

Shell Commands in IPython

You can use any command that works at the command line in IPython by prefixing it with the `!` character. For example, the `ls`, `pwd`, and `echo` commands can be run as follows:

```
In [1]: !ls
myproject.txt

In [2]: !pwd
/home/jake/projects/myproject

In [3]: !echo "printing from the shell"
printing from the shell
```

Passing Values to and from the Shell

Shell commands can not only be called from IPython, but can also be made to interact with the IPython namespace. For example, you can save the output of any shell command to a Python list using the assignment operator:

```
In [4]: contents = !ls

In [5]: print(contents)
['myproject.txt']

In [6]: directory = !pwd

In [7]: print(directory)
['/Users/jakevdp/notebooks/tmp/myproject']
```

Note that these results are not returned as lists, but as a special shell return type defined in IPython:

```
In [8]: type(directory)
IPython.utils.text.SList
```

This looks and acts a lot like a Python list, but has additional functionality, such as the `grep` and `fields` methods and the `s`, `n`, and `p` properties that allow you to search, filter, and display the results in convenient ways. For more information on these, you can use IPython's built-in help features.

Communication in the other direction—passing Python variables into the shell—is possible through the `{varname}` syntax:

```
In [9]: message = "hello from Python"

In [10]: !echo {message}
hello from Python
```

The curly braces contain the variable name, which is replaced by the variable's contents in the shell command.

Shell-Related Magic Commands

If you play with IPython's shell commands for a while, you might notice that you cannot use `!cd` to navigate the filesystem:

```
In [11]: !pwd  
/home/jake/projects/myproject
```

```
In [12]: !cd ..
```

```
In [13]: !pwd  
/home/jake/projects/myproject
```

The reason is that shell commands in the notebook are executed in a temporary sub-shell. If you'd like to change the working directory in a more enduring way, you can use the `%cd` magic command:

```
In [14]: %cd ..  
/home/jake/projects
```

In fact, by default you can even use this without the `%` sign:

```
In [15]: cd myproject  
/home/jake/projects/myproject
```

This is known as an automagic function, and this behavior can be toggled with the `%automagic` magic function.

Besides `%cd`, other available shell-like magic functions are `%cat`, `%cp`, `%env`, `%ls`, `%man`, `%mkdir`, `%more`, `%mv`, `%pwd`, `%rm`, and `%rmdir`, any of which can be used without the `%` sign if `automagic` is on. This makes it so that you can almost treat the IPython prompt as if it's a normal shell:

```
In [16]: mkdir tmp
```

```
In [17]: ls  
myproject.txt tmp/
```

```
In [18]: cp myproject.txt tmp/
```

```
In [19]: ls tmp  
myproject.txt
```

```
In [20]: rm -r tmp
```

This access to the shell from within the same terminal window as your Python session means that there is a lot less switching back and forth between interpreter and shell as you write your Python code.