nestpy: How to update

Greg Rischbieter – July 2025

Introduction and Overview:

Unfortunately, updating nestpy is really a case-by-case basis. But this document will serve to provide some base guidance that will work in the vast majority of situations. Any differences here will likely arise from the complexity of the changes to NEST, and/or changes to github's available workflow testing architecture.

I will document how I update nestpy. While there may be easier ways to do this, this is what I've done reliably for several years. This documentation assumes that you are trying to sync NESTCollaboration/nestpy to the current master version of NESTCollaboration/nest. If you are trying to sync with a specific tag/commit from NEST that isn't the current master branch, the change will likely come in the "git submodel update" commands below, but I do not know those readily at the time of writing.

Getting the Code and Preparing for the Update:

To begin, you want to be working in an environment with Python3, preferably 3.6 or greater. I've used 3.8.5 for a long time.

Clone nestpy:

```
git clone git@github.com:NESTCollaboration/nestpy.git
```

I like to make a copy of this original unchanged version where I test updating nestpy:

```
cp -r nestpy testpy; cd testpy
```

Then, inside of this "testpy" directory, we can try building the code with the modern nest version and see where it breaks.

```
git submodule update --init --recursive #preps the submodules
git submodule update --remote --merge #updates the submodels with the
modern branches
pip install .
```

This will almost always fail to build. But the output errors are helpful (albeit very cryptic).

Dealing with Changes to NEST Functions:

Often, the number of input arguments to a NEST function will change, meaning that we need to update the bindings. Not every function in NEST is py-bound, and not every bound function has explicit argument definitions, which makes this easier.

Get a list of differences between the modern NEST functions and the previous versions nestpy used:

```
diff lib/nest/include/NEST/NEST.hh
../nestpy/lib/nest/include/NEST/NEST.hh > nest.diff
vim nest.diff #or emacs nest.diff if you like that more
```

You need to check the lines that have changes to how functions are called, and update the relevant bindings in “src/nestpy/bindings.cpp” and in the two python scripts under “tests”.
For example, the 2025 nestpy update saw the GetYields(...) function lose the boolean input argument for “oldERmodel”. So that argument needed to be removed from the python bindings. Meanwhile, SetDriftVelocity(...) received an argument for detector pressure, but doesn’t have its inputs specifically listed in bindings.cpp, so no changes were needed there. However, SetDriftVelocity *is* called in one of the test scripts in “tests”, so it had to be updated with a sensible value.

You will also want to check other files in NEST that have py-bound functions, such as TestSpectra:

```
diff lib/nest/include/NEST/TestSpectra.hh
../nestpy/lib/nest/include/NEST/TestSpectra.hh > testspectra.diff
```

You can check that you have made all the necessary changes by retrying to build the code:

```
pip install .
```

And then try running the test scripts:

```
python tests/core_nest_tests.py
python tests/example_tests.py
```

Addressing CMake Errors (not NEST related):

I will not claim to be savvy with CMake. I usually need to spend time googling the errors I see or ask ChatGPT for help. Fortunately, there’s only been one case in my experience with this. All of the CMake compile commands are included at line 45 of "setup.py". The last time we had an issue here, all that was needed was another flag added to the list of CMake commands.

Committing to github:

If you've been following the way I've previously done this, you've been working in a "testpy" directory, as opposed to the "nestpy" directory you initially cloned. I do this so that none of the build files accidentally get pushed to github.

However, you must go back to your nestpy directory and copy over all of the changed files from "testpy" to "nestpy". These files will likely include (but may differ on a case-by-case basis):

- src/nestpy/bindings.cpp
- setup.py
- tests/core_nest_tests.py
- tests/example_tests.py

Once all of the changed files have been copied over, you want to **update the HISTORY.md file** so that there are at least *some* notes about what changed.
If you plan on incrementing the nestpy version number (which you should), you'll also want to go to "setup.py" and make sure that you increment the "version" flag upwards to whatever you want the next version to be.

