

Reading sending data and automatic simulations from-to Matlab/Python

RTDS training course

2018

Intelligent Electrical Power Grids

Electric Sustainable Energy

TU Delft

Outlines

- Introduction
- Connecting with a MATLAB M file
- Connecting with Python 3

Introduction

From time to time, some RTDS users want is to be able to **send out small number of commands to the simulation from an external application such as MATLAB™ as well as retrieve the corresponding simulation results.** The incoming results from RTDS can be utilized in the external application. Usually the necessary interaction speed in such circumstance is not too high, i.e., in the range of hundreds of milliseconds.

- A script function in RSCAD/Runtime.
 - 'ListenOnPort()' command
- a TCP/IP socket communication: server + client

Figure 1
'ListenOnPort()' command
RSCAD/Runtime
Help

The screenshot shows the RTDS Runtime Help window. The left pane displays a tree view of the help content, with 'ListenOnPort Function' selected under 'Miscellaneous Functions'. The right pane shows the documentation for the 'ListenOnPort Script Command'. The title bar of the window reads 'RTDS Runtime Technologies Power Systems Simulation Software'. The main content area contains the following text:

ListenOnPort Script Command

The ListenOnPort script command provides a way for an external process to control RSCAD by sending regular script commands over a tcp/ip connection. There are two forms of this command, the first form accepts one integer parameter which specifies the port number to [listen](#) on.

Syntax:

```
ListenOnPort(int portNumber);
```

The second form of this command accepts two parameters, the integer port number and a second integer value which indicates if errors are to be passed back through the tcp/ip connection. This second parameter is essentially a boolean value, where zero indicates false and non-zero indicates true. The pre-defined strings "true" and "false" are accepted.

Syntax:

```
ListenOnPort(int portNumber, int passErrorsBack);
```

```
ListenOnPort(4567,true); //will pass the errors back through the socket connection
```

When executed, a server socket is opened on this port and the script waits for an external process to connect to it. When a connection is established, script commands are read from this tcp connection until the connection is closed. One script command at a time is read and executed, and then the script blocks waiting for more input. This behavior is unlike the processing of regular file scripts, which are completely parsed and processed prior to the script execution. Because the tcp commands are executed as soon as they are received, it significantly complicates the processing of commands that affect the flow of control (such as while loops, for loops and if statements). Because of these complications, commands that affect flow of control are not executed until the entire statement has been received by the RSCAD [listener](#). For example, if the script commands below were received on a tcp port:

```
if (test1 == 5)
{
printf("about to run test 5\n");
}
else if (test1 == 7)
{
printf("about to run test 7\n");
}
else
{
printf("No valid test number received!\n");
}
```

RSCAD would wait until the complete "if" statement had been received (which includes all of these statements) before starting execution. This is important to keep in mind because RSCAD will block waiting for the completion of this IF statement, and this blocking behavior needs to be anticipated.

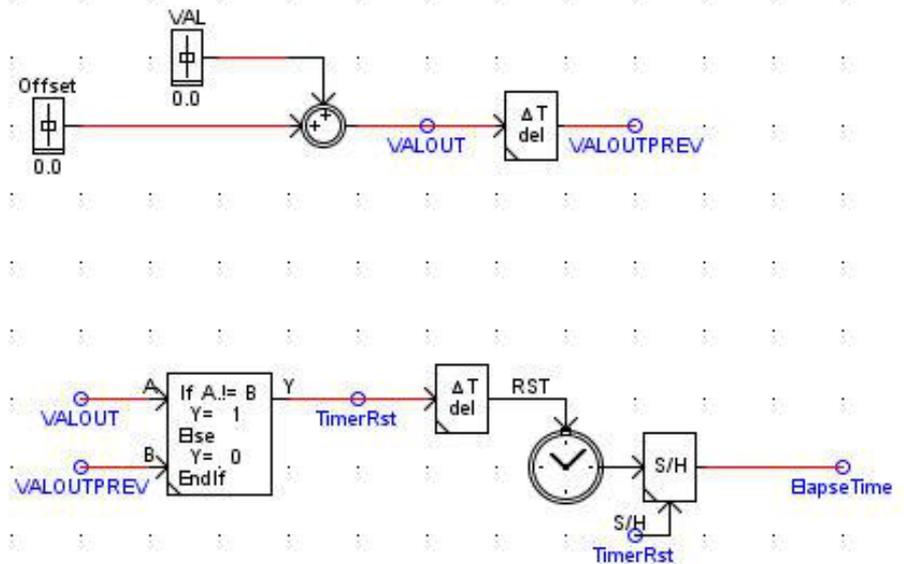
Here is a simple example script, along with the code for a Java process which will send commands to RSCAD:

Script code:

Built the connection with a MATLAB M file

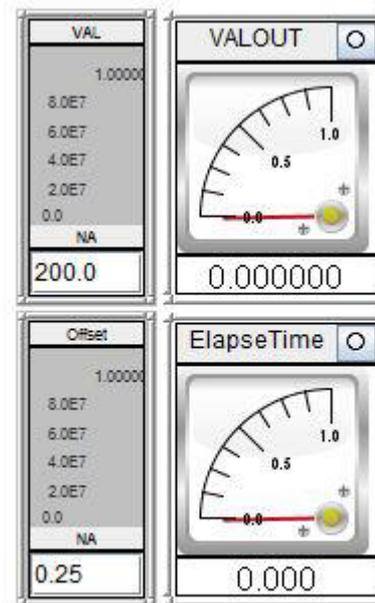
- The MATLAB M file is dependent on an external library regarding the TCP communication capability.
 - The 'JTCP.m' file is the library.
(<http://www.mathworks.com/matlabcentral/fileexchange/24524-tcpip-communications-in-matlab>).

RSCAD Draft Circuit



Built the connection with a MATLAB M file

RSCAD Runtime



- The 'VAL' slider would be used as input to the simulation case
- The external application which connects to the RSCAD/Runtime would control the slider
- The external application would use the meter value as input.
- The second meter ('ElapseTime') displays the time between each change made to the 'VAL' slider.

Built the connection with a MATLAB M file

- *RSCAD Runtime script*

```
1 float temp_float;  
2 string temp_string;  
3  
4 fprintf(stdmsg, "Initialization of RTDS simulation");  
5  
6 ListenOnPort(4575, true);  
7  
8 fprintf(stdmsg, "Execution of script is done\n");
```

The most important command in this script file is at the line number 6. It is the 'ListenOnPort()' command with the port number passed as a parameter. Regarding the port number, note that many standard or well-established network applications have already reserved many port numbers, which are known as 'Well-Known Port Numbers' (http://www.tcpipguide.com/free/t_CommonTCPIPApplicationsandAssignedWellKnownandRegi-2.htm).

As explained in the introductory section, once this command is executed, the port is open and the RSCAD/Runtime becomes a TCP server.



Built the connection with a MATLAB M file

- *Open the related M file in MATLAB*

Pay attention:

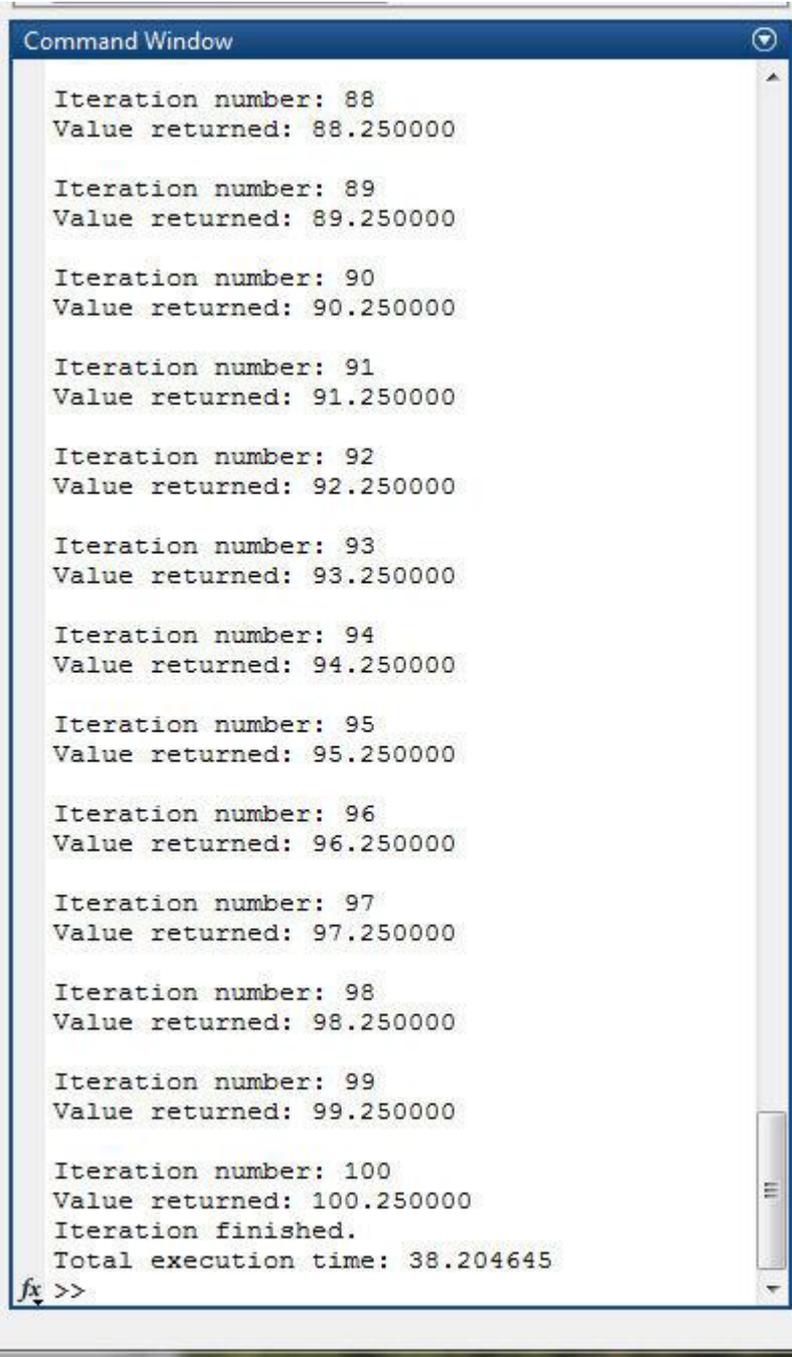
- a. The command at line number 8 declares a variable with port number. This port number should match the port number given in the RSCAD/Runtime script file
- b. The regular expression is utilized as a rigorous measure in order to make sure the transmission is complete and the token string is ready to be parsed at the client
- c. The line number 12 tries to connect to the RSCAD/Runtime with connection request. The second and third parameters of this function call are the IP address and port number of the TCP server, which is RSCAD/Runtime in this demonstration
 1. '127.0.0.1' is used under the assumption that the same computer which runs RSCAD/Runtime also runs this MATLAB M file.
- d. The 'for' loop, the lines between 17 and 58, send out 100 values with an interval of 0.2 seconds.
- e. The output of the RTDS simulation is captured by the two commands at line number 31 and 32
- f. The output of the RTDS simulation is captured by the two commands at line number 31 and 32



How to run the demonstration

1. Compile the RTDS simulation case ('Pingpong.dft') in RSCAD/Draft. Make sure that there is no error or warning. Load the simulation case in RSCAD/Runtime. Don't start the case yet. The case will be started by the external application (MATLAB M file).
2. Load the script file. Go to the 'Script/Open' menu and select the 'ListenOnPort.scr' script file. Start the script by selecting the 'Run Script' button in the script toolbar. Now, the RSCAD/Runtime becomes a TCP server and listens to the port number given by the script command.
 - In MATLAB, go to the directory where the MATLAB M code files ('Pingpong.m' and 'jtcp.m') are stored and run the 'Pingpong.m' M file from the MATLAB command window.
3. Then it can be seen that the RTDS simulation case starts and bi-directional communication, sending out a value from MATLAB to RTDS and receiving the corresponding result from RTDS to MATLAB, can be monitored at the MATLAB command window

