

Developing with Python on iMX Developer's Kits

Embedded Artists AB

Davidshallsgatan 16
SE-211 45 Malmö
Sweden

<http://www.EmbeddedArtists.com>

Copyright 2016 © Embedded Artists AB. All rights reserved.

No part of this publication may be reproduced, transmitted, transcribed, stored in a retrieval system, or translated into any language or computer language, in any form or by any means, electronic, mechanical, magnetic, optical, chemical, manual or otherwise, without the prior written permission of Embedded Artists AB.

Disclaimer

Embedded Artists AB makes no representation or warranties with respect to the contents hereof and specifically disclaim any implied warranties or merchantability or fitness for any particular purpose. Information in this publication is subject to change without notice and does not represent a commitment on the part of Embedded Artists AB.

Feedback

We appreciate any feedback you may have for improvements on this document. Send your comments by using the contact form: www.embeddedartists.com/contact.

Trademarks

All brand and product names mentioned herein are trademarks, services marks, registered trademarks, or registered service marks of their respective owners and should be treated as such.

Table of Contents

1	Document Revision History	4
2	Introduction	5
2.1	Conventions.....	5
3	Quick Start.....	6
3.1	Hello World	6
3.2	File Example	6
3.3	DisplayHelper	7
4	Target Requirements	9
4.1	Adding Packages Using Yocto.....	9
4.2	Adding Packages Using PIP.....	9
5	Develop.....	11
5.1	Development and Debugging on a PC	11
5.2	Remote Debugging Using Eclipse	11
5.2.1	Prepare Target	11
5.2.2	Prepare PC.....	12
5.2.3	Setup a Connection to Target.....	17
5.2.4	Create a Project	20
5.2.5	Debug.....	20
5.2.6	Breakpoints	21
5.2.7	Exiting Eclipse	22
5.2.8	Remote Shell.....	23
5.2.9	Target Changed IP Address	23
6	Further Expansion	25
7	Troubleshooting.....	26
7.1	Cannot get an SSH connection to the target	26
7.1.1	Solve "Access denied" problem	26

1 Document Revision History

<i>Revision</i>	<i>Date</i>	<i>Description</i>
A	2016-12-07	First release

2 Introduction

Python is a widely used high-level interpreted programming language that has been around since 1991. Python interpreters are available on many operating systems including Linux, Windows and Mac OS.

This document provides you with step-by-step instructions to start writing Python scripts for the Embedded Artists iMX based COM boards. It is assumed that the reader is familiar with Python as this is no course but rather a getting started guide.

The instructions have been tested on PCs running both Windows and Ubuntu. As Eclipse is a multi-platform IDE the experience is very similar in both operating systems. Differences have been noted in the documentation when it impacted the outcome (e.g. handling of breakpoints in section 5.2.6).

Additional documentation you might need is.

- The *Getting Started* document for the board you are using.
- The *Working with Yocto* document

2.1 Conventions

A number of conventions have been used throughout to help the reader better understand the content of the document.

Constant width text – is used for file system paths and command, utility and tool names.

```
$ This field illustrates user input in a terminal running on the  
development workstation, i.e., on the workstation where you edit,  
configure and build Linux
```

```
# This field illustrates user input on the target hardware, i.e.,  
input given to the terminal attached to the COM Board
```

```
This field is used to illustrate example code or excerpt from a  
document.
```

```
This field is used to highlight important information
```

3 Quick Start

All pre-built file system images for the Embedded Artists iMX based COM boards come with a python interpreter already installed.

3.1 Hello World

The very first example in most programming languages is a program that writes "Hello World" on the terminal. In python this can be done with very little code. Start by booting into Linux and then open a new file with an editor:

```
# nano hello.py
```

Copy and paste the following into the editor:

```
#!/usr/bin/python
print "Hello World"
```

Save the file and exit the editor (in `nano` it is `Ctrl-X` followed by `Y` and `Enter`).

Try the script with:

```
# python hello.py
Hello World
```

3.2 File Example

This example shows how to pass arguments to a script and some very basic file handling. The script does the same as the system command `wc`, that is it counts the number of lines, words, and bytes in the specified file.

Open an editor:

```
# nano fileinfo.py
```

Copy and paste the following into the editor:

```
#!/usr/bin/python

import sys
import os

def printUsage():
    print "Usage:"
    print "    python", sys.argv[0], "<file>"
    exit(0)

def parse(fname):
    lineCount = 0
    wordCount = 0
    byteCount = os.path.getsize(fname)
    with open(fname, 'r') as f:
        lines = f.readlines()
        lineCount = len(lines)
        for line in lines:
            wordCount += len(line.strip().split())
```

```

    return (lineCount, wordCount, byteCount)

if __name__ == "__main__":
    if len(sys.argv) != 2:
        printUsage()

    fname = sys.argv[1]
    if not os.path.exists(fname):
        print "File does not exist:", fname
        print
        printUsage()

    (lines, words, bytes) = parse(fname)

    print " " * len(fname), "  Lines  Words  Bytes"
    print "%s  %5d  %5d  %5d" % (fname, lines, words, bytes)

```

Run the script with the `/var/log/messages` file as input:

```
# python fileinfo.py /var/log/messages
```

If there is any problem with the copy-pasting then the script can be downloaded like this instead:

```
# wget http://imx.embeddedartists.com/python/fileinfo.py
```

3.3 DisplayHelper

This is a script that will:

- Extract information about configured displays (actually the frame buffers)
- Draw a pattern on each frame buffer
- Cycle through the backlight settings for each backlight controller to fade the display in and out three times

To download the example (it is too large to include in this document):

```
# wget http://imx.embeddedartists.com/python/DisplayHelper.py
```

Run the script with:

```

# python DisplayHelper.py

Found framebuffer: ['/dev/fb0', '/dev/fb1']
Detected /dev/fb0 - available
Detected /dev/fb1 - available
Testing:
  Device:      /dev/fb0
  Size:        800 x 480
  Type:        RGB 888
  Byte/Pixel: 4
  Blank:       /sys/devices/soc0/fb.20/graphics/fb0/blank
Unblanking
Drawing 4bpp gradient, h = 120

```

```
Testing:
  Device:      /dev/fb1
  Size:        240 x 320
  Type:        RGB 888
  Byte/Pixel: 4
  Blank:       /sys/devices/soc0/fb.20/graphics/fb1/blank
Unblinking
Drawing 4bpp gradient, h = 80
Found backlights:
  path: /sys/class/backlight/backlight2.22
  actual_brightness: 7
    bl_power: 0
    brightness: 7
  max_brightness: 7
    type: raw
    uevent: bin(0)
Waiting 5 seconds before backlight tests...
Starting backlight manipulation..
Adjusting brightness for backlight2.22
  loop= 0 lvl= 0
  loop= 0 lvl= 1
  loop= 0 lvl= 2
  ...
```

4 Target Requirements

All pre-built file system images for the Embedded Artists iMX based COM boards come with a Python interpreter already installed. Additional packages can be added either using Yocto when setting up the file system or in runtime using the pip package manager.

