR_FlexiVision_Plug_In.py - \\ars-dati\Pubblic\Flexibowl\FlexiBowl 2020\

```
'''
Created on 26

@author: Franc
'''
#!/bin/env python
```

Script

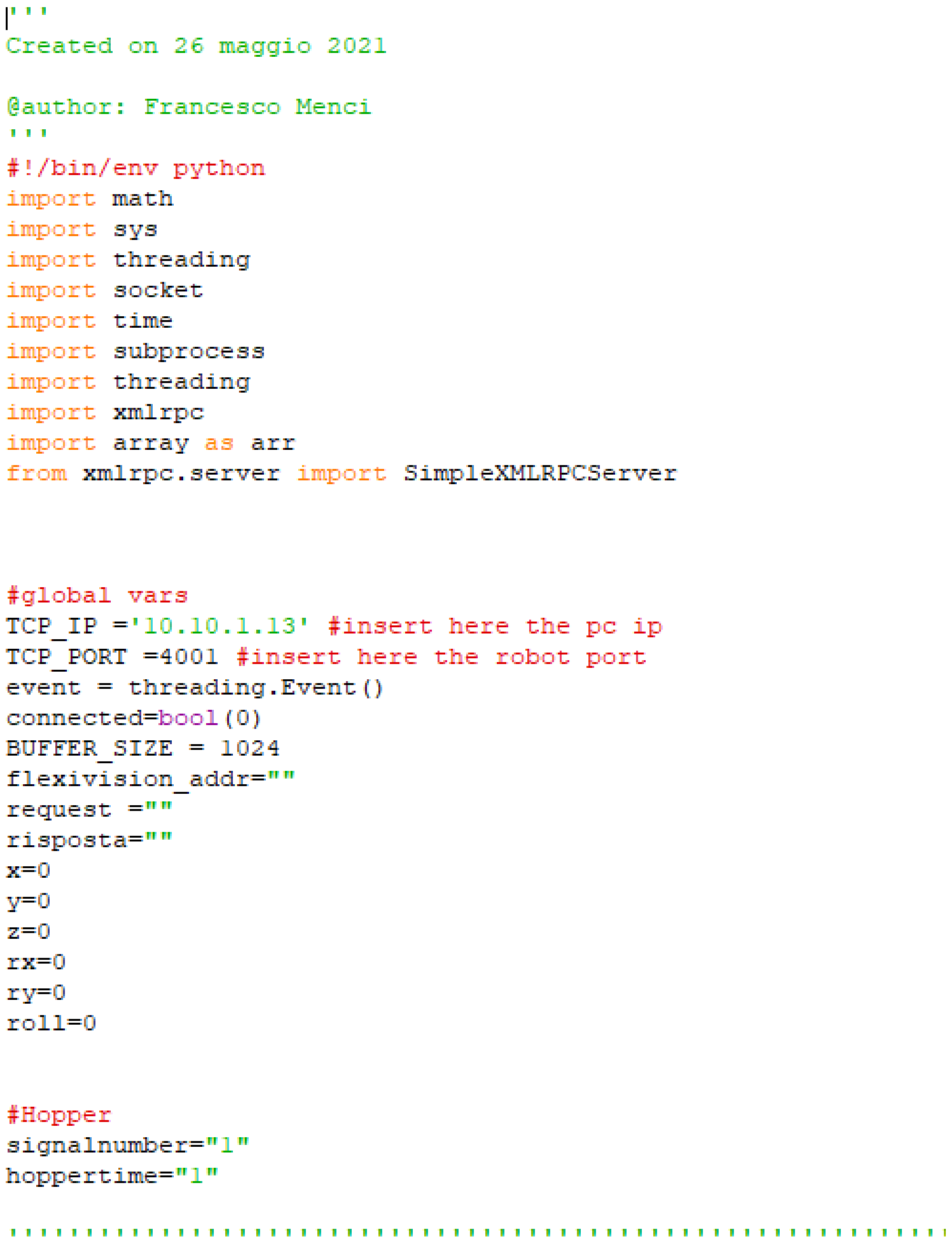
Let us analyse the main script



```
'''
Created on 26 maggio 2021

@author: Francesco Menci
'''
#!/bin/env python
import math
import sys
import threading
import socket
import time
import subprocess
import threading
import xmlrpc
import array as arr
from xmlrpc.server import SimpleXMLRPCServer



#global vars
TCP_IP ='10.10.1.13' #insert here the pc ip
TCP_PORT =4001 #insert here the robot port
event = threading.Event()
connected=bool(0)
BUFFER_SIZE = 1024
flexivision_addr=""
request =""
risposta=""
x=0
y=0
z=0
rx=0
ry=0
roll=0


#Hopper
signalnumber="1"
hoppertime="1"

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
```

