Python Errors

Precompiled Libraries

While not common practice, Python add-ons can be distributed with their own precompiled libraries. Unlike regular Python scripts, these are not portabl between different platforms.

It is possible the library is incompatible with your Blender installation (attempting to load a library built for a different version of Python, or loading a 32-b library on a 64-bit system).

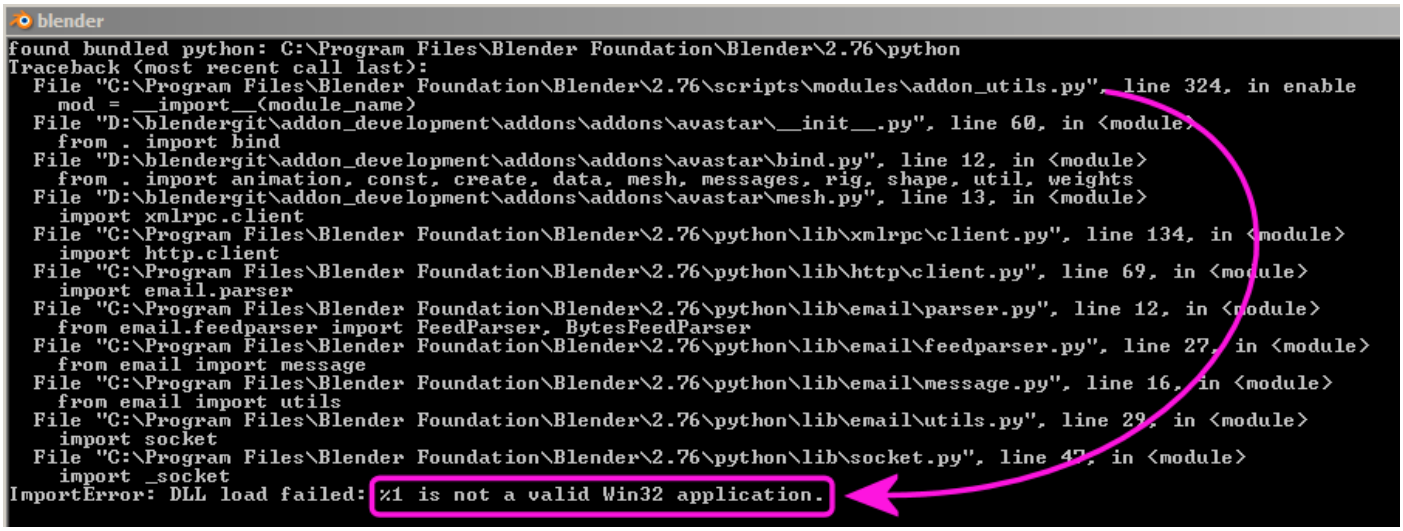
If the add-on contains `.pyd` or `.so` files, check that the distribution is compatible with your operating system.

Platform Specific

Windows

Mixed Python Libraries (DLLs)

If Python is raising errors or you have an add-on that just fails when enabled with an error – e.g: `... is not a valid Win32 application.` – this may be caused by some inconsistency in the Python libraries. While Blender comes with its own bundled Python interpreter, duplicate, incompatible libraries can cause problems.



A Python traceback.

To find out which Python Library caused the Problem check the error message.

This is normally reported somewhere around the bottom line of the traceback. With the error above you see the problem is caused while trying to import `_socket`. This corresponds to either a file named `_socket.py` or `_socket.pyd`.

To help troubleshoot this problem, the following script can be pasted into the Text editor and run to check for duplicate libraries in your search path. (Th output will show in Command Line Window.)

```
import os
import sys

# Change this based on the library you wish to test
test_lib = "_socket.pyd"

def GetSystemDirectory():
    from ctypes import windll, create_string_buffer, sizeof
    GetSystemDirectory = windll.kernel32.GetSystemDirectoryA
    buffer = create_string_buffer(260)
```

```
    GetSystemDirectory(buffer, sizeof(buffer))
    return os.fsdecode(buffer.value)

def library_search_paths():
    return (
        # Windows search paths
        os.path.dirname(sys.argv[0]),
        os.getcwd(),
        GetSystemDirectory(),
        os.environ["WINDIR"],  # GetWindowsDirectory
        *os.environ["PATH"].split(";"),

        # regular Python search paths
        *sys.path,
        )

def check_library_duplicate(libname):
    paths = [p for p in library_search_paths()
             if os.path.exists(os.path.join(p, libname))]

    print("Library %r found in %d locations:" % (libname, len(paths)))
    for p in paths:
        print("- %r" % p)

check_library_duplicate(test_lib)
```