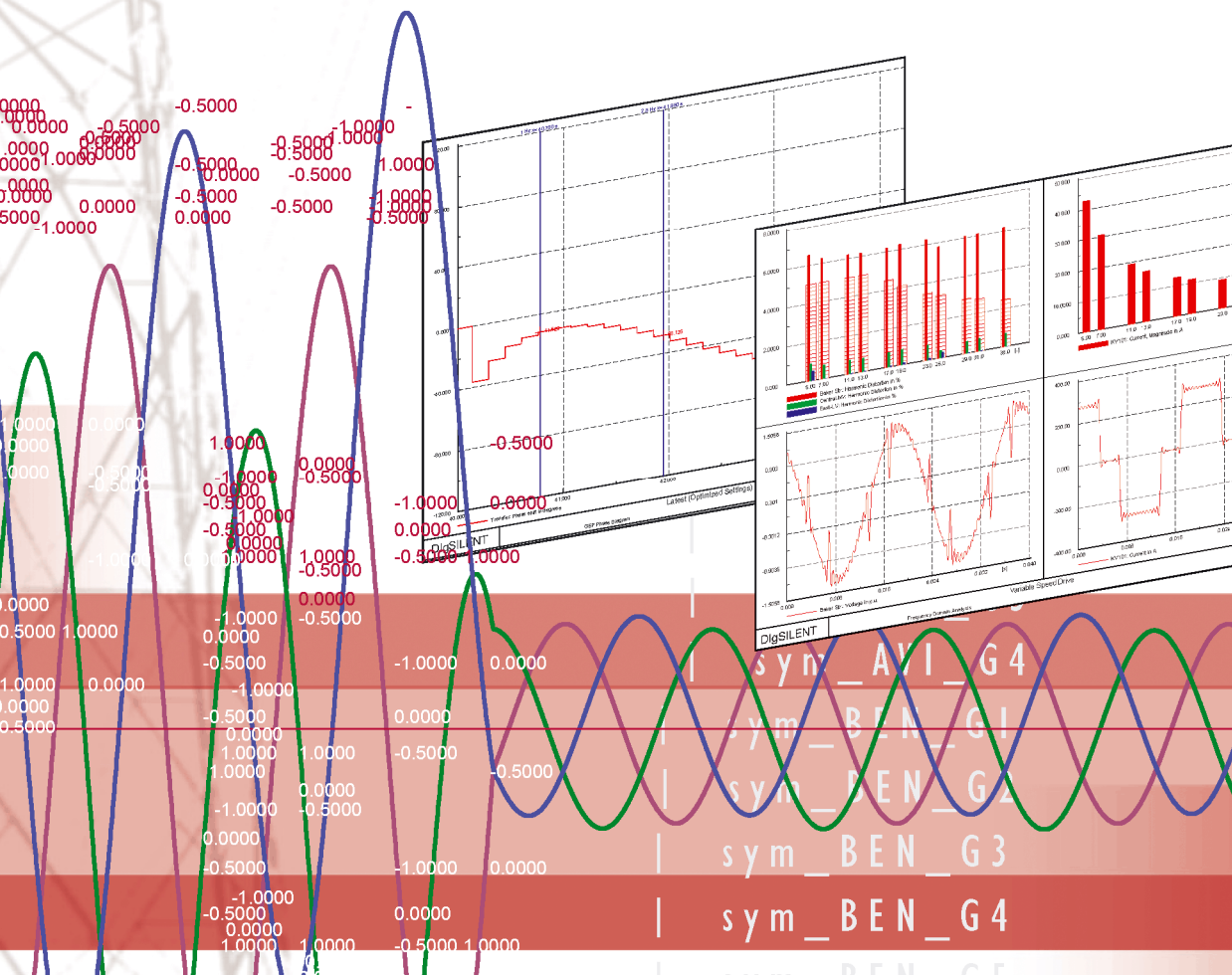


DigSILENT *PowerFactory* Training Material

Python-Debugging



$T_p = 0.9$
41300 ALD---
ARG_3.3-G
41811 AVI---G
41812 AVI---G
41813 AVI---G
41814 AVI---G
42011 BEN16--
42017 BEN16--
42011 BEN16--
42012 BEN16--
42013 BEN16--



DigSILENT GmbH

Heinrich-Hertz-Str. 9
72810 - Gomaringen
Germany

T: +49 7072 9168 00
F: +49 7072 9168 88

<http://www.digsilent.de>
info@digsilent.de

Revision r1288

Copyright ©2013, DigSILENT GmbH. Copyright of this document belongs to DigSILENT GmbH. No part of this document may be reproduced, copied, or transmitted in any form, by any means electronic or mechanical, without the prior written permission of DigSILENT GmbH.

