Upgrade Open in app

Published in Towards Data Science · Follow

You have **3** free member-only stories left this month. Upgrade for unlimited access.

Frank Andrade · Follow
May 26, 2021 · 6 min read

# How to Easily Convert a Python Script to an Executable File (.exe)

Two simple ways to create a Python executable file.

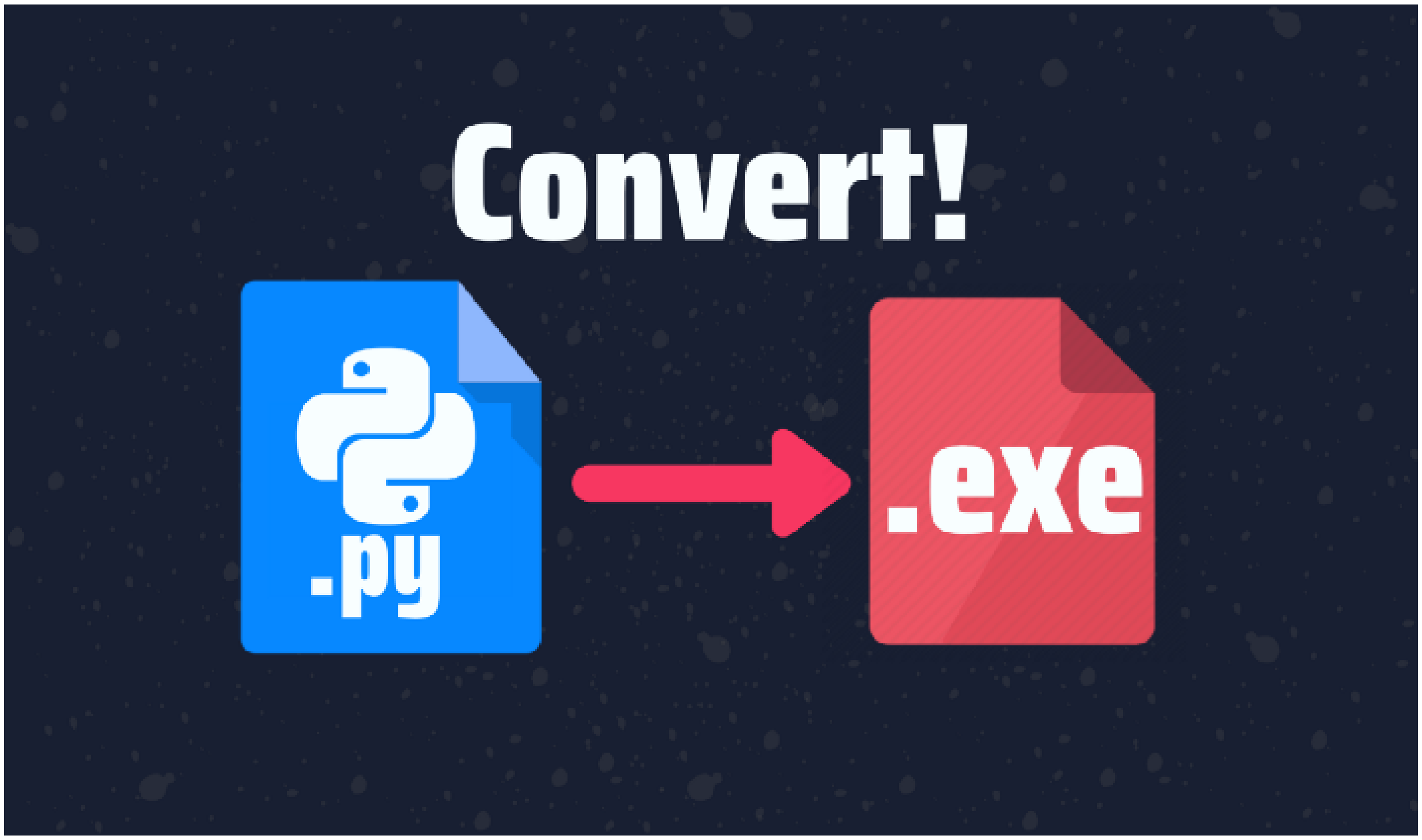


Image by author

Although running a Python script using the terminal or your favorite text editor is straightforward, there are some situations in which you will prefer to hide all the code written in the script (.py) inside an executable file (.exe).

