

How to program instruments using Python

Tao Song

Department of Electrical and Electronic Engineering
Xi'an Jiaotong - Liverpool University

Introduction

Instruments can be programmed by various programming language e.g. VB, C++, C#, Python, MATLAB. Programming languages serve as a “jacket” that contains the commands for you to talk to instruments. The commands are in the format of SCPI (Standard Commands for Programmable Instruments) which is an entire different language for PC – Instruments communications. Please be noted that instruments can only recognize commands using SCPI while it has nothing to do with the chosen programming language. This is the reason why instruments programming could be achieved using various programming languages. Therefore, when we say instruments programming, it is all about how to insert the commands into your program. Different programming languages have different ways to insert the commands. Python is a powerful and understandable programming language that contains a large number of useful packages to solve various problems. This guide is based on Python which integrates the package PyVisa for a clear way to program. This guide will show you the basic instrument programming techniques based on Python using Keysight instruments.

Pre-requisite

Download Python: <https://www.python.org/downloads/>

Download codes and instructions: <https://github.com/CollinsSong/PyVisa>

Download Keysight Instrument Control Bundle:

<https://www.keysight.com/main/software.jsp?cc=CN&lc=chi&ckey=2664810&nid=-33330.977662&id=2664810>

When you download Python from its official website. I recommend you to download Python 3 since all the codes were written for Python 3 (Python 2 is slightly different with Python 3 on syntactic). There are various versions of Python on its official website, you don't really need to download the latest version since it may not be compatible with all the packages (In my case, right now the latest version of Python is 3.7, while I cannot find a satisfied version for package matplotlib to plot graphs).

You are also required to download Keysight Instrument Control Bundle which contains I/O Libraries Suite, Command Expert and BenchVue Platform (we will not use this). To follow the instructions from this guide, Python needs to be downloaded and its environmental variables should be configured at beginning. The environmental variables configuration of Keysight Instrument Control Bundle should also be done (for other brands, you will have the corresponding software suite to include everything). See Appendix A if you have troubles with

the configuration of environmental variables.