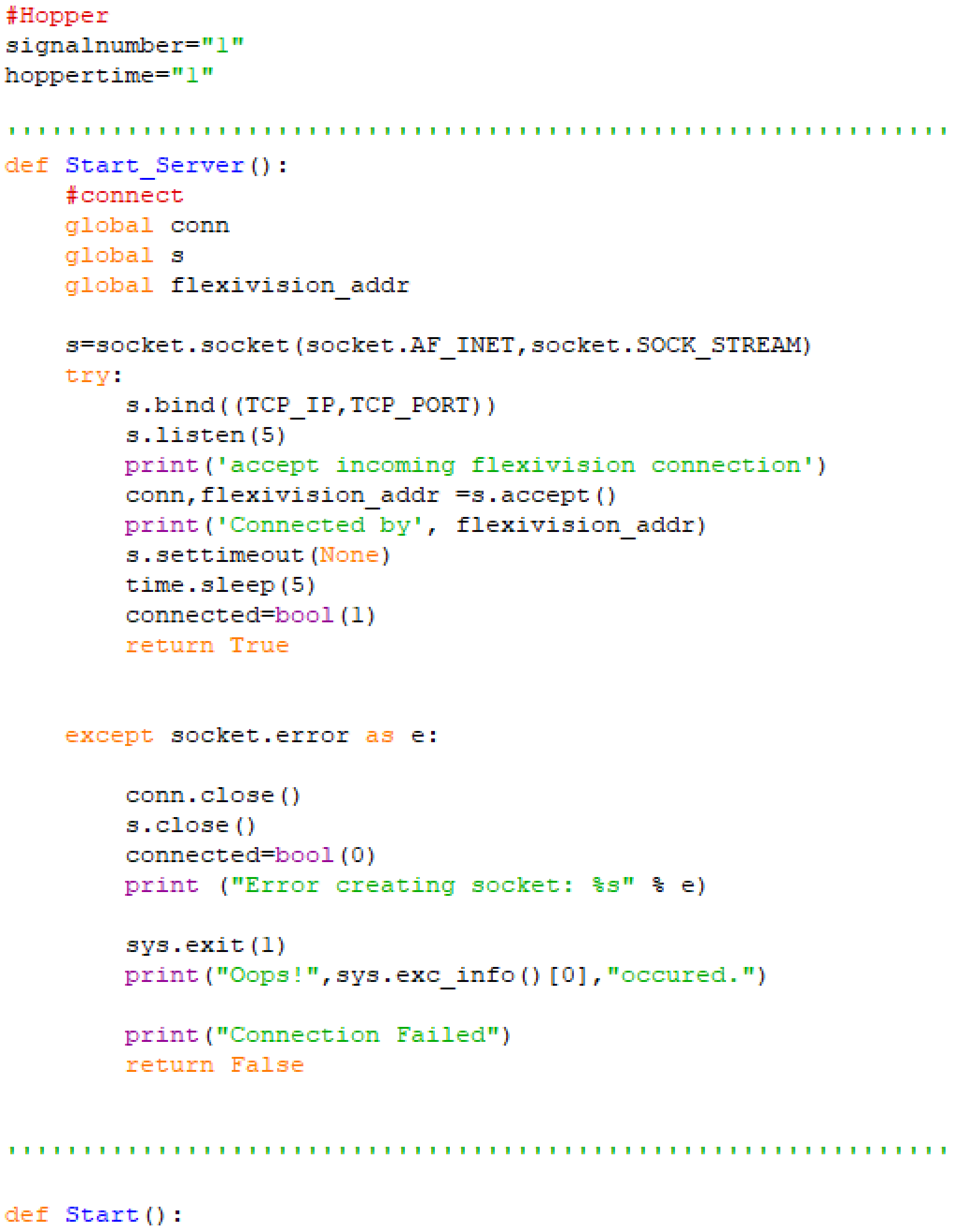
Script

Let us analyse the main script



```
'''
Created on 26 maggio 2021

@author: Francesco Menci
'''
#!/bin/env python
import math
import sys
import threading
import socket
import time
import subprocess
import threading
import xmlrpc
import array as arr
from xmlrpc.server import SimpleXMLRPCServer



#global vars
TCP_IP ='10.10.1.13' #insert here the pc ip
TCP_PORT =4001 #insert here the robot port
event = threading.Event()
connected=bool(0)
BUFFER_SIZE = 1024
flexivision_addr=""
request =""
risposta=""
x=0
y=0
z=0
rx=0
ry=0
roll=0


#Hopper
signalnumber="1"
hoppertime="1"

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
```

