


Mak.py

By James Reynolds

Been doing this since 2000...



- Assimilator now free! 
- Requires Mac OS 7.x, 8.x or 9.x
- Will not be updated to 10.x
- <http://www.stairways.com/main/assimilator>

My head



Photo courtesy of puukibeach

I switched to Jamf!



So much free time!

- Now I can do other things like watch the Jordan Peterson interview with Cathy Newman again!



Right?

- Right?...

Jamf is missing so much

- I'm still writing tons of scripts
- "Uploaded to Jamf Nation"
 - 131 package manifests
 - 81 licensed software templates
 - 56 managed preference manifests
 - 113 scripts (majority are SH scripts)
 - 315 extension attributes (majority are SH scripts)

<rant>

- A little detour

Shell scripting? Really?

- Variables?
 - Only data type is string and it is essentially a copy-paste
 - Non-quoted empty strings wreck anything that expects arguments
 - "Internal field separator"
 - No built-in string manipulation tools (required sed and awk)
 - Only has 1D arrays and no hashes/dictionaries
- Documentation
 - Has anyone here read the man page for sh?

Shell scripting? Really?

- Must learn subcommands (sed, awk, /bin/*, /sbin/*, etc)
 - Platform compatibility is mostly ok except when it isn't
 - (~~Linux `killall` vs macOS `killall`...~~) [edit, I was wrong]
- Some syntax is impossible to understand
 - Is Perl really worse than the shell?
 - How do you do a web search for strange symbols?

Shell scripting? Really?

- So many shells and all are slightly different
 - What's the differences between sh and bash?
- ALL errors print to STDERR and then are stepped over
 - This is the most insane thing ever
- Spaces must be escaped or quoted or else: destruction...
 - iTunes 2.0 installer erased hard drives because of this!

Don't get me wrong

- SH (1977) is so much easier than...
 - Assembly language (1949)
 - FORTRAN (1957)
 - COBOL (1959)
 - BASIC (1964)
 - Pascal (1970)
 - Forth (1970)
 - C (1972)




But...

- Perl (1987) saved us from SH (1977)
 - Included functionality of sed (1974) and awk (1977)
 - Included real data types (with caveats)
 - Influenced Python (1991), PHP (1995), Ruby (1995), JavaScript (1995), and Windows PowerShell (2006)
- Why did we go back to SH (1977)?????

Why back to SH?

- Perl (1987) has been criticized since 1993
- A lot of newbies wrote bad Perl (1987) in the 1990's
 - (Similar thing happened to PHP (1995))
- Perl includes a lot of C and SH 'isms
- Linux (1991)
- Mac OS X (2001)
- And this...



*nihil sub sole novum nec valet quisquam
dicere ecce hoc recens est iam enim
praecessit in saeculis quae fuerunt ante nos*

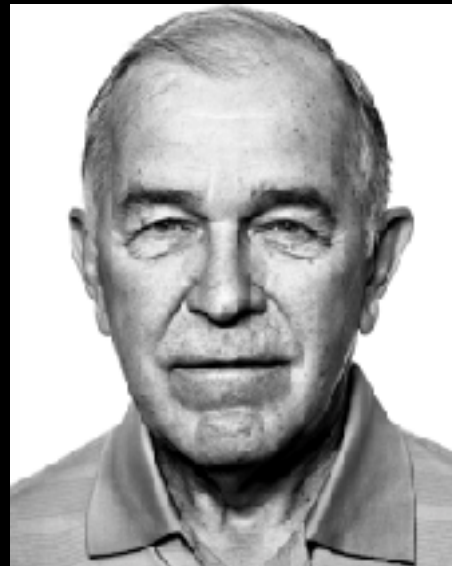
Specifically

- Ken Thompson
- Dennis Ritchie (1941-2011)
- Joe Ossanna
- Douglas McIlroy
- Peter Neumann
- Brian Kernighan
- Rudd Canaday

