16. 附录

16.1. 交互模式

交互式 REPL 有两个变种版本。 经典的基本解释器在所有平台上受到支持，具有最小化的行控制功能。

在 Windows 上，或在具有 curses 支持的类 Unix 系统上，从 Python 3.13 开始默认会使用一个新的交互式 shell。 它支持彩色、多行编辑、历史浏览和粘贴模式。 要禁用彩色，请参阅 控制颜色 了解详情。 功能键将提供一些附加功能。 F1 是进入交互式帮助浏览器 pydoc。 F2 允许浏览不带输出也不带 >>> 和 ... 提示符的命令行历史。 F3 是进入"粘贴模式"，这可以更方便地粘贴大段代码。 按 F3 将返回常规提示符。

当使用新的交互式 shell 时，可通过键入 exit 或 quit 退出 shell。 不再需要在这些命令之后添加代表调用的圆括号。

如果不想要新的交互式 shell，可以通过 PYTHON_BASIC_REPL 环境变量禁用它。

16.1.1. 错误处理

当发生错误时，解释器会打印错误消息和栈回溯。 在交互模式下，将返回到主提示符；当输入是来自文件的时候，它将在打印栈回溯之后退出并附带一个非零的退出状态码。 （由 try 语句中 except 子句所处理的异常在此上下文中不属于错误。） 有些错误属于无条件致命错误，会导致程序附带非零状态码退出；这适用于内部一致性丧失以及某些内存耗尽的情况等。 所有错误消息都将被写入到标准错误流；来自被执行命令的正常输出测会被写入到标准输出。

将中断字符（通常为 Control-C 或 Delete ）键入主要或辅助提示符会取消输入并返回主提示符。 [1] 在执行命令时键入中断引发的 KeyboardInterrupt 异常，可以由 try 语句处理。

16.1.2. 可执行的Python脚本

在 BSD 等类Unix系统上，Python 脚本可以像 shell 脚本一样直接执行，通过在第一行添加：

```
#!/usr/bin/env python3
```

（假设解释器位于用户的 PATH ）脚本的开头，并将文件设置为可执行。 `#!` 必须是文件的前两个字符。在某些平台上，第一行必须以Unix样式的行结尾（`'\n'`）结束，而不是以Windows（`'\r\n'`）行结尾。注意，"散列字符"，或者说"磅字符"， `'#'` ，在Python中代表注释开始。

可以使用 **chmod** 命令为脚本提供可执行模式或权限。

```
$ chmod +x myscript.py
```

在Windows系统上，没有"可执行模式"的概念。 Python安装程序自动将 `.py` 文件与 `python.exe` 相关联，这样双击Python文件就会将其作为脚本运行。扩展也可以是 `.pyw` ，在这种情况下，会隐藏通常出现的控制台窗口。

16.1.3. 交互式启动文件

当您以交互模式使用 Python 时，您可能会希望在每次启动解释器时，解释器先执行几条您预先编写的命令，然后您再以交互模式继续使用。您可以通过将名为 `PYTHONSTARTUP` 的环境变量设置为包含启动命令的文件的文件名来实现。这类似于 Unix shell 的 `.profile` 功能。

Python 只有在交互模式时，才会读取此文件，而非在从脚本读指令或是将 `/dev/tty` 显式作为被运行的 Python 脚本的文件名时（后者反而表现得像一个交互式会话）。这个文件与交互式指令共享相同的命名空间，所以它定义或导入的对象可以在交互式会话中直接使用。您也可以在该文件中更改提示符 `sys.ps1` 和 `sys.ps2`。

如果您想 *从当前目录中* 读取一个额外的启动文件，您可以在上文所说的全局启动文件中编写像 `if os.path.isfile('.pythonrc.py'): exec(open('.pythonrc.py').read())` 这样的代码。如果要在脚本中使用启动文件，则必须在脚本中显式执行此操作：

```
import os
filename = os.environ.get('PYTHONSTARTUP')
if filename and os.path.isfile(filename):
    with open(filename) as fobj:
        startup_file = fobj.read()
    exec(startup_file)
```

16.1.4. 定制模块

Python 提供了两个钩子供你进行自定义: sitecustomize 和 usercustomize。 要了解它是如何工作的，首先需要找到用户 site-packages 目录的位置。 启动 Python 并运行以下代码:

```
>>> import site
>>> site.getusersitepackages()
'/home/user/.local/lib/python3.x/site-packages'
```

现在，您可以在该目录中创建一个名为 `usercustomize.py` 的文件，并将所需内容放入其中。它会影响Python的每次启动，除非它以 `-s` 选项启动，以禁用自动导入。

sitecustomize 的工作方式相同，但通常由计算机管理员在全局 site-packages 目录中创建，并在 usercustomize 之前导入。 更多细节请参阅 `site` 模块的文档。

备注

[1] GNU Readline 包的问题可能会阻止这种情况。