Practical Python for Sysadmins

Hello, PSU MacAdmins Conference 2013!

Practical Python for Sysadmins

PSU MacAdmins Conference 2013

Nate Walck
@natewalck

Jeremy Reichman
@jaharmi

Tamman Technologies, Inc.
Philadelphia, Pennsylvania

Assumptions

- You will be using OS X
- You will not install Python yourself
- Feel free to follow along in Terminal

Examples

- Text file with commands, try them yourself
- goo.gl/dgMjW

Quick demo

Practical Python for Mac Admins

No need to be afraid

Why Python?

Pros

- Escapes symbols/spaces for you (mostly)
- Easy to reuse code
- No need to reinvent the wheel

Cons

- Cannot bash things (you must shell-out)
- Different from command prompt
- Can be hard to switch mindset from bash

The Python Interpreter

Python Interpreter

- Interactive (Like bash, except not)
- Starts in current working directory
- Inherits your shell environment
- ‘>>>’ is your command prompt

Python Interpreter

Get Current Working Directory

```
>>> import os
>>> os.getcwd()
```

Python Interpreter

Show Environment Variables

```
>>> import os
>>> print(os.environ)
>>> for k, v in os.environ.items():
>>>     print(k, v)
>>>
```

PEP 8 and style

- 4 spaces per indentation level
- Wrap long lines within parentheses
- \ to break a line
- Words with underscores
  - this_is_the_style
  - ThisIsNotTheStyle

Indenting

- Applies to code blocks
- Starts and ends a block
- Whitespace must be consistent

Variables

- Names of arbitrary length
  - Letters, numbers, symbols
  - Must begin with a letter
  - Some names reserved
- Values
  - Any object

Data Types

Data Types

- Dynamically typed
- Strongly typed
- Almost everything is an object…But you don’t need to care
- Many helpful types of objects are built-in

Data Types

- type() tells you what an object's type is
  - type(*some_object*)
- help() gives docs on objects
  - help(*some_object*)

Numbers

- Integers
- Floats
  - Floating point
  - Decimal numbers
- Not quoted
- help(int), help(float)

Numbers

```
number1 = 7
number2 = .256
number3 = -1024

number1 + number2 + number3
number1 < number2
number2 == number3
```