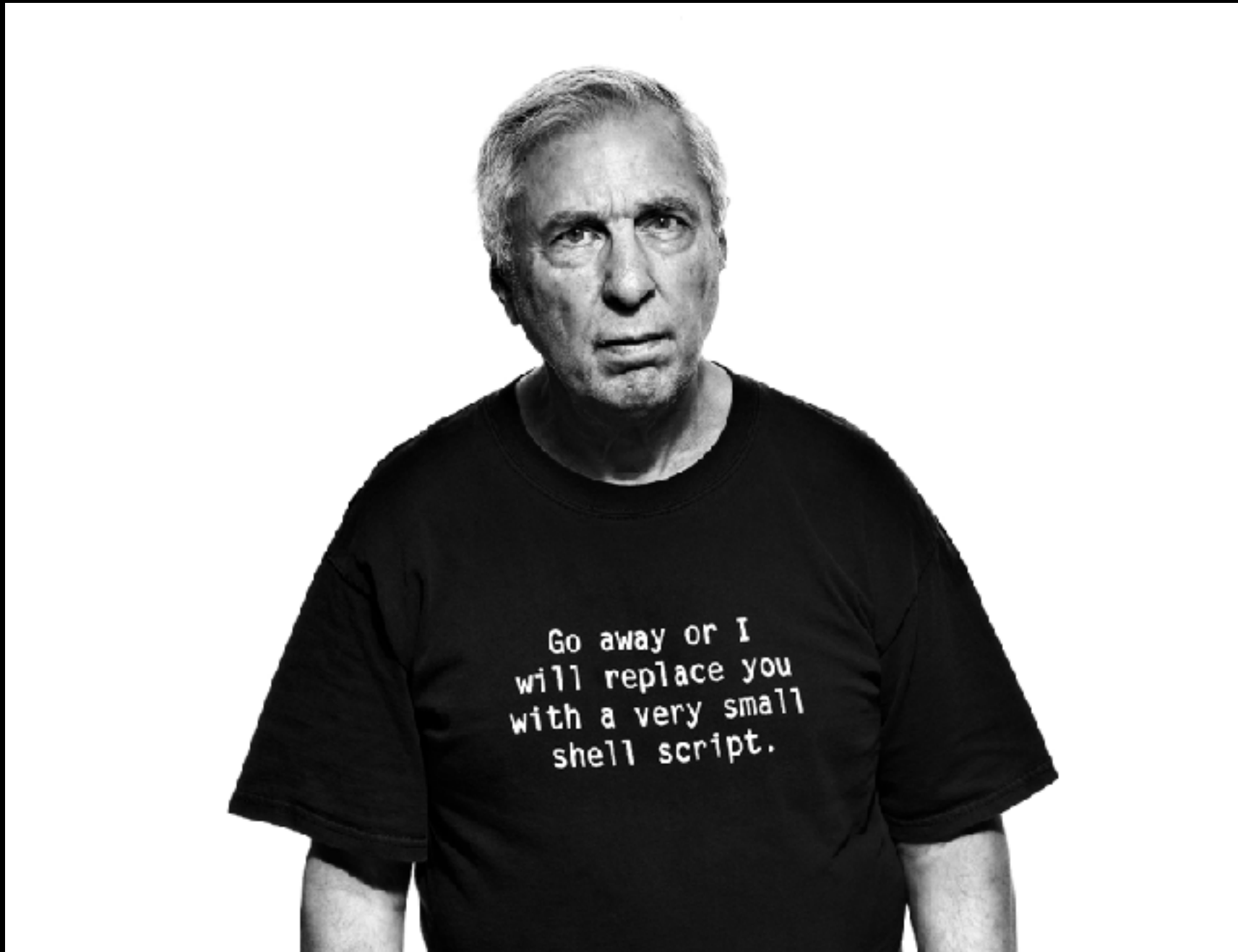


Specifically

- Michael Lesk
- Bill Joy
- *Stephen Bourne*
- *David Korn*
- Andrew Tanenbaum
- Gordon Bell
- And so many more...



Be afraid, very afraid



The Church of Bell Labs (aka Unix)



- With an the ancient orthodoxy written by geniuses
 - Surely they knew better than us
- I would flee right now if any of them were in the room...
- But still...

