

Convert .py to .exe

A demonstration of how to package a python script into an executable file. This tutorial includes compiling to one file, no console, how to add an icon and adding other files to the final package.

- [PIP](#)
- [PyInstaller](#)
- [Basic Compiling](#)
- [No Console](#)
- [Onefile](#)
- [Adding an Icon](#)
- [Adding Other Files](#)
- [End Notes](#)
- [Auto PY to EXE](#)
- [Common Issues and Questions](#)
 - [My script runs fine in IDLE but won't run when packaged to exe](#)
 - [zipimport.ZipImportError: can't find module 'encodings'](#)
 - ["Open command window here" isn't shown when I shift right click?](#)
 - ['pip' is not recognized as an internal or external command](#)
 - [Fatal error in launcher: Unable to create process using...](#)
 - ['pyinstaller' is not recognized as an internal or external command](#)
 - [When editing the PATH variable, I can only edit the variable](#)
 - [The exe does not work on another computer](#)
 - [My antivirus detected my exe as a virus](#)
 - [I get lots of WARNINGS when running pyinstaller](#)
 - [Will this add my other scripts? / Will this work with external Python modules?](#)
 - [is it available in Walmart?????](#)

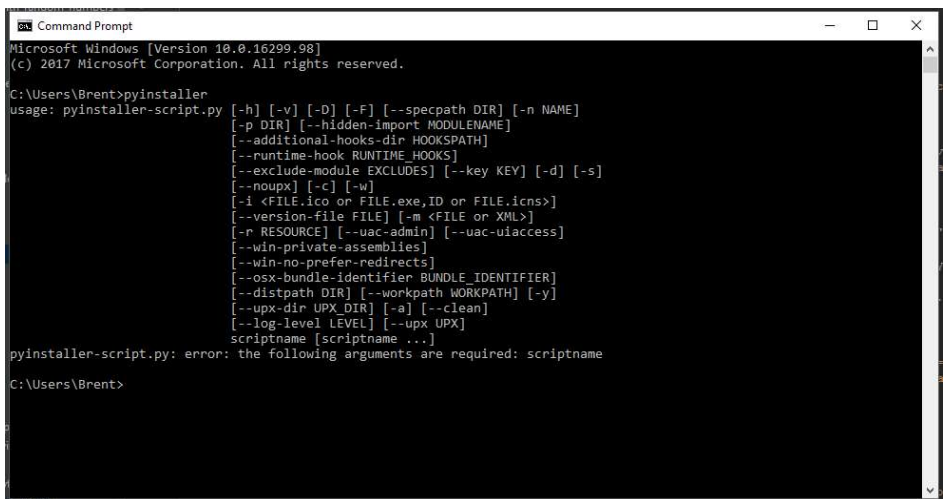
PIP

If you haven't used or setup pip before, go to my tutorial at [how-to-setup-pythons-pip](#) to setup pip.

PyInstaller

Now that pip has been set up, execute the command `pip install pyinstaller` in cmd. Make sure to check the output for errors as if there are errors it would not have installed successfully. PyInstaller now supports Python 2.7 - 3.7 including Python 3.7.

To make sure it installed properly, type `pyinstaller` in cmd and make sure no errors appeared.



```
Command Prompt
Microsoft Windows [Version 10.0.16299.98]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\Brent>pyinstaller
usage: pyinstaller-script.py [-h] [-v] [-D] [-F] [--specpath DIR] [-n NAME]
                             [-p DIR] [--hidden-import MODULENAME]
                             [--additional-hooks-dir HOOKSPATH]
                             [--runtime-hook RUNTIME_HOOKS]
                             [--exclude-module EXCLUDES] [--key KEY] [-d] [-s]
                             [--noupX] [-c] [-w]
                             [-i <FILE.ico or FILE.exe,ID or FILE.icns>]
                             [--version-file FILE] [-m <FILE or XML>]
                             [-r RESOURCE] [--uac-admin] [--uac-uiaccess]
                             [--win-private-assemblies]
                             [--win-no-prefer-redirects]
                             [--osx-bundle-identifier BUNDLE_IDENTIFIER]
                             [--distpath DIR] [--workpath WORKPATH] [-y]
                             [--upx-dir UPX_DIR] [-a] [--clean]
                             [--log-level LEVEL] [--upx UPX]
                             scriptname [scriptname ...]

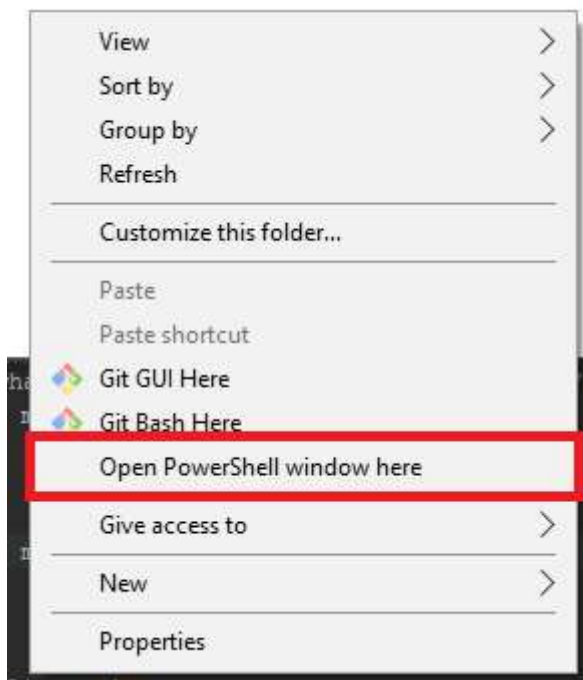
pyinstaller-script.py: error: the following arguments are required: scriptname

C:\Users\Brent>
```