Contents

1 Debugging of the Python Scripts	3
1.1 Adjustment of the Python Script for Debugging	4
2 Example	5

1 Debugging of the Python Scripts

Eclipse is a specialised open source application that is often used for debugging of Python scripts. To debug Python scripts inside of eclipse user has to add a Python module PyDev.

Both Eclipse and PyDev are open source and can be downloaded from official sites

(www.eclipse.org, www.pydev.org)

How to prepare the Eclipse for debugging Python scripts:

- Install Eclipse Standard from www.eclipse.org/downloads/ on your computer
- Open the Eclipse and select “Install New softwer...” located in “Help” menu.
- Add the repository <http://pydev.org/updates> and install PyDev (see Figure 1.1).

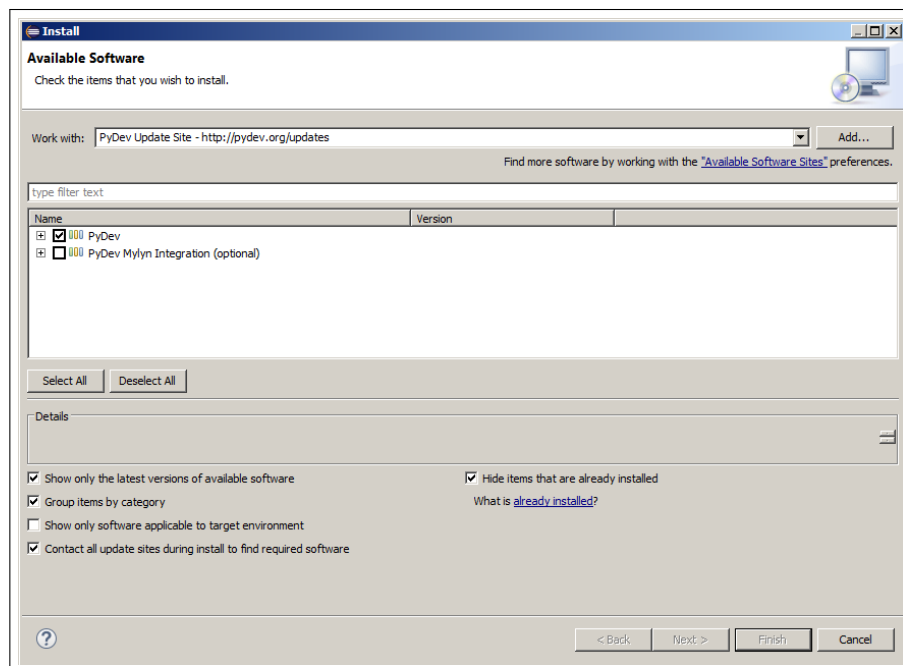


Figure 1.1: Installation of PyDev

How to start debugging a Python script:

- Open Eclipse
- Open perspective and select Debug (see Figure 1.2)