There is a better way

- Learn to read SH (1977), like C (1972), it's here for good
- But quit writing new scripts in shell!
 - Unless it is a throw away script with a few lines
 - That means don't publish it!
- Someone tell the Jamf community!!!!

Better ways

(10.12 sizes)

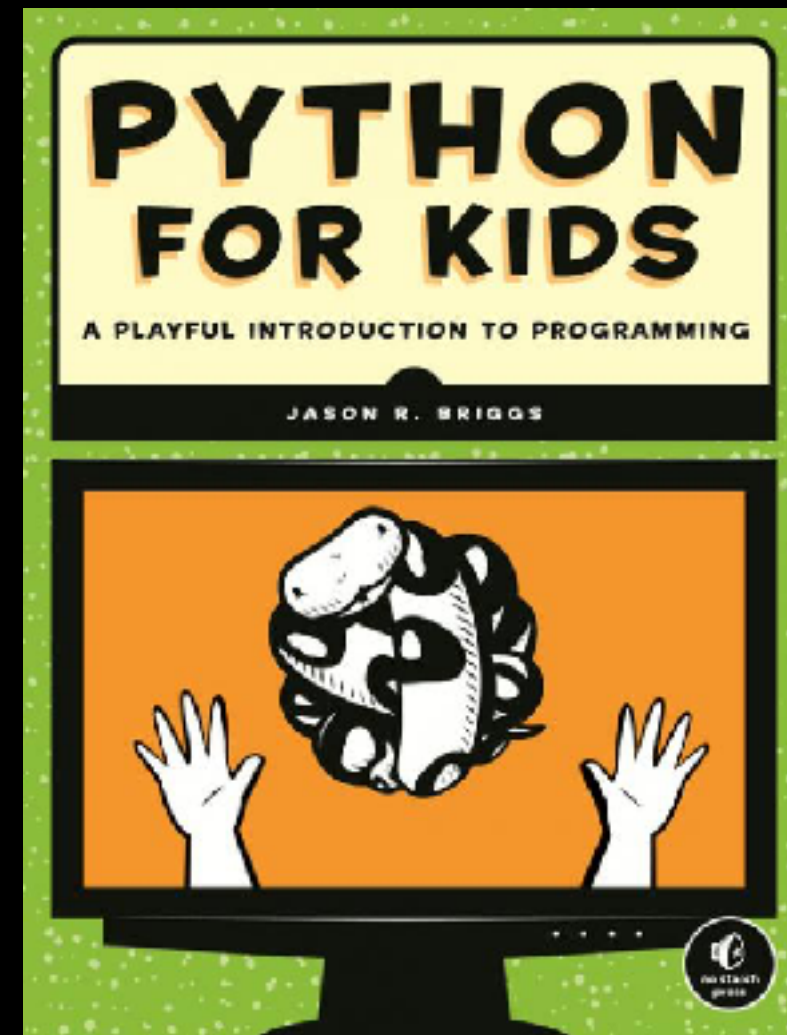
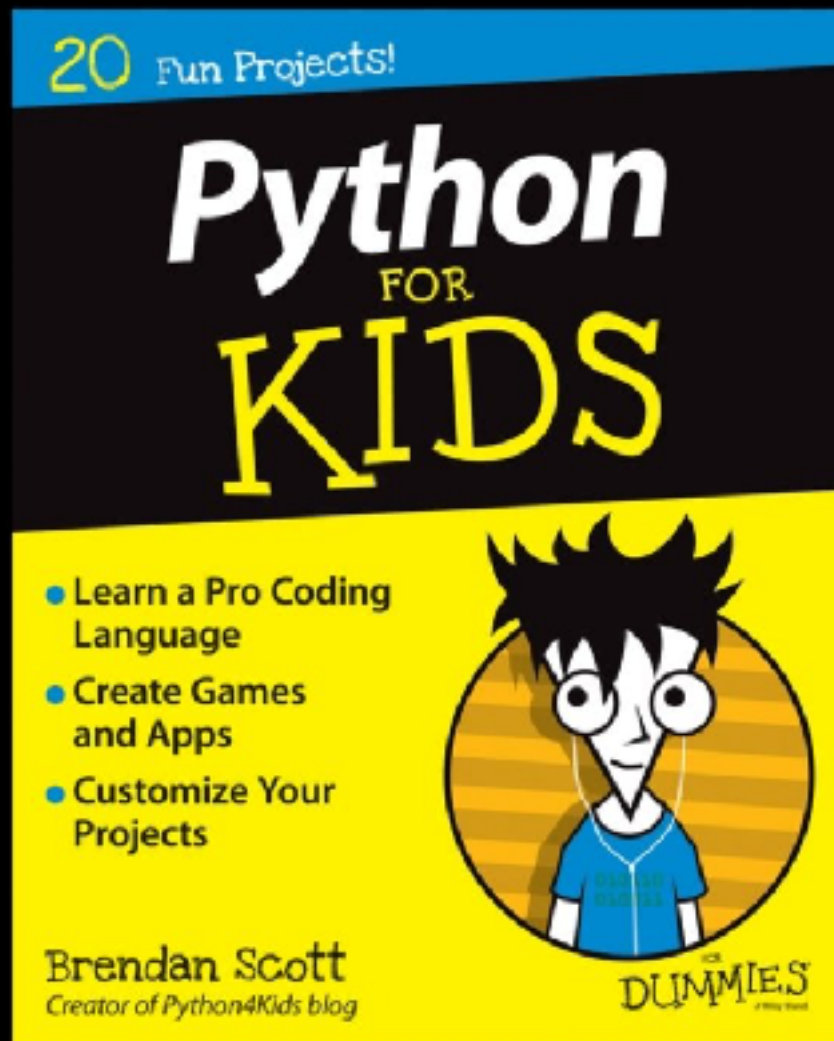
- In the future: Swift (2014) - 7 MB but ties into the system
- Ruby (1995) - 46 MB of Ruby stuff (Ruby was almost cool)
- Perl (1987) - 187 MB of Perl stuff (Perl is not cool)
- Python (1991) - 384 MB of Python stuff (Python is cool)
 - The biggest single thing in macOS (except speech)