4.1 Adding Packages Using Yocto

At the time this document was written the following packages could be added using Yocto:

python-async	python-imaging	python-pyparted
python-autobahn	python-jinja2	python-pyqt
python-backports-ssl	python-ldap	python-pyrex
python-certifi	python-lxml	python-pyserial
python-cffi	python-m2crypto	python-pytz
python-cheetah	python-mako	python-pyudev
python-cloudeebus	python-markupsafe	python-pyusb
python-cmd2	python-matplotlib	python-pyyaml
python-cython	python-mccabe	python-pyzmq
python-dateutil	python-mock	python-requests
python-dbus	python-msgpack	python-scons
python-dbusmock	python-nose	python-simplejson
python-decorator	python-numeric	python-six
python-distribute	python-numpy	python-slip-dbus
python-django	python-pep8	python-smartpm
python-django-south	python-pexpect	python-smbus
python-docutils	python-pip	python-smmmap
python-enum34	python-prettytable	python-snakefood
python-epydoc	python-psutil	python-sqlalchemy
python-feedparser	python-pyalsaaudio	python-tornado
python-flufl-enum	python-pycairo	python-twisted
python-futures	python-pycparser	python-ujson
python-gdata	python-pycurl	python-urllib3
python-gevent	python-pyflakes	python-vobject
python-git	python-pygments	python-webdav
python-gitdb	python-pygtk	python-zopeinterface
python-greenlet	python-pyopenssl	
python-gst	python-pyparsing	

The instructions on how to find available packages as well as how to include them in the image build is explained in the *Working with Yocto* document.

4.2 Adding Packages Using PIP

Pip is a package management system used to install software packages written in Python. Pip can be used to install Python extensions from an online repository containing more than 80000 packages. With the correct extension installed almost anything can be done in Python.

Pip can be installed by including the `python-pip` package when building the file system as explained in the previous section. It has already been installed in the prebuilt images from Embedded Artists.

Pip handles packages from <https://pypi.python.org/pypi> which claims to have more than 80000 packages. To install the `prettytable` package:

```
# pip install prettytable
```

```
Downloading/unpacking prettytable
  Downloading prettytable-0.7.2.zip
  Running setup.py
(path:/var/volatile/tmp/pip_build_root/prettytable/setup.py)
egg_info for package prettytable

Installing collected packages: prettytable
  Running setup.py install for prettytable

Successfully installed prettytable
Cleaning up...
```

Note that a lot of pip packages have dependencies and if not met the package cannot be installed. A careful inspection of the output from `pip` will give a clue to what is missing.

5 Develop

There are several ways to write, test and debug python scripts:

- **Everything done on target.** Using a terminal and a text editor (`nano` and `vi` are installed by default) to write the script and then execute the script to test it. This works very well for small scripts or small changes of large scripts. See section 3.1 for a description of this.
- **Development and debugging on a PC.** There are many graphical editors or development environments that can be used on a PC to write and debug python scripts. When the script is working then it can be transferred to the target for testing. This setup works well for scripts that don't use target-specific instructions. For example, paths can be changed while debugging on the PC and then changed back before executing on target.
- **Remote debugging.** Using an IDE on a PC to write the code and a remote debugger to step through the code on the target. More complex to setup but in the end the most powerful way.

5.1 Development and Debugging on a PC

This alternative has very little to do with the target as everything is developed on the PC. There are a lot of options on development environments and editors on PCs and it all comes down to the preferences of you the developer.

The developed program can be transferred to the target in a lot of different ways including:

- USB Memory Stick
- An uSD card
- FTP (requires extra software on the target)
- SCP (secure copy over SSH)
- If it is a single file then opening it in an editor and copy-pasting from the editor on the PC works.

5.2 Remote Debugging Using Eclipse

This chapter describes how to setup the Eclipse IDE with python support and the extra bits needed to remotely debug code running on the Embedded Artists iMX based COM boards.

5.2.1 Prepare Target

SSH Server

The PC software requires an SSH connection to the target in order to control it and transfer files to/from it. The SSH server can be installed on the target by including the `ssh-server-openssh` feature when building the image as explained in the *Working with Yocto* document. It has **already been installed** in the prebuilt images from Embedded Artists.

The images built with Yocto have two users: **tester** with no password and **root** with the password **pass**. To gain access to all files and folders, use the root user when developing.

Use an SSH client on the PC and try to connect to the target. Make sure that it works before continuing on with the steps below to avoid problems down the road. If you have problems creating an SSH connection to the target see section 7.1 for more instructions.

PyDev Debugger (pydevd)

PyDev Debugger is the backend (running on the target) to the PyDev Eclipse plugin that will be used on the PC. To install the package:

```
# pip install pydevd
Downloading/unpacking pydevd
  Downloading pydevd-0.0.6.zip (1.1MB): 1.1MB downloaded
  Running setup.py
  ...
  Plain-python version of pydevd installed (cython speedups not
  available).
Successfully installed pydevd
Cleaning up...
```

The output of the pip command above has been edited as it included some warnings. The installed `pydevd` will work despite the warnings.

5.2.2 Prepare PC

The PC needs to have Python installed as well as the Eclipse IDE with the PyDev and Remote System Explorer plugins. This will be covered below.

