Technische Universität Berlin

Faculty of Electrical Engineering and Computer Science

Institute of Software Engineering and Theoretical Computer Science

PySimLib - Python Simulation Library

User Guide

Release date: November 12, 2016

This is a work in progress version

Contents

List of Tables

List of Figures

Listings

1 Introduction

This user guide is aimed to aid the user in setting up the Python Simulation Library (in the following abbreviated with PySimLib), run it, and explain the libraries features. After reading this document the user should be able to simulate models using the PySimLib and be able to use it in Python in order to create own simulation descriptions.
If you have any questions or issues, please feel free to write an e-mail to *a.mehlhase@tu-berlin.de*. In case you find bugs in our software we would be pleased for a report at our repository `https://gitlab.tubit.tu-berlin.de/a.mehlhase/PySimulationLibrary`.

2 Setting up PySimLib

The following will show you how to set up PySimLib, while mainly focusing on Windows platforms. Installation instructions for systems based on Ubuntu are also given. Due to the variety of Linux systems we can't provide installation instructions for all of them. Unfortunately we currently have no access to a Mac computer and can therefore not provide installation instructions, nor do we know if PySimLib works on Mac at all. We would be pleased to hear feedback on that.

At first we will give a brief introduction on how to open a command line terminal or console, which will be necessary for the setup process described here. The first installation step then will be to install Python, as PySimLib is a Python library. In order to simulate models using the PySimLib you will need a simulator that can simulate your models. Note that PySimLib is not a simulator itself but can communicate with several ones and provides a common interface to do so. The second step of installation will be to install a simulator. After these preparations PySimLib can be installed and finally we will describe how to configure PySimLib correctly.

2.1 Command line

The command line terminal or console is a program by which the user can issue commands in text-form. Interaction is realized in the following way: The program writes text to the screen that the user can read. The user writes text commands that are confirmed by the "Enter" or "Return" keys and are read and executed by the program.

Usually operating systems come with a command line program, in order to control settings, files etc. Specific commands may not be executed by any user but by a system administrator, who is allowed to run commands with advanced privileges, such as installing a program or making any other change to the system.

2.1.1 Windows

In Microsoft Windows, the utility cmd.exe is included, also called. In order to open it, click the "Start"-Button or the Windows-Button. In the search widget, type "cmd" without the quotation marks. The program "Command Prompt", "cmd.exe" or something similar should appear. Click it in order to start it. A window similar to that shown in figure 1 should open up. In order to run the console in adminstrator mode, repeat the above process but instead of doing a left click, click right and press "Run as adminstrator".

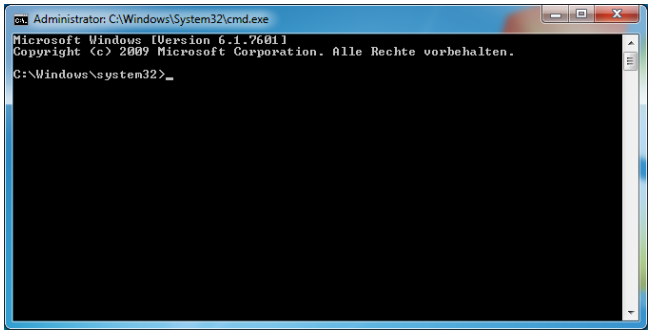

Figure 1: Windows command prompt under Windows 7 in Administrator mode

2.2 Unix-like

Usually a command line program, usually called Terminal in Unix world, is included with the operating system. You should look for it in your program launcher possibly in a category "System tools" or something close.

You can run a command with advanced privileges by entering the "sudo" command before it. Say you want to run the command "help" with advanced privileges enter "sudo help" without quotation marks. This might ask you for the password of your operating systems' user account.

2.3 Installing Python

The first step in setting up PySimLib is installing Python. PySimLib was developed for Python 3, in particular version 3.5. The library might also work with previous versions of Python 3 but surely not with Python 2. By us, PySimLib has only been tested in Python 3.5 and we would recommend you to use it. In general it is more likely that we will support a newer version than an older one than Python 3.5.

In this guide we will expect users to install apart from Python also "pip" (this will be shown later). Pip is Pythons' package management utility and allows to easily install extensions to Python. This is not required, as extensions can be also installed manually but this is rather cumbersome and we don't encourage it. In case you don't want to use pip, you can download the mentioned packages and install them manually. We will not give support on this process.