Methodology

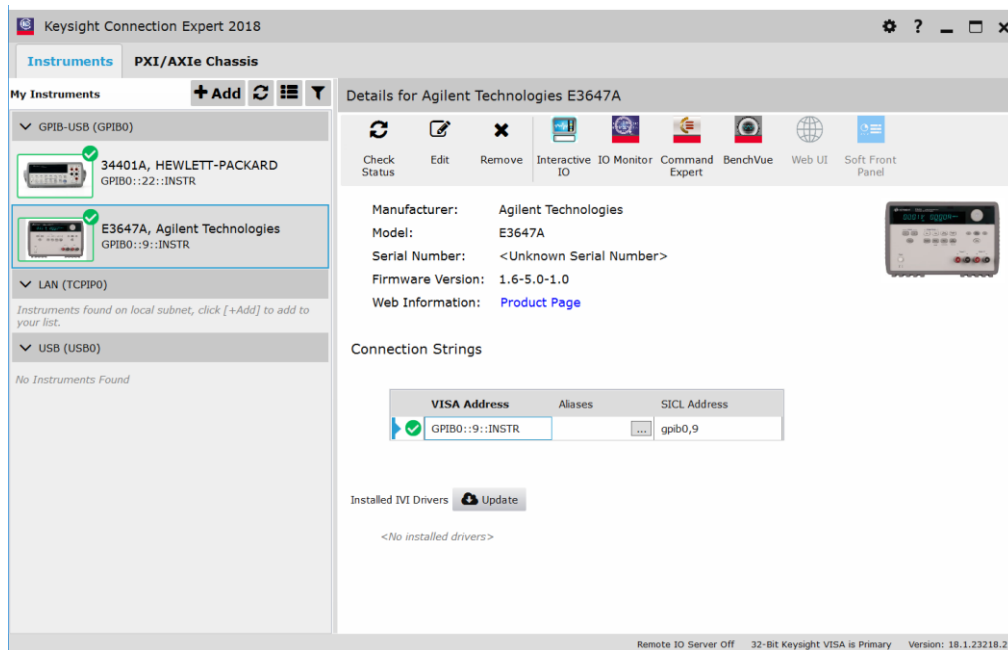
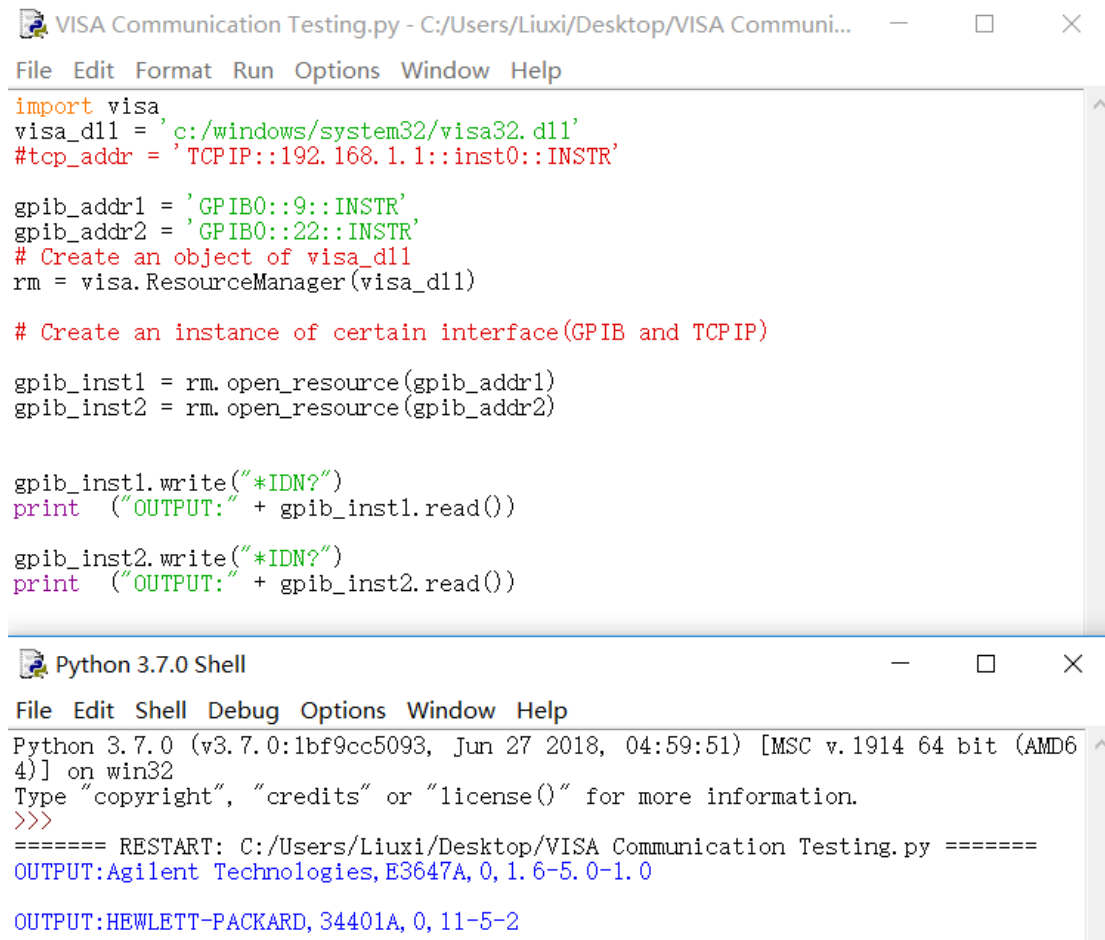


Figure 1 Keysight Connection Expert

There is a software coming together with your downloaded package, namely Keysight Connection Expert. Connect your instruments as described in the BenchVue instruments programming guide.

Once you have successfully installed this package, open Connection Expert. You will have an user interface as in Figure 1, in which you can check your VISA address if your instruments are well connected. Otherwise, try to reconnect your instruments to your PC to see if Connection Expert can detect your instruments. Check your instruments in *My Instruments* to see the connection status. The VISA address of each instrument is shown below the instrument name or shown in the details (for instruments from other brands, you should also be able to find the corresponding VISA address for your instruments).

Find file VISA Communication Testing.py to test the communications of your PC and your instruments. I suppose you cannot run this file at the first time since you need to install the package for Python. As shown in Figure 2, the code is started with `import visa`. This is a basic step to involve PyVisa package into your program.



The image shows two windows from a Windows operating system. The top window is titled 'VISA Communication Testing.py - C:/Users/Liuxi/Desktop/VISA Communi...' and contains a Python script. The script imports the 'visa' module, sets the 'visa_dll' path to 'c:/windows/system32/visa32.dll', and defines two TCP/IP addresses: 'tcp_addr = 'TCPIP::192.168.1.1::inst0::INSTR''. It then creates a 'ResourceManager' object and opens two GPIB resources: 'gpi_b_inst1' at address '9::INSTR' and 'gpi_b_inst2' at address '22::INSTR'. Finally, it sends an '*IDN?' query to both instruments and prints the responses. The bottom window is titled 'Python 3.7.0 Shell' and shows the execution of the script. It displays the Python version (3.7.0), the file path, and the output of the IDN queries: 'Agilent Technologies,E3647A,0,1.6-5.0-1.0' and 'HEWLETT-PACKARD,34401A,0,11-5-2'.

```

import visa
visa_dll = 'c:/windows/system32/visa32.dll'
#tcp_addr = 'TCPIP::192.168.1.1::inst0::INSTR'

gpi_b_addr1 = 'GPIB0::9::INSTR'
gpi_b_addr2 = 'GPIB0::22::INSTR'
# Create an object of visa_dll
rm = visa.ResourceManager(visa_dll)

# Create an instance of certain interface(GPIB and TCP/IP)

gpi_b_inst1 = rm.open_resource(gpi_b_addr1)
gpi_b_inst2 = rm.open_resource(gpi_b_addr2)

gpi_b_inst1.write("*IDN?")
print ("OUTPUT:" + gpi_b_inst1.read())

gpi_b_inst2.write("*IDN?")
print ("OUTPUT:" + gpi_b_inst2.read())