The command window:
Communication between RTDS
and MATLAB: Result



```
Command Window

Iteration number: 88
Value returned: 88.250000

Iteration number: 89
Value returned: 89.250000

Iteration number: 90
Value returned: 90.250000

Iteration number: 91
Value returned: 91.250000

Iteration number: 92
Value returned: 92.250000

Iteration number: 93
Value returned: 93.250000

Iteration number: 94
Value returned: 94.250000

Iteration number: 95
Value returned: 95.250000

Iteration number: 96
Value returned: 96.250000

Iteration number: 97
Value returned: 97.250000

Iteration number: 98
Value returned: 98.250000

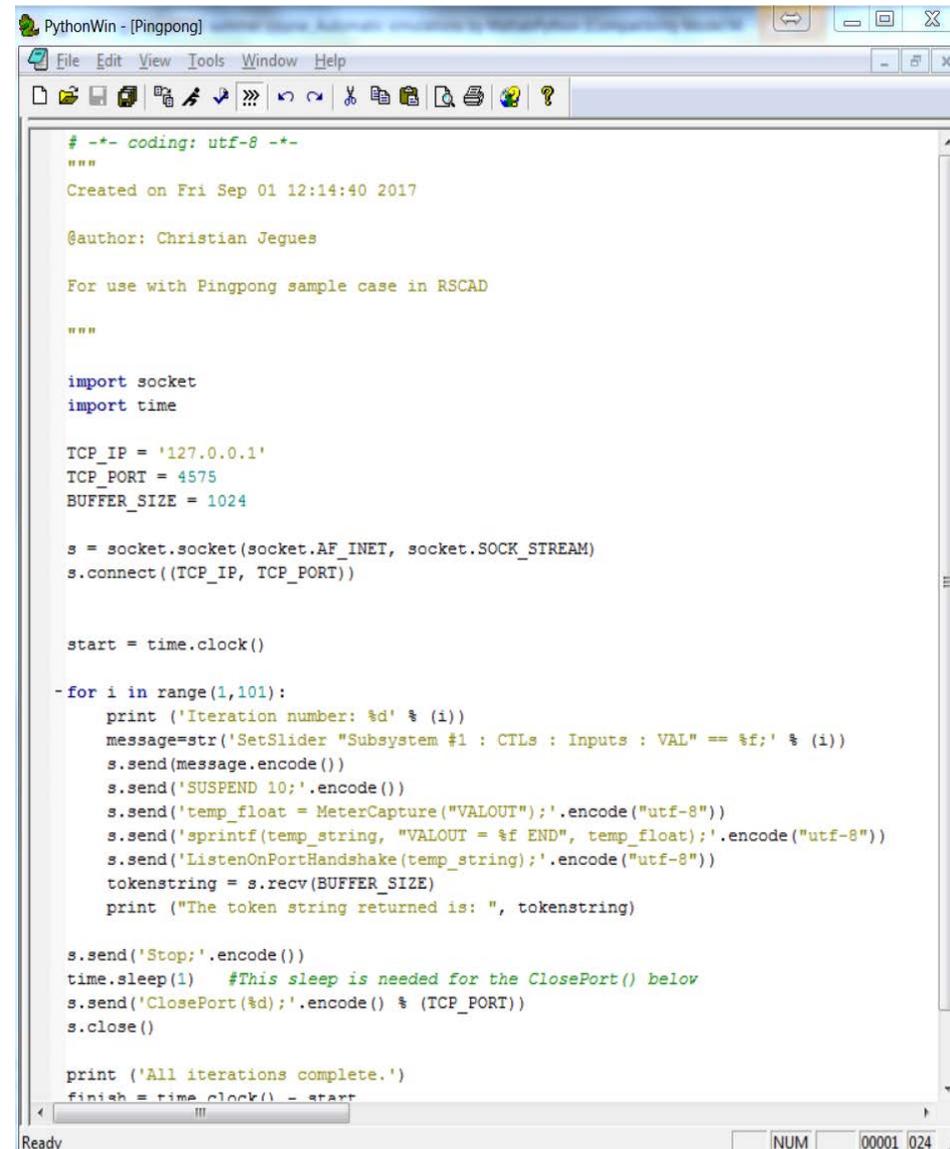
Iteration number: 99
Value returned: 99.250000

Iteration number: 100
Value returned: 100.250000
Iteration finished.
Total execution time: 38.204645

fx >>
```