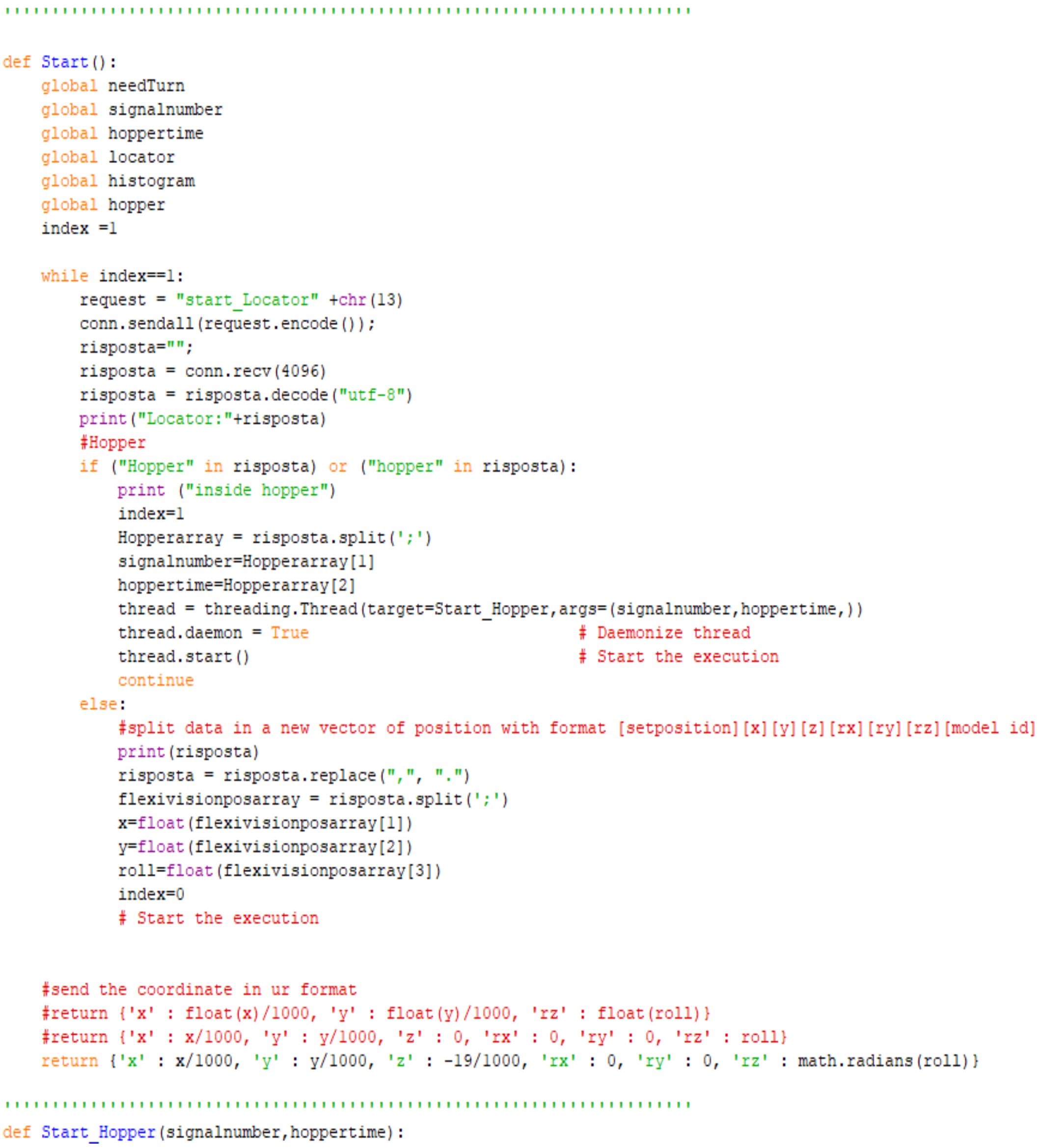
Script



```
#Hopper
signalnumber="1"
hoppertime="1"

'''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
def Start_Server():
    #connect
    global conn
    global s
    global flexivision_addr

    s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
    try:
        s.bind((TCP_IP,TCP_PORT))
        s.listen(5)
        print('accept incoming flexivision connection')
        conn,flexivision_addr =s.accept()
        print('Connected by', flexivision_addr)
        s.settimeout(None)
        time.sleep(5)
        connected=bool(1)
        return True


    except socket.error as e:

        conn.close()
        s.close()
        connected=bool(0)
        print ("Error creating socket: %s" % e)

        sys.exit(1)
        print("Oops!",sys.exc_info()[0],"occured.")

        print("Connection Failed")
        return False


'''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

def Start():
```

