Python / Assembly Language

It is possible to call your assembly language code from Python. In fact, it can be insanely easy. It can also be difficult. What differentiates the two are the arguments you need to pass.

Python stores data ina very different way than C and therefore assembly language. All Python data types are objects, which for sets and dicts can make things difficult and outside the scope of this book.

If you want to explore the subject deeply, we suggest this link

Simple EXample

Take this trivial C function:

```
int square(int x) {
    return x * x;
}
```

It could be written in this way using our Apple / Linux convergence macros:

```
#include "apple-linux-convergence.S"

/*
    gcc -fPIC -shared -o my_square.so function.S
*/
        .p2align    2
        .text
        GLABEL      square

#if defined(__APPLE__)
_square:
#else
square:
#endif

        mul     x0, x0, x0
        ret

        .end
```

This function is a leaf and requires no interaction with the stack.

Notice there is no `main()`.

Using the following command line, turn your assembly language module into a "shared object" or .so file:

```
gcc -fPIC -shared -o my_square.so function.S
```

`-fPIC` means generate position independent code. This allows the library to be relocated wherever the program loading the library wants it.

Then, in Python:

```
from ctypes import *

so_file  = "./my_square.so"
my_funcs = CDLL(so_file)
print(my_funcs.square(10))
```

It just works.

More to Come Here