When to use SH vs Python

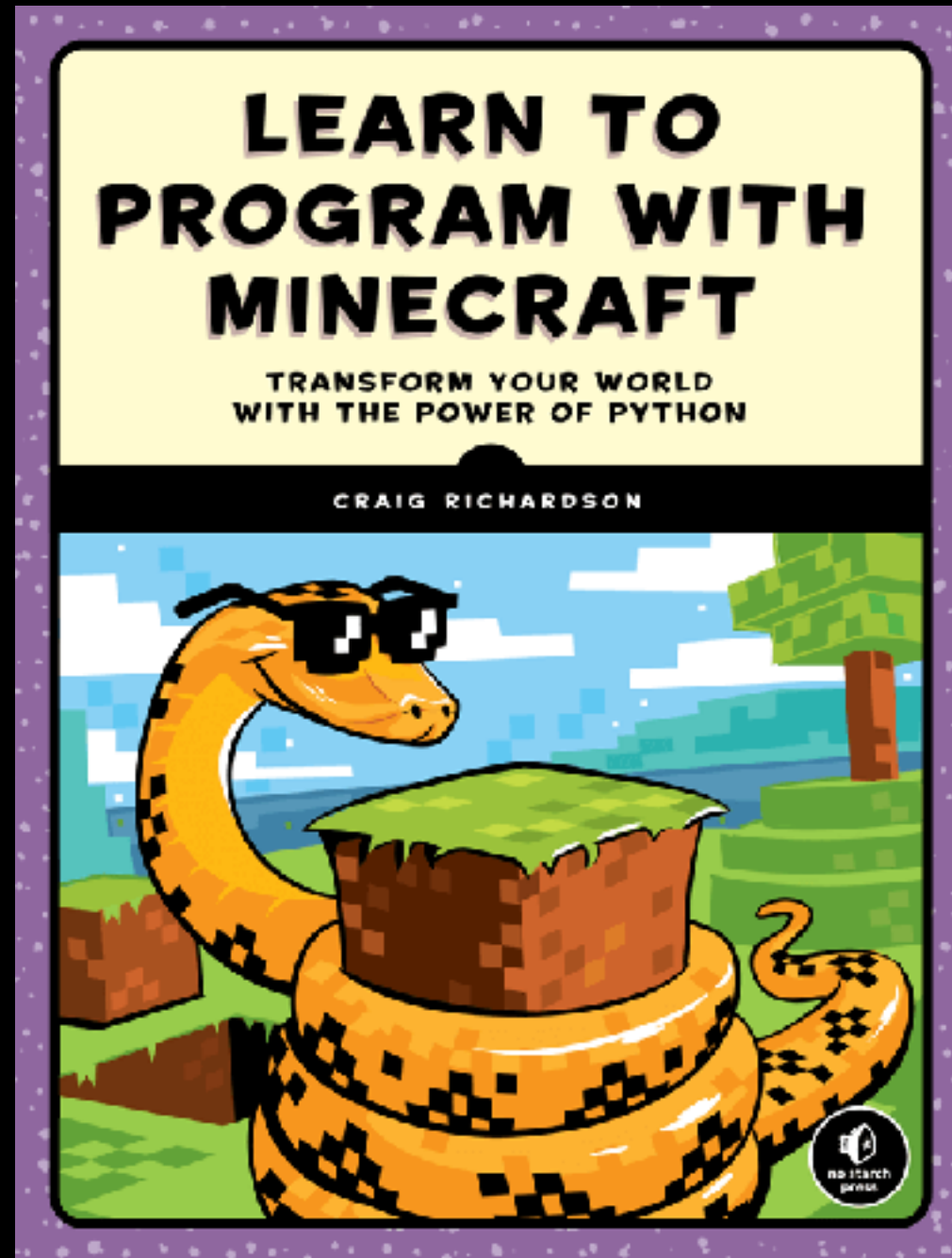
- If it requires more than
 - 1 function or
 - any type of control structure
 - (like a loop, or
 - if/then/else statement),
 - you should probably use Python instead of shell