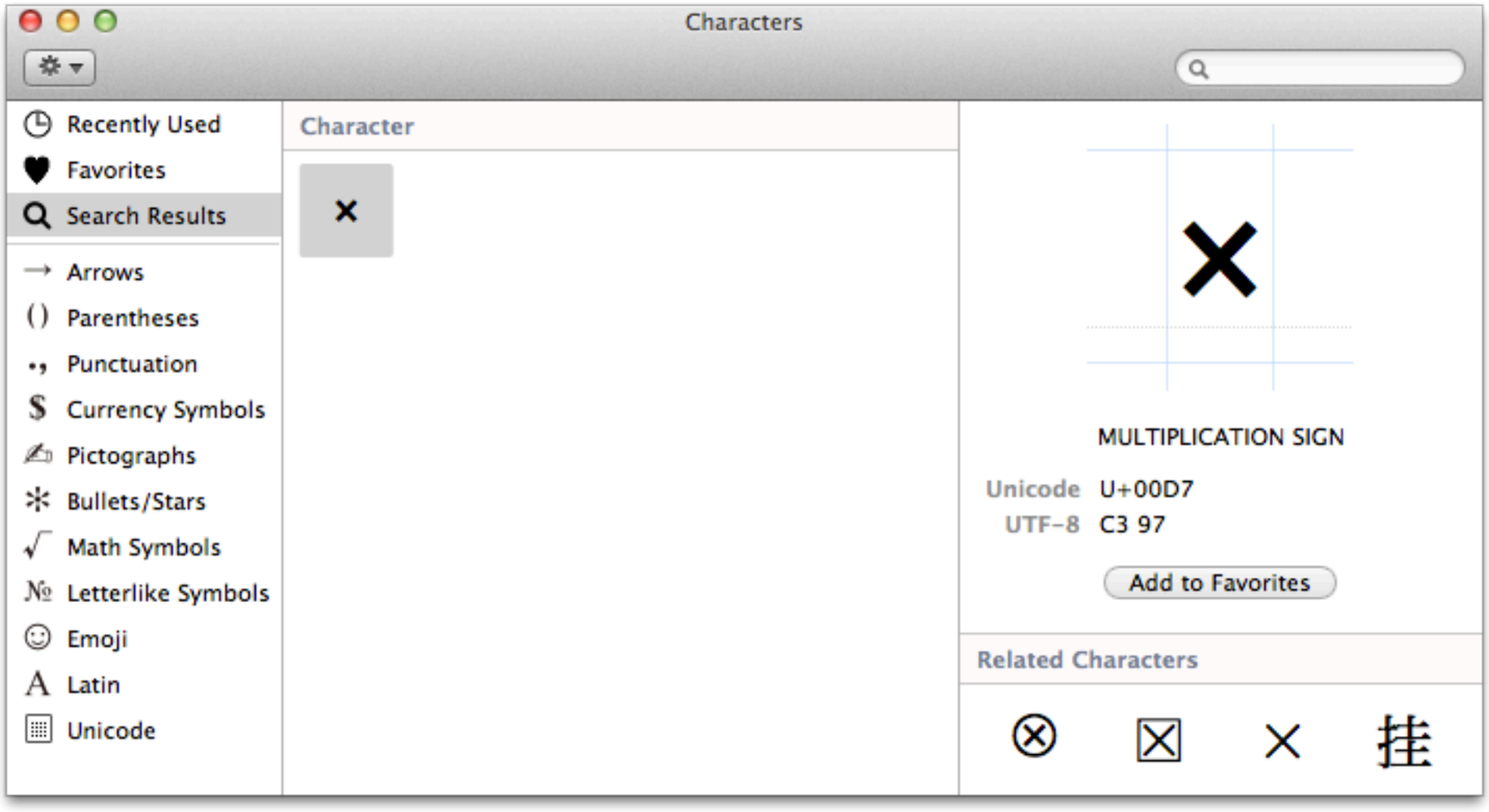
Strings

- Alphanumeric, punctuation, whitespace, character codes
- Quoted
  - Single or double
  - Triple
- help(str)

Strings

```
string1 = "Hello world"
string2 = '''Lorem ipsum dolor sit
  amet, consectetur adipisicing
  elit, sed do eiusmod tempor
  incididunt ut labore et
  dolore magna aliqua.'''
```

```
string3 = u'1024 \u00D7 768'
```



Edit > Special Characters …

```
print(string3)
1024 × 768
```

String format

Substitute Values (or Objects) into a String

```
print "Some text %s" % substitute_this

print “A number %d”.format(substitute_this)
```

Lists

- Arrays or sequences
  - Order is important
  - Contains any data type
  - Square brackets, []
  - help(list) or help([])

Booleans

	True	False
Bool	True	False, None
String	'a', 'Any text'	''
Int, Float	1, 7, 256, -1024	0
List	[1, 2, 3]	[]

Dictionaries

- Key-value pairs
- Keys
  - Case-sensitive
  - No duplicates
  - New value overwrites
- Curly braces, {}

Dictionaries

- Values
  - Any data type
    - Mix and match types
- Unordered
- help(dict) or help({})

Control and Logic

- Some are very similar to bash
- Others are more powerful than bash logic

for loops

- Use “for” reserved keyword
- If output is an array, consider using a list comprehension

for loop comparison

Bash

```
for model in ${model_list[@]}
do
    echo $model
done
```

for loop comparison

Bash

```
for model in ${model_list[@]}
do
    echo $model
done
```

Python

```
for model in model_list:
    print(model)
```

if...then

- Use “if,” “elif,” and “else” reserved keywords
- First true statement wins

if...then

Bash

```
if [ "$motivation" -gt "0" ]; then
    echo "Script all the things"
fi
```

if...then

Bash

```
if [ "$motivation" -gt "0" ]; then
    echo "Script all the things"
fi
```

Python

```
if motivation > 0:
    print("Script all the things")
```

case

- No such thing in Python
- Use the if...elif...else structure to accomplish the same thing

case statement

Bash

```
case $model in
    "MacBookAir")
        echo "Thin and Light"
        ;;
    "MacBookPro")
        echo "Thick and Heavy"
        ;;
    "MacPro")
        echo "Forever alone?"
        ;;
    *)
        echo "Everything else"
esac
```

case statement

Bash

```
case $model in
    "MacBookAir")
        echo "Thin and Light"
        ;;
    "MacBookPro")
        echo "Thick and Heavy"
        ;;
    "MacPro")
        echo "Forever alone?"
        ;;
    *)
        echo "Everything else"
esac
```

case statement

Bash

```
case $model in
    "MacBookAir")
        echo "Thin and Light"
        ;;
    "MacBookPro")
        echo "Thick and Heavy"
        ;;
    "MacPro")
        echo "Forever alone?"
        ;;
    *)
        echo "Everything else"
esac
```