Python

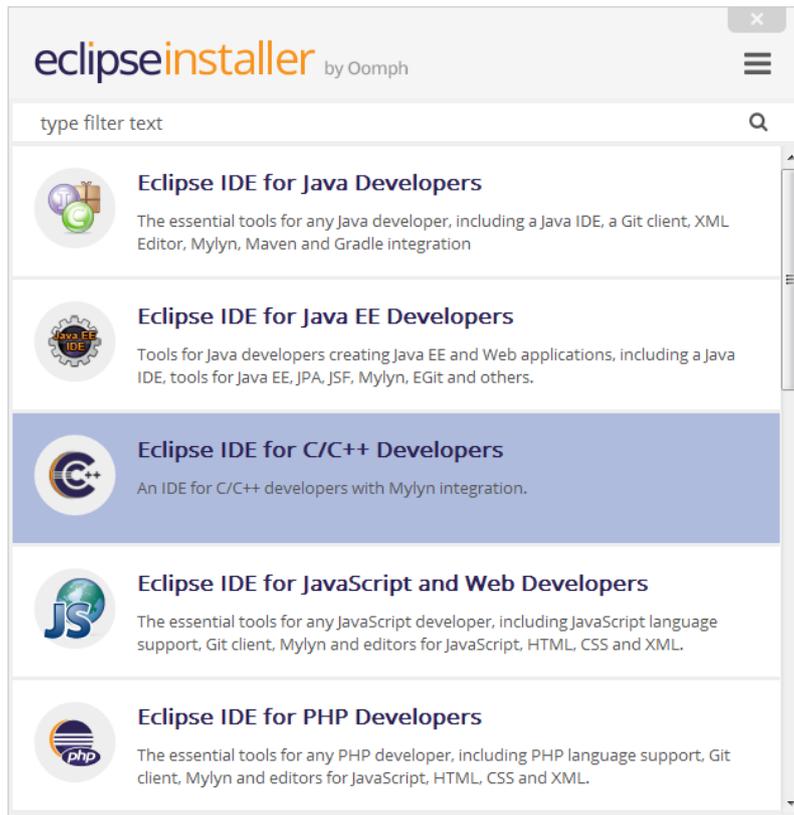
The first thing is to install Python. It is best to choose the same version as the one used on the target. To check which version that is used run the following command in a terminal:

```
# python -V
Python 2.7.9
```

A Python installer can be downloaded from <https://www.python.org/downloads/>.

Eclipse

The next thing to do is to download and run the Eclipse installer from (<https://www.eclipse.org/downloads/>). When this document was written the latest version was called Neon. Select the "Eclipse IDE for C/C++ Developers" version when prompted:

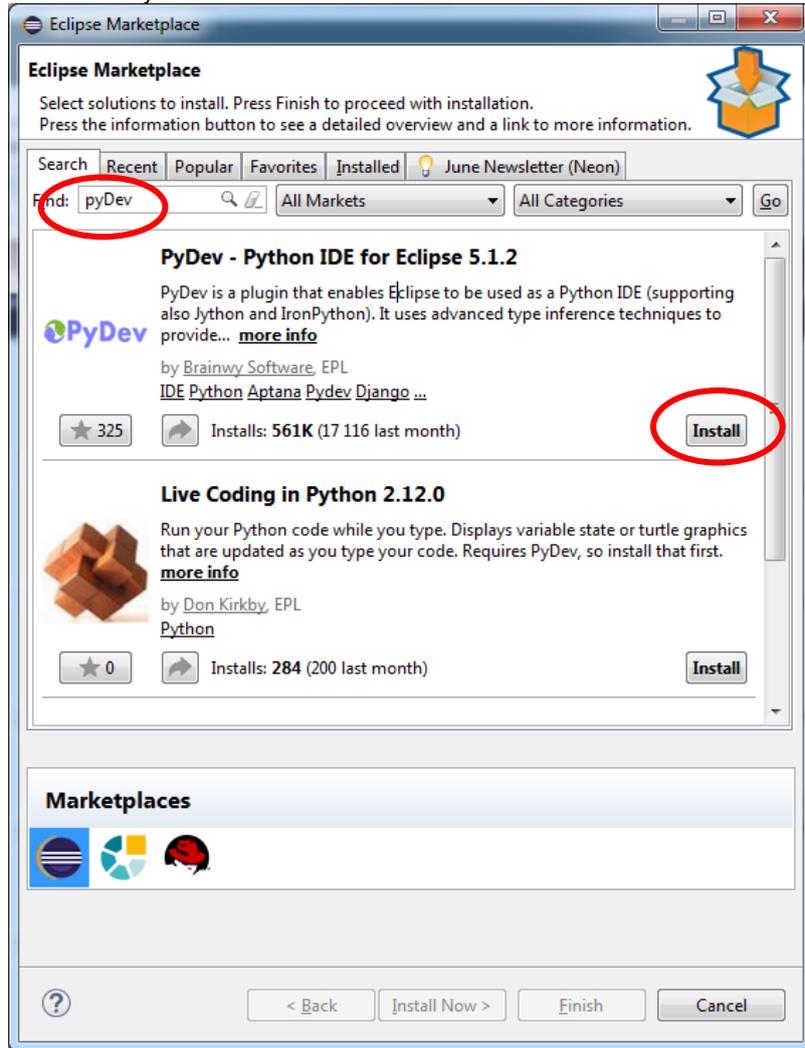


Complete the wizard, accept the license agreement and wait for the installation to complete.

Install PyDev plugin in Eclipse

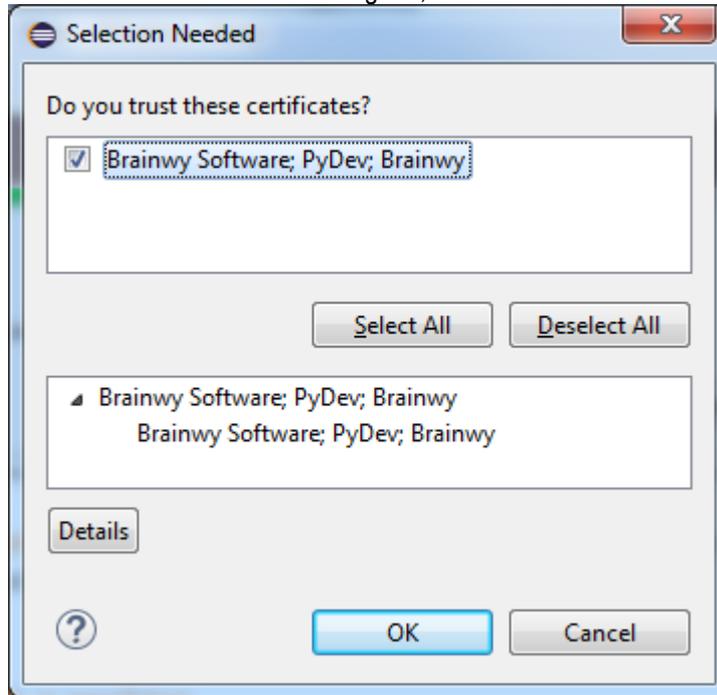
- 1) Start Eclipse
- 2) Go to Help→Eclipse Marketplace