-http://databio.org/posts/shell_scripts.html

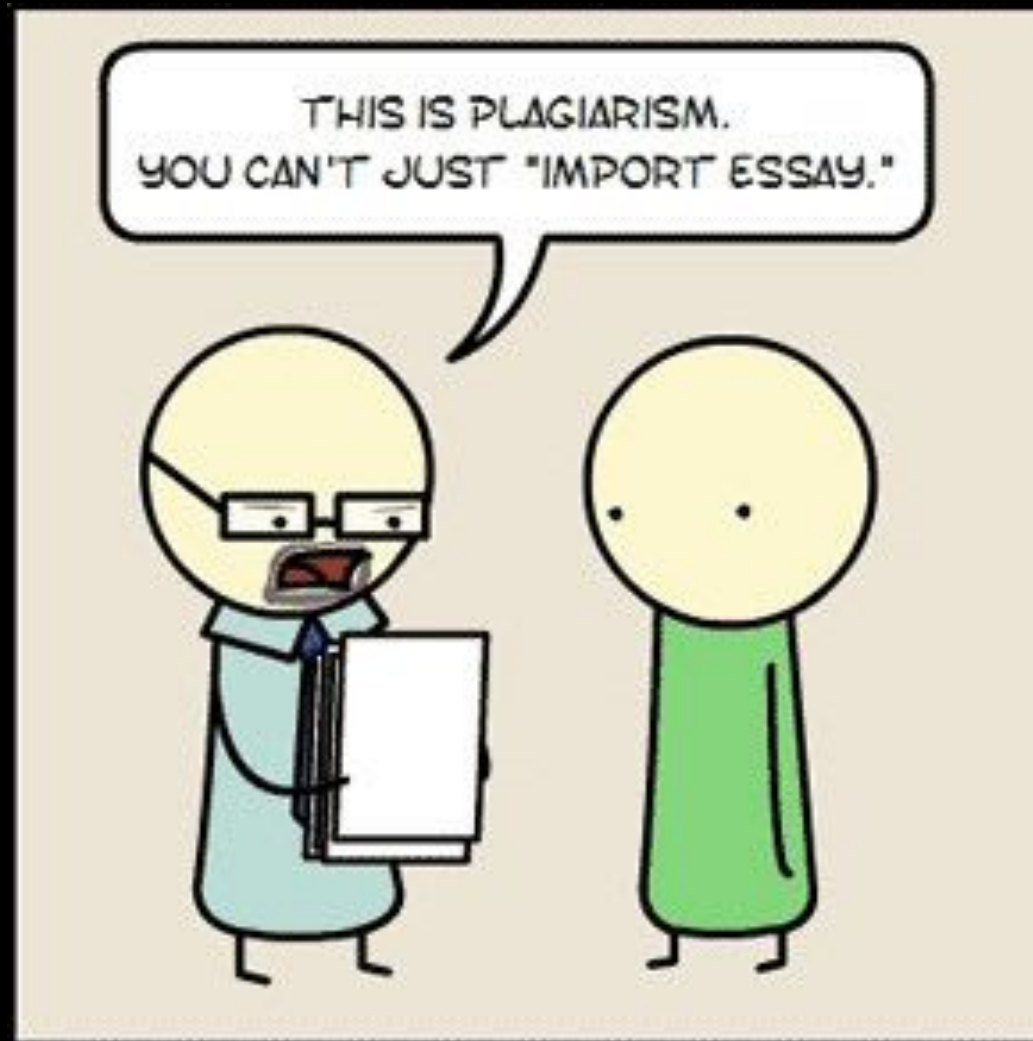
Little kids learn Python



You can script Minecraft with Python



Python is cool



So learn it and use it!

Even I learned Python

- Not my favorite
- I think Lua is amazing
- I'm looking forward to Swift
 - (not ready yet because of versioning and distribution)

</end rant>

- Ok, back to my presentation
- So I switched to Jamf
- I am still writing a lot of scripts
- They did something I liked
 - `/usr/local/bin/jamf`

/usr/local/bin/jamf

Usage: **jamf** verb [options]

verb is one of the following:

about	Displays information about the jamf binary
bind	Binds this computer to a directory service
bless	Blesses a System or a NetBoot Server
changePassword	Changes a local user's password
checkJSSConnection	Checks the availability of the JSS
createAccount	Creates a new local account on the system

jamf

createConf	Creates a configuration file that the jamf binary uses
createHooks	Creates and configures login/logout hooks
createSetupDone	Ensures the Setup Assistant does not launch next boot
createStartupItem	Creates a startup script to contact the JSS
deleteAccount	Deletes a local account from the local dscl database
deletePrinter	Deletes a printer from the system
deleteSetupDone	Causes the Setup Assistant to launch on the next boot
displayMessage	Displays a message to the current user

jamf

<code>enablePermissions</code>	Enables permissions on a volume
<code>enroll</code>	Enrolls this computer with the JSS
