# Getting Started Guide for Pyzo & FreeCAD on MS Windows

## Steps to Start

- First of all, create the start_pyzo.py file in a path, for example *C:\Users\user\AppData\Roaming\pyzo\start_pyzo.py* (more info in section later) with the following content:

```
import FreeCAD
import FreeCADGui
from PySide2 import QtCore
QtCore.QCoreApplication.setAttribute(QtCore.Qt.AA_ShareOpenGLContexts, True)
import sys
stdout_backup = sys.stdout
stderr_backup = sys.stderr
FreeCADGui.showMainWindow()
sys.stdout = stdout_backup
sys.stderr = stderr_backup
```

- Start Pyzo and add a new shell configuration: *Shell / Edit shell configurations... / Add Configuration*
- Configure the shell as follows:

  **- name**: freecad_version
  **- exe**: C:\Program Files\FreeCAD_version\bin\python.exe [path to python.exe in your FreeCAD installation]
  **- gui**: PySide2
  **- startupScript**: Check "File to run at startup" and enter the file path: *C:\Users\user\AppData\Roaming\pyzo\start_pyzo.py*
  **- environ**:
  - PYZO_PROCESS_EVENTS_WHILE_DEBUGGING=1 (to prevent FreeCAD from freezing while in Debug mode)
  - LC_NUMERIC=en_US.UTF-8 (American format regional number configuration)

- In the Shells Window, select "*New Shell...*" and choose the "freecad_version" configuration
- An instance of FreeCAD will open with the name "python" displayed in the top-left corner of the window instead of "FreeCAD" . This instance is connected to Pyzo
- Open the file you want to debug in Pyzo: *File / Open*
- Add the desired breakpoints
- Use *Run / Execute file* to begin debugging

## Usage options

- *Run / Execute selection*: Executes the selected code. If nothing is selected, it executes the line where the cursor is located
- *Run / Execute cell*: executes the code between two lines that start with “##” and where the cursor is located
- *Run / Run file as script*: runs the file as if it were a macro in FreeCAD, with the inconvenience that it restarts the shell beforehand to provide a clean environment
- Debug postmortem: *Shell / Postmortem: debug from last traceback*

## Notes

- Interactive debug: While the program is paused at a breakpoint or any point, It's possible to interact with the FreeCAD instance from both the GUI and the Python console as well as from the Pyzo’s shell. This is due to the setting of the key PYZO_PROCESS_EVENTS_WHILE_DEBUGGING=1 which will continue to run the GUI event loop when the interpreter stopped.
- Running the "__file__" variable inside a script with *Run/Execute file* will throw a NameError. To work correctly, it should be executed with *Run/Run file as script* if possible; if not, a workaround is to add two lines at the beginning of the script:

```
import inspect
__file__ = inspect.getfile(inspect.currentframe())
```

## About start_pyzo.py

This file contains the initial lines that Pyzo will run at startup of the FreeCAD instance.
Some commets about the script:

1. import FreeCAD # Mandatory
2. import FreeCADGui # Mandatory
3. from PySide2 import QtCore # for the next line
4. QtCore.QCoreApplication.setAttribute(QtCore.Qt.AA_ShareOpenGLContexts, True) # This allows sharing OpenGL contexts among several PyQt/PySide windows and prevents possible OpenGL-related issues in FreeCAD's GUI. It fixes some messages at the beginning of the FreeCAD instance:
   "Qt WebEngine seems to be initialized from a plugin. Please set Qt::AA_ShareOpenGLContexts using QCoreApplication::setAttribute before constructing QguiApplication."
   "WebEngineContext used before QtWebEngine::initialize() or OpenGL context creation failed."
   It is a problem of the current state, something temporary.
5. import sys # for the next lines
6. stdout_backup = sys.stdout # Backs up the original standard output (stdout)
7. stderr_backup = sys.stderr # Backs up the original standard error output (stderr)
8. FreeCADGui.showMainWindow() # Mandatory
9. sys.stdout = stdout_backup #  Restores the original standard output
10. sys.stderr = stderr_backup # Restores the original standard error output

    # Backups of lines 6 and 7 and the restorations of lines 9 and 10 are done to prevent the standard output and error output of Pyzo's console from being redirected to the FreeCAD Report view

## About Outputs **(incomplete and potentially incorrect)**

Standard output is, for example, the output of print("hello"), and standard error output is any error message thrown while running the code. The outputs when running code can be sent to Pyzo's shell only, FreeCAD's Report view only, or Both Pyzo's console and FreeCAD's Report view.

Currently, in this guide, the configuration is set in the 'start_pyzo.py' code (view the 'About' section), where both standard and error outputs are directed exclusively to Pyzo's own shell. Thus, *print("hello")* will be printed in Pyzo's shell, while *App.Console.PrintMessage("hello")* will be printed in the FreeCAD Report view. Error outputs will always be printed in Pyzo's shell.

If different behaviors are desired, there are variables that can be adjusted, although at the moment, I have not been able to achieve consistent results:

- PYTHONUNBUFFERED=1:  To print standard outputs of Pyzo's console both in Pyzo's own console and in the FreeCAD Report view. By default, it is set to 0.
- PYZO_STDIO_SINK=file (console – file – pipe): To direct console outputs to a file. By default, it is set to *console.*

## Sources

- https://forum.freecad.org/viewtopic.php?t=78047&start=10
- https://github.com/pyzo/pyzo
- https://pyzo.org/