Testing and documentation

Simon Pfreunschuh

September 16, 2020

1 Getting and understanding the `weather_app` code.

- Clone the repository `https://github.com/simonpf/weather_app`, which provides an interface to the current SMHI weather forecast. The project folder contains the following subfolders and files:

```
weather_app/
├── weather_app/
│   ├── __init__.py
│   └── api.py
└── test
    ├── test_api.py
    └── test_weather_app.py
```

- Have a look at the code in the `weather_app/api.py` source flie. It should be sufficiently well documented to figure out how to use it.
- Use the `weather_app.api` module to plot predicted temperature and precipitation.

2 Test driven development

- Install the pytest package using `pip`:

```
pip install pytest
```

- Run `pytest` on the tests in the `test` subfolder. You will see that one of the tests fail.
- Implement the required functionality to make the test pass.

3 Turning your code into a package

- Add the required files to turn you code into a package. Use `weather_app_<your_name>` as the package name to avoid name clashes when uploading the package.
- Add the required non-standard library packages to the install\$_{\text{dependencies}}$ in the `setup.py`.
- Install your package locally using `pip`

4 Uploading your code into a package

- Build wheels for you package following the instruction from the lecture
- Create an account on `test.pypi.org`
- Upload your binary distribution package to the `test.pypi` index.

5 Testing your package in a virtual environment

- Create a new folder, change into it and create a virtual environment
- Install your uploaded `weather_app` package from PyPI. Note that you have to specify the index explicitly `https://test.pypi.org`
- Use your package to plot the temperature forecast for the next 24 hours.