FL Studio MIDI Scripting:

Beginner's Guide

1. Setting up the work environment.

1.0.0. Download Visual Studio Code

Download Visual Studio Code as this is the optimal IDE for writing our python scripts. It is user friendly with built-in features which would otherwise have to be installed manually. Visual Studio Code will be able to provide code completion and type hinting which is essential for avoiding logical programmer errors and mistypes of names.

Download Visual Studio Code - Mac, Linux, Windows

1.0.1. Download Visual Studio Code Python Extensions

After downloading and installing visual studio code we need to install a few extensions before starting to work with python.

- **Python for VSCode**: rich support for the Python language (for all actively supported versions of the language: >=3.7), including features such as IntelliSense (Pylance), linting, debugging, code navigation, code formatting, refactoring, variable explorer, test explorer, and more! Python - Visual Studio Marketplace
- **Pylance**: Include in the Python for VSCode package. Should it fail to install or needs an update you can install it as a separate package as well.Pylance - Visual Studio Marketplace
- **Pylint**:The Pylint extension provides a series of features to help your productivity while working with Python code in Visual Studio Code. Pylint - Visual Studio Marketplace

According to the Google Python Coding Standard section 2.1.4 states: “Make sure you run pylint on your code. Suppress warnings if they are inappropriate so that other issues are not hidden. To suppress warnings, you can set a line-level comment:

```
def do_PUT(self):  # WSGI name, so pylint: disable=invalid-name  ...”
```

Some additional optional but useful extensions:

- **Dragan Color Theme:** A theme with syntax highlighting for python that is very clear and high contrast. Dragan Color Theme - Visual Studio Marketplace
- **Python Indent:** Correct python indentation. Saves time by avoiding fiddling with indents in the code base. Python Indent - Visual Studio Marketplace
- **Auto Docstring:** Increases the speed of writing documentation strings for objects in python by providing auto-completion. autoDocstring - Python Docstring Generator - Visual Studio Marketplace

1.0.2. Download Python 3.9

There may be a python version installed on your system or a python version shipped with VSCode but we need to make sure that we are using the same python version as the FL Studio Python environment. The FL Studio Midi Scripting python environment(PyBridgex64.dll) is a severely limited version of python 3.9(more details on this later..), so we will download python 3.9.13 so that our extensions can provide the appropriate feedback for our version of python.

Python Release Python 3.9.13 | Python.org

After downloading and installing python make sure to restart. Now we can open VSCode and choose our python interpreter by typing >python interpreter in the top bar. Then choose our python version from the drop down list.

1.0.4. Test the python environment to make sure everything is setup correctly.

2.0.0. Opening our script in FL Studio

Now that our IDE is setup and ready to write python we can create the first file of our script. This python file has to be placed in the FL scripts folder. It also must have a prefix of “device_” in its name. It must be a python file. It must have a directive at the top of the file # name=name of script. For example, this script will output “Hello Fl Studio” to the FL Studio Scripting Terminal:

device_myscript.py

```
# name=MyScript
# receiveFrom=MyScript

print(“Hello Fl Studio!”)
```

Fl Studio will scan all folders and sub-folders in files that match these requirements. Therefore it is essential to not name any other python files “device_”. Furthermore, do not declare a # name directive for the files either – use a standard python docstring instead. More guidelines and recommendations on project structure later.

2.0.1. Debuging our script in FL Studio

Once we opened our script we can view the output of the python environment in FL studio by clicking view script. Here we will be able to see errors that occur in the script as it is running inside the fl python bridge. When coding in the IDE and writing python scripts it will be natural to search for guides on the internet. Unfortunately due to the nature of FL Studio’s python environment, those guides and suggestions may not translate to FL Studio’s python – especially solutions which utilize the python standard library which is mostly excluded from the Fl scripting environment. So it is essential to debug the code through the FL Studio Scripting View.

WARNING: FL will NOT report errors in imported modules if they occur during import of the module. This is a very important point because in python you usually want to organize your code in small bits and modules that carry a designated responsibility. This is not possible in FL Studio Scripting while in development. Should you make any adjustment to the module that will result in it not loading, FL will simply stop the script and indicate the file does not exist. A module would have to be tested and finalized, then ran as the main script file (device_) to test for any errors in the FL Evironment. Then you would be able to import it as a separate module without fear of your code-base being unmaintainable in the future.

So what is the solution?

- **Option 1. Write everything in one file.**

  Write your entire script in one large file. This is the best solution for simpler scripts. And even larger scripts still in development stages. The disadvantage is that the code will be exponentially verbose as the code-base gets larger due to the inability to load modules/separate concerns.

- **Option 2:**

  Write and test your modules and finalize them before utilizing them in your main script. This may be mixed with option 1 while in the development stages then fully converted to option 2 upon finalization of the code-base. The advantage is you will be able re-use code between scripts and reduce name verbosity. Disadvantage is that this testing and finalizing process takes a long time, and it's not always possible to finalize a module while still in development.

- **Option 3.**

  Create a virtual FL Studio python environment and test your code in that space. Then transfer code to FL Studio to assert its functionality. This option is for large scale projects because creating the environment will be a complex and time consuming project on its own. The advantage is this is the best option for code maintainability and portability between different versions of FL Studio (or

different operating systems). Also this allows you not be dependant on the FL studio scripting library but merely use it as an interface.

Here is a good resource/example of a virtual FL Studio Environment. MiguelGuthridge/FL-Studio-API-Stubs: Stub code, documentation, and basic emulation for the FL Studio Python API (github.com) Unfortunately this project is not for the current version of Fl studio scripting and is not maintained. Nonetheless is it a golden resource of information about fl studio scripting not provided by the Fl documentation:

Appendix.

1. Useful Resources:

- MiguelGuthridge/FL-Studio-API-Stubs: Stub code, documentation, and basic emulation for the FL Studio Python API (github.com)
- List of python modules available in fl studio : Using custom Python modules | Forum (image-line.com)