13th Jan 2022 | 8 minutes read

The Python Requirements File and How to Create it

Xavier Rigoulet

Python Basics

Python requirements files are a great way to keep track of the Python modules. It is a simple text file that saves a list of the modules and packages required by your project. By creating a Python `requirements.txt` file, you save yourself the hassle of having to track down and install all of the required modules manually.

In this article, we will learn how to create Python requirements files along with the best practices and the benefits of using them. It's often used together with virtual environments in Python projects, but that is outside the scope of this article.

Before we go into the details on how to create a Python requirements file, make sure to check our list of Python IDEs and code editors HERE if you are serious about LEARNING PYTHON. It makes your life easier and increases your productivity.

Using a Python requirements file comes with a lot of benefits.

First, it allows you to keep track of the Python modules and packages used by your project. It simplifies the installation of all of the required modules on any computer without having to search through online documentation or Python package archives. It is used to install all of the dependencies on another computer so that they are compatible with one another.

Second, it makes it easy to share your project with others. They install the same Python modules you have listed in your requirements file and run your project without any problems.

Third, if you ever need to update or add a Python module to your project, you simply update the requirements file rather than having to search through all of your code for every reference to the old module.

Next, let's learn how to create one!

How to Create a Python Requirements File

It is just a text file with all of the required modules for your Python project. Start by navigating to your Python project directory and creating a new `.txt` document.

Make sure it is named `requirements.txt`, and then save it in the same directory as your `.py` files for this project.

There's not much else we need to do to create a Python requirements file at this point, but we will cover how to install specific packages manually in the terminal.

You can also generate a Python `requirements.txt` file directly from the command line with:

Code

```
pip freeze > requirements.txt
```

`pip freeze` outputs a list of all installed Python modules with their versions.

Adding Modules to Your Python Requirements File

Now that we have created a Python requirements file, it's time to start adding some modules! The first step is to open the text document and add the names of the modules you would like to install.

For example, if I want to install the `tensorflow` library into my project, I type in `tensorflow` on its own line along with the required version. Let's type an example in your newly created Python `requirements.txt` file:

Code

```
tensorflow==2.3.1
uvicorn==0.12.2
fastapi==0.63.0
```

Once you add all of the modules you need, save the document and exit!

Installing Python Packages From a Requirements File

Now that our Python requirements file is all set up, let's take a look at how to install packages from it. To do this, we will use the pip package manager.

The `pip` utility is used to install, upgrade, and uninstall Python packages. It is also used to manage Python virtual environments and more.

To start, open up a terminal or a command prompt and navigate to the directory of your Python project. Once you are there, type the following command:

Code

```
pip install -r requirements.txt
```

This installs all of the modules listed in our Python requirements file into our project environment.

`Output:`

Code

```
Successfully installed absl-py-1.0.0 astunparse-1.6.3 cachetools-4.2.4
certifi-2021.10.8 charset-normalizer-2.0.9 click-7.1.2 fastapi-0.63.0 gast-
0.3.3 google-auth-2.3.3 google-auth-oauthlib-0.4.6 google-pasta-0.2.0 grpcio-
1.42.0 h11-0.12.0 h5py-2.10.0 idna-3.3 importlib-metadata-4.8.2 keras-
preprocessing-1.1.2 markdown-3.3.6 numpy-1.18.5 oauthlib-3.1.1 opt-einsum-
3.3.0 protobuf-3.19.1 pyasn1-0.4.8 pyasn1-modules-0.2.8 pydantic-1.8.2
requests-2.26.0 requests-oauthlib-1.3.0 rsa-4.8 six-1.16.0 starlette-0.13.6
tensorboard-2.7.0 tensorboard-data-server-0.6.1 tensorboard-plugin-wit-1.8.0
tensorflow-2.3.1 tensorflow-estimator-2.3.0 termcolor-1.1.0 typing-
extensions-4.0.1 urllib3-1.26.7 uvicorn-0.12.2 werkzeug-2.0.2 wheel-0.37.0
wrapt-1.13.3 zipp-3.6.0
```

It is a good practice to set a new environment before installing packages with your Python requirements file.

If you ever need to remove a module for any reason, use the same command but with the `uninstall` keyword instead of `install`. Also, use upgrade instead of

`install` to update previously installed Python packages.