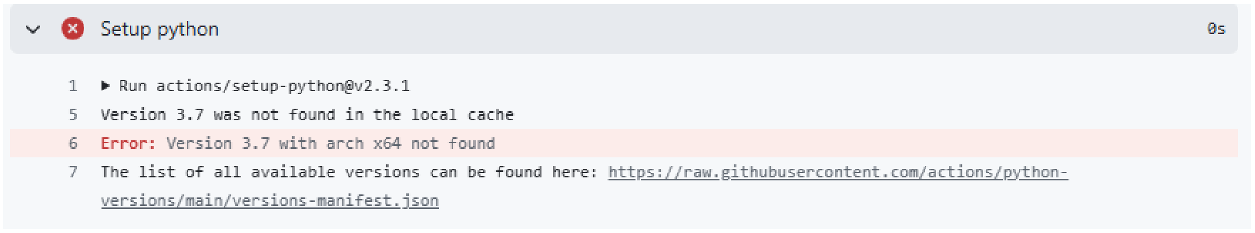
Now, you're ready to commit.

```
git checkout -b <new_branch_name> #switch to a new branch
git submodule update --init --recursive #preps the submodules
git submodule update --remote --merge #updates the submods to modern
branches
git add .
git commit -m "message detailing the commit"
git push origin master "new_branch_name"
```

You can now go to github.com/NESTCollaboration/nestpy to create a Pull Request with this new commit/branch. And then tag that version as a new release on github from the web browser.

Updating the Github Workflows if Tests fail:

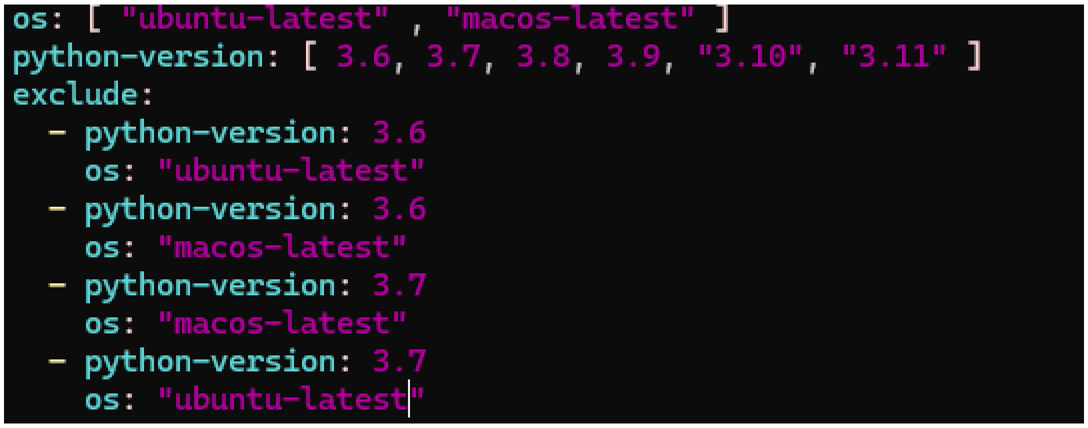
It's good practice to only merge branches that pass all of the workflow tests. However, twice in the past we've seen checks immediately fail with the error:



```
Setup python                                                                 0s

▶ Run actions/setup-python@v2.3.1
Version 3.7 was not found in the local cache
Error: Version 3.7 with arch x64 not found
The list of all available versions can be found here: https://raw.githubusercontent.com/actions/python-
versions/main/versions-manifest.json
```

This is because github occasionally changes which architectures are testable with a free account. In the example above, Python 3.7 on Ubuntu was no longer available. To fix this, an exclusion must be added to the ".github/workflows/test_package.yml" file, such as:



```
os: [ "ubuntu-latest" , "macos-latest" ]
python-version: [ 3.6, 3.7, 3.8, 3.9, "3.10", "3.11" ]
exclude:
  - python-version: 3.6
    os: "ubuntu-latest"
  - python-version: 3.6
    os: "macos-latest"
  - python-version: 3.7
    os: "macos-latest"
  - python-version: 3.7
    os: "ubuntu-latest"
```

You can add/exclude more architectures by editing those lines of that test_package script.

Commit those changes with:

```
git add .github/workflows/test_package.yml
git commit -m "edited workflows"
git push origin <branch_name>
```

The tests for your pull request will immediately restart with the new commit.