|  |
| --- |
| Python API Design Guideline |
| VER 1.0 Draft 1 |
| Tianjian Chen |

1. **背景Background**

本标准文档应用于封装C库为Python Extension，以达成提供Python API供Python编码使用的目的。该标准适用于NLP、PS se-proc、PS spider、INF Com等组织所维护的基础库接口封装。

1. **标准细节 Specifications**
   1. Python Extension编译环境： 统一为Python 2.7.2版本
   2. 设计模式

鉴于绝大部分基础库提供的是C风格接口，在接口封装时，建议Python接口设计上采用一一对应原则进行封装。假设原C库L中存在函数为FuncA，则对应Python Extension Module命名为L，Python函数接口也命名为FuncA或者相同含义的命名。Python Extension被调用时使用语句：

L.FuncA()

各个C库应当分别封装在自身对应的Python Module中，比如libsegment 对应 wordseg.so，避免出现单个Python Module规模过大的情况。

* 1. 内存管理原则

尽量保持原有C函数库的可重入特性，避免在封装程序中使用静态变量。

直接应用malloc分配的内存，应当在同一个函数过程内释放或通过配对的初始化清除函数释放内存。

使用Python库申请的内存Object，则应当尽量采取借用应用(Borrowed Reference)和引用传递的方法，将引用计数交由上层Python逻辑管理。

* 1. 函数参数，返回值和异常

由于以Python Object作为参数，所以常见的buffer length等参数可以选择不出现在函数参数列表之中。

尽可能使用Python标准数据对象类型作为返回值，如PyInt PyDouble PyString PyList PyDict等

原C库调用表示异常的返回值，需要以Python ExceptionObject的形式向Python逻辑抛出，以方便Python逻辑加强健壮性。

* 1. 接口联用原则

有些C库在使用中存在相互依赖关系，比如NLP的wordseg postag 和wordner，再比如ULlib的sign dict和odict，经常性在一起使用，因此在各自库的Python接口设计中，要考虑到联用的需求，比如wordseg的结果，可以被postag使用，postag的结果可以继续传入wordner标注。

1. **参考代码 Reference Code**

参考SVN路径

RP封装的PyLib

<https://svn.baidu.com/st/rp/trunk/pylib>

NLP封装的PyNLP

<https://svn.baidu.com/public/trunk/pynlp>