```

```

Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:59:51) [MSC v.1914 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Users/Liuxi/Desktop/VISA Communication Testing.py =====
OUTPUT:Agilent Technologies,E3647A,0,1.6-5.0-1.0

OUTPUT:HEWLETT-PACKARD,34401A,0,11-5-2

```

Figure 2 VISA communication testing

Open your Command window (cmd), once you have successfully set your environmental variables, you are supposed to get the details of pip when you enter “pip” in Command. Problems may happen in Command with pip if you have installed both Python 2 and Python 3, then you need to use pip3 instead of pip. To install PyVisa packages for Python, enter “pip install visa”. Note that the package name in the code when you install them are exactly same as import XX. For instance, when you run a program that has “import XXXX (name of package)”, if you haven’t got the corresponding package installed, the error details will show you “no module named XXXX”, then you need to go to Command to enter “pip install XXXX (name of package)” to install the package. Therefore, once you have successfully installed visa package, using the VISA Communication Testing.py file or using the codes in Figure 2, you will have the information of your connected instruments presented. The codes in Figure 2 are the basic steps of connecting your instruments. The pattern gpi_b_addr should connect to your GPIB address obtained from the connection expert for GPIB communication. While if you are using TCP/IP, you can refer to tcp_addr. If you have already got the expected information of your instruments presented, it means you have a well connection between your PC and your instruments.

```

C:\Users\Liuxi>pip
Usage:
  pip <command> [options]

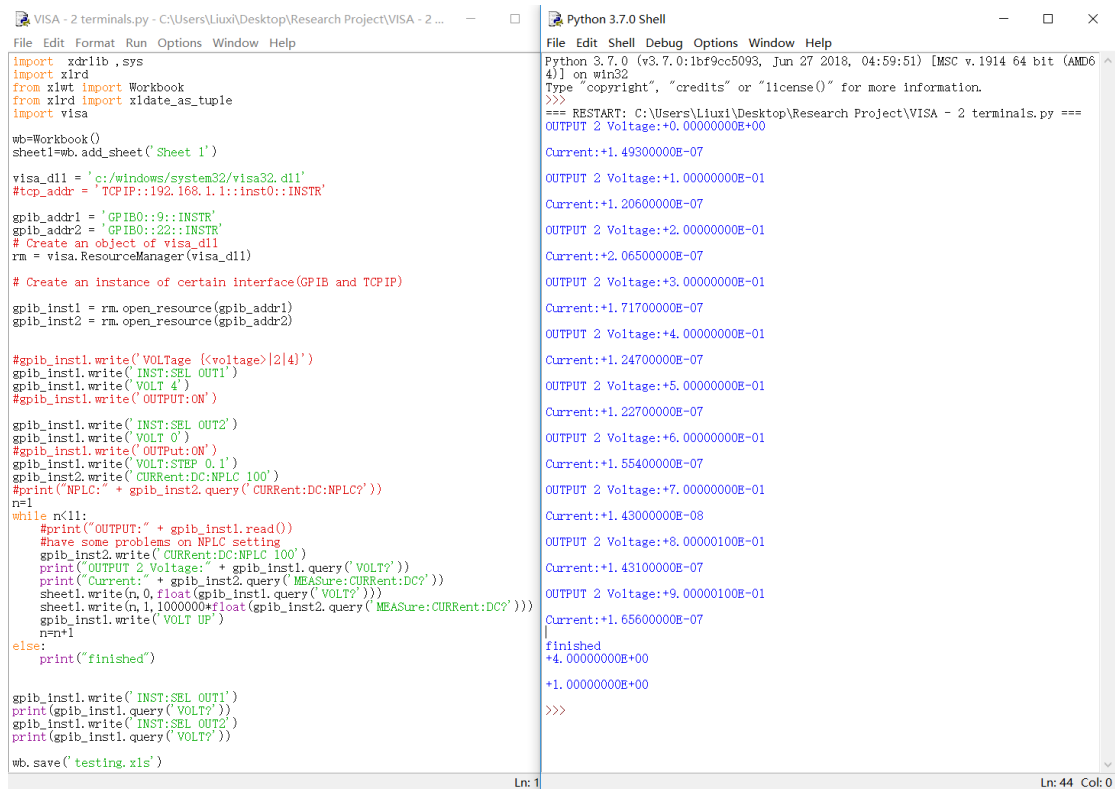
Commands:
  install          Install packages.
  download         Download packages.
  uninstall        Uninstall packages.
  freeze           Output installed packages in requirements format.
  list             List installed packages.
  show             Show information about installed packages.
  check            Verify installed packages have compatible dependencies.
  config           Manage local and global configuration.
  search           Search PyPI for packages.
  wheel            Build wheels from your requirements.
  hash            Compute hashes of package archives.
  completion       A helper command used for command completion.
  help            Show help for commands.

General Options:
  -h, --help            Show help.
  --isolated            Run pip in an isolated mode, ignoring environment variables and user configuration.
  -v, --verbose         Give more output. Option is additive, and can be used up to 3 times.
  -V, --version         Show version and exit.
  -q, --quiet           Give less output. Option is additive, and can be used up to 3 times (corresponding to
                        WARNING, ERROR, and CRITICAL logging levels).
  --log <path>         Path to a verbose appending log.
  --proxy <proxy>       Specify a proxy in the form [user:passwd@]proxy.server:port.
  --retries <retries>   Maximum number of retries each connection should attempt (default 5 times).
  --timeout <sec>       Set the socket timeout (default 15 seconds).
  --exists-action <action> Default action when a path already exists: (s)witch, (i)gnore, (w)ipe, (b)ackup,
                        (a)bort).
  --trusted-host <hostname> Mark this host as trusted, even though it does not have valid or any HTTPS.
  --cert <path>         Path to alternate CA bundle.
  --client-cert <path> Path to SSL client certificate, a single file containing the private key and the
                        certificate in PEM format.
  --cache-dir <dir>     Store the cache data in <dir>.
  --no-cache-dir        Disable the cache.
  --disable-pip-version-check Don't periodically check PyPI to determine whether a new version of pip is available for
                        download. Implied with --no-index.
  --no-color            Suppress colored output

