Installation

Before you can install pyCPA, you must set up Python 2.7 and all required python packages (Step 1).

In a second step, pyCPA will be installed in a manual procedure.

The guide below explains both steps. If you have already set up a sophisticated Python environment, you may directly proceed with Step 2.

Step 1: Setting up prerequisites

In this step, we are going to install a compatible version of Python on your system along with additional Python packages required by pyCPA. Please note, that the preferred version is Python 2.7, but most parts of pyCPA are also compatible with Python 3.

- Required Python packages: setuptools, argparse, pygraphviz, matplotlib
- Optional Python packages: numpy, simpy, xlrd

Linux

We assume, Linux users will be familiar with installing any packages from their distribution's repositories. Most likely, Python will already be installed on your system and if not it will typically be installed automatically (as a dependency) when installing the additional Python packages.

Before proceeding, you might want to check the status of your Python installation, i.e. what version is installed (if at all) and what Python packages are already available, using the following commands:

```
$ python --version
$ pydoc modules
```

If you use Ubuntu/Debian, you can install missing packages using the following commands:

```
$ sudo apt-get install python-setuptools python-argparse python-pygraphviz python-
matplotlib python-numpy python-simpy python-xlrd
```

If your distribution does not provide similar packages, you can use `pip` to install additional python packages.

```
$ pip install simpy
```

Windows

We recommend to use Python(x,y), which includes a comprehensive set of scientific Python libraries and tools as well as related documentation. Most of the required Python modules such as matplotlib are already included, but some (SimPy) have to be installed by hand if required. It is recommended to uninstall any other Python distribution before installing Python(x,y).

In order to safe disk space, you may choose the recommended setting during installation and additionally check xlrd and pygraphviz. Python(x,y) comes with several interactive consoles (based on IPython), editors and applications. For your first hands-on experience, we recommend using `Spyder` as an IDE. You can also run a command prompt via the Python(x,y) icon on the Desktop and choosing `IPython (sh)` as an interactive console.

You can install missing packages using the recommended installer for python packages, `pip`, from the command prompt. `pip` is included in Python(x,y).

```
$ python -m pip install simpy
```

For a detailed documentation regarding package installation you may consult the official website.

Step 2: Downloading and setting up pyCPA

For downloading the pyCPA source code, you have two options:

A. **Easisest**: Download and extract the latest pyCPA release from Bitbucket.
B. For experts: If you are familiar with mercurial, you can alternatively clone the repository using tortoise-hg or installing mercurial using your Linux distribution's tools and running the following command:

```
$ hg clone https://bitbucket.org/pycpa/pycpa/
```

Depending on how you want to use pyCPA, there are two ways of making pyCPA available to your python installation:

A. **Easiest**: Install pyCPA into your python installation using the command prompt from the pyCPA directory. This is for people who just want to use pyCPA as it comes.

```
$ python setup.py install
```

B. For expert: Leave pyCPA where it is and tell Python to use the module in place. This is for people who want to modify pyCPA or use different versions in parallel. You achieve this by setting the PYTHONPATH variable to the pyCPA directory. For command line users, this is done as follows:

```
$ export PYTHONPATH="/path/to/pyCPA:$PYTHONPATH"
```

Note that you must *NOT* specify the subdirectory `pycpa` within the pyCPA directory. If you prefer using an IDE, please refer to Using an IDE: PyDev.

Step 3: Testing and using pyCPA

Congratulations, you have installed pyCPA!

In order to test pyCPA, you may want to run the examples which are provided with the distribution. The quickest way to do this is to run the following on the command prompt (e.g. `IPython (sh)` on Windows):

```
$ python /path/to/pycpa/examples/spp_test.py
```

If you want to know what this examples does and how it works checkout the Static Priority Preemtive Example.

Depending on your personal preferences, you may also use an IDE of which we give a more detailed account in the following sections.

Using an IDE: Spyder (Windows)

Spyder is installed with Python(x,y). Simply open one of the example files (e.g. spp_test.py) and click the `Run` button.

Using an IDE: PyDev

You may also use Eclipse with PyDev as IDE, which can be installed by the following steps:

1. Make sure that you have installed Python 2.7 *BEFORE* you install Eclipse.
2. Download from http://www.eclipse.org/downloads/eclipse-packages/ the current Eclipse release for Windows 32 bit (!). Extract the zip-file, execute `eclipse.exe` and follow the installation instructions.
3. Open Eclipse and specify a workspace. If you open a workspace for the first time, you will have to close the Welcome tab, before proceeding to your workspace.
4. Select the menu item `Help -> Install New Software`, search for the site http://pydev.org/updates. Select and install the item “PyDev” which will be displayed in the list of available software.

Now, you can set up a pyCPA project as follows:

1. Open the PyDev-Perspective by selecting in the main menu `Window -> Open Perspective -> Other -> PyDev`
2. Select in the main menu `File -> New -> PyDev Project`.
3. In the PyDev-Project Window specify a project name; the project will be saved to your workspace unless specified otherwise.
4. Choose the project type “Python” and select the 2.7 interpreter version.
5. Click on “Please configure an interpreter before proceeding”.
    1. Select `Manual Config` in the pop-up window.
    2. In the settings for the Python interpreter click `New...` and specify an interpreter name, e.g. Python27, and the path to the interpreter executable (e.g. `C:\myPathToPython\python.exe`). In the appearing pop-up window select all options.
    3. In the tab `Libraries`, select `New Folder` and specify the path to the pyCPA-folder (e.g. `C:\MyPathTo\pycpa`).
    4. Close the preferences window by clicking ok.
6. Back in the PyDev-Project Window, click `add project directory to PYTHONPATH` and then the button `Finish`.
7. You may now add a Python file to your project (right-click on your project in the PyDev Package Explorer -> New... -> File) and write a Python program (e.g. test.py) which uses pyCPA.
8. To run test.py, right-click on `test.py` and select `Run as -> Python Run`. If you want to modify your run settings in order to e.g. specify arguments, select `Run as -> Run Configurations` and adapt the settings as needed before clicking `Run` in the Run Configurations Window.
9. You may also try out the examples which are provided with pyCPA such as the Static Priority Preemtive Example.