Basic Compiling

Created a new folder and put your python file in it and any other modules or files it may need. Then hold shift and right-click in the folder, in the menu that popped up, click "Open PowerShell Window Here" or

"Open command window here" for older versions of windows. If this option doesn't appear, try again or open cmd and type `cd {folder location}` to move to that folder.



Now that cmd is in the right location, execute `pyinstaller {the name of your python file}` make sure to add .py or whatever extension it has. Wait for it to finish and check if any errors have appeared. If none have you can close cmd and look at the three folders generated.

Look in the 'dist' folder and you should see {the name of your script}.exe. If you run that your script should execute as an exe.

⤴ No Console

When running the compiled script, you will notice a console window will appear. If you do not want this, add the 'w' flag to the statement when creating the script.

Thus the new statement will be `pyinstaller -w {the name of your python file}`. Now when you run the .exe, the console will not appear.

› Onefile

If you want all the files to be packed into one .exe you will need to add the 'F' flag. Thus the new statement will be `pyinstaller -F {the name of your python file}`. Note that this will not work for all python scripts due to third party libraries or how the script works. You will find out if it works or not by running the .exe in the 'dist' folder.

If you want to use onefile mode with external files, it will pay to read [this](#)

› Adding an Icon

To add an icon to the final .exe (normal or onefile) add the 'i' flag and then the location of the .ico file (do not rename a .png/.jpg/.bmp to a .ico - I have had dumb people in the comments do this). You can find some nice icons [here](#).

Thus the new statement will be `pyinstaller -i {icon location} {python file}`, remember to not forget about extensions.

› Adding Other Files

Since pyinstaller only searches for Python files, you need to add data files like images, databases and JSON files yourself. To do this, simply provide the `--add-data` flag with the source and destination of your file as many times as required.

As described in the documentation, to use this flag, it must be formatted as `--add-data SRC;DST`. SRC is the source file path of the file you want to add, for example `C:\Users\me\Desktop\my-image.jpg`, this can be either an absolute or relative path to the current working directory. DST is the

destination in the packaged folder/executable. I recommend using a `.` to put a file in the root path. You can also specify a folder, for example, `images` which will then create a folder named `images` and copy the file to there.

An example of this is when I want to put the file `C:\my-file.png` into the `img\` folder in the packaged folder/executable. To do this I would use `--add-data "my-file.png";"img/"` (I used `"` so spaces in the paths don't cause errors). Now in Python, I will need to look in the `img\` folder in the root of the packaged folder/executable to get the image.

If you can't get this to work, I recommend trying out my project called `auto-py-to-exe` mentioned below. The application takes care of moving extra files and many other things so you don't have to.