Maybe you need to send the script to someone who doesn't code at all or you might need to schedule a job that runs a `.exe` at a specific time on your computer. Whatever the situation, in this guide, I will show you 2 ways to create an executable file. The first (auto-py-to-exe) has a friendly interface that will help beginners to easily create executables, while the second (PyInstaller) offers a straightforward way to create executables through the terminal.

**Table of Contents**

## Making an Executable file with auto-py-to-exe

The first option offers a nice GUI (graphical user interface) that takes care of all the stuff necessary to convert your Python script into an executable file.

By the way, you can also watch my YouTube video to learn how to convert a .py to .exe, in case you prefer watching the steps below rather than reading them.

### Installing with pip

To install the last version of auto-py-to-exe, just open up a terminal and run the following command.

```
pip install auto-py-to-exe
```

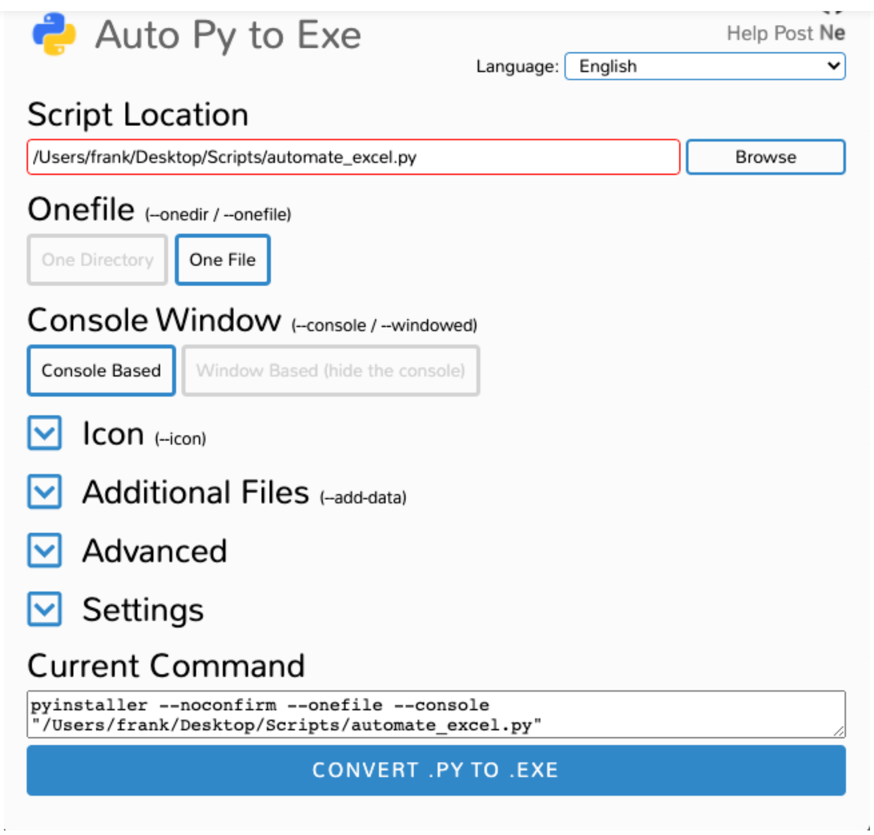
*Note: Make sure that the working environment in which you're installing* `auto-py-to-exe` *contains all the libraries that your script needs to run.*

### Running auto-py-to-exe

Once you install auto-py-to-exe, making an executable file is as easy as writing the following command.

```
auto-py-to-exe
```

After running the command, the following GUI application will open.


Auto Py to Exe
Help Post Ne
Language: English
Script Location
/Users/frank/Desktop/Scripts/automate_excel.py
Browse
Onefile (--onedir / --onefile)
One Directory
One File
Console Window (--console / --windowed)
Console Based
Window Based (hide the console)
Icon (--icon)
Additional Files (--add-data)
Advanced
Settings
Current Command
CONVERT .PY TO .EXE

```
pyinstaller --noconfirm --onefile --console
"/Users/frank/Desktop/Scripts/automate_excel.py"
```

Image by author

I will walk you through each option to properly create an executable file.

