Python在Windows上的常见问题

目录

我怎样在Windows下运行一个Python程序?

这不一定是一个简单的问题。如果你已经熟悉在Windows的命令行中运行程序的方法，一切都显而易见；不然的话，你也许需要额外获得些许指导。

除非你使用某种集成开发环境，否则你最终会在所谓的 "命令提示窗口 "中 *输入* Windows命令。 通常情况下，你可以在搜索栏中搜索 `cmd` 来创建这样一个窗口。你应该能够发现你已经启动了这样一个窗口，因为你会看到一个 Windows "命令提示符"，它通常看起来像这样。

```
C:\>
```

前面的字母可能会不同，而且后面有可能会有其他东西，所以你也许会看到类似这样的东西：

```
D:\YourName\Projects\Python>
```

出现的内容具体取决与你的电脑如何设置和最近用它做的事。 当你启动了这样一个窗口后，就可以开始运行Python程序了。

Python 脚本需要被另外一个叫做 Python *解释器* 的程序来处理。 解释器读取脚本，把它编译成字节码，然后执行字节码来运行你的程序。 所以怎样安排解释器来处理你的 Python 脚本呢?

首先，确保命令窗口能够将“py”识别为指令来开启解释器。 如果你打开过一个命令窗口， 尝试输入命令 `py` 然后按回车：

```
C:\Users\YourName> py
```

然后你应当看见类似类似这样的东西：

```
Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017, 06:04:45) [MSC v.1900 32 bit (Intel)] c
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

解释器已经以“交互模式”打开。这意味着你可以交互输入Python语句或表达式，并在等待时执行或评估它们。这是Python最强大的功能之一。输入几个表达式并看看结果：

```
>>> print("Hello")
Hello
>>> "Hello" * 3
'HelloHelloHello'
```

许多人把交互模式当作方便和高度可编程的计算器。 想结束交互式Python会话时，调用 `exit()` 函数，或者按住 `Ctrl` 键时输入 `Z` ，之后按 `Enter` 键返回Windows命令提示符。

你可能发现在开始菜单有这样一个条目 开始 ‣ 所有程序 ‣ Python 3.x ‣ Python (命令行)， 运行它后会出现一个有着 `>>>` 提示的新窗口。 在此之后，如果调用 `exit()` 函数或按 `Ctrl-Z` 组合键后窗口将会消失。 Windows 会在这个窗口中运行一个“python”命令，并且在你终止解释器的时候关闭它。

现在我们知道 `py` 命令已经被识别，可以输入 Python 脚本了。 你需要提供 Python 脚本的绝对路径或相对路径。 假设 Python 脚本位于桌面上并命名为 `hello.py`，并且命令提示符在用户主目录打开，那么可以看到类似于这样的东西:

```
C:\Users\YourName>
```

那么现在可以让 `py` 命令执行你的脚本，只需要输入 `py` 和脚本路径:

```
C:\Users\YourName> py Desktop\hello.py
hello
```

我怎么让 Python 脚本可执行?

在 Windows 上，标准 Python 安装程序已将 .py 扩展名与文件类型 (Python.File) 相关联，并为该文件类型提供运行解释器的打开命令 (`D:\Program Files\Python\python.exe "%1" %*`)。 这足以使脚本在命令提示符下作为“foo.py”命令被执行。 如果希望通过简单地键入“foo”而无需输入文件扩展名来执行脚本，则需要将 .py 添加到 PATHEXT 环境变量中。

为什么有时候 Python 程序会启动缓慢?

通常，Python 在 Windows 上启动得很快，但偶尔会有错误报告说 Python 突然需要很长时间才能启动。更令人费解的是，在其他配置相同的 Windows 系统上，Python 却可以工作得很好。

该问题可能是由于计算机上的杀毒软件配置错误造成的。当将病毒扫描配置为监视文件系统中所有读取行为时，一些杀毒扫描程序会导致两个数量级的启动开销。请检查你系统安装的杀毒扫描程序的配置，确保两台机它们是同样的配置。已知的， McAfee 杀毒软件在将它设置为扫描所有文件系统访问时，会产生这个问题。

我怎样使用 Python 脚本制作可执行文件?

请参阅 如何由 Python 脚本创建能独立运行的二进制程序? 查看可用来生成可执行文件的工具清单。

*.pyd 文件和 DLL 文件相同吗?

是的， .pyd 文件也是 dll ，但有一些差异。如果你有一个名为 `foo.pyd` 的DLL，那么它必须有一个函数 `PyInit_foo()` 。 然后你可以编写 Python 代码 “import foo”， Python 将搜索 foo.pyd （以及 foo.py 、 foo.pyc ）。如果找到它，将尝试调用 `PyInit_foo()` 来初始化它。你不应将 .exe 与 foo.lib 链接，因为这会导致 Windows 要求存在 DLL 。

请注意， foo.pyd 的搜索路径是 PYTHONPATH ，与 Windows 用于搜索 foo.dll 的路径不同。此外， foo.pyd 不需要存在来运行你的程序，而如果你将程序与 dll 链接，则需要 dll 。 当然，如果你想 `import foo` ，则需要 foo.pyd 。在 DLL 中，链接在源代码中用 `__declspec(dllexport)` 声明。在 .pyd 中，链接在可用函数列表中定义。

我怎样将 Python 嵌入一个 Windows 程序?

在 Windows 应用程序中嵌入 Python 解释器可以总结如下：

1. 请 **不要** 直接将 Python 编译到你的 .exe 文件中。 在 Windows 上，Python 必须是一个 DLL 以便处理导入本身就是 DLL 的模块。 （这是首先要知道的未写入文档的关键事实。） 正确的做法，应该是链接到 `pythonNN.dll`；它通常安装在 `C:\Windows\System` 中。 *NN* 是 Python 的版本号，例如数字 "33" 代表 Python 3.3。

   你可以通过两种不同的方式链接到 Python 。加载时链接意味着链接到 `pythonNN.lib` ，而运行时链接意味着链接 `pythonNN.dll` 。 （一般说明： `python NN.lib` 是所谓的“import lib”，对应于 `pythonNN.dll` 。它只定义了链接器的符号。）

   运行时链接极大地简化了链接选项，一切都在运行时发生。你的代码必须使用 Windows 的 `LoadLibraryEx()` 程序加载 `pythonNN.dll` 。代码还必须使用使用 Windows 的 `GetProcAddress()` 例程获得的指针访问 `pythonNN.dll` 中程序和数据（即 Python 的 C API ）。宏可以使这些指针对任何调用 Python C API 中的例程的 C 代码都是透明的。

2. 如果你是使用 SWIG，那么很容易创建一个将使得应用的数据和方法可供 Python 使用的 "扩展模块"。 SWIG 将为你处理所有繁琐的细节。 结果是让你链接 *置入* 你的 .exe 文件当中的 C 代码 (!) 你 **无需** 创建一个 DLL 文件，而这也简化了链接过程。

3. SWIG 将创建一个 init 函数（一个 C 函数），其名称取决于扩展模块的名称。例如，如果模块的名称是 leo ，则 init 函数将被称为 initleo() 。 如果您使用 SWIG 阴影类，则 init 函数将被称为 initleoc() 。这初始化了一个由阴影类使用的隐藏辅助类。

   你可以将步骤 2 中的 C 代码链接到 .exe 文件的原因是调用初始化函数等同于将模块导入 Python ！ （这是第二个关键的未记载事实。）

4. 简而言之，你可以用以下代码使用扩展模块初始化 Python 解释器。