3) Search for PyDev and then click Install

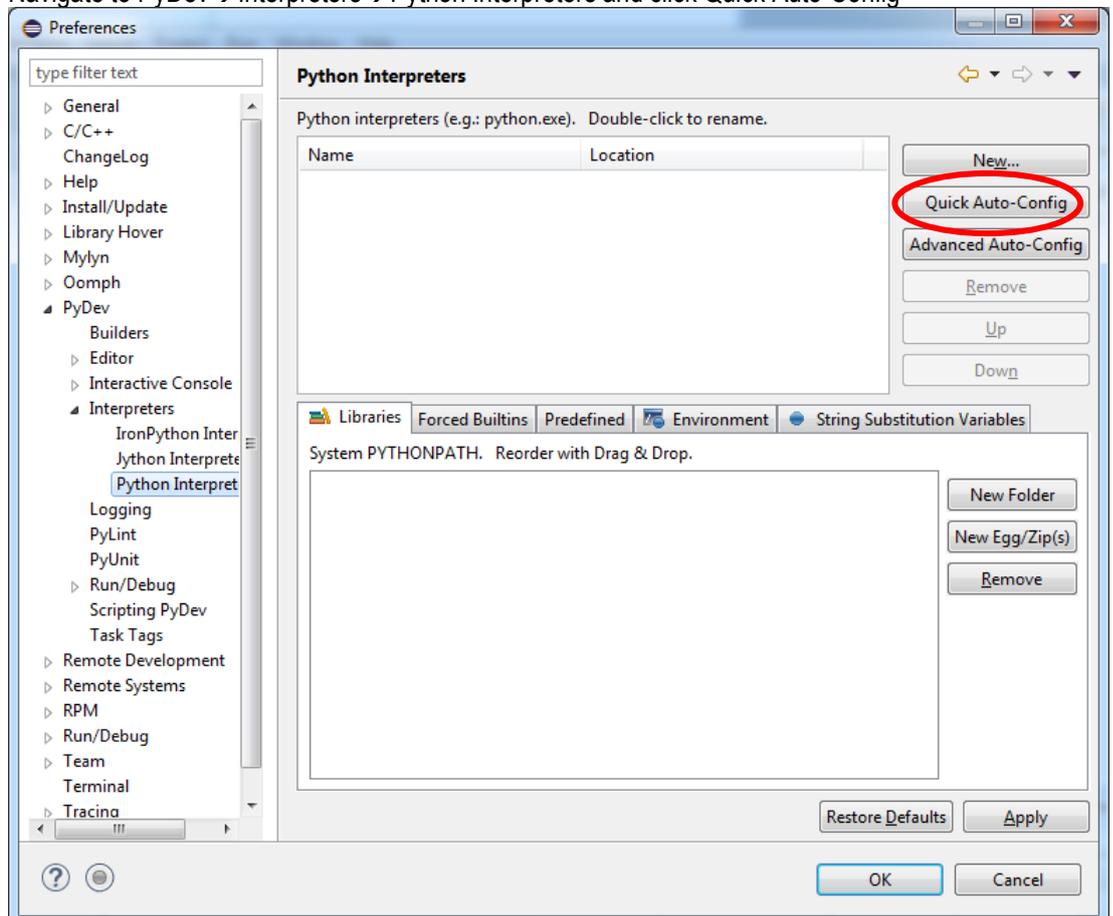


4) Accept license agreements to continue with the installation

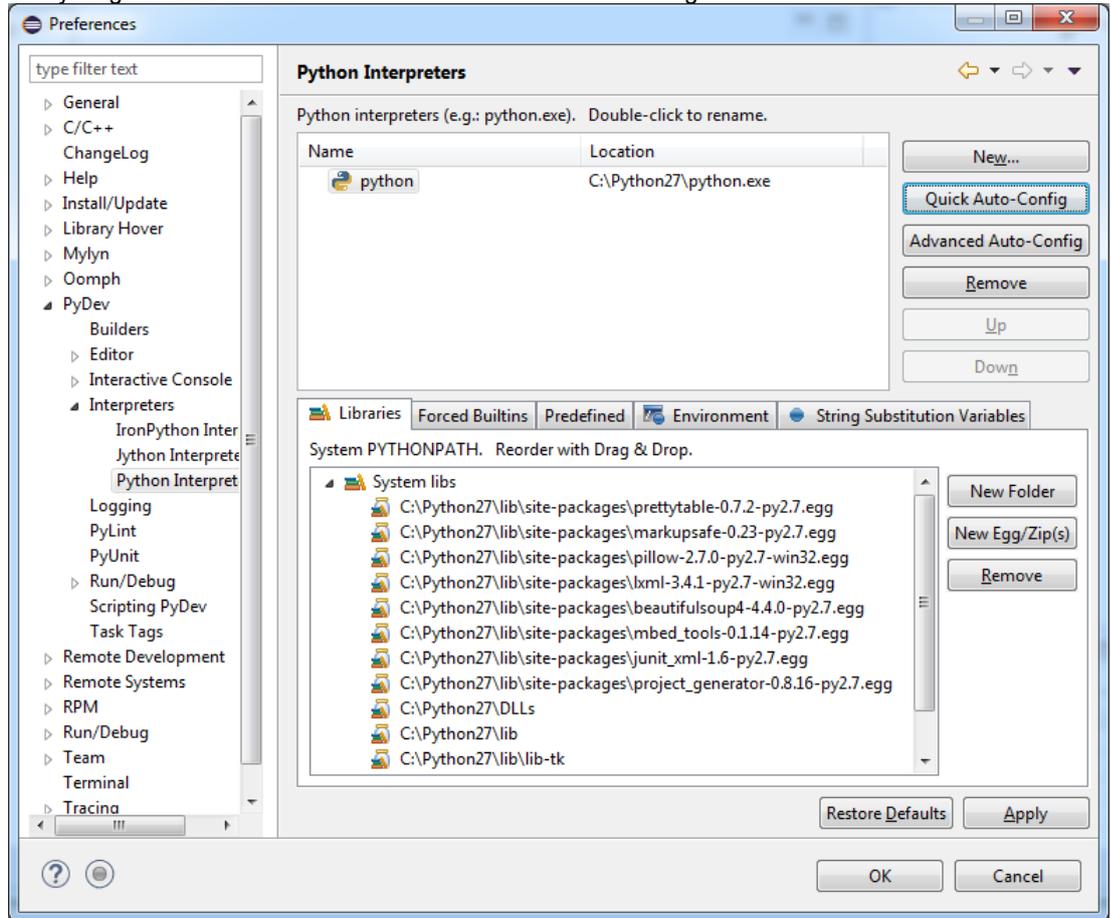
- 5) When presented with the "Selection Needed" dialog it is important to mark the Brainwy Software as trusted before clicking OK, otherwise the installation may fail.