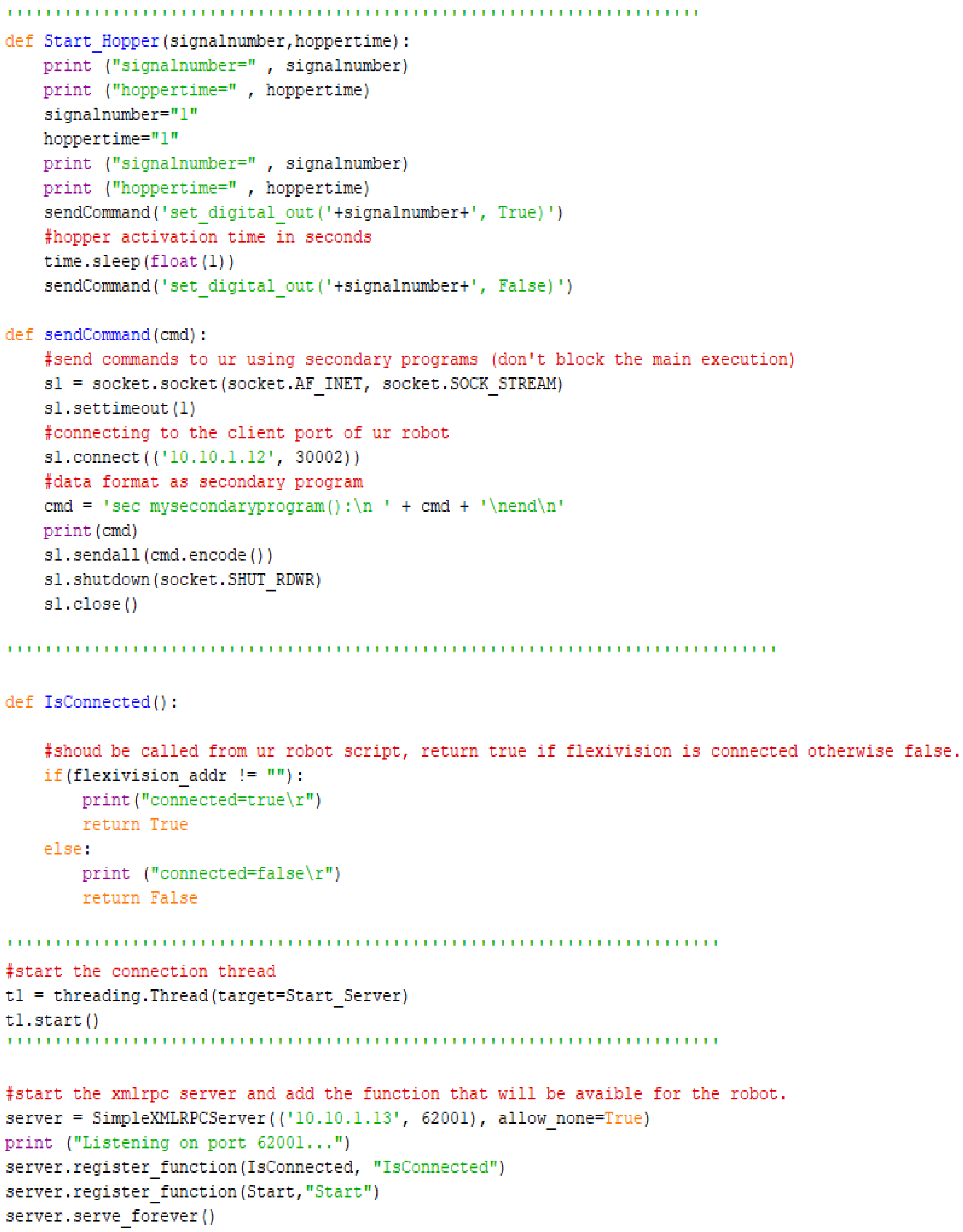
Script

TCP_SERVER



```
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

def Start():
    global needTurn
    global signalnumber
    global hoppertime
    global locator
    global histogram
    global hopper
    index =1

    while index==1:
        request = "start_Locator" +chr(13)
        conn.sendall(request.encode());
        risposta="";
        risposta = conn.recv(4096)
        risposta = risposta.decode("utf-8")
        print("Locator:"+risposta)
        #Hopper
        if ("Hopper" in risposta) or ("hopper" in risposta):
            print ("inside hopper")
            index=1
            Hopperarray = risposta.split(';')
            signalnumber=Hopperarray[1]
            hoppertime=Hopperarray[2]
            thread = threading.Thread(target=Start_Hopper,args=(signalnumber,hoppertime,))
            thread.daemon = True                            # Daemonize thread
            thread.start()                                  # Start the execution
            continue
        else:
            #split data in a new vector of position with format [setposition][x][y][z][rx][ry][rz][model id]
            print(risposta)
            risposta = risposta.replace(",", ".")
            flexivisionposarray = risposta.split(';')
            x=float(flexivisionposarray[1])
            y=float(flexivisionposarray[2])
            roll=float(flexivisionposarray[3])
            index=0
            # Start the execution


    #send the coordinate in ur format
    #return {'x' : float(x)/1000, 'y' : float(y)/1000, 'rz' : float(roll)}
    #return {'x' : x/1000, 'y' : y/1000, 'z' : 0, 'rx' : 0, 'ry' : 0, 'rz' : roll}
    return {'x' : x/1000, 'y' : y/1000, 'z' : -19/1000, 'rx' : 0, 'ry' : 0, 'rz' : math.radians(roll)}

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
def Start_Hopper(signalnumber,hoppertime):
```

Script



```
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

def Start_Hopper(signalnumber,hoppertime):
    print ("signalnumber=" , signalnumber)
    print ("hoppertime=" , hoppertime)
    signalnumber="1"
    hoppertime="1"
    print ("signalnumber=" , signalnumber)
    print ("hoppertime=" , hoppertime)
    sendCommand('set_digital_out('+signalnumber+', True)')
    #hopper activation time in seconds
    time.sleep(float(1))
    sendCommand('set_digital_out('+signalnumber+', False)')

def sendCommand(cmd):
