PyInstaller - Distributing Python Code

Python 2.x 2.7 , Python 3.x 3.3–3.4

Examples

Bundling to a Single File

`pyinstaller myscript.py -F`

The options to generate a single file are `-F` or `--onefile` . This bundles the program into a single `myscript.exe` file.

Single file executable are slower than the one-folder bundle. They are also harder to debug.

Bundling to One Folder

When PyInstaller is used without any options to bundle `myscript.py` , the default output is a single folder (named `myscript`) containing an executable named `myscript` (`myscript.exe` in windows) along with all the necessary dependencies.
The app can be distributed by compressing the folder into a zip file.

One Folder mode can be explictly set using the option `-D` or `--onedir`

`pyinstaller myscript.py -D`

Advantages:

One of the major advantages of bundling to a single folder is that it is easier to debug problems. If any modules fail to import, it can be verified by inspecting the folder.
Another advantage is felt during updates. If there are a few changes in the code but the dependencies used are *exactly* the same, distributors can just ship the executable file (which is typically smaller than the entire folder).

Disadvantages

The only disadvantage of this method is that the users have to search for the executable among a large number of files.
Also users can delete/modify other files which might lead to the app not being able to work correctly.

Installation and Setup

Pyinstaller is a normal python package. It can be installed using pip:

```
pip install pyinstaller
```

Installation in Windows
For Windows, pywin32 or pypiwin32 is a prerequisite. The latter is installed automatically when pyinstaller is installed using pip.

Installation in Mac OS X
PyInstaller works with the default Python 2.7 provided with current Mac OS X. If later versions of Python are to be used or if any major packages such as PyQT, Numpy, Matplotlib and the like are to be used, it is recommended to install them using either MacPorts or Homebrew .

Installing from the archive
If pip is not available, download the compressed archive from PyPI .
To test the development version, download the compressed archive from the *develop* branch of PyInstaller Downloads page.

Expand the archive and find the `setup.py` script. Execute `python setup.py install` with administrator privilege to install or upgrade PyInstaller.

Verifying the installation
The command `pyinstaller` should exist on the system path for all platforms after a successful installation.
Verify it by typing `pyinstaller --version` in the command line. This will print the current version of pyinstaller.

Using Pyinstaller

In the simplest use-case, just navigate to the directory your file is in, and type:

`pyinstaller myfile.py`

Pyinstaller analyzes the file and creates:

- A **myfile.spec** file in the same directory as `myfile.py`
- A **build** folder in the same directory as `myfile.py`
- A **dist** folder in the same directory as `myfile.py`
- Log files in the **build** folder

The bundled app can be found in the **dist** folder

Options
There are several options that can be used with pyinstaller. A full list of the options can be found here .

Once bundled your app can be run by opening 'dist\myfile\myfile.exe'.

Syntax

```
pyinstaller [options] script [script ...] | specfile
```

Parameters

Remarks

PyInstaller is a module used to bundle python apps in a single package along with all the dependencies. The user can then run the package app without a python interpreter or any modules. It correctly bundles many major packages like numpy, Django, OpenCv and others.

Some important points to remember:

- Pyinstaller supports Python 2.7 and Python 3.3+
- Pyinstaller has been tested against Windows, Linux and Mac OS X.
- It is **NOT** cross compiler. (A Windows app cannot be packaged in Linux. You've to run PyInstaller in Windows to bundle an app for Windows)

Homepage Official Docs