The process I used for embedding the Python interpreter into a C++ runtime was relatively straightforward, given I have a fairly strong grasp of the compilation and linking process for a C++ program.

After acquiring the appropriate library, it was mostly a matter of adding the include directories to have access to the function definitions and linking the libraries in order to properly link the implementations. I did the process both using Visual Studio, and CMake. I verified my CMake solution on both Windows and Mac OSX. Very likely it would work as is on Linux assuming a proper Python installation.

The major roadblocks:

1. Windows requires the inclusion of the Python3x.dll / Python3x\_d.dll in the application’s working directory. In CMake, I copied them directly out the user’s install directory. With Visual Studio, it was simpler to embed the libraries into a /vendor directory and copy them from there.
2. Visual Studio is annoying to configure. As to avoid absolute paths, I embedded the Python library include and libs directory into a /vendor directory alongside the dlls in order to make relative paths.
3. On Mac OSX (and presumably Linux), when running the CPP application the Python interpreter doesn’t automatically have the current running directory on its system path with the result that it cannot find the local modules. I suspect this is, at its root, related to the .dll being in the same directory. I suspect if we added the Python dlls to our system32 directories we would have the same issue.

Overall, this was quite enjoyable to do as I had always meant to attempt something similar.

Screenshots:



