Python Plumbum 简介：用 Python 来写脚本

作者：Tomer Filiba 译者：LCTT qhh0205 | 2018-03-22 10:11 收藏: 2

Plumbum：Shell 组合器

你是否曾希望将 shell 脚本紧凑地融入到**真正**的编程语言里面？ 那么可以了解下 Plumbum Shell 组合器。Plumbum （lead 的拉丁语，以前用来制作管道）是一个小型但功能丰富的类库，用于以 Python 进行类似 shell 脚本编程。该库的理念是 “**永远不要再写 shell 脚本**”，因此它试图合理地模仿 shell 语法（shell 组合器），同时保持 **Python 特性和跨平台**。

除了 **类似 shell 的语法** 和 **便捷的快捷方式** 之外，该库还提供本地和 **远程** 命令执行（通过 SSH）、本地和远程文件系统 **路径**、简单的工作目录和环境 **操作**、快捷访问 ANSI 颜色，以及 **编程命令行接口（CLI）** 应用程序工具包。现在让我们看一些代码！

其最新版本 1.6.6 发布于 2018 年 2 月 12 日。

快捷使用指南

基本使用

```
>>> from plumbum import local
```

```
>>> ls
LocalCommand(<LocalPath /bin/ls>)
>>> ls()
u'build.py\ndist\ndocs\nLICENSE\nplumbum\nREADME.rst\nsetup.py\ntests\ntodo.txt\n'
>>> notepad = local["c:\\windows\\notepad.exe"]
>>> notepad()                                   # Notepad window pops up
u''                                             # Notepad window is closed by user, command returns
```

不需要为每个你想使用的命令写 `xxx = local["xxx"]`，你可以**导入命令行**：

```
>>> from plumbum.cmd import grep, wc, cat, head
>>> grep
LocalCommand(<LocalPath /bin/grep>)
```

参见**本地命令行**。

管道

```
>>> chain = ls["-a"] | grep["-v", "\\.py"] | wc["-l"]
>>> print chain
/bin/ls -a | /bin/grep -v '\.py' | /usr/bin/wc -l
>>> chain()
u'13\n'
```

重定向

```
>>> ((cat < "setup.py") | head["-n", 4])()
u'#!/usr/bin/env python\nimport os\n\ntry:\n'
>>> (ls["-a"] > "file.list")()
u''
>>> (cat["file.list"] | wc["-l"])()
u'17\n'
```

参见 **输入/输出重定向** 。

工作目录操作

```
>>> local.cwd
<Workdir /home/tomer/workspace/plumbum>
>>> with local.cwd(local.cwd / "docs"):
...     chain()
...
u'15\n'
```

参见 **路径** 和 **本地对象** 。

前台后和后台执行

```
>>> from plumbum import FG, BG
>>> (ls["-a"] | grep["\\.py"]) & FG         # The output is printed to stdout directly
build.py
.pydevproject
setup.py
>>> (ls["-a"] | grep["\\.py"]) & BG         # The process runs "in the background"
<Future ['/bin/grep', '\\.py'] (running)>
```

参见 **前台和后台** 。

命令行嵌套

```
>>> from plumbum.cmd import sudo
>>> print sudo[ifconfig["-a"]]
/usr/bin/sudo /sbin/ifconfig -a
>>> (sudo[ifconfig["-a"]] | grep["-i", "loop"]) & FG
lo        Link encap:Local Loopback
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
```

参见 **命令行嵌套** 。

远程命令（通过 SSH）

```
>>> from plumbum import SshMachine
>>> remote = SshMachine("somehost", user = "john", keyfile = "/path/to/idrsa")
>>> r_ls = remote["ls"]
>>> with remote.cwd("/lib"):
...     (r_ls | grep["0.so.0"])()
...
u'libusb-1.0.so.0\nlibusb-1.0.so.0.0.0\n'
```

参见 远程 。

CLI 应用程序

```
import logging
from plumbum import cli

class MyCompiler(cli.Application):
    verbose = cli.Flag(["-v", "--verbose"], help = "Enable verbose mode")
    include_dirs = cli.SwitchAttr("-I", list = True, help = "Specify include directories")

    @cli.switch("-loglevel", int)
    def set_log_level(self, level):
        """Sets the log-level of the logger"""
        logging.root.setLevel(level)

    def main(self, *srcfiles):
```

```
        print "Compiling:", srcfiles

if __name__ == "__main__":
    MyCompiler.run()
```

