SPECIAL TOPIC: CALLING C ROUTINES

Eric Prebys

General

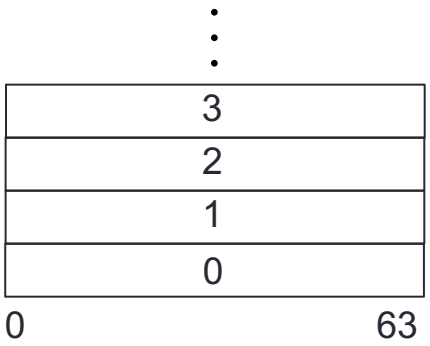
- As we said before, interpreted Python is extremely slow
- If you are doing something complex for which an existing library doesn't exist, you may need to write your own.
- Luckly, this is pretty easy
  - Write your code
  - compile it as a "dynamic link library"
  - Use the "ctypes" library in Python to load the library and define the input and output variable.
- System dependence
  - MacOS and Linux:
    - Use the built-in gcc compiler to create a .dylib or .so library, respectively
  - Windows
    - Install the free Community version of Visual Studio, including C/C++ development tools
    - Open a developer window (be sure it's 64-bit!)
      - Use, eg. "x64 Native Tools Command Prompt for VS 2022"
    - Use command line tools to creat a .DLL library
- NB This will also work with FORTRAN, which believe it or not you may need to know.

Variable Types

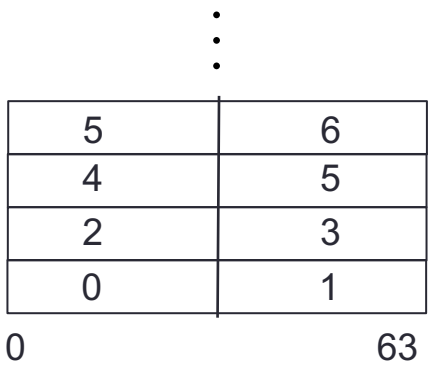
- For individual variables, passed by value, the ctype library will do the appropriate conversions, within reason.
- For arrays, the variable types in the numpy array MUST match the variable types in the C program
  - All that will be passed to the route in the pointer to the beginning of the array data.
  - It has no way of “figuring” out what type of data it is
  - Remember
    - Python:
      - float = 64-bit floating point (on modern systems)
      - int = bigint (array must use a C type)
    - C
      - float = 32-bit floating point
      - double = 64-bit floating point (default for numpy float)
      - int = 32 bit integer on all modern systems
      - long = 64 bit integer

Consequences of Mistakes

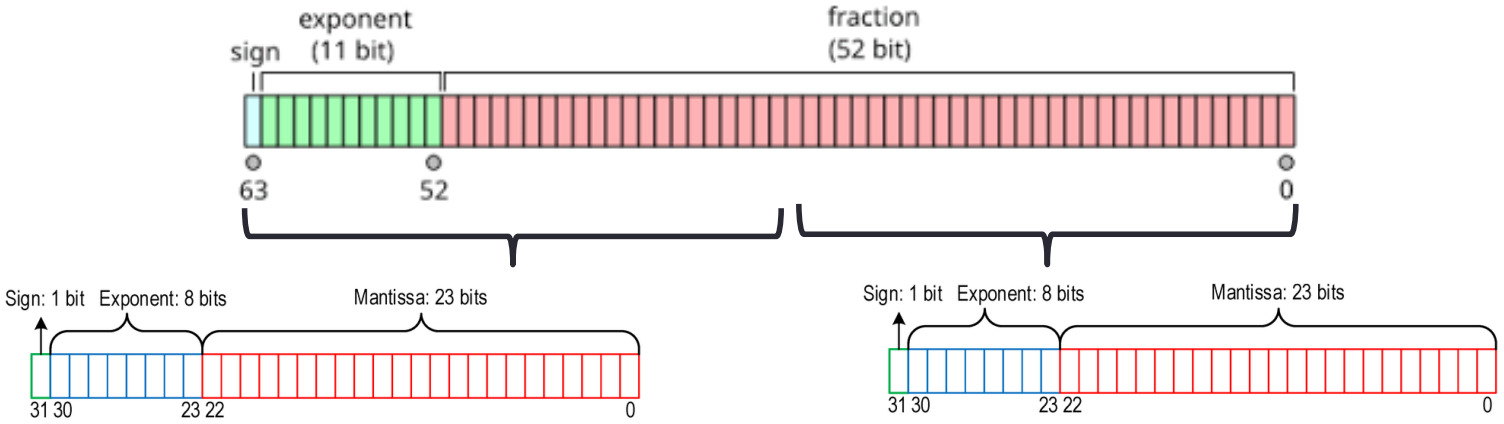
- If I create a np.float64 array, the data will fill a region of memory 64 bits at time.



- The pointer that will get passed to the C routine points to the first memory location
- If the C routine is expecting a “float” (32-bit)
- array, it will step through this like



- which will yield random nonsense!



C++

- C++ functions can “overloaded”; that is, the same function name can refer to different functions depending on the arguments and argument types.
- It handles this by “mangling” the function name to encode the argument list, which is VERY hard to deal directly with in Python.
- There are two ways to deal with this:
  - Easy: Undo the mangling (and the overloading) by declaring the routing ‘extern “C”’, eg

    extern “C” double find_max(double *arr, int n) {

  - Harder (but arguably better): use the Python binding utilities to create proper wrappers, including overloading.
    - Beyond the scope of this course