- 6) Agree to restart Eclipse for the changes to take effect
- 7) Go to Window → Preferences
- 8) Navigate to PyDev → Interpreters → Python Interpreters and click Quick Auto-Config



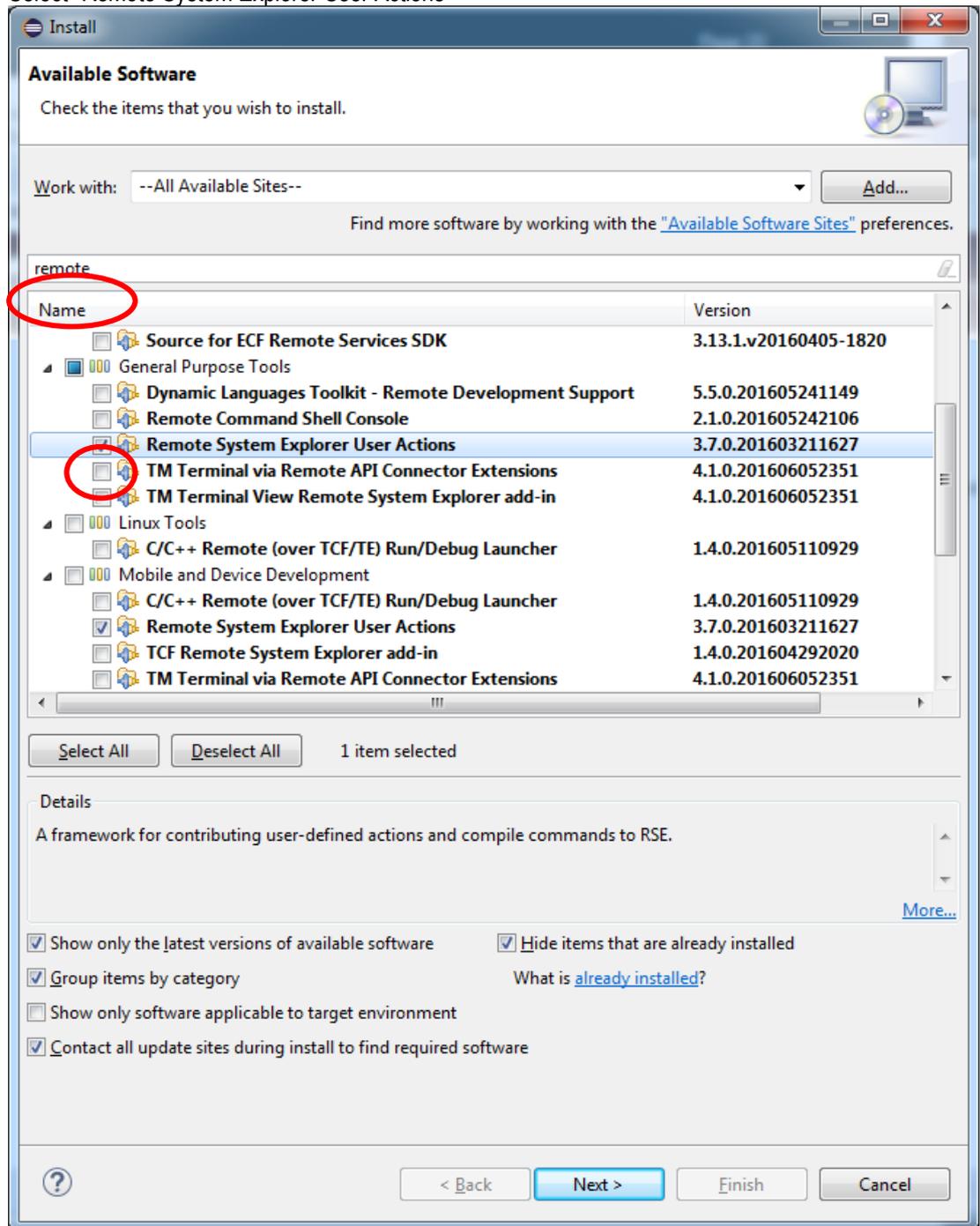
- 9) Everything should now be filled in so click OK to save the settings



Install Remote System Explorer Plugins

- 1) Go to Help → Install New Software
- 2) Change to work with "-- All Available Sites --" and wait for the list to load
- 3) Enter "remote" as filter

- 4) Select "Remote System Explorer User Actions"



- 5) Click Next, accept licences and finally restart Eclipse

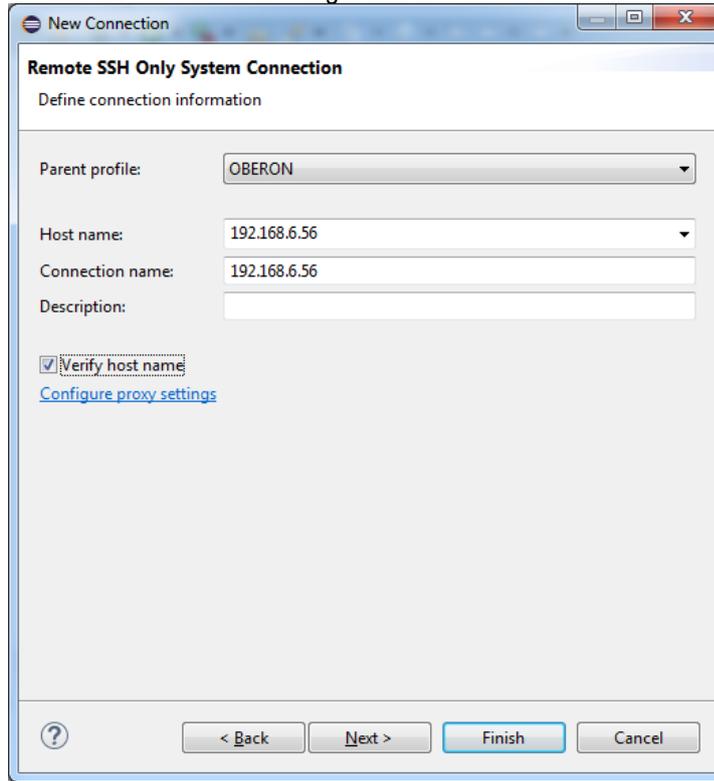
The PC is now ready.

5.2.3 Setup a Connection to Target

It is time to configure Eclipse so that it knows how to connect to the target. This is needed for debugging as well as adding/updating/deleting files.