As mentioned before, use the `pip freeze` command to output a list of the Python modules installed in your environment.

How to Maintain a Python Requirements File

If you created a Python `requirements.txt` file at one point but have failed to maintain it for some reason, fear not! You can do it as follows.

Step 1: Output a list of outdated packages with `pip list --outdated` .

`Output:`

Code

```
Package              Version Latest
Type
____________________ _______ ______ ____
_
click                7.1.2   8.0.3
wheel
fastapi              0.63.0  0.70.0
wheel
gast                 0.3.3   0.5.3
wheel
h5py                 2.10.0  3.6.0
wheel
numpy                1.18.5  1.21.4
wheel
pip                  20.0.2  21.3.1
wheel
setuptools           44.0.0  59.5.0
wheel
starlette            0.13.6  0.17.1
wheel
tensorflow           2.3.1   2.7.0
wheel
tensorflow-estimator 2.3.0   2.7.0
wheel
uvicorn              0.12.2  0.15.0 wheel
```

Step 2: Upgrade the required package with `pip install -U PackageName` .

As an example, let's update `fastapi` :

Code

```
pip install -U fastapi
```

`Output:`

Code

```
Successfully installed anyio-3.4.0 fastapi-0.70.0 sniffio-1.2.0 starlette-0.16.0
```

It is also possible to upgrade everything with `pip install -U -r requirements.txt` .

Step 3: Check to see if all of the tests pass.

Step 4: Run `pip freeze > requirements.txt` to update the Python requirements file.

Step 5: Run `git commit` and `git push` to the production branch.

Freezing all your dependencies helps you have predictable builds.

If you need to check for missing dependencies, you can do so with the following command:

Code

```
python -m pip check
```

`Output:`

Code

```
No broken requirements found.
```

In our case, we are good to go!

How to Create Python Requirements File After Development

While it is possible to create it manually, it is a good practice to use the `pipreqs` module. It is used to scan your `imports` and build a Python requirements file for you.

According to THE DOCUMENTATION, once installed with the following command:

Code

```
pip install pipreqs
```

running `pipreqs` in the command line generates a `requirements.txt` file automatically:

Code

```
$ pipreqs /home/project/location
Successfully saved requirements file in
/home/project/location/requirements.txt
```

Why You Should Use a Python Requirements File

Create a Python `requirements.txt` file when starting a new data science project. It is always a good idea to include one in your project, particularly in the context of version control.

If you are unsure about version control, read more about it HERE. And if you are interested in writing better Python code, you can find more information HERE.

Using Python requirements files is among Python development best practices. It dramatically reduces the need for managing and supervising different libraries. Managing your library dependencies from one place makes it easier, more convenient, and faster. It helps keep everything organized and easy for everyone involved.

Compared to pasting a list of dependency paths into the command line every time you want to install or update them, it makes installing your Python applications on another system easier. It is a great way to ensure you have all the necessary dependencies installed for your project.

Also, GitHub provides automated vulnerability alerts for dependencies in your repository. By uploading a `requirements.txt` with your code, GITHUB checks for any conflict and sends an alert to the administrator if it detects any. It can even resolve the vulnerabilities automatically!

Best Practices for Using a Python Requirements File

There are several best practices to follow in using a Python `requirements.txt` file:

- Always use the `pip freeze` command to generate a list of Python modules and packages installed in the virtual environment of your project. This ensures the list is up to date and accurate.
- Only list the Python modules and packages your project needs. Do not include

- Save the Python `requirements.txt` file in the project source code repository so that other developers can easily install all of the Python modules and packages when they clone or check out your project.
- Use the `pip install -r requirements.txt` command to install all of the Python modules and packages listed in your `requirements.txt` file. This saves time and effort.
- Keep your Python `requirements.txt` files up to date and accurate. This ensures your project always uses the latest versions of the Python modules and packages.

Looking for data science project ideas to experiment with creating and maintaining a Python requirements file? Feel free to check THIS ARTICLE to find some inspiration!

Closing Thoughts on the Python Requirements File

In this article, we've learned what a Python `requirements.txt` file is and how to create it. We've also learned about its benefits and the Python community best practices for using a requirements file.

Thank you for reading! I hope this article was helpful and informative. You can find out more on LEARNPYTHON.COM.

Tags: Python Basics

You may also like