# Python model adapters within FEWS

When running Python model adapters within FEWS, Python should be installed on your system and the dependencies (Python packages) of your adapter as well. The proposed way to do this is to:

- work in virtual environment
- organize model adapter code as python package
- make sure that all dependencies are listed in the `setup.py` of the package, to allow straightforward installation

## Working with virtual environment

A Python virtual environment, put simply, is an isolated working copy of Python which allows you to work on a specific project without worry of affecting other projects. You can have multiple virtual environments on your system, all having their own specific (versions of) packages available. The virtualenv package provides the basic needs, but virtualenvwrapper makes life a bit easier.

More information on installation and use can be found in the documentation:

- virtualenv

install with:

```
pip install virtualenv
```

- virtualenvwrapper

install with:

```
pip install virtualenvwrapper
```

Use **virtualenvwrapper-win** on a windows system (https://pypi.python.org/pypi/virtualenvwrapper-win). Install with:

```
pip install virtualenvwrapper-win
```

## Create Python package

The python scaffold package (https://pypi.python.org/pypi/Scaffold/0.1.3) helps to create a proper directory structure for your package, following the structure as explained in http://www.scotttorborg.com/python-packaging/minimal.html

Each project you scaffold will create the following directory structure:

```
/[projectname]/
/[projectname]/setup.py
/[projectname]/bin
/[projectname]/docs
/[projectname]/[projectname]
/[projectname]/[projectname]/__init__.py
/[projectname]/tests
/[projectname]/tests/__init__.py
/[projectname]/tests/[projectname]_tests.py
```

Both `setup.py` and `[projectname]_tests.py` are set up automatically to reference your project name as a module. The rest is up to you! You should put your code file(s) in the

`/[projectname]/[projectname]/`

directory and your tests in

`/[projectname]/tests/`

### Installing Scaffold

You can view the scaffold package on PyPi here: http://pypi.python.org/pypi/Scaffold/0.1.3

To install, simply use

```
pip install scaffold
```

## Running Scaffold

You can run scaffold as a python module:

```
python -m scaffold -p "projectname" [-d {base directory}]
```

By default, the project is created as a child of the current working directory. The optional argument

```
-d {base directory}
```

allows to create the project somewhere else.

## Picking A Name

Python module/package names should generally follow the following constraints:

- All lowercase
- Unique on pypi, even if you don't want to make your package publicly available (you might want to specify it privately as a dependency later)
- Underscore-separated or no word separators at all (don't use hyphens)

## Installing the package

Now we can install the package locally (for use on our system), with:

```
python setup.py install
```

For packages under development, we can also install the package with a symlink, so that changes to the source files will be immediately available to other users of the package on our system:

```
python setup.py develop
```

## Adjusting setup.py

The `setup.py` file is used for meta information of the package and for the requirements. The initial `setup.py` file, as created by scaffold, looks like this:

```
try:
    from setuptools import setup
except ImportError:
    from distutils.core import setup

config = {
    'description': 'My Project',
    'author': 'My Name',
    'url': 'URL to get it at.',
    'download_url': 'Where to download it.',
    'author_email': 'My email.',
    'version': '0.1',
    'install_requires': ['nose'],
    'packages': ['projectname'],
    'scripts': [],
    'name': 'projectname'
}

setup(**config)
```

The config dictionary has to be adjusted in order to describe the actual information on the package and the author(s). The field `install_requires` should include a list with the requirements of the package. All packages listed in there will be installed when installing your package with:

```
python setup.py develop
```