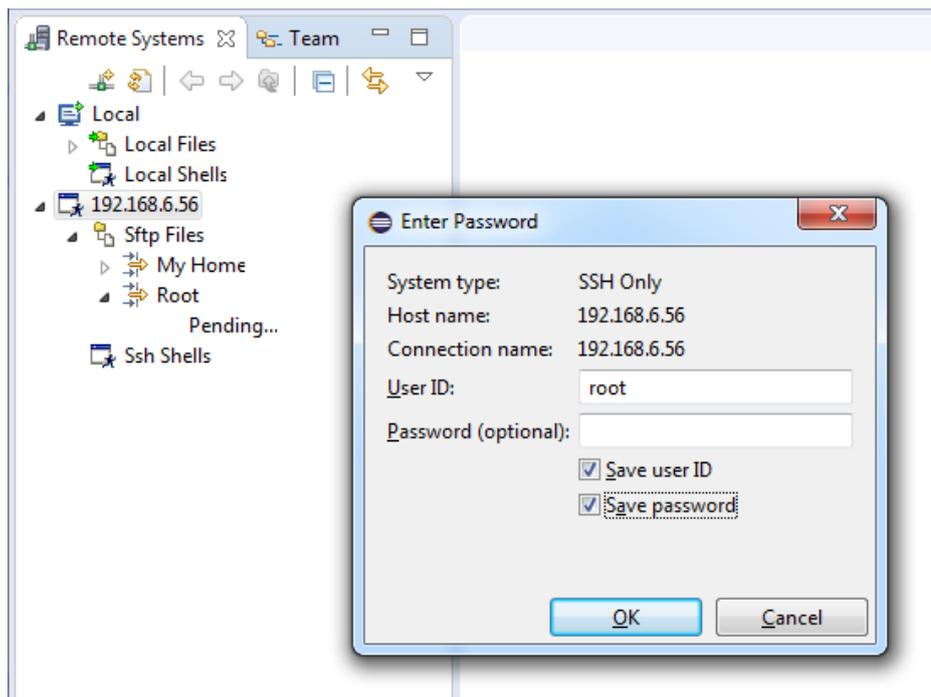
- 1) Go to Window → Perspective → Open Perspective → Other...

5) Fill in the IP number of the target and click Finish



There will now be a new item in the Remote System View. Expand the Sftp Files node to see parts of the file system on the target.

An Enter Password dialog will appear when attempting to expand the Root node. If the password was changed or a user was added in section 5.2.1 above then fill in that information, otherwise use the default which is **root** as user and with **pass** as password.



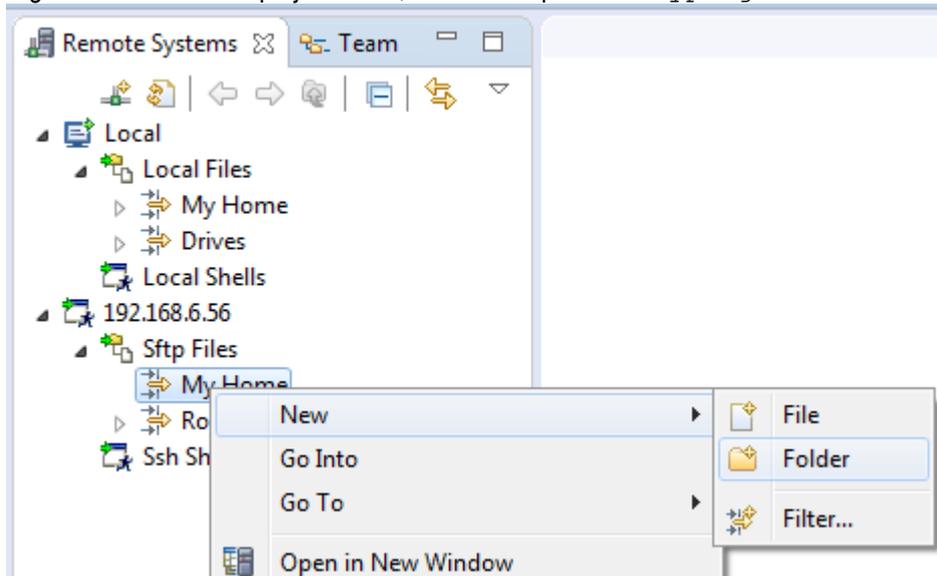
A warning about an RSA key fingerprint may appear. Click Yes to continue.

It should now be possible to browse the target's file system without any more dialogs appearing.

5.2.4 Create a Project

Now that everything has been prepared it is finally time to create the python project. This is done in the Remote System View.

- 1) Expand the target node down to the My Home node
- 2) Right-click and add the project folder, in this example called `myproj1`



- 3) Now turn that folder into a project by right-clicking on it and selecting the Create Remote Project option. It will seem like nothing happens.
- 4) Add a new file to the `myproj1` folder, in this example called `test.py`
- 5) Double-click the `test.py` file and then copy-paste the following into it:

```
#!/usr/bin/python

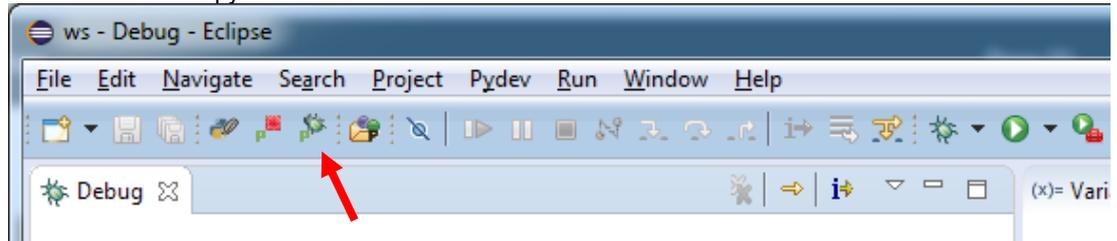
import sys
import pydevd
pydevd.settrace('192.168.6.54') # replace IP with address
                                # of Eclipse host machine

i = 3
p = 'Hello!' * i
print p
```

5.2.5 Debug

- 1) Open the Debug perspective in Eclipse by going to the Window→Perspective→Open Perspective→Other...
- 2) Select Debug from the list and click OK