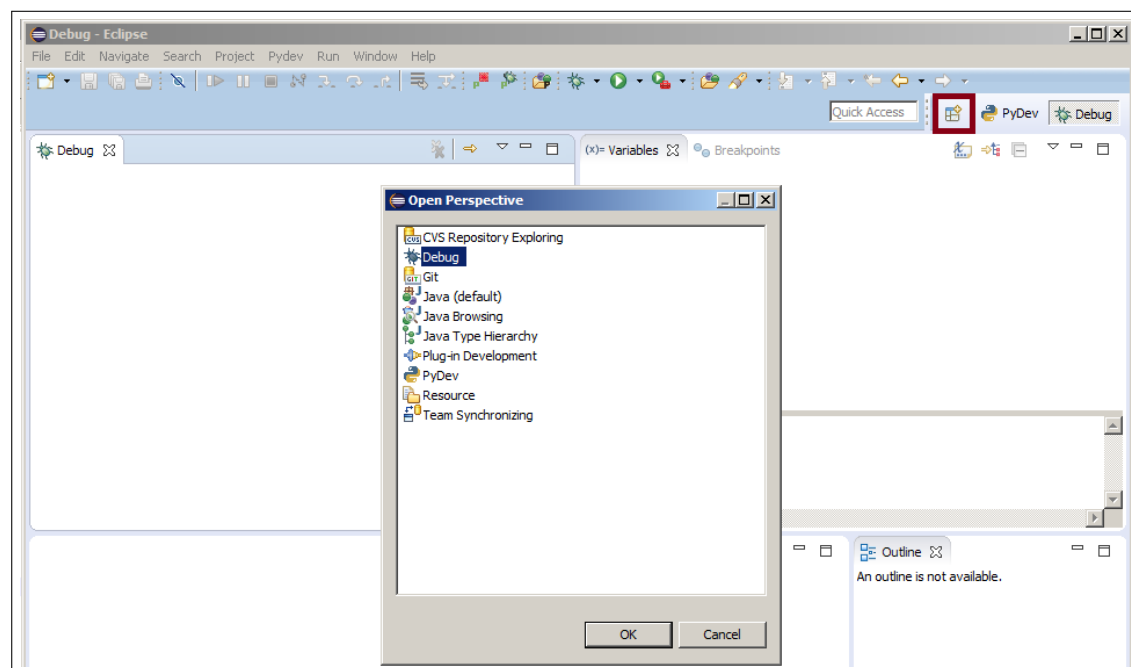


Figure 1.2: Activating of the Debug perspective

- Start the remote debugger server by selecting “Start Debug Server” in the “PyDev” menu, or by clicking on the green button pointed on the Figure 1.3 .

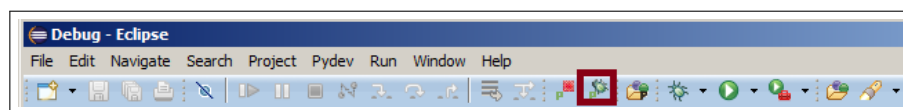


Figure 1.3: Installation of PyDev

- After starting the Remote debugger server a confirming message (e.g Debug Server at port: 5678) will be displayed in Debug Server window.
- When Remote debugger was started go to *PowerFactory* .
- Open the Python script you would like to debug adjust it for debugging and execute it.
- Change to Eclipse and wait for the remote debugger server.

1.1 Adjustment of the Python Script for Debugging

To be able to debug a Python Script via Eclipse you will have to import `pydevd` module that is located inside of Eclipse directory

(e.g. `eclipse\plugins\org.python.pydev_2.8.2.2013090511\pysrc`).

After importing of the module start the debugger by calling `.settrace()` method.

An example of the described code adjustment:

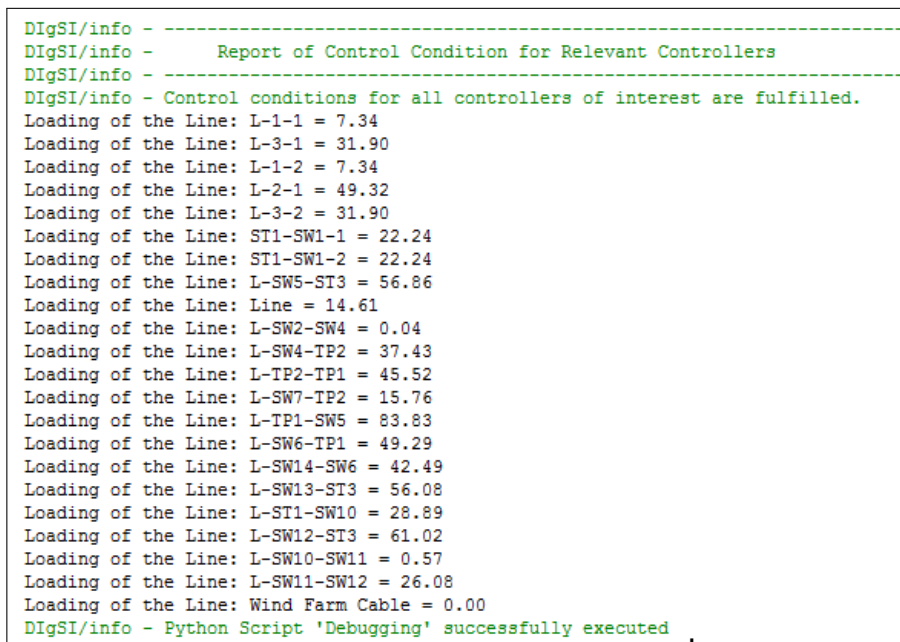
```
import sys
sys.path.append(r"C:\Program Files\eclipse\plugins\
                org.python.pydev_2.8.2.2013090511\pysrc")

import pydevd
#start debugger
pydevd.settrace()
```

