13.6 执行外部命令并获取它的输出¶

问题¶

你想执行一个外部命令并以Python字符串的形式获取执行结果。

解决方案¶

使用 `subprocess.check_output()` 函数。例如：

```
import subprocess
out_bytes = subprocess.check_output(['netstat','-a'])
```

这段代码执行一个指定的命令并将执行结果以一个字节字符串的形式返回。 如果你需要文本形式返回，加一个解码步骤即可。例如：

```
out_text = out_bytes.decode('utf-8')
```

如果被执行的命令以非零码返回，就会抛出异常。 下面的例子捕获到错误并获取返回码：

```
try:
    out_bytes = subprocess.check_output(['cmd','arg1','arg2'])
except subprocess.CalledProcessError as e:
    out_bytes = e.output       # Output generated before error
    code      = e.returncode   # Return code
```

默认情况下，`check_output()` 仅仅返回输入到标准输出的值。 如果你需要同时收集标准输出和错误输出，使用 `stderr` 参数：

```
out_bytes = subprocess.check_output(['cmd','arg1','arg2'],
                                    stderr=subprocess.STDOUT)
```

如果你需要用一个超时机制来执行命令，使用 `timeout` 参数：

```
try:
    out_bytes = subprocess.check_output(['cmd','arg1','arg2'], timeout=5)
except subprocess.TimeoutExpired as e:
    ...
```

通常来讲，命令的执行不需要使用到底层shell环境（比如sh、bash）。 一个字符串列表会被传递给一个低级系统命令，比如 `os.execve()` 。 如果你想让命令被一个shell执行，传递一个字符串参数，并设置参数 `shell=True` . 有时候你想要Python去执行一个复杂的shell命令的时候这个就很有用了，比如管道流、I/O重定向和其他特性。例如：

```
out_bytes = subprocess.check_output('grep python | wc > out', shell=True)
```

需要注意的是在shell中执行命令会存在一定的安全风险，特别是当参数来自于用户输入时。 这时候可以使用 `shlex.quote()` 函数来将参数正确的用双引用引起来。

讨论¶

使用 `check_output()` 函数是执行外部命令并获取其返回值的最简单方式。 但是，如果你需要对子进程做更复杂的交互，比如给它发送输入，你得采用另外一种方法。 这时候可直接使用 `subprocess.Popen` 类。例如：

```
import subprocess

# Some text to send
text = b'''
hello world
this is a test
goodbye
'''
```

```
# Launch a command with pipes
p = subprocess.Popen(['wc'],
          stdout = subprocess.PIPE,
          stdin = subprocess.PIPE)

# Send the data and get the output
stdout, stderr = p.communicate(text)

# To interpret as text, decode
out = stdout.decode('utf-8')
err = stderr.decode('utf-8')
```

`subprocess` 模块对于依赖TTY的外部命令不合适用。 例如，你不能使用它来自动化一个用户输入密码的任务（比如一个ssh会话）。 这时候，你需要使用到第三方模块了，比如基于著名的 `expect` 家族的工具（pexpect或类似的）