```

Figure 3 Installing package in Command

Once you have got a well connection between your PC and instruments, we can go ahead to look at how to control the instruments by your codes. As I mentioned before, we use SCPI to make the instruments understand our language. The SCPI commands that can be used for the instruments can be found by searching their programming guide with their model number on google. I have attached two programming guides for the instruments that I am using with this guide. You can refer to Figure 4 for an idea about how to combine Python and SCPI. Note that if you don't define anything of DMM, it will use the default setting for testing. If you do need accurate measurements, please refer to Figure 5 to set the DMM. You can refer to the programming guide for SCPI to see what is the meaning of the two MINs for DMM. This method is changing the accuracy of measurements by changing the integration time. Therefore, accurate measurements will take more time than the default measurements.



```

File Edit Format Run Options Window Help
import xrdlib.sys
import xlrd
from xlwt import Workbook
from xlrd import xldate_as_tuple
import visa

wb=Workbook()
sheet1=wb.add_sheet('Sheet 1')

visa_dll = 'c:/windows/system32/visa32.dll'
#tcp_addr = 'TCP/IP::192.168.1.1::inst0::INSTR'

gpi_b_addr1 = 'GPIB0::9::INSTR'
gpi_b_addr2 = 'GPIB0::22::INSTR'
# Create an object of visa_dll
rm = visa.ResourceManager(visa_dll)

# Create an instance of certain interface(GPIB and TCP/IP)

gpi_b_inst1 = rm.open_resource(gpi_b_addr1)
gpi_b_inst2 = rm.open_resource(gpi_b_addr2)

#gpi_b_inst1.write('VOLTage {<voltage>|2|4}')
gpi_b_inst1.write('INST:SEL OUT1')
gpi_b_inst1.write('VOLT 4')
#gpi_b_inst1.write('OUTPUT:ON')

gpi_b_inst1.write('INST:SEL OUT2')
gpi_b_inst1.write('VOLT 0')
#gpi_b_inst1.write('OUTPUT:ON')
gpi_b_inst1.write('VOLT:STEP 0.1')
gpi_b_inst2.write('CURRent:DC:NPLC 100')
#print('NPLC:' + gpi_b_inst2.query('CURRent:DC:NPLC?'))
n=1
while n<11:
    #print('OUTPUT:' + gpi_b_inst1.read())
    #Have some problems on NPLC setting
    gpi_b_inst2.write('CURRent:DC:NPLC 100')
    print('OUTPUT 2 Voltage: ' + gpi_b_inst1.query('VOLT?'))
    print('Current:' + gpi_b_inst2.query('MEASure:CURRent:DC?'))
    sheet1.write(n,0,float(gpi_b_inst1.query('VOLT?')))
    sheet1.write(n,1,1000000*float(gpi_b_inst2.query('MEASure:CURRent:DC?')))
    gpi_b_inst1.write('VOLT UP')
    n=n+1
else:
    print("finished")

gpi_b_inst1.write('INST:SEL OUT1')
print(gpi_b_inst1.query('VOLT?'))
gpi_b_inst1.write('INST:SEL OUT2')
print(gpi_b_inst1.query('VOLT?'))

wb.save('testing.xls')
Ln: 1