2 Example

Following simple script calls all objects of the class `ElmLne`, executes load flow calculation and writes results in *PowerFactory* Output window.

```
import powerfactory
app=powerfactory.GetApplication()
ldf=app.GetFromStudyCase("ComLdf")
Lines=app.GetCalcRelevantObjects("*.ElmLne")
ldf.Execute()
for Line in Lines:
    value=Line.GetAttribute("c:loading")
    name=Line.loc_name
    app.PrintPlain("Loading of the Line: %s = %.2f " %(name,value))
```



```
DIgSI/info - -----
DIgSI/info -      Report of Control Condition for Relevant Controllers
DIgSI/info - -----
DIgSI/info - Control conditions for all controllers of interest are fulfilled.
Loading of the Line: L-1-1 = 7.34
Loading of the Line: L-3-1 = 31.90
Loading of the Line: L-1-2 = 7.34
Loading of the Line: L-2-1 = 49.32
Loading of the Line: L-3-2 = 31.90
Loading of the Line: ST1-SW1-1 = 22.24
Loading of the Line: ST1-SW1-2 = 22.24
Loading of the Line: L-SW5-ST3 = 56.86
Loading of the Line: Line = 14.61
Loading of the Line: L-SW2-SW4 = 0.04
Loading of the Line: L-SW4-TP2 = 37.43
Loading of the Line: L-TP2-TP1 = 45.52
Loading of the Line: L-SW7-TP2 = 15.76
Loading of the Line: L-TP1-SW5 = 83.83
Loading of the Line: L-SW6-TP1 = 49.29
Loading of the Line: L-SW14-SW6 = 42.49
Loading of the Line: L-SW13-ST3 = 56.08
Loading of the Line: L-ST1-SW10 = 28.89
Loading of the Line: L-SW12-ST3 = 61.02
Loading of the Line: L-SW10-SW11 = 0.57
Loading of the Line: L-SW11-SW12 = 26.08
Loading of the Line: Wind Farm Cable = 0.00
DIgSI/info - Python Script 'Debugging' successfully executed .
```

Figure 2.1: *PowerFactory* output window results

Debugging Process:

- Adjust Script for example:

```
import sys
sys.path.append(r"C:\Program Files\eclipse-standard-luna-R-win32-x86_64
    \eclipse\plugins\org.python.pydev_3.7.1.201409021729\pysrc")
import pydevd
#start debugger
pydevd.settrace()
```

- Start Eclipse and activate Debug Server as described in Chapter 1.
- Run the Script and wait until Eclipse and *PowerFactory* are connected (see Figure 2.2).
- Press F8 just to resume your script. By pressing the F8 the results of the script will be presented in *PowerFactory* output window. Script is now finished.

- Set a breakpoint inside of script code field (see Figure 2.3) by just double mouse click on the empty space on the left side of the line number.
- Start the same script once again.
- After reconnection by pressing F6 you will be able to run step by step through the script code. Please feel free to use *variable section* (see Figure 2.3) to follow the variables. Look for the variable *Line* and notes how it changes every time compiler enters the *for* loop.

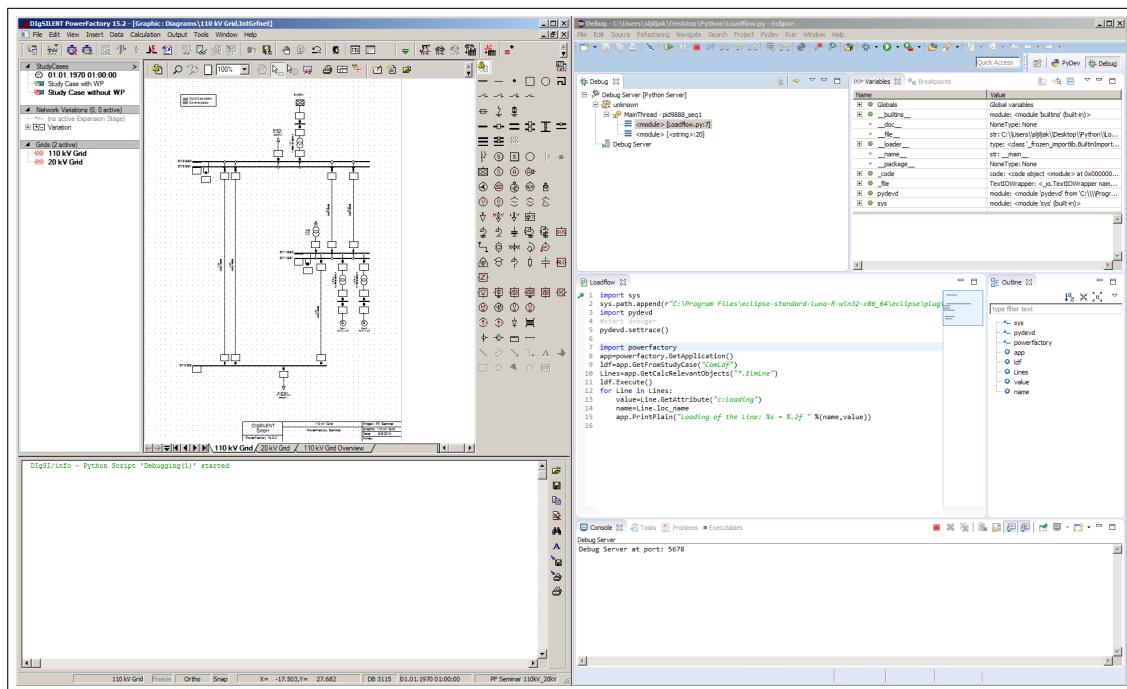


Figure 2.2: PowerFactory and Eclipse connection

Inside of Eclipse Window user can find following sections.