Python

```
if model == "MacBookAir":
    print("Thin and Light")
elif model == "MacBookPro":
    print("Thick and Heavy")
elif model == "MacPro":
    print("Forever alone?")
else:
    print("Everything else")
```

Functions

- Use “def” reserved keyword
- May use “return” keyword

Functions

Bash

```
function hello_world (){
    echo "Hello World!"
}

hello_world
```

Functions

Bash

```
function hello_world (){
    echo "Hello World!"
}

hello_world
```

Python

```
def hello_world():
    print("Hello World!")

hello_world()
```

Working with Files

- Building paths
- Pathname components

Working with Files

Joining Paths

```
import os
silverlight_plugin_path = os.path.join("/", \
    "Library", \
    "Internet Plug-Ins", \
    "Silverlight.plugin")

print(silverlight_plugin_path)
/Library/Internet Plug-Ins/Silverlight.plugin
```

Working with Files

Manipulating Paths

```
os.path.basename(silverlight_plugin_path)
'Silverlight.plugin'

os.path.dirname(silverlight_plugin_path)
'/Library/Internet Plug-Ins'

os.path.splitext("com.apple.Safari.plist")
('com.apple.Safari', '.plist')
```

Tests on Files

- Does the path exist?
- What kind of object is at that path?

Tests on Files

```
os.path.exists(silverlight_plugin_path)
True

os.path.isdir(silverlight_plugin_path)
True

os.path.islink("/etc")
True
```

glob

- Equivalent to shell globbing
- Returns matching path(s)

glob uses the fnmatch module

glob

```
import glob
osx_install = glob.glob("/Applications/" \
    "Install*OS X*.app")

print(osx_install)
['/Applications/Install Mac OS X Lion.app',
 '/Applications/Install OS X Mountain
Lion.app']
```

Version numbers

- distutils.version supports version objects
  - StrictVersion
  - LooseVersion
- setuptools.pkg_resources
  - parse_version

Version numbers

```
from distutils import version
version.LooseVersion('10.4.1') > \
version.StrictVersion('10.4.11a4')

False
```

Version numbers

```
from pkg_resources import parse_version as V

V('10.4.1') > V('10.4.11a4')

False
```

PyObjC

- You can run Objective-C code via a Python bridge (Stop the madness!)
- Allows access to native methods for manipulating the OS.