```

```

Python 3.7.0 Shell
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:59:51) [MSC v.1914 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
=== RESTART: C:\Users\Liuxi\Desktop\Research Project\VISA - 2 terminals.py ===
OUTPUT 2 Voltage:+0.00000000E+00
>>>
Current:+1.49300000E-07
OUTPUT 2 Voltage:+1.00000000E-01
Current:+1.20600000E-07
OUTPUT 2 Voltage:+2.00000000E-01
Current:+2.06500000E-07
OUTPUT 2 Voltage:+3.00000000E-01
Current:+1.71700000E-07
OUTPUT 2 Voltage:+4.00000000E-01
Current:+1.24700000E-07
OUTPUT 2 Voltage:+5.00000000E-01
Current:+1.22700000E-07
OUTPUT 2 Voltage:+6.00000000E-01
Current:+1.55400000E-07
OUTPUT 2 Voltage:+7.00000000E-01
Current:+1.43000000E-08
OUTPUT 2 Voltage:+8.00000100E-01
Current:+1.43100000E-07
OUTPUT 2 Voltage:+9.00000100E-01
Current:+1.65600000E-07
finished
+4.00000000E+00
+1.00000000E+00
>>>
Ln: 44 Col: 0

```

Figure 4 Controlling the instruments

```

gpi_b_inst1.write('INST:SEL OUT1')
print("VD=" + gpi_b_inst1.query('VOLT?'))
#print("ID=" + gpi_b_inst2.query('MEASure:CURRent:DC?'))
sheet1.write(m, 0+2*n, float(gpi_b_inst1.query('VOLT?')))
sheet1.write(m, 1+2*n, float(gpi_b_inst2.query('MEASure:CURRent:DC? MIN, MIN'))))
gpi_b_inst1.write('VOLT UP')
m=m+1

```

Figure 5 DMM measurements setting

```

gpi_b_inst1.timeout=5000
gpi_b_inst2.timeout=5000

```

Figure 6 Time out setting

If you run the program, an excel file with the name of your setting will be generated in the same folder with your codes once you have finished the testing. Your testing results are shown in the excel file.





 testing	2018/7/10 9:23
 [REDACTED]	2018/7/9 15:17
 VISA - 2 terminals	2018/7/10 9:24
 VISA - 3 terminals	2018/7/9 14:54

Figure 7 Excel generated

	A	B	C
1			
2	0	0.184	
3	0.1	0.1677	
4	0.2	0.1513	
5	0.3	0.139	
6	0.4	0.1227	
7	0.5	0.1452	
8	0.6	0.1309	
9	0.7	0.1349	
10	0.8	0.1656	
11	0.9	0.1513	
12			
13			