- **1. Variables section:** This section will be empty and as interpreter goes through the lines of the code each variable that it meets will be placed here. Variable representation contains information on name, value, type of the variable and values of all it's attributes.
- **2. Script code:** Presents code of the script. It is possible to follow the interpreter while it passes through the lines of the code. One of the ways to control interpreter steps is by using F6 keyboard button.
- **3. Outline:** Overview over variables that are used in script. By selecting one of them all positions in 2. Section where selected variable is used will be marked.
- **4. Console, Task, Problems, Error Log, ...:** Section contains more subsections sorted in tabs. Each of subsections has its own task and presents results independently.

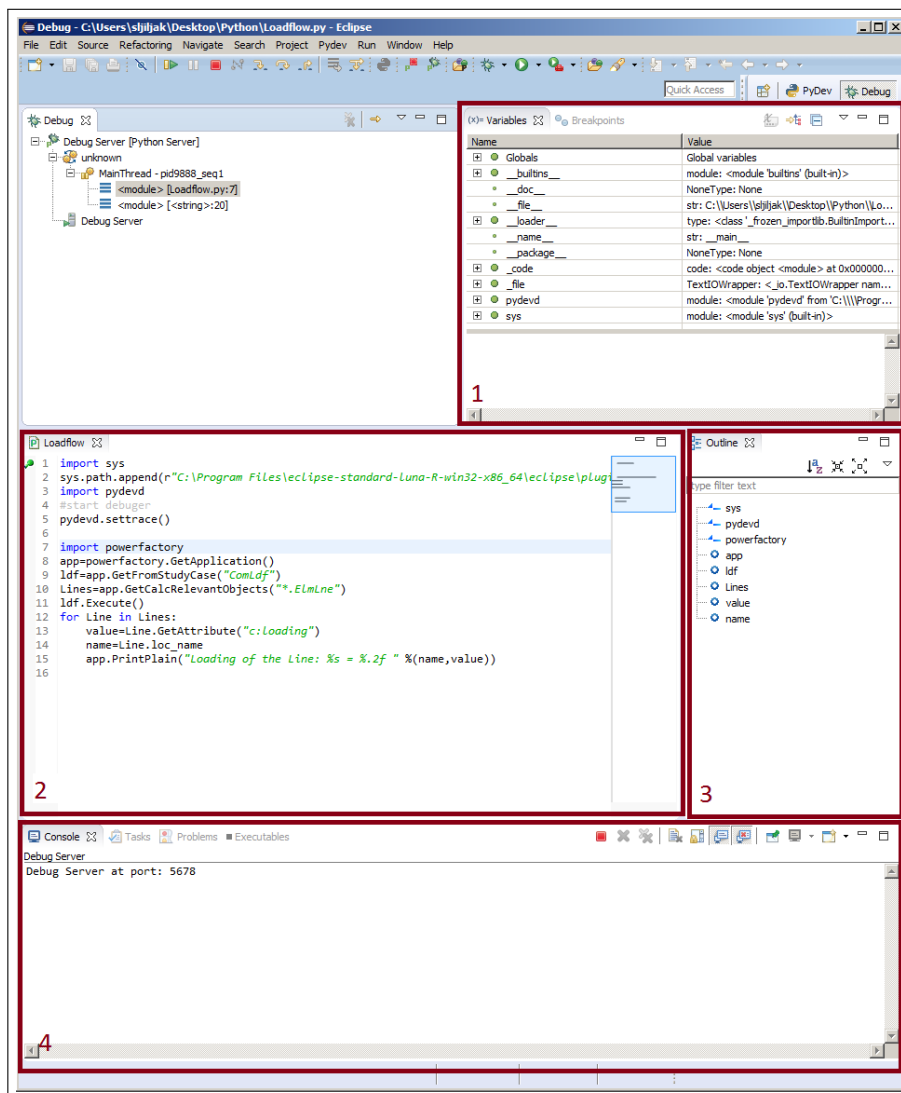


Figure 2.3: Debugging of the Python