2.3.1 Windows

Go to Pythons' website, download the installer and run it. You should see something like in figure 2. You don't need to customize the installation, however we encourage you to check the "Add Python 3.5 to PATH" box. This allows you to execute Python from any directory, which is not necessarily required but makes working with Python easier. Don't select this in case you have other versions of Python installed and one of them is added to the PATH environment variable. However, in this guide we will expect that the user added Python 3.5 to the PATH variable. If you did not do that you have to specify the full path to Python whenever you want to use it.

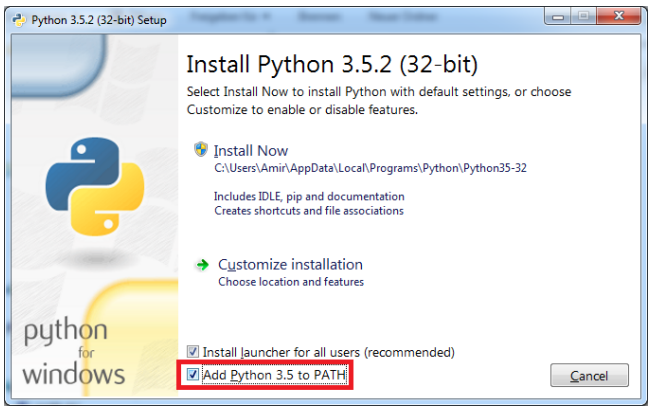

Figure 2: Python 3.5 Windows setup

In case you want to customize the installation, make sure, like in figure 3, that you select at least "pip" and "tcl/tk and IDLE", which is required by PySimLib in order to draw and show plots.

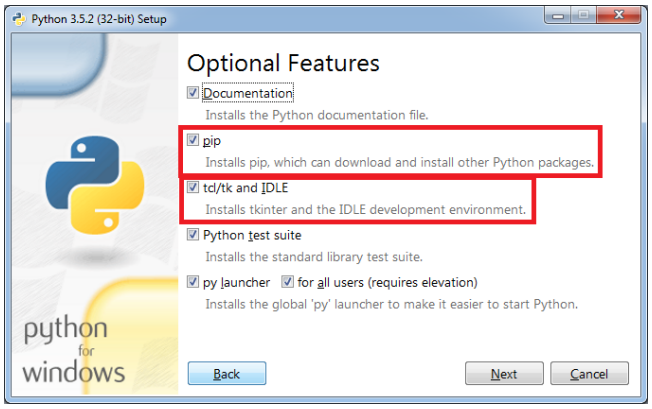

Figure 3: Python 3.5 Windows customized setup

2.3.2 Ubuntu

Python should already be installed on Ubuntu. If not, install it by running the following commands:

```
sudo apt-get install python3
sudo apt-get install python3-pip
sudo apt-get install python3-tk
```

If you are unsure whether Python is installed or not, run the commands anyways. In case Python is already installed, apt-get is going to inform you and nothing is going to be done.

2.4 Installing a Simulator

PySimLib currently supports the simulators listed in table 1.

Dymola
OpenModelica
MATLAB/Simulink

Table 1: List of simulators supported by PySimLib

Please follow the manual of the simulator of your choice in order to set it up correctly. Be sure that it is working properly before continuing with installing PySimLib.

It should be noted that these simulators exist in different versions, which differentiate in terms of features etc. We have listed the constellations of operating system and simulator version under which we tested PySimLib in table 2.

	Windows 7 64 bit	Windows 10 64 bit	Ubuntu 14.10	Ubuntu 16.04
Dymola 2013	✓			
OpenModelica 1.9.6			✓	✓
MATLAB/Simulink R2013a				✓

Table 2: Testing Constellations of Operating Systems and Simulators using PySimLib

In case you want to use Dymola under Windows you need to install the package "pywin32" from `https://sourceforge.net/projects/pywin32/`. PySimLib can also work with a demo version of Dymola but a lot of features will then be unavailable so make sure you have a fully licensed version.

You can install (or remove) additional simulators any time but whenever you do that, you have to reconfigure PySimLib. See subsection 2.6 for configuring PySimLib.