输出样例：

```
$ python simple_cli.py -v -I foo/bar -lspam/eggs x.cpp y.cpp z.cpp
Verbose: True
Include dirs: ['foo/bar', 'spam/eggs']
Compiling: ('x.cpp', 'y.cpp', 'z.cpp')
```

参见 **命令行应用程序** 。

颜色和风格

```
from plumbum import colors
with colors.red:
    print("This library provides safe, flexible color access.")
    print(colors.bold | "(and styles in general)", "are easy!")
print("The simple 16 colors or",
      colors.orchid & colors.underline | '256 named colors,',
      colors.rgb(18, 146, 64) | "or full rgb colors",
      'can be used.')
```

输出样例：

```
This library provides safe color access.
Color (and styles in general) are easy!
The simple 16 colors, 256 named colors, or full hex colors can be used.
Unsafe color access is available too.
```

参见 **颜色** 。

开发和安装

该库在 **Github** 上开发，非常乐意接受来自用户的 **补丁** 。请使用 GitHub 的内置 **issue 跟踪器** 报告您遇到的任何问题或提出功能上的需求。该库在 **IMIT 许可** 下发布。

要求

Plumbum 支持 Python 2.6-3.6 和 PyPy，并且通过 **Travis CI** 和 **Appveyor** 持续地在 Linux，Mac 和 Windows 机器上测试。Plumbum 在任何类 Unix 的平台应该可以正常工作，但是在 Windows 上，你也许需要安装一个合适的 **coreutils** 环境并把其加入到你的 `PATH` 环境变量中。我推荐 **mingw** （与 **Windows Git** 捆绑在一起），但是 **cygwin** 应该也可以。如果你仅仅是使用 Plumbum 代替 Popen 来运行 Windows 程序，那么就不需要 Unix 工具了。 注意远程命令的执行，你需要一个 openSHH 兼容的客户端（同样与 Windows Git 捆绑在一起）和一个 bash 兼容的 shell，也需要在主机上有一个 coreutils 环境。

下载

你可以在 **Python Package Index** （多种格式）下载该库，或者直接运行 `pip install plumbum` 。如果你使用 Anaconda，你可以使用 `conda install -c conda-forge plumbum` 从 `conda-forge` 通道获取。

用户指南

用户指南涵盖了 Plumbum 大部分功能，拥有大量的代码片段，你可以不用花多少时间即可开始使用。该指南逐渐介绍概念和"语法"，因此推荐你按照顺序阅读。一个 **有效的快速参考指南** 。 略。。。

关于

Plumburn 最初的目的是让本地和远程程序轻松地执行，假设没有比老的 ssh 更时髦的东西了。在此基础上，设计了一个文件系统抽象层，以便能够无缝地处理[illegible]远程文件。 我有这个想法一段时间了，直到我必须要个给我当前工作的项目写一个构建脚本，我决定使用 shell 脚本，现在是实现它的时候了。Plumbum 诞生[illegible] `Path` 类的片段和我为 **RPyC** 写的 `SshContext` 和 `SshTunnel` 类。 `Path` 类是我为前面说的构建系统写的。当我将两者与 shell 组合器（因为 shell 脚本在[illegible]确实有优势）组合在一起时，奇迹就发生了，便产生了Plumbun。

致谢

该项目受到了 **Andrew Moffat** 的 **PBS**（现在被称作 **sh**）启发，并且借用了他的一些思想（即像函数一样看待程序，导入命令行的技巧）。然而我感觉在 PBS 中有太多的神秘的东西，当我编写类 shell 程序时，语法不是我想要的。关于这个问题我联系了 Andrew，但是他想让 PBS 保持这种状态。除此之外，两个库走不同的方向，Plumbum 试图提供一种更合理的方法。

Plumbum 也向 **Rotem Yaari** 致敬，他为特定的目的建议了一个代号为 `pyplatform` 的库，但是尚未实现过。

上一篇：用 Python 构建你自己的 RSS 提示系统
下一篇：如何用 Python 解析 HTML

LCTT 译者

qhh0205
共计翻译：1.0 篇 | 共计贡献：24 天
贡献时间：2018-02-16 -> 2018-03-11
访问我的 LCTT 主页 | 在 GitHub 上关注

相关阅读

Shell 更多标签

学习 Shell 脚本编程的免费资源 2020-07-04
基础：如何在 Linux 中运行一个 Shell 脚本 2021-02-10
[免费训练营] Shell 编程训练营，它来了 2021-08-18

如何在 Bash shell 脚本中解析命令行选项 2021-08-29
Xonsh Shell：在 Linux 终端结合 Bash Shell 2022-03-24