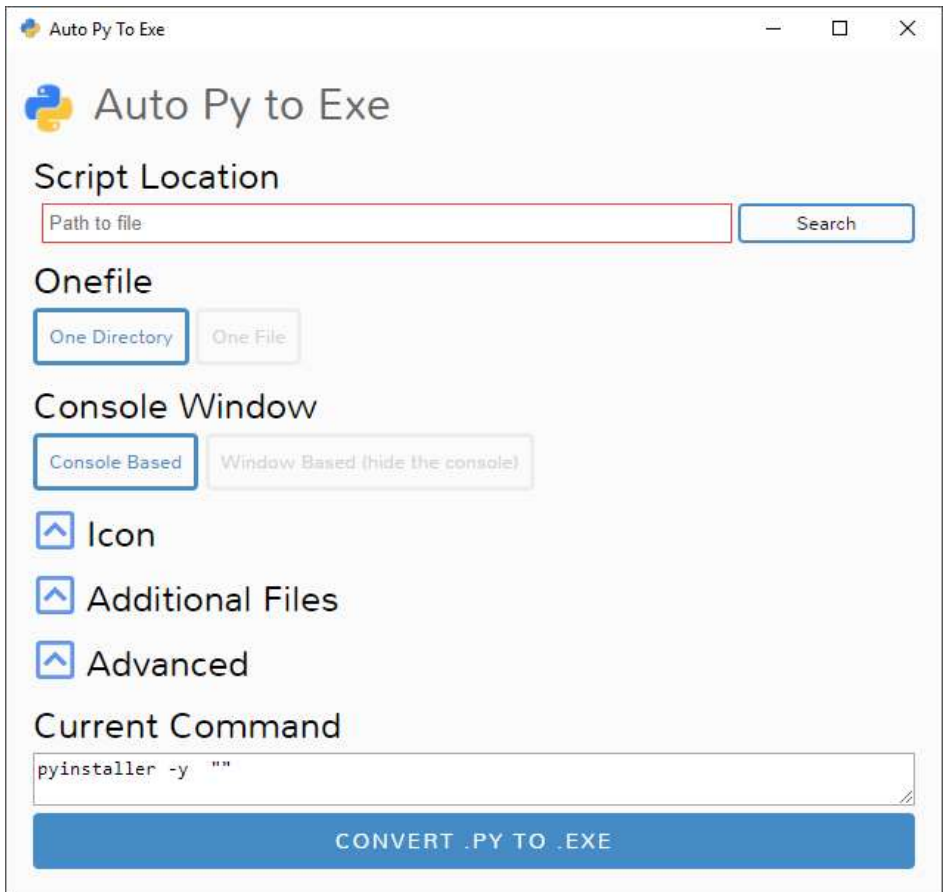
The separator between the SRC and DST can be different on different platforms. Windows uses `;` whereas most other operating systems like OSX and Linux use `:`.

› End Notes

You can combine these flags to make things like onefile executables with no console and an icon by using a statement like: `pyinstaller -w -F -i {icon location} {python file}`

› Auto PY to EXE

In March 2018 I created an application that allows you to create executables easily from Python scripts. It is a simple graphical interface built with Eel in Python and uses `pyinstaller` in the background.



Common Issues and Questions

My script runs fine in IDLE but won't run when packaged to exe

Regarding that the python script runs properly by itself then this would have been caused by an incorrect configuration or third-party modules. The best way to identify the issue is to add `-d all` to the command and then re-package it. This will mean the exe is now in a debugging mode. Open up cmd and then run the exe using cmd e.g. `"C:/folder/path/myexe.exe"`. Any errors will be preserved in the console which you were previously missing.

↳ **zipimport.ZipImportError: can't find module 'encodings'**

Please upgrade PyInstaller to 3.4 or above using: `python -m pip install --upgrade PyInstaller`

↳ **"Open command window here" isn't shown when I shift right click?**

Make sure you are holding down shift. If you are using new versions of Windows, this has been replaced by "Open PowerShell Window Here". Using this method will work the same for this tutorial, so go ahead and use PowerShell.

↳ **'pip' is not recognized as an internal or external command**

Please follow the pip setup again, you have done something wrong. Alternatively, you can try to use `python -m pip {command}`.

↳ **Fatal error in launcher: Unable to create process using...**

Try executing `python -m pip install pyinstaller` in cmd.

↳ **'pyinstaller' is not recognized as an internal or external command**

Go back to the PyInstaller heading, you have not installed pyinstaller, remember to test it.

↳ **When editing the PATH variable, I can only edit the variable**

Add a ';' to the end and then put in the folder location, then apply/save it. This input is like this because you are using an older version of Windows.

↳ **The exe does not work on another computer**

This may be an architecture issue. PyInstaller will create an executable with the architecture of the machine it was built with. You are most likely using a 64bit machine if you are asking this question to compile the .py; thus it will create a 64bit executable. As with any other programs, you cannot run 64bit on 32bit but you can run 32bit on 64bit. Thus I recommend using 32bit python or compiling on a 32bit machine so it will work on both architectures.

↳ **My antivirus detected my exe as a virus**

This is your anti-virus vendors fault. Check out [this](#).

↳ **I get lots of WARNINGS when running pyinstaller**

These warnings can be ignored in most cases. I have not currently found a situation where these are an issue, after all, they are only warnings.

↳ **Will this add my other scripts? / Will this work with external Python modules?**

If your main script imports your others scripts, then yes. PyInstaller looks at imports to figure out what to bundle, so it will add your other scripts just like if you were to import os or time.