**目标**

提供有足够弹性的架构，可供用户撰写自己的扩展，能自行定义

* 命令。类似谷歌拼音的 i 模式。这个模式可用于输入用户自定义的一些映射，比如日期，中文标点等。比如用户输入 iqianming，就会自动调用 fortune，输出名人名言。
* 自动匹配。这类似 Emacs 的 abbrev 展开功能，可以让用户无论何时只要输入rj，就输出”人间大炮一级准备“。
* 输入法。举例来说：
  + 双拼码表。让用户能自定义双拼的映射，从而通过编写简单的脚本，得到符合自己习惯的双拼输入法。
  + 甚至能调用外部输入模块，和 SunPinyin 协同合作。比如实现简单的英文辅助输入功能。

**初步构想**

为了实现 command 和 abbrev 功能，就需要一个支持脚本的 wrapper。这个 wrapper 实现了

* 基本的 session 管理
  + 原来的 CIMIView 是 sunpinyin 的高层数据结构，它持有其他对象的引用。但是引入了 wrapper 之后，wrapper 会持有 CIMIView 的引用，所以在 Python binding 里面，我们可以沿用 incref 和 decref 的机制来维护 CIMIView 的生命周期。可以考虑用 boost::shared\_ptr 来实现之。
* 加载用户脚本
* 在合适的时候，调用脚本

因此，我们需要能调用脚本。目前，我们将目标语言暂定为 Python。所以 wrapper 会

1. 初始化 python 的运行时，
2. 加载用户脚本
3. 在截获用户输入时，在合适的时机，把用户输入发送给脚本进行处理，将结果放到 preedit 区域，或者直接上屏。

为了实现更进一步的输入法自定义功能，wrapper 还需要把现有的 sunpinyin 接口暴露出来，让 Python 脚本调用。下面列举了一些输入法用到的类，按照重要程度从高到低排序：

* CIMIView （维护着输入的状态）
* CKeyEvent （用户输入）
* IPreeditString（用于表示 preedit 字串）
* CIMIContext （build lattice，和用户词库有关）
* CSimplifiedChinesePolicy (标点映射）
* QuanpinSchemePolicy/ShuangpinSchemePolicy (全拼双拼的配置)
* CHotkeyProfile （快捷键的设置，比如翻页键）
* CSunpinyinSessionFactory (负责配置和生成 CIMIView，不考虑）
* CIMIWinHandler (和输入法平台相关，不考虑)

接下来就是为需要暴露的接口编写 pythonic 的 python 绑定。