- 3) Click the "Start the pydev server" button

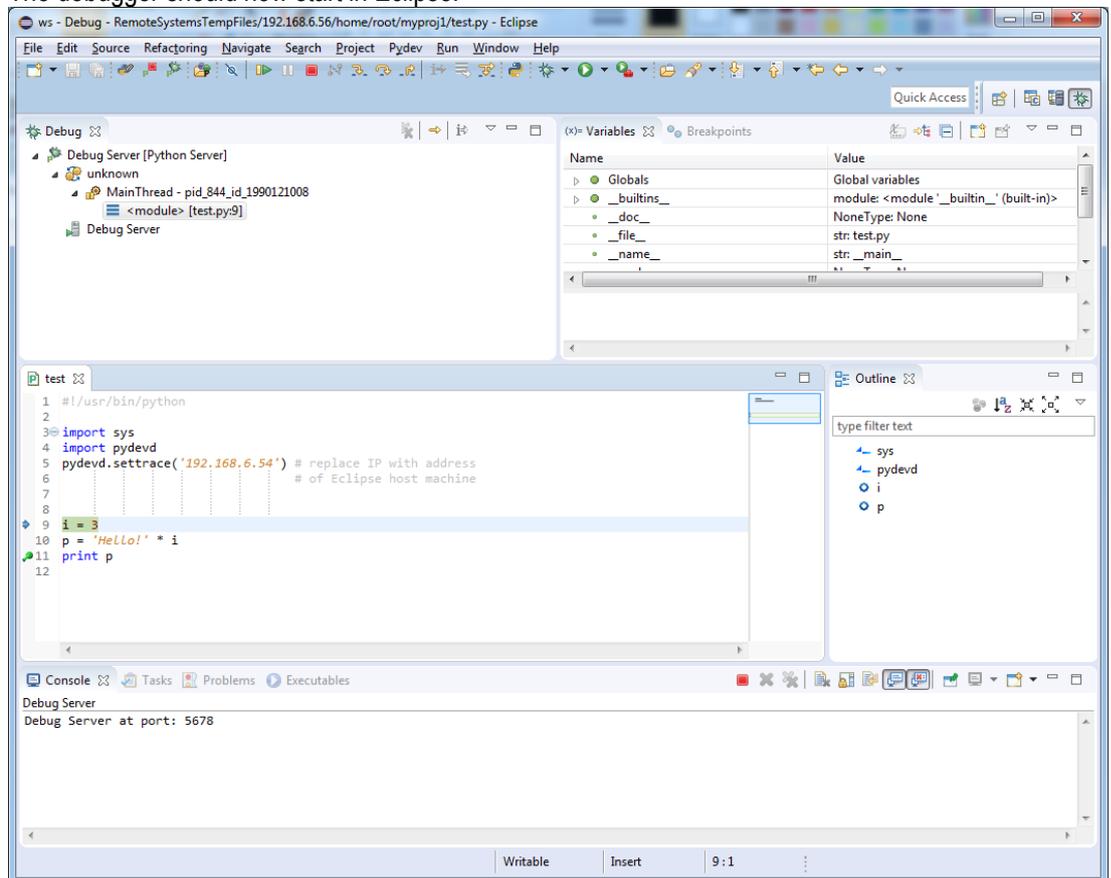


don't confuse the button with the normal debug button. The server button has a small **p** on it.

- 4) Now start the program on the target. Type the following in the terminal:

```
# cd myproj1/
# python test.py
```

- 5) The debugger should now start in Eclipse:



The debugger has options to set breakpoints, inspect/change variable values, single step and run through code. The debug server will keep running even if the test script reaches its end. Just run again from step 4 as many times as is needed.

5.2.6 Breakpoints

Due to the way that PyDev handles paths, breakpoints may not work. To fix this (only needed one time per target):

- 1) Double-click on the area to the left of the line number in test.py to set a breakpoint
- 2) Start the program

```
# cd myproj1/
# python test.py

warning: Debugger speedups using cython not found. Run
'"/usr/bin/python" "/usr/lib/python2.7/site-
packages/setup_cython.py" build_ext --inplace' to build.

pydev debugger: warning: trying to add breakpoint to file that
does not exist: /home/root/myproj1/C:/temp/2016-07-
05/ws/RemoteSystemsTempFiles/192.168.6.56/home/root/myproj1/te
st.py (will have no effect)
```

- 3) To fix this use the Remote Systems View in Eclipse and browse to Sftp Files → Root → usr → lib → python2.7 → site-packages
- 4) Double-click on pydev_file_utils.py to open it
- 5) Scroll down to the line with

```
PATHS_FROM_ECLIPSE_TO_PYTHON = []
```

- 6) Change that line to

```
PATHS_FROM_ECLIPSE_TO_PYTHON = [
(r'C:\temp\2016-07-05\ws\RemoteSystemsTempFiles\192.168.6.56\home\root',
 r'/home/root')
]
```

Substitute the path with what your warning pointed to (marked in green in step 2 above) as it depends on where your workspace is located.

- 7) Save the file
- 8) Stop and restart the debugging

When tested on a PC running Ubuntu 14.04 the breakpoints worked without the fix above and the statement should be kept at the default which is

```
PATHS_FROM_ECLIPSE_TO_PYTHON = []
```

5.2.7 Exiting Eclipse

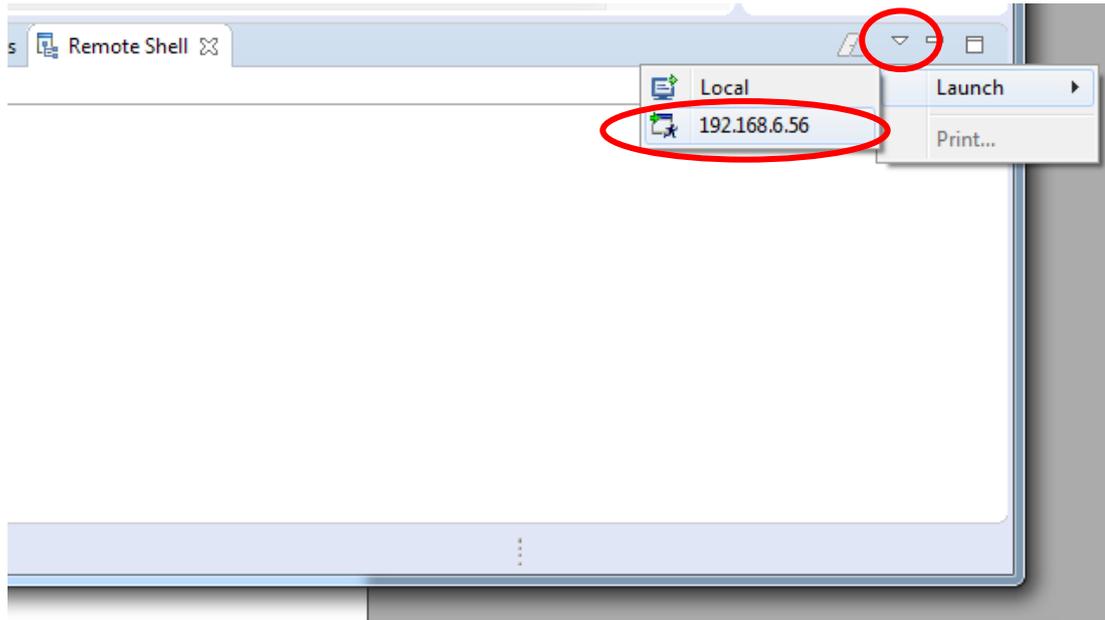
The best way to exit eclipse after having used the debugger is:

- 1) Stop any running debug session
- 2) Stop the debug server
- 3) Save changes in open files
- 4) Disconnect from the Remote System

- 5) Exit eclipse
- 6) Shut down the target (if no longer needed)