PyObjC

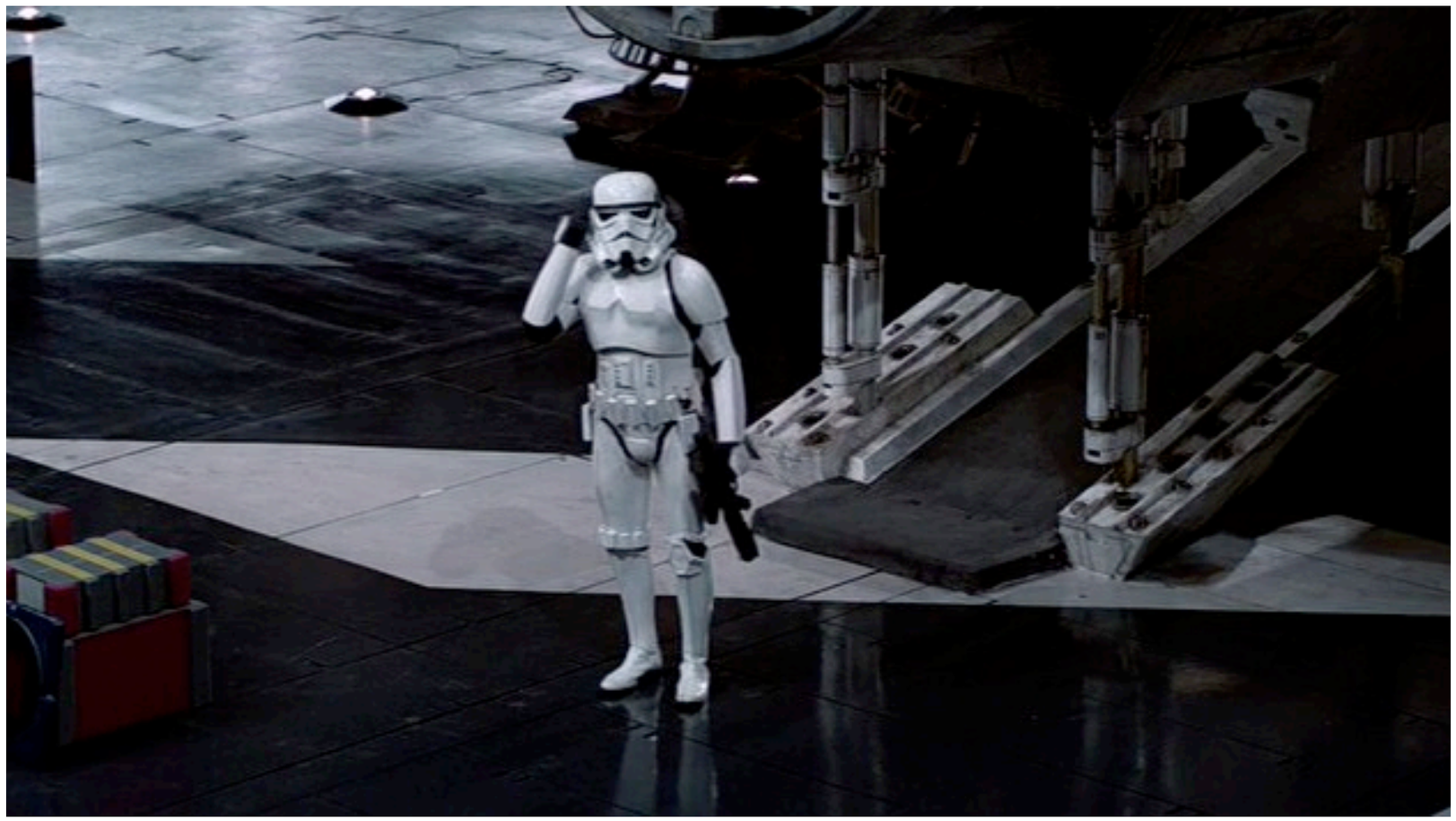
Read plist via CFPreferences

```
from Foundation import CFPreferencesCopyAppValue

preference_value = CFPreferencesCopyAppValue( \
    'AutoBackup', 'com.apple.TimeMachine')

print(preference_value)
True
```

PyObjC

Get Screen Resolution

```
from AppKit import NSScreen
width = NSScreen.mainScreen().frame().size.width
height = NSScreen.mainScreen().frame().size.height
print(width, height)
(1440.0, 900.0)
```



Is this thing working?

syslog

- Send messages to system log
  - Facility (sender)
  - Priority (level)

syslog

```
import syslog
syslog.openlog("TEST")
syslog.syslog(syslog.LOG_NOTICE,
              "No nonsense found.")
syslog.closelog()

5/12/13 4:54:12.689 PM TEST[47885]: No
nonsense found.
```

Running shell commands in Python?

Sure, why not

Running Commands

- Subprocess module
- Call external shell commands

	Returns
subprocess.call	Return code
subprocess.check_call	Return code or exception
subprocess.check_output	Output string or exception

Running Commands

- Arguments
  - Command as a list of strings
  - Example: [“echo”, “Hello”]
- Using shell=True
  - Command as a single string
  - Executed directly through shell
  - **Strongly** Discouraged

Running Commands

```
import subprocess
cmd_str = '/usr/bin/dsmemberutil checkmembership \
-U jeremy -G admin'

cmd = cmd_str.split()

retcode = subprocess.call(cmd)
retcode = subprocess.check_call(cmd)
output = subprocess.check_output(cmd)
```

Helpful Resources

Because Learning is fun

Learning More

- Think Python - goo.gl/G53Pa
- Code Academy - goo.gl/exxcw
- Dive Into Python - goo.gl/h9QHZ
- Python.org - goo.gl/Zpr3t
- Learn Python the Hard Way - goo.gl/6Mw6u

GitHub

- Search Github to see if it already exists
- Clone a project and use it
- Tweak if necessary
- Read their code/documentation to make sure you understand how it works

irc.freenode.net

- Official Python Channel - #python
- Several helpful Python + Mac users on ##osx-server