```
#include <Python.h>
...
Py_Initialize();  // 初始化 Python。
initmyAppc();  // 初始化（导入）辅助类。
PyRun_SimpleString("import myApp");  // 导入影子类。
```

5. Python C API 存在两个问题，如果你使用除 MSVC 之外的编译器用于构建 python.dll ，这将会变得明显。

   问题 1: 接受 `FILE *` 参数的所谓的 "极高层级" 函数在多编译器环境中将不起作用，因为每个编译器中 `struct FILE` 的概念都会是不同的。 从实现的角度看来这些都是极低层级的函数。

   问题2：在为 void 函数生成包装器时，SWIG 会生成以下代码：

```
Py_INCREF(Py_None);
_resultobj = Py_None;
return _resultobj;
```

   Py_None 是一个宏，它扩展为对 pythonNN.dll 中名为 _Py_NoneStruct 的复杂数据结构的引用。同样，此代码将在多编译器环境中失败。将此类代码替换为：

```
return Py_BuildValue("");
```

   有可能使用 SWIG 的 `%typemap` 命令自动进行更改，但我无法使其工作（我是一个完全的SWIG新手）。

6. 使用 Python shell 脚本从 Windows 应用程序内部建立 Python 解释器窗口并不是一个好主意；生成的窗口将独立于应用程序的窗口系统。相反，你（或 wxPythonWindow 类）应该创建一个“本机”解释器窗口。将该窗口连接到Python解释器很容易。你可以将 Python的 i/o 重定向到支持读写的 _任意_ 对象，因此你只需要一个包含 read() 和 write() 方法的 Python 对象（在扩展模块中定义）。

如何让编辑器不要在我的 Python 源代码中插入 tab ？

本 FAQ 不建议使用制表符， Python 样式指南 **PEP 8** ，为发行的 Python 代码推荐 4 个空格；这也是 Emacs python-mode 默认值。

在任何编辑器下，混合制表符和空格都是一个坏主意。 MSVC 在这方面没有什么不同，并且很容易配置为使用空格： 点击 Tools ‣ Options ‣ Tabs，对于文件类型“Default”，设置“Tab size”和“Indent size”为 4 ，并选择“插入空格”单选按钮。

如果混合制表符和空格导致前导空格出现问题， Python 会引发 `IndentationError` 或 `TabError` 。你还可以运行 `tabnanny` 模块以批处理模式检查目录树。

如何在不阻塞的情况下检查按键？

使用 `msvcrt` 模块。 这是一个标准的 Windows 专属扩展模块。 它定义了一个函数 `kbhit()` 用于检查是否有键盘中的某个键被按下，以及 `getch()` 用于获取一个字符而不将其回显。

我该如何解决缺失 api-ms-win-crt-runtime-l1-1-0.dll 错误?

这将在使用未安装全部更新的 Windows 8.1 或更旧的系统时发生于 Python 3.5 及之后的版本上。 首先请确保你的操作系统受支持并且已经更新补丁，如果此问题仍未解决，请访问 Microsoft support page 获取有关手动安装 C 运行时更新补丁的指导。