Connecting with Python

- Any programming environment which supports TCP socket communication can be used when writing the necessary external application
- The programming environment selected for this demonstration is Python 3 programming environment (Pythonwin)
- The procedures of demonstration is as follows:
 1. Find the Pingpong.py in the related folder, and open it.
 2. Compile the RTDS simulation case (Pingpong.dft) in RSCAD/Draft without errors.
 3. Load the simulation case in RSCAD/Runtime and start it to run.

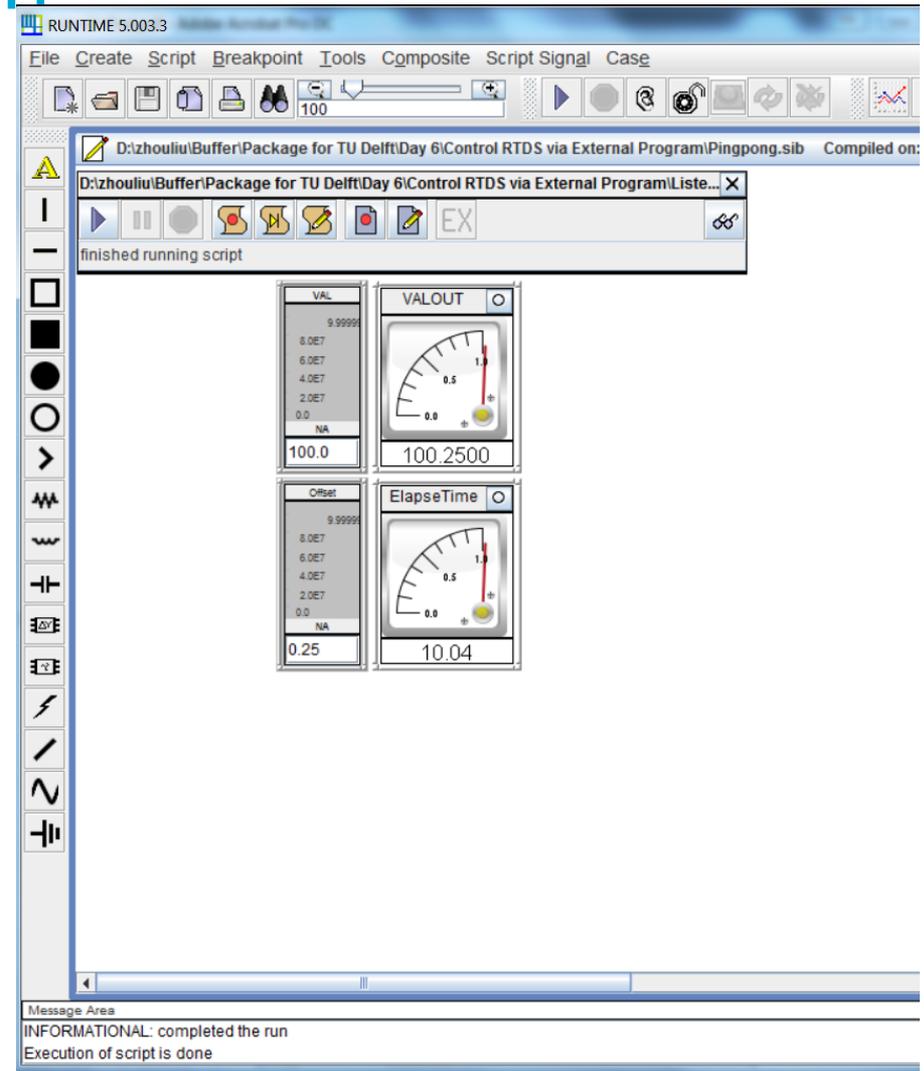


```
PythonWin - [Pingpong]
File Edit View Tools Window Help
Created on Fri Sep 01 12:14:40 2017
@author: Christian Jegues
For use with Pingpong sample case in RSCAD
import socket
import time
TCP_IP = '127.0.0.1'
TCP_PORT = 4575
BUFFER_SIZE = 1024
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((TCP_IP, TCP_PORT))
start = time.clock()
for i in range(1,101):
    print ('Iteration number: %d' % (i))
    message=str('SetSlider "Subsystem #1 : CTLs : Inputs : VAL" == %f;' % (i))
    s.send(message.encode())
    s.send('SUSPEND 10;'.encode())
    s.send('temp_float = MeterCapture("VALOUT");'.encode("utf-8"))
    s.send('sprintf(temp_string, "VALOUT = %f END", temp_float);'.encode("utf-8"))
    s.send('ListenOnPortHandshake(temp_string);'.encode("utf-8"))
    tokenstring = s.recv(BUFFER_SIZE)
    print ("The token string returned is: ", tokenstring)
s.send('Stop;'.encode())
time.sleep(1) #This sleep is needed for the ClosePort() below
s.send('ClosePort(%d);'.encode() % (TCP_PORT))
s.close()
print ('All iterations complete.')
finish = time.clock() - start
```