5.2.8 Remote Shell

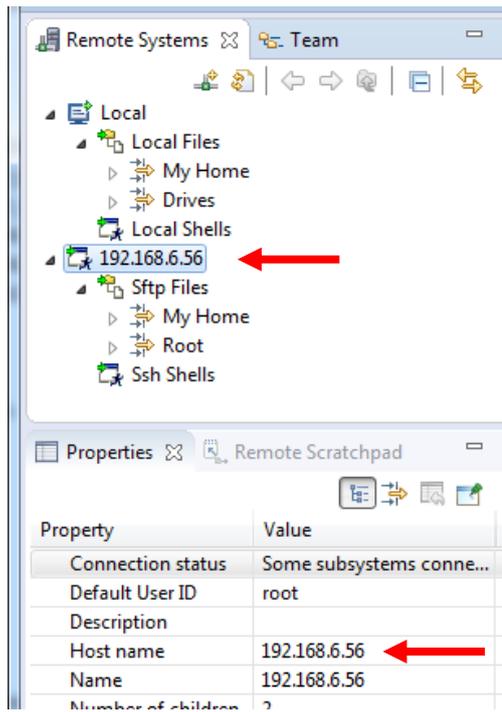
It is always possible to enter commands on the target using either a terminal program or an SSH program. The Remote System Explorer plugin adds another way - the Remote Shell. To enable it go to the Window→Show View menu and then select the Remote Systems->Remote Shell option in the dialog. Use the menu button in the Remote Shell tab and select the target:



5.2.9 Target Changed IP Address

The project that was created in section 5.2.4 was set to use a fixed IP address (192.168.6.56) and if that IP address changes after a reboot of the target then Eclipse cannot connect. This can be fixed either by changing the target from a dynamic to a fixed IP address (not covered by this guide) or by changing the IP number in Eclipse. The IP number is found by selecting the target node in the Remote Systems View.

Click on the current IP number (192.168.6.56) in the field next to Host Name and change it to the new one. Don't change the Name as that will cause problems with the local copy of the remote project.



6 Further Expansion

The links in this section are to give some starting points when searching for ways to extend your Python scripts or to learn more about Python. We have not tested or verified any of the frameworks, packages or plugins and cannot guarantee that they will work.

Beginner: Introductory Books

<https://wiki.python.org/moin/IntroductoryBooks>

Beginner: Examples and Sample Code

<https://wiki.python.org/moin/BeginnersGuide/Examples>

Beginner: Links to more books, tutorials, and lessons

<https://wiki.python.org/moin/BeginnersGuide/Programmers>

Other IDEs?

There are a lot of alternative to using Eclipse, just look at the list on the python wiki: <https://wiki.python.org/moin/IntegratedDevelopmentEnvironments>.

Available GUI package(s)?

There are a lot of options when it comes to GUI development using python:

<https://wiki.python.org/moin/GuiProgramming>

Some of the more frequently mentioned ones are (in no particular order):

- Kivy - <http://kivy.org/>
- PyQt - <https://wiki.python.org/moin/PyQt>
- PyGUI - http://www.cosc.canterbury.ac.nz/greg.ewing/python_gui/
- libavg - <https://www.libavg.de/site/>
- wxPython - <http://wxpython.org/>

Other modules/libs that are good to have?

<https://wiki.python.org/moin/UsefulModules>

7 Troubleshooting

7.1 Cannot get an SSH connection to the target

If you have problems getting an SSH connection to the target follow the steps below. PuTTY will be used as SSH client in these steps. The client can be downloaded from www.putty.org.

1. Get the IP address of the target. Enter the command in a terminal application connected to the target (as described in the Getting Started manual). In the example below the IP address is 192.168.6.50.

```
# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:1A:F1:10:00:9A
          inet addr:192.168.6.50  Bcast:192.168.6.255  Mask:255.255.255.0
          inet6 addr: fe80::21a:f1ff:fe10:9a/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
```

2. Start PuTTY and enter the IP address into the Host Name field and then click the Open button.

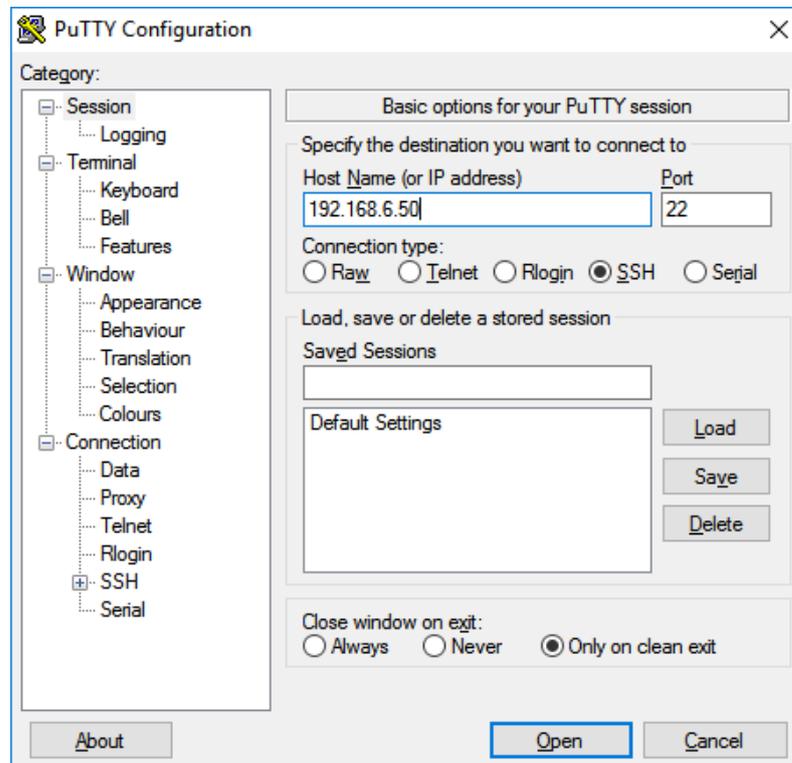


Figure 1 - PuTTY SSH Client

3. You will be asked to enter a username. Enter **root**.
4. You will then be asked to enter a password. Enter **pass**.
5. Now you are logged into the target via an SSH connection. If you get an "Access denied" message please follow the instructions below.

7.1.1 Solve "Access denied" problem

If you get an "Access denied" message when trying to login you have to modify the SSH configuration. By default SSH may not allow the user root to login. The instructions below assume that you have a terminal application connected to the target.

1. Open the configuration file for the SSH server

```
# nano /etc/ssh/sshd_config
```

2. Find the line that starts with #PermitRootLogin and remove the '#' (hash) character. If you cannot find this line just add it to the file (without the hash)

```
PermitRootLogin yes
```

3. Save the file and exit the editor (in `nano` it is Ctrl-X followed by Y and Enter).
4. Restart the SSH server

```
# /etc/init.d/sshd restart
```

5. Now try to connect to the target again.