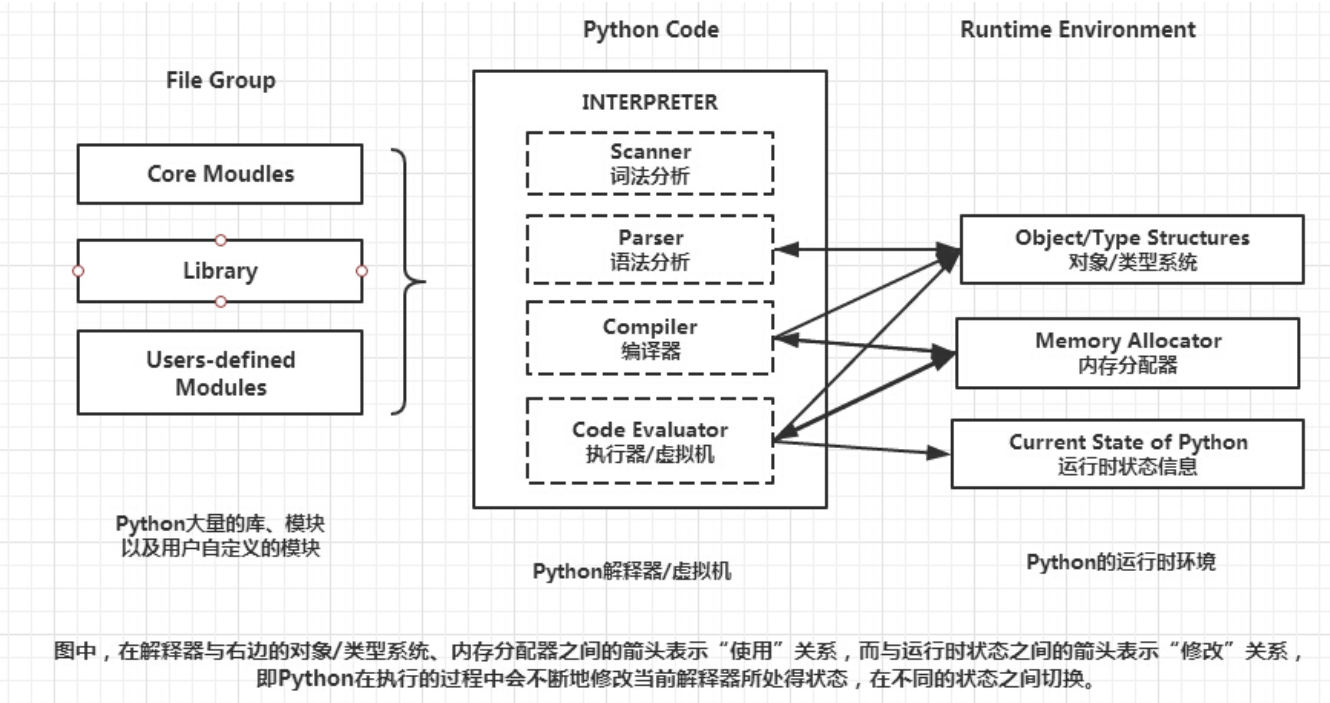
python 解释器及底层运行过程

解释器种类

1. **CPython**：官方版本的解释器。这个解释器是用C语言开发的，所以叫CPython。CPython是使用最广的Python解释器。我们通常说的、下载的、讨论的、使用的都是这个解释器。
2. **Ipython**：基于CPython之上的一个交互式解释器，在交互方式上有所增强，执行Python代码的功能和CPython是完全一样的。CPython用>>>作为提示符，而IPython用In [序号]:作为提示符。
3. **PyPy**：一个追求执行速度的Python解释器。采用JIT技术，对Python代码进行动态编译（注意，不是解释），可以显著提高Python代码的执行速度。绝大部分CPython代码都可以在PyPy下运行，但还是有一些不同的，这就导致相同的Python代码在两种解释器下执行可能会有不同的结果。
4. **Jython**：运行在Java平台上的Python解释器，可以直接把Python代码编译成Java字节码执行。
5. **IronPython**：和Jython类似，只不过IronPython是运行在微软.Net平台上的Python解释器，可以直接把Python代码编译成.Net的字节码。

运行过程



图中，在解释器与右边的对象/类型系统、内存分配器之间的箭头表示“使用”关系，而与运行时状态之间的箭头表示“修改”关系，即Python在执行的过程中会不断地修改当前解释器所处得状态，在不同的状态之间切换。

- pyc文件
  - py文件的解释结果保存下来的文件
  - 直接使用pyc文件就可以了，这无疑大大提高了程序运行速度
  - **对于当前调用的主程序不会生成pyc文件**；
  - **以import xxx或from xxx import xxx等方式导入主程序的模块才会生成pyc文件**；
  - 每次使用pyc文件时，都会根据pyc文件的创建时间和源模块进行对比，如果源模块有修改，则重新创建pyc文件，并覆盖先前的pyc文件，如果没有修改，直接使用pyc文件代替模块；
  - pyc文件统一保存在模块所在目录的**pycache**文件夹内

GIL全局解释锁

- 起初原因：为了解决多线程之间的数据完整性和状态同步问题
- 设计功能：在任意时刻只有一个线程在解释器中运行
- 实现原理：分时复用
- 不去除原因：Cpython 当大家试图去拆分和去除GIL的时候，发现大量库代码开发者已经重度依赖GIL而非常难以去除
- 规避方法：
    - 用multiprocess（多进程）替代Thread（推荐）【多进程而不是多线程】
    - 用其他解析器（不推荐）【小众】