Connecting with Python

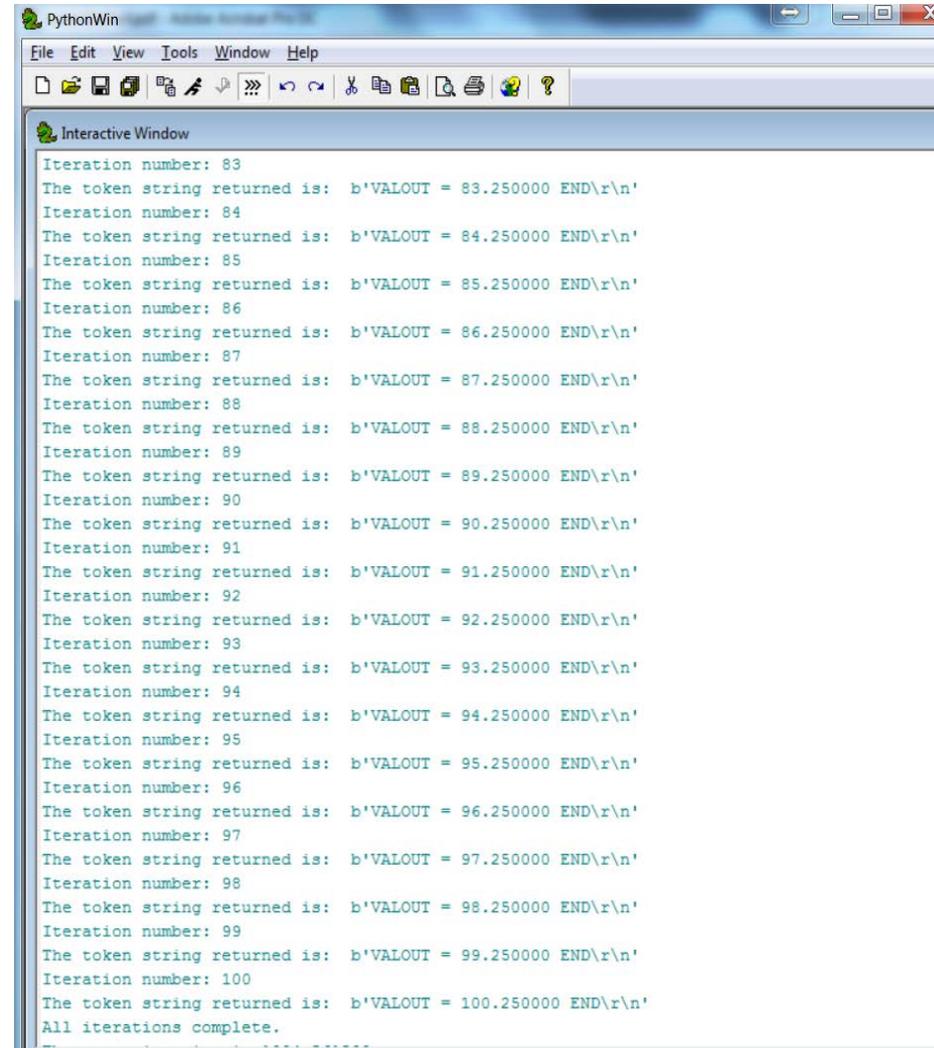
4. Load the script file. Go to the 'Script/Open' menu and select the 'ListenOnPort.scr' script file.
5. Start the script by selecting the 'Run Script' button in the script toolbar

