Compiling python code

1. run script in IDE

Some code is better run interactively to execute separate chunks sequentially. In this case, jupyter notebook or IDE like spyder works fine

1. run script from terminal

You could also run python code directly from the terminal with ´´´run python myfile.py´´´,

but you should cd to the correct directory first or type the absolute path to myfile.py. This can be tedious sometimes. Similarly, you must remember to run your code in the right environment if you are using specific dependencies…

A shortcase I sometimes use is to write a .sh file with the terminal command above and make the file executable (rightclick… or code). Then you only need to doubleclick the file to run the python code intended.

1. create a setup.py and install your package

If the process above is to tiring, which does not necessarily look like it but can really be, you can properly install your code as a python package in your environment.

First, you need to write a setup.py file with requirements necessary

then run ´´´pip install -e .´´´. the ‘.’ here refers to the current working directory, which I assume to be the directory where the setup.py can be found. The -e flag specifies that we want to install in *editable mode*, which means that when we edit the files in our package we do not need to re-install the package before the changes come into effect.

now you can import the package from other python scripts, or even call it from the terminal without the python command or setting the working directory

1. compile with pyinstaller

If the solutions above still not fit your idea of how to run your new python code, you can still compile it. This will make an executable file out of your code, which can be run without the need of checking for dependencies, environments, or even needing to install python! This is probably the safest way to go to make your code widely accessible.

pip install pyinstaller (sticks with 3.6?)

pip install - - upgrade pyinstaller to 4.5

pyinstaller --add-binary ffmpeg.exe;. --add-binary ffmpeg.exe;. --onefile --icon=logo.ico VideoPyToolbox.py

--onefile flag keeps the dist folder clean of dependencies, but takes longer to start the exe

-w flag suppresses terminal window when running your exe

--icon flag sets an icon for the exe file

--add-binary flag adds existing binaries to your compilation

for matplotlib error see here <https://stackoverflow.com/questions/67345287/matplotlib-directory-not-found-while-using-pyinstaller-to-create-exe-from-py-fil>

1. for mac OS

pyinstaller --onefile [--windowed](https://pyinstaller.readthedocs.io/en/stable/usage.html#cmdoption-w) --icon=logo.ico VideoPyToolbox.py