2.5 Installing PySimLib

PySimLib can be installed very simply using pip. In order to determine the correct pip version try entering the following commands:

```
pip3.5
pip3
pip
```

Remember the first one that worked and take it for all following commands where "pipxyz" is used.
To install PySimLib execute the following commands (**you might need advanced privileges**):

```
pipxyz install zmq
pipxyz install numpy
pipxyz install PySimLib
```

2.6 Configuring PySimLib

In order to have PySimLib communicating with simulators, it must be told certain info, for example which simulator is installed, where is it installed etc. This is realized by using a global configuration file. In order to configure PySimLib a normal text editor is sufficient. **Important: Only configure simulators that you have installed on your system!**

The name of the configuration file is `PySimLib.cfg` (beware of the case on Unix-like systems). On Windows systems, the configuration file must be stored in the `%LOCALAPPDATA%` directory, for Unix-like systems under `/.config`. `%LOCALAPPDATA%` is an environment variable that expands to a specific folder in your user directory, so is " " a short cut to the user directory. You can use the command line and the "echo" command in order to expand them to fully qualified paths, like so (similar for Unix-like):

```
echo %LOCALAPPDATA%
```

Note that these folders might be hidden by the file explorer.

2.6.1 Windows

In order to create the configuration file, **in case you haven't created it already as else this might overwrite your configuration file**, enter in console:

```
echo. 2>%LOCALAPPDATA%\PySimLib.cfg
```

To edit the configuration file, type in console:

```
notepad %LOCALAPPDATA%\PySimLib.cfg
```

2.6.2 Unix-like

Call your text editor with the configuration file path as argument. For instance, when using Ubuntus text editor "gedit":

```
gedit ~/.config
```

2.6.3 Configuration file Syntax

The configuration file consists of sections and key-value pairs attached to them. A section is introduced by a text in squared brackets, for instance `[Dymola]` for the "Dymola" section. Following the sections are the key-value pairs, the options of the section. Both key and value are text and they are separated by an equal-sign. Only one key-value pair per line is allowed. **Be aware that the key-value pairs are in general case-sensitive!**

An example of a configuration file with all values for all simulators is shown in listing 2. Recap to only configure simulators that are installed and working on your system. The meaning of the configuration values are for the specific simulators are explained in the following subsections. In general, when asked for a path, the fully qualified path should be preferred, as did in the example for the Dymola and Simulink tools.

```
[Dymola]
PathExe=C:\Program Files (x86)\Dymola 2013 FD01\bin\dymola.exe
StartupDelay=8
PathAlist=C:\Program Files (x86)\Dymola 2013 FD01\bin\alist.exe
SimByExe=true

[OpenModelica]
PathExe=omc

[Simulink]
PathExe=/home/amir/MATLAB/R2013a/bin/matlab
```

Listing 1: An example configuration file

2.6.4 Configuration Values for Dymola

Configuration key	Value type	Value explanation
PathExe	Path	The path to the Dymola executable or a command in order to run it.
StartupDelay	Integer	The duration in seconds that PySimLib should wait for Dymola to start up before trying to communicate with it. This setting is highly dependent on the machine you're running PySimLib on.
PathAlist	Path	The path to the alist executable or a command in order to run it. This utility is shipped with Dymola.
SimByExe	{true, false}	This should in general be set to true. Only set this to false if you're running Dymola with a demo license. Apart from the general drawbacks of Dymolas demo mode, not all features from PySimLib work. Use this if you want to test working with PySimLib and Dymola together. For efficient working with PySimLib in conjuction with Dymola, you should have a license such that you can run the "dymosim" executables, that Dymola generates for models, outside of Dymola.

Table 3: Configuration values for Dymola

2.6.5 Configuration Values for OpenModelica

Configuration key	Value type	Value explanation
PathExe	Path	The path to the OpenModelica compiler executable "omc" or a command in order to run it.

Table 4: Configuration values for OpenModelica

2.6.6 Configuration Values for MATLAB/Simulink

Configuration key	Value type	Value explanation
PathExe	Path	The path to the MATLAB executable or a command in order to run it.

Table 5: Configuration values for MATLAB/Simulink

2.7 Updating PySimLib

You can use Pip again to update PySimLib to its newest version. Simply execute the following command to do so:

```
1 pipxyz install --upgrade PySimLib
```

You may again need advanced privileges.

TODO: weiter