Now, the RSCAD/Runtime becomes a TCP server and listens to the port number given by the script command.



Connecting with Python

6. Then let the program in Pythonwin run by click the related button
7. The slider of variable “VAL” has been changed every 0.2s from 1 to 100.
8. The returned results can be seen from the shell window of related Python programming environment.



The screenshot shows a PythonWin window with an 'Interactive Window' pane. The pane displays the output of a program running 100 iterations. Each iteration prints the iteration number and a token string. The token string is a byte string containing the iteration number, a floating-point value, and the string 'END'. The floating-point value increases by 0.25 in each iteration, starting from 83.250000 and ending at 100.250000. The output ends with 'All iterations complete.'

```
PythonWin
File Edit View Tools Window Help
Interactive Window
Iteration number: 83
The token string returned is: b'VALOUT = 83.250000 END\r\n'
Iteration number: 84
The token string returned is: b'VALOUT = 84.250000 END\r\n'
Iteration number: 85
The token string returned is: b'VALOUT = 85.250000 END\r\n'
Iteration number: 86
The token string returned is: b'VALOUT = 86.250000 END\r\n'
Iteration number: 87
The token string returned is: b'VALOUT = 87.250000 END\r\n'
Iteration number: 88
The token string returned is: b'VALOUT = 88.250000 END\r\n'
Iteration number: 89
The token string returned is: b'VALOUT = 89.250000 END\r\n'
Iteration number: 90
The token string returned is: b'VALOUT = 90.250000 END\r\n'
Iteration number: 91
The token string returned is: b'VALOUT = 91.250000 END\r\n'
Iteration number: 92
The token string returned is: b'VALOUT = 92.250000 END\r\n'
Iteration number: 93
The token string returned is: b'VALOUT = 93.250000 END\r\n'
Iteration number: 94
The token string returned is: b'VALOUT = 94.250000 END\r\n'
Iteration number: 95
The token string returned is: b'VALOUT = 95.250000 END\r\n'
Iteration number: 96
The token string returned is: b'VALOUT = 96.250000 END\r\n'
Iteration number: 97
The token string returned is: b'VALOUT = 97.250000 END\r\n'
Iteration number: 98
The token string returned is: b'VALOUT = 98.250000 END\r\n'
Iteration number: 99
The token string returned is: b'VALOUT = 99.250000 END\r\n'
Iteration number: 100
The token string returned is: b'VALOUT = 100.250000 END\r\n'
All iterations complete.
```

Thank you for your attention!

Questions ?

IEPG, ESE, TU Delft