**First Class Functions:**

First class functions: If a function can be assigned to a variable or passed as object/variable to other function, that function is called as first class function.

First-Class Functions" (FCF) are functions which are treated as so called "First-Class Citizens" (FCC). FCC's in a programming language are objects (using the term "objects" very freely here) which:

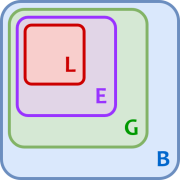
* Can be used as parameters
* Can be used as a return value
* Can be assigned to variables
* Can be stored in data structures such as hash tables, lists, ...

Actually, *very* roughly and simply put, FCF's are variables of the type 'function' (or variables which point to a function). You can do with them everything you can do with a 'normal' variable.

Knowing this, both this\_is\_another\_example(myarg) and this\_is\_example(myarg1) are First-Class Functions, since *all* functions are First-Class in certain programming languages.

**Namespaces in Python**

A namespace is a collection of currently defined symbolic names along with information about the object that each name references. You can think of a namespace as a [dictionary](https://realpython.com/python-dicts) in which the keys are the object names and the values are the objects themselves. Each key-value pair maps a name to its corresponding object.

[](https://files.realpython.com/media/t.fd7bd78bbb47.png)

1. **Local**: If you refer to x inside a function, then the interpreter first searches for it in the innermost scope that’s local to that function.
2. **Enclosing**: If x isn’t in the local scope but appears in a function that resides inside another function, then the interpreter searches in the enclosing function’s scope.
3. **Global**: If neither of the above searches is fruitful, then the interpreter looks in the global scope next.
4. **Built-in**: If it can’t find x anywhere else, then the interpreter tries the built-in scope.

**Global Interpreter Lock (GIL)**

The GIL is a single lock on the interpreter itself which adds a rule that execution of any Python byte code requires acquiring the interpreter lock. This prevents deadlocks (as there is only one lock) and doesn’t introduce much performance overhead. But it effectively makes any CPU-bound Python program single-threaded.

The GIL can obviously be removed and this has been done multiple times in the past by the developers and researchers but all those attempts broke the existing C extensions which depend heavily on the solution that the GIL provides ( this will break the thread safe concept)

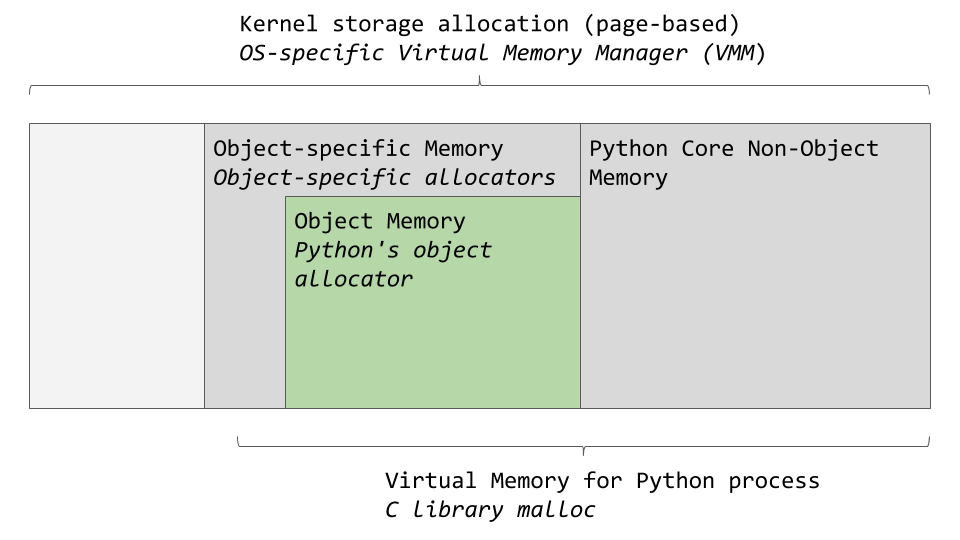
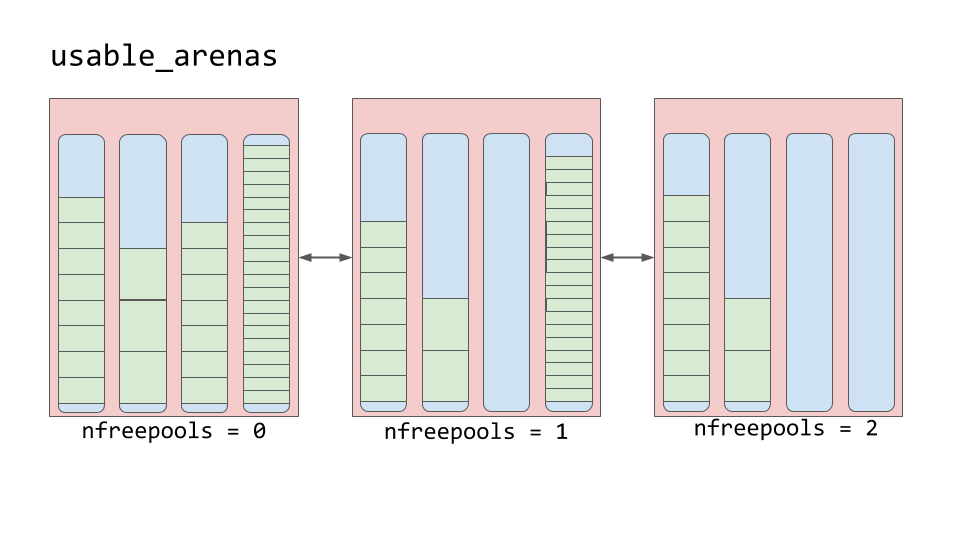
**Memory Management:**

Memory management is the process by which applications read and write data. A memory manager determines where to put an application’s data. Since there’s a finite chunk of memory, like the pages in our book analogy, the manager has to find some free space and provide it to the application. This process of providing memory is generally called memory **allocation.**

Garbage Collection / Free memory removing waste

Python allows you to inspect the current reference count of an object with the sys module. You can use sys.getrefcount(numbers), but keep in mind that passing in the object to getrefcount() increases the reference count by 1.

In any case, if the object is still required to hang around in your code, its reference count is greater than 0. Once it drops to 0, the object has a specific deallocation function that is called which “frees” the memory so that other objects can use it.

[](https://files.realpython.com/media/memory_management.92ad564ec680.png) [](https://files.realpython.com/media/memory_management_6.60e9761bc158.png)