## Steps to create an executable file

**Step 1: Add the script location**

Browse the script you wish to convert and add it to the "Script Location" field. In this example, I'm going to choose a script that automates Excel reports (you can find my guide to automate Excel in the link below)

**A Simple Guide to Automate Your Excel Reporting with Python**

Use openpyxl to automate your Excel reporting with Python.

towardsdatascience.com

Feel free to choose any script you want. **However, if your script needs to read a path make sure that you use absolute paths since relative paths won't behave as you might expect with executable files.** If necessary, include the following line of code below to know where the executable file is located and make the necessary changes to your script so you read/export files on the right directory.

```
application_path = os.path.dirname(sys.executable)
```

**Step 2: Choosing "One Directory" or "One File"**

Now we have to choose whether we want to create "one directory" or "one file." The first creates a directory with all the dependencies your script needs to run (including the executable file), while the second creates only a single executable file.

the executable file, which is recommended if your script generates console-based outputs. However, if you don't want to show the console outputs when running the executable file, choose "Window Based"

My script needs the name of the Excel spreadsheet to be introduced as input in order to create my Excel report, so I'm going to choose "Console Based."

**4 Free and Paid Web Scraping Courses Every Data Scientist Should Take**
Acquire this must-have skill every data scientist should have.
medium.com

**Step 4: Advanced options (e.g., output directory, additional imports)**

You can add an icon, add files that your script needs to run, and more! However, for this example, I'll only modify the path where the executable file will be exported. To do so, click on the "Setting" option and browse the output directory you wish.

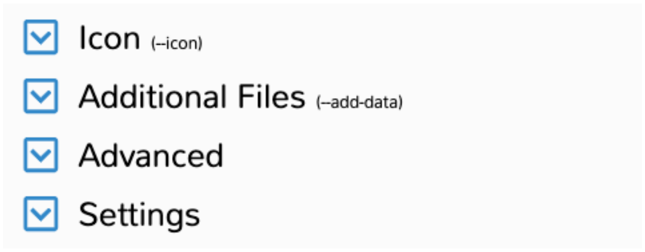
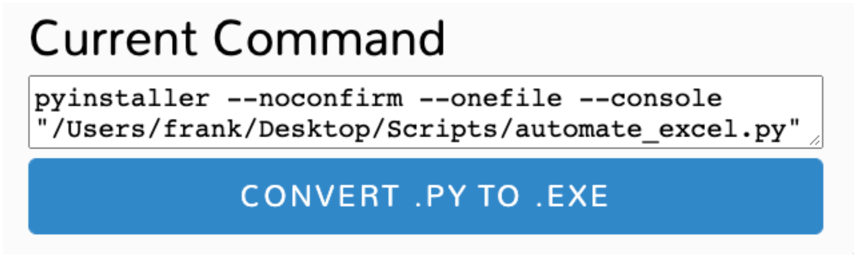


Image by author

*Note: If you see an error like this "ModuleFoundNotError: Not module named 'name_of_module'" after double-clicking on the executable file created, you'll have to repeat from step 1 again, but now in the "Advanced" option write the module name is missing inside the "hidden-import" field.*

**Step 5: Convert the file**

To convert the .py file to .exe just click the blue button you see below.



```
pyinstaller --noconfirm --onefile --console
"/Users/frank/Desktop/Scripts/automate_excel.py"
```

Image by author

Something really important that `auto-py-to-exe` shows above the convert button is the code that `pyinstaller` (the main library and second option in this guide to make .exe files) needs to create an executable file behind that fancy GUI you see on the screen.

Once the process is finished the executable file should be located in the output directory you set in step 4!

## Making an Executable file with PyInstaller

This option fits better for those who prefer to quickly create an executable file running a command on the terminal.

- **Step 2:** Using the terminal, go to the directory where your script is located (use the `cd` command)
- **Step 3:** Once you're in the right directory, write a command with the following syntax `pyinstaller --onefile name_of_script.py` in the terminal to make the script executable.

The command used in step 3 is similar to the code shown in the step 5 picture for the `auto-py-to-exe` option. You can play a little bit with the GUI offered by `auto-py-to-exe` to get used to the many options you can add to this command.

After running the command, you should see a message that says "completed successfully." In the directory where your script is located, a folder named "dist" should have been created. Inside the folder, you'll find the standalone executable!

*Congratulations! Now your Python script has been converted to an executable file. In case you want to schedule when this file will run on your computer check this guide.*

**How to Easily Automate Your Python Scripts on Mac and Windows**
Use crontab and the task scheduler to automate your scripts and save time
towardsdatascience.com

*Below you can find some guides I made on libraries used in data science (Matplotlib/Seaborn, Pandas, and Scikit-Learn).*

- A Pandas Guide for Excel Users
- How to Make Beautiful Visualizations with Matplotlib and Seaborn
- A Simple Guide to Scikit-Learn — Build Your First Machine Learning Model in Python

**Join my email list with 3k+ people to get my Python for Data Science Cheat Sheet I use in all my tutorials (Free PDF)**