Figure 8 Testing results for 1 output system

C	D	E	F	G	H	I	J
0		1		2		3	
0	-5.93E-08	0	-1.64E-08	0	-4.91E-08	0	-6.1E-09
0.1	3.48E-08	0.1	3.27E-08	0.1	-5.93E-08	0.1	-7.56E-08
0.2	-7.16E-08	0.2	2.04E-08	0.2	-1.02E-08	0.2	0.00000002
0.3	-5.93E-08	0.3	-5.32E-08	0.3	2.66E-08	0.3	-1.02E-08
0.4	-8.2E-09	0.4	-1.84E-08	0.4	1.23E-08	0.4	-1.23E-08
0.5	-1.84E-08	0.5	3.07E-08	0.5	-4.91E-08	0.5	1.84E-08
0.6	-5.32E-08	0.6	-4.29E-08	0.6	3.27E-08	0.6	0.000000047
0.7	0	0.7	-2.45E-08	0.7	4.1E-09	0.7	5.11E-08
0.8000001	1.125E-07	0.8000001	0	0.8000001	-1.43E-08	0.8000001	-3.48E-08
0.9000001	-1.43E-08	0.9000001	1.84E-08	0.9000001	-7.56E-08	0.9000001	-8.2E-09
1	7.77E-08	1	-5.72E-08	1	1.02E-08	1	4.1E-09
1.1	-3.88E-08	1.1	-4.29E-08	1.1	1.02E-08	1.1	2.66E-08
1.2	-1.64E-08	1.2	1.23E-08	1.2	-1.64E-08	1.2	-2.45E-08
1.3	5.52E-08	1.3	3.48E-08	1.3	1.02E-08	1.3	-3.88E-08
1.4	2.86E-08	1.4	3.07E-08	1.4	6.75E-08	1.4	4.1E-09
1.5	2.678E-07	1.5	3.128E-07	1.5	3.087E-07	1.5	1.799E-07
1.6	1.4925E-06	1.6	1.4762E-06	1.6	1.3576E-06	1.6	1.4864E-06
1.7	4.6391E-06	1.7	4.7005E-06	1.7	4.6719E-06	1.7	4.5819E-06
1.8	9.9346E-06	1.8	0.000009953	1.8	9.8405E-06	1.8	9.9182E-06
1.9	1.64036E-05	1.9	1.63893E-05	1.9	1.63484E-05	1.9	1.63464E-05
2	2.35228E-05	2	2.35719E-05	2	0.000023484	2	2.35126E-05
2.1	3.12268E-05	2.1	3.11552E-05	2.1	3.11103E-05	2.1	3.11696E-05
2.2	3.92191E-05	2.2	3.91455E-05	2.2	3.92109E-05	2.2	0.000039123
2.3	4.74403E-05	2.3	4.73769E-05	2.3	0.000047426	2.3	4.73054E-05
2.4	5.56636E-05	2.4	5.56902E-05	2.4	5.55409E-05	2.4	5.55695E-05
2.5	0.000064122	2.5	6.41159E-05	2.5	6.40647E-05	2.5	6.40791E-05
2.6	7.26581E-05	2.6	7.26152E-05	2.6	7.25681E-05	2.6	7.26152E-05
2.7	8.13394E-05	2.7	8.13046E-05	2.7	8.12433E-05	2.7	8.12576E-05
2.799999	9.00247E-05	2.799999	8.99389E-05	2.799999	8.99204E-05	2.799999	0.000089947
2.899999	9.87183E-05	2.899999	9.86978E-05	2.899999	9.85342E-05	2.899999	9.86181E-05
2.999999	0.000107404	2.999999	0.000107379	2.999999	0.000107389	2.999999	0.0001074

Figure 9 Testing results for 2 output system

Appendix

Appendix A: How to configure Environmental Variables

Environmental Variables are super important for all types of programming, it gives your OS an idea about where to refer and how to understand your codes.

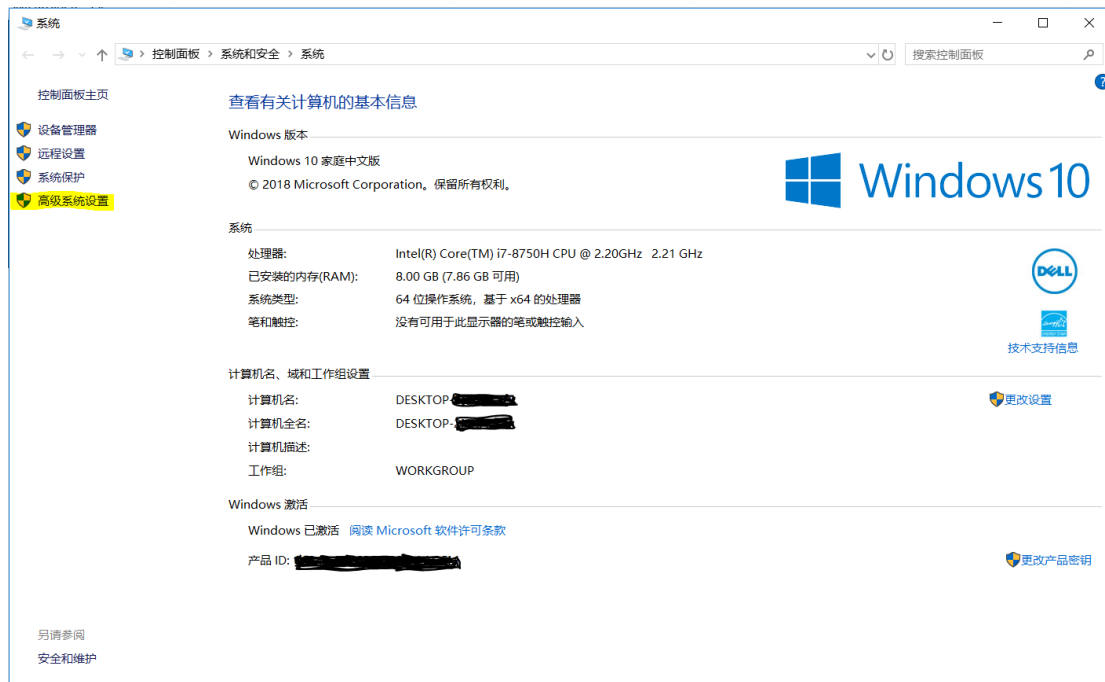


Figure A1

Firstly, go to the interface of your system by right click PC (shown in Figure A1). Click Advanced Setting in the left side to check environmental variables (shown in Figure A2). In Figure A3, double click PATH in the BOTTOM side then you are able to see your existing environmental variables.



Figure A2

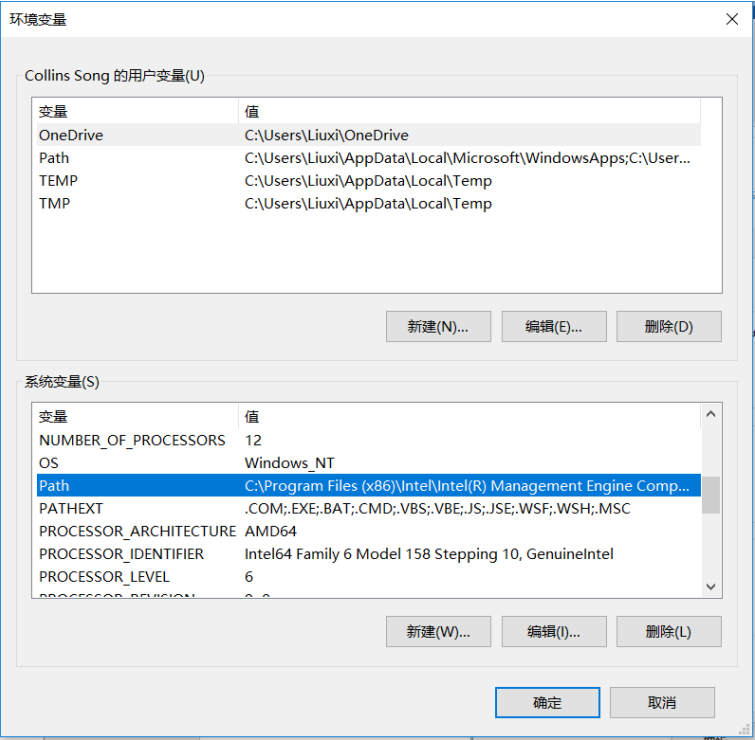


Figure A3

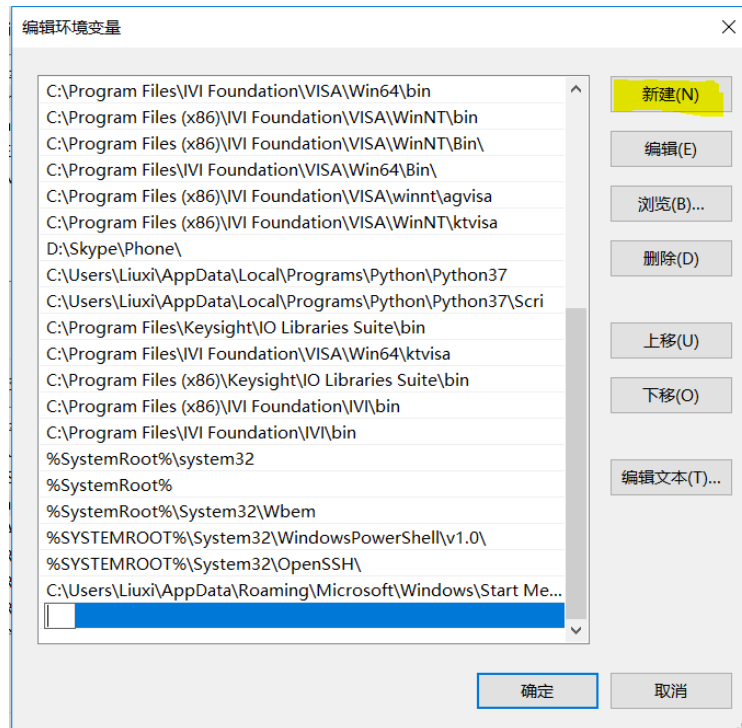


Figure A4

Click New in your environmental variables setting to add the paths of Python into your setting. Find your installed path of Python which is inside the folder of Python37 (for Python 3.7). Copy and paste its path into your environmental variables (as shown in Figure A5).

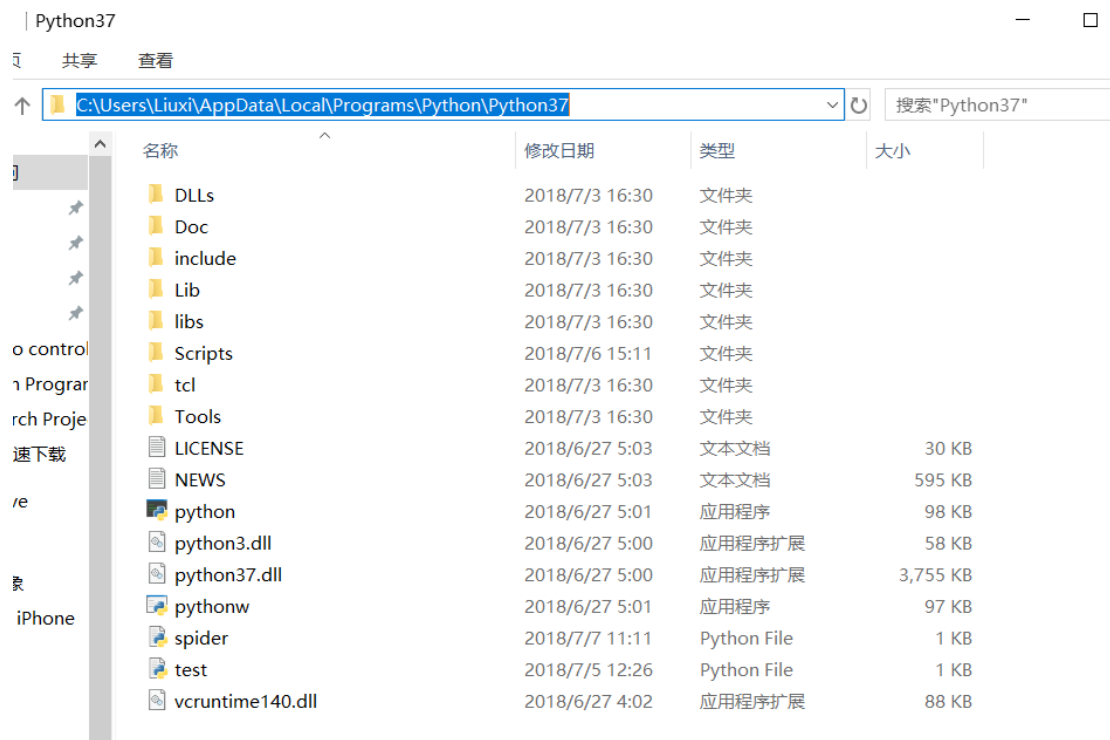


Figure A5

名称	修改日期	类型	大小
__pycache__	2018/7/5 15:45	文件夹	
chardetect	2018/7/5 15:52	应用程序	101 KB
easy_install	2018/7/3 16:30	应用程序	101 KB
easy_install-3.7	2018/7/3 16:30	应用程序	101 KB
exchangelib-1.11.4-py2.py3-none-an...	2018/7/6 11:41	WHL 文件	147 KB
f2py	2018/7/3 16:44	Python File	1 KB
futurize	2018/7/5 15:52	应用程序	73 KB
futurize-script	2018/7/5 15:52	Python File	1 KB
pasteurize	2018/7/5 15:52	应用程序	73 KB
pasteurize-script	2018/7/5 15:52	Python File	1 KB
pip	2018/7/3 16:30	应用程序	101 KB
pip3.7	2018/7/3 16:30	应用程序	101 KB
pip3	2018/7/3 16:30	应用程序	101 KB
pyvisa-info	2018/7/3 17:10	应用程序	73 KB
pyvisa-info-script	2018/7/3 17:10	Python File	1 KB
pyvisa-shell	2018/7/3 17:10	应用程序	73 KB
pyvisa-shell-script	2018/7/3 17:10	Python File	1 KB
runxldr	2018/7/5 15:45	Python File	16 KB
yagmail	2018/7/6 15:11	应用程序	101 KB

Figure A6

Then open the Scripts folder to copy paste its path to environmental variables as well (as shown in Figure A6). Do the same thing to add the paths of Keysight Instrument Control Bundle if you didn't see it in your environmental variables. The configured paths are shown in Figure A7, where yellow paths are for Python, the rest five paths labeled in red are for Keysight Instrument Control Bundle (make sure that you have the correct path, otherwise it will not work).

```

C:\Users\Liuxi\AppData\Local\Programs\Python\Python37
C:\Users\Liuxi\AppData\Local\Programs\Python\Python37\Scripts
C:\Program Files\Keysight\IO Libraries Suite\bin
C:\Program Files\IVI Foundation\VISA\Win64\ktvisa
C:\Program Files (x86)\Keysight\IO Libraries Suite\bin
C:\Program Files (x86)\IVI Foundation\IVI\bin
C:\Program Files\IVI Foundation\IVI\bin

```

Figure A7