# Working with packages and environments in Spyder

Jump to bottom

C.A.M. Gerlach edited this page on Sep 12, 2018 · 18 revisions

While relatively straightforward once you're familiar with it, the interaction between Spyder and other packages and environments can sometimes be confusing for first-time users. Spyder 4 will make this process much easier with integrated, interactive GUI package and environment management, but in the meantime—particularly with the changes released in Spyder 3.3.0—we'd like to clarify how that relationship works.

## The most common problem: Using newly-installed packages inside Spyder

After installing a package (let's call it `foo`) outside Spyder, users may encounter an error trying to import it inside the IDE:

```
In [1]: import foo
Traceback (most recent call last):

  File "<ipython-input-4-7f58dd7fb72e>", line 1, in <module>
    import foo

ModuleNotFoundError: No module named 'foo'
```

This happens because `foo` was installed (with either `conda` or `pip`) in a different conda or venv/virtualenv environment than the one in which Spyder is currently running.

To confirm this is the problem, you need to:

1. Activate the environment (*e.g.* `myenv`) in which you installed the package `foo` (*e.g.* with `source activate myenv` on macOS/Linux or `activate myenv` on Windows, `workon myenv` for virtualenv/venv, *etc*)
2. Start a Python interpreter there by running the command `python`.
3. Run the following command inside the Python interpreter:

```
import sys; sys.executable
```

4. Start Spyder and run the same command shown in Step 3 in a Console.
5. If the resulting paths are the same, then Spyder and the package are in the same environment, and `import foo` shouldn't produce an error (or else there is likely an unrelated issue with your installation).
6. If the resulting paths are different, then you have three choices:
   - Activate the environment in which Spyder is installed and install your package on it (see next section). If you try to install future packages in another environment (like `myenv`), you'll get the same `ModuleNotFoundError`.
   - Install Spyder into the existing `myenv` environment, or any other you'd like to work in, and run it from there (see following section). This is a little simpler than the third option and has the same effect, but more overhead and is less flexible.
   - Install just the `spyder-kernels` package into the `myenv` environment, and set your Python interpreter path in Spyder's Preferences to point to `myenv`'s Python executable (see the final section. This requires Spyder >=3.3.0 and one more initial step, but requires less maintenance in the long run and avoids duplicate Spyder installs.

## Installing packages into the same environment as Spyder

Spyder is a Python package just like any other you may be used to, and so you can `import` any package within its Console or Editor as you could from a regular Python or IPython terminal launched in Spyder's environment:

- If Spyder is installed with Anaconda (as we recommend) and launched via a shortcut, from Anaconda Navigator or from Anaconda Prompt without modifying anything, this will be the default `base` Anaconda environment.
- If Spyder is installed via `pip` (experts only) and not into a `virtualenv` / `venv`, this will usually be whatever Python installation `pip` itself belongs to.
- If you use a system package manager (`apt-get`, `dnf`, `emerge`, etc) to install Spyder, this will typically be your system Python and its library of packages.
- If you installed Spyder into a specific environment (`conda-env` or `venv`), or it came with a pre-configured one (like those for Keras or TensorFlow) and launched it from there, it will only have access to packages from that environment.

Therefore, if you'd like to use a package with your existing Spyder install (*e.g.* `import`'ing it into your scripts, packages or a Spyder IPython console), the simplest way to do so is to install the package into the same environment in which you installed Spyder, typically by the same means you installed Spyder (`conda`, `pip`, package manager, etc). However, if you're installing packages with `pip`, `conda-forge`, Github, or custom channels, working on multiple major projects at once, using prebuilt environments, or otherwise have more sophisticated needs, you'll likely want to use one or more separate environments for your packages. If so, the next section explains how.

## Working with other environments and Python installations

If you have an existing, pre-configured environment (such as for Keras or TensorFlow), are managing multiple environments (such as for development or testing purposes), or even would like to work within a totally separate Python installation as that in which Spyder is installed (such as a system-installed Spyder with a separate Anaconda installation, or vice-versa), you have two main options:

### The naive approach

To use Spyder with another environment, the most straightforward way is to just install `spyder` into the environment from which you'd like to use the packages in, and run it from there. This works with all Spyder versions and should require no extra configuration once the IDE is installed; however, it results in multiple installations to manage and isn't as flexible or configurable as the alternative. Therefore, when dealing with multiple environments, we recommend the modular approach.

### The modular approach

Starting with Spyder **3.3.1**, you can install the modular `spyder-kernels` package into any Python environment (`conda` environment, `virtualenv/venv`, system Python, WinPython, *etc*) in which you wish to work, and then change the Python interpreter used by Spyder on its IPython consoles to point to the Python executable of that environment.

This takes a small amount of preparation and configuration, but is much "lighter" and quicker than a full Spyder installation into that environment, avoids dependency conflicts, and opens up new workflow possibilities.

To achieve this, follow these steps:

1. Activate the environment (*e.g.* `myenv`) in which you'd like to work (*e.g.* with `source activate myenv` on macOS/Linux or `activate myenv` on Windows, `workon myenv` for virtualenv/venv, *etc*)

2. Install the `spyder-kernels` package there, with the command:

   ```
   conda install spyder-kernels=0.*
   ```

   if using conda/Anaconda, or

```
pip install spyder-kernels==0.*
```

if using pip/virtualenv.

3. After installing via either method, run the following command inside the same environment:

   ```
   python -c "import sys; print(sys.executable)"
   ```

   and copy the path returned by that command (it should end in `python`, `pythonw`, `python.exe` or `pythonw.exe`, depending on your operating system).

4. Deactivate that environment, activate the one in which Spyder is installed (if you've installed it in its own environment) and start Spyder as you normally would.

5. After Spyder has started, navigate to `Preferences > Python Interpreter > Use the following interpreter` and paste the path from Step 3 into the text box.

6. Start a new IPython console. All packages installed in your `myenv` environment should be available there.

## Be Social 👍

Connect with **Spyder** through our social media channels and stay up to date with current developments!

- Google+ Community
- Google+ Page
- Facebook Page
- Twitter
- Youtube
- Gitter Chat Room

## Clone this wiki locally

```
https://github.com/spyder-ide/spyder.wiki.git
```