<code>fixByHostFiles</code>	Fixes the ByHost files
<code>fixDocks</code>	Repairs docks that have ? after certain OS Updates
<code>fixPermissions</code>	Fixes the permissions on a given target
<code>flushCaches</code>	Flush cache files for the system and/or users
<code>flushPolicyHistory</code>	Flush the policy history on the JSS
<code>getARDFields</code>	Displays the ARD Fields on a volume

jamf

getComputerName	Displays the computer name on a volume
heal	Run self healing for all packages on this computer
help	Displays this message or details on a specific verb
install	Installs a package
installAllCached	Installs all packages that are cached
listUsers	Lists all the users on the computer
log	Log the IP address, action, and username to the JSS
manage	Enforces the entire management framework

jamf

mapPrinter	Maps a printer
mcx	Apply Managed Preferences
modifyDock	Installs or removes items in all users docks
mount	Mounts a file share
notify	Checks the JSS for new notification messages
policy	Checks for policies on the JSS
reboot	Reboots the computer
recon	Runs Recon to update the inventory in the JSS

jamf

removeFramework	Removes the JAMF Binary and associated files
removeSWUSettings	Remove settings that point SWU at internal servers
resetPassword	Resets a local user account password
runScript	Runs a script
runSoftwareUpdate	Run Software Update
setARDFields	Sets the ARD Fields
setComputerName	Sets the computer name
setHomePage	Sets the default home page for users