    #send commands to ur using secondary programs (don't block the main execution)
    s1 = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s1.settimeout(1)
    #connecting to the client port of ur robot
    s1.connect(('10.10.1.12', 30002))
    #data format as secondary program
    cmd = 'sec mysecondaryprogram():\n ' + cmd + '\nend\n'
    print(cmd)
    s1.sendall(cmd.encode())
    s1.shutdown(socket.SHUT_RDWR)
    s1.close()

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''


def IsConnected():

    #shoud be called from ur robot script, return true if flexivision is connected otherwise false.
    if(flexivision_addr != ""):
        print("connected=true\r")
        return True
    else:
        print ("connected=false\r")
        return False

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
#start the connection thread
t1 = threading.Thread(target=Start_Server)
t1.start()
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

#start the xmlrpc server and add the function that will be avaible for the robot.
server = SimpleXMLRPCServer(('10.10.1.13', 62001), allow_none=True)
print ("Listening on port 62001...")
server.register_function(IsConnected, "IsConnected")
server.register_function(Start,"Start")
server.serve_forever()
```

FlexiVision Command List

To send the command to FlexiVision you must modify the value of the “command” string.

N_Mission	Command	Action
1	“start_Locator"	Starts the parts localization process by recalling the FlexiBowl handling routine in case there are no parts that can be picked up. ***Return:*** “Pattern1;x;y;r”.
2	“stop_Locator"	Stops the process of locating the object with the aid of the FlexiBowl.
3	“turn_Locator"	If no parts are picked up, by this command the operator can make the FlexiBowl rotate and the “start_Locator” routine start. ***Return:*** “Pattern1;x;y;r”.
4	“test_Locator"	Starts the process of locating the object without the aid of the FlexiBowl. ***Return:*** “Pattern1;x;y;r”.
5	“start_Control"	Starts the inspection cycle. ***Return:*** “Control1;x;y;r”.
6	“state_Locator"	Locator status diagnostics is shown: ***Return:*** “Locator is Running” “Locator is in Error” “Locator is not Running”.
7	“start_Empty"	Start the FlexiBowl® Quick-Emptying sequence. ***Return:*** “start_Empty ended”
8	“get_Recipe"	The name of the recipe currently loaded on FlexiVision is shown. ***Return:*** “recipe name”.
9	“set_Recipe=recipe name"	The recipe corresponding to the sent "recipe name" is loaded.