Learn the Python interpreter

COMMAND LINE AUTOMATION IN PYTHON

Noah Gift

Lecturer, Northwestern & UC Davis & UC Berkeley | Founder, Pragmatic AI Labs

Three Laws of Automation

1. Any task that is talked about being automated, will eventually be automated
2. If it isn't automated it is broken
3. If a human is doing it, a machine eventually will do it better

Major Learning Objectives

- IPython shell commands
- Shell commands with subprocess
- Walking the file system
- Command-line functions

Using IPython with shell commands

- The `!` syntax executes shell commands

```
# Show free disk
!df -h
```

```
Filesystem      Size  Used Avail Use% Mounted on
overlay         335G  176G  159G  53% /
tmpfs            64M     0   64M   0% /dev
tmpfs           7.7G     0  7.7G   0% /sys/fs/cgroup
/dev/nvme0n1p1  335G  176G  159G  53% /etc/hosts
```

Capturing output from IPython shell commands

- Output of command can be assigned to a variable

```
ls = !ls
```

  - The type is an SList

```
type(ls)
```

```
IPython.utils.text.SList
```

Pure Python vs IPython

- The `!` character will throw a syntax error in Python

```
?  ~ python foo.py
  File "foo.py", line 1
    !ls
    ^
SyntaxError: invalid syntax
```

  - The subprocess module can perform equivalent actions

Passing programs to the Python interpreter

- Two ways to execute Python code to an interpreter
- Passing a script to the Python interpreter

```
python hello.py
```

  - Passing a program to the Python interpreter via `-c`

```
python -c "import datetime;print(datetime.datetime.utcnow())"
```

```
2019-04-01 01:04:47.17224
```

Practicing with the IPython shell

COMMAND LINE AUTOMATION IN PYTHON

Capture IPython Shell output

COMMAND LINE AUTOMATION IN PYTHON

Noah Gift

Lecturer, Northwestern & UC Davis & UC Berkeley | Founder, Pragmatic AI Labs

Unix Pipes

- Unix Philosophy
- Simple Tools
- Combine for Sophisticated Solutions

Understand Unix Pipes

- Using Unix Pipes to count the size of python files

```
# Sum them up using `awk`
ls -l | awk '{ SUM+=$5} END {print SUM}'
8040
```

```
# Pipe multiple outputs using Pipe operators
ls -l | grep .py | awk '{ SUM+=$5} END {print SUM}'
3776
```

Capturing shell output with bash magic function

- Magic function `%%bash --output`

```
%%bash --out output
ls -l | awk '{ SUM+=$5} END {print SUM}
```

  - The type of this command is a string

```
type(output)
```

```
`str`
```

```
output
```

Capturing shell output with ! Syntax

- Alternate method of invoking shell commands
- The `!` operator invokes shell commands in IPython

```
ls_count = !ls -l | awk '{ SUM+=$5} END {print SUM}'
```

  - The type of this command is an SList

```
type(ls_count)
IPython.utils.text.SList
ls_count
['8070']
```

Bash and STDERR

- This is a command that will create output to STDERR

```
%%bash --out output
ls --turbo
```

    - STDERR isn't captured

Capture both STDOUT and STDERR

- `%%magic` allows STDOUT and STDERR capture

```
%%bash --out output --err error
ls -l | awk '{ SUM+=$5} END {print SUM}'
echo "no error so far" >&2
```

- The output of error

```
error
'no error so far\n'
```

Practicing with the Captured Output

COMMAND LINE AUTOMATION IN PYTHON

Automate with SList

COMMAND LINE AUTOMATION IN PYTHON

Noah Gift

Lecturer, Northwestern & UC Davis & UC Berkeley | Founder, Pragmatic AI Labs

SList methods

- Three main methods
- `fields`
- `grep`
- `sort`

Using SList fields

- List the items in a directory and save the variable

```
ls = !ls -l /usr/bin
```

  - Collect whitespace-separated fields

```
ls.fields(1,5)[1:4]
['1 Jan', '1 Jul', '1 Sep']
```

Using SList grep

- Assign `ls` output to an `SList`

```
ls = !ls -l /usr/bin
```

  - Grep a pattern

```
ls.grep("kill")
```

  - Only results matching pattern are displayed

```
['lrwxrwxrwx 1 root root          5 May 14  2018 pkill -> pgrep',
 '-rwxr-xr-x 1 root root      26704 May 14  2018 skill']
```

Using SList sort

- Capture `df` unix command

```
disk_usage = !df -h
```

  - Sort by usage

```
disk_usage.sort(5, nums = True)
```

```
['/dev/nvme0n1p1  335G  177G  158G  53% /etc/hosts',
 'Filesystem      Size  Used Avail Use% Mounted on',
 'overlay         335G  177G  158G  53% /',
 'shm              64M   24K   64M   1% /dev/shm']
```

SList and regular Python lists

- An SList can be popped using `.pop()`

```
var = ls.pop()
print(var)
'pear84.txt'
```

  - slicing operations work on SLists

```
ls[-4:]
['pear5.txt', 'pear52.txt', 'pear56.txt', 'pear6.txt']
```

Wrapping up SList

- SList to list workflow

```
type(ls)
'IPython.utils.text.SList'
newls = list(ls)
'list'
```

- SList to set

```
sls = set(ls)
```

- SList to dictionary

```
dls = dict(vals=ls)
```

Practicing with SList

COMMAND LINE AUTOMATION IN PYTHON