

14. 交互式编辑和编辑历史

某些版本的 Python 解释器支持编辑当前输入行和编辑历史记录，类似 Korn shell 和 GNU Bash shell 的功能。这个功能使用了 [GNU Readline](#) 来实现，一个支持多种编辑方式的库。这个库有它自己的文档，在这里我们就不重复说明了。

14.1. Tab 补全和编辑历史

在解释器启动的时候，补全变量和模块名的功能将 [自动打开](#)，以便在按下 Tab 键的时候调用补全函数。它会查看 Python 语句名称，当前局部变量和可用的模块名称。处理像 `string.a` 的表达式，它会求值在最后一个 `'.'` 之前的表达式，接着根据求值结果对象的属性给出补全建议。如果拥有 `__getattr__()` 方法的对象是表达式的一部分，注意这可能会执行程序定义的代码。默认配置下会把编辑历史记录保存在用户目录下名为 `.python_history` 的文件。在下次 Python 解释器会话期间，编辑历史记录仍旧可用。

14.2. 默认交互式解释器的替代品

Python 解释器与早期版本的相比，向前迈进了一大步；无论如何，还有些希望的功能：如果能在编辑连续行时建议缩进（解析器知道接下来是否需要缩进符号），那将很棒。补全机制可以使用解释器的符号表。有命令去检查（甚至建议）括号，引号以及其他符号是否匹配。

一个可选的增强型交互式解释器是 [IPython](#)，它已经存在了有一段时间，它具有 `tab` 补全，探索对象和高级历史记录管理功能。它还可以彻底定制并嵌入到其他应用程序中。另一个相似的增强型交互式环境是 [bpython](#)。

© 版权所有 2001-2022, Python Software Foundation.

This page is licensed under the Python Software Foundation License Version 2.

Examples, recipes, and other code in the documentation are additionally licensed under the Zero Clause BSD License.

See [History and License](#) for more information.

The Python Software Foundation is a non-profit corporation. [Please donate.](#)

最后更新于 9月 15, 2022. [Found a bug?](#)

Created using [Sphinx](#) 3.4.3.



3.10.7

🔍

转向