telitpython: strategytodevelop

A possible strategy to develop Python code for Telit modules

1. **edit** the source code with **Pythonwin** editor distributed by Telit accessible at Telit Download zone (www.telit.com/en/products/download-zone.php) (username and password required) or asking to ts-emea@telit.com

See the **INSTALL** Python IDE section for Windows or for Linux and the **EDIT** Python scripts section for Windows or for Linux for more details

Maybe a good method to learn the Python for Telit modules is to read the Easy Script in Python user guide, then develop the Python code and test it on the module directly starting by the Examples which are available in the Download zone (ask to you distributor if you dont' have access: he should download the scripts for you). Very useful references are the manuals of the 1.5.2+ version available with PythonWin package and the test of little code or instructions you can do with the Pythonwin tool in its Interactive window.
It's important to test also small parts of code working on PC on the module direclty because there are some differences between the 1.5.2+ version for PC installed with Pythonwin and the same version of Python engine built for Telit modules (the Easy Script in Python is a reference for this)

2. put more **print** instructions in the source code to get later a your customized debug.
To visualize the messages in the print instructions arguments, you need to have available the debug port that is accessible in different ways according to different kind of modules (basically: or the Tx_trace and Rx_trace lines are available on your module or you should use TelitSerialPortMux application and in this case the most part of the module serial lines (Tx,Rx, RTS,CTS,DTR,DSR,GND) must be available on your board to make available a debug port

If you have a USB connector at PC side you could implement or buy some USB to serial adapter cable

Alternatively (or in combination with print) you could use the **SER.send** method to visualize on AT commands port your debug messages

If you can't put all the serial lines on your module and if you want to debug your code, you could **forward the print outputs to the AT command serial port** instead of debug port using this trick to forward the print messages to the AT command port:

```
import sys
import SER
import MDM
```

```
import MOD



class SERstdout:
    def __init__(self):
        SER.set_speed("115200","8N1")
    def write(self,s):
        SER.send(s)


if(sys.platform != "win32"):       #
    sys.stdout = SERstdout()    # Redirect print statements to SERIAL ASC0
    sys.stderr = SERstdout()     # Redirect errors to SERIAL ASC0
```

3. select the **Check** icon on the menu bar (or Ctrl+Shift+C) to **check the syntax** of the code if it's correct or not

4. select the **.py file** you have created in your PC folder and then click on the right key of the mouse and select **"Compile"**

5. you should get a **.pyo file** on your folder. Arrange to don't have any .py file downloaded in the module and to don't have any application on PC which is using the COM port.

6. Then,

- supply the EVK2 (www.telit.com/en/products/gsm-gprs.php?p_id=12&p_ac=show&p=17) evaluation kit board

The EVK comes with 2 DB9 serial com ports (PROG/DATA), 1 USB port converting 2 serial ports (only DB9 or USB can be selected using the jumpers)

- put a RS232 serial cable between the COM port of your PC and the upper port (*PROG*) of the evk2 (ASC0)

  OR

  put a USB to serial cable between the USB port of your PC and the upper port (*PROG*) of the evk2 (ASC0)

  OR

  put the USB to USB cable provided with the EVK2 between the USB port of your PC and the USB port of the EVK2

- then select the pyo file to **download**, click the right key of the mouse and then select **Download**. The downloading should be done

(see also the file *Python's New Tool DRAFT.pdf*)

7. **enable** the script you have downloaded on the module with the command at#escript (e.g. **at#escript="<filename>.pyo"**)

8. **run** your script in one of the following ways:

8.1 at switch on, according to the DTR status (if at#startmodescr=0)
8.2 at switch on, regardless the DTR status (if at#startmodescr=1 or 2)
8.3 on user choice, with at#execscr command (from the firmware ver. 7.02.02) (I suggest this for debug on module)

It's suggested to run the Python script off-line (emulation mode, running it from PythonWin IDE on PC) only when you need to carry out some comparisons with the Python running on the module. This is because:
- the timing of the communications between Python code running on Pc and module could be different
- not all the Python libraries (e.g. IIC and SPI) implemented for the 1.5.2+ Python environment in the Telit module are implemented for the 1.5.2+ Python enivronment in the PC and viceversa: FLOATING POINT IS NOT SUPPORTED for the 1.5.2+ Python environment in the Telit module !!!
- you could risk to use for PC a Python version different from 1.5.2+ which is the version embedded in the modules and edit Python sources not compatible

9. **debug** your script.

The best thing to debug Python code for your module is to debug the code while the script is running on the module.
The alternative is to debug your code with the help of the PythonWin IDE, but in this case you should pay attention to the same reasons which discourage to run the Python code in emulation mode (from PythonWIn) : many error messages could result from a mismatching between the Python environment in the module and Python environment in the PC.

For debugging a good solution is to use TelitSerialPortMux application, deselect Python flag on the Setup window, connect one hyperterminal session to the Virtual Com #1 (to send AT commands) and one hyperterminal session to the Virtual Com #4 (debug).
Set: **at#selint=2**, **at#cmuxscr=0, at#startmodescr=0** > Disconnect the hyperteminal windows > Switch off the module > switch on the module > reconnect the hyperterminal sessions to the above virtual Coms > issue on hyperterminal session connected to VCOM#1: **at#escript="<filename>.pyo"** > **at#execscr** > you should observe on the hyperterminal session connected to VCOM#4 the debug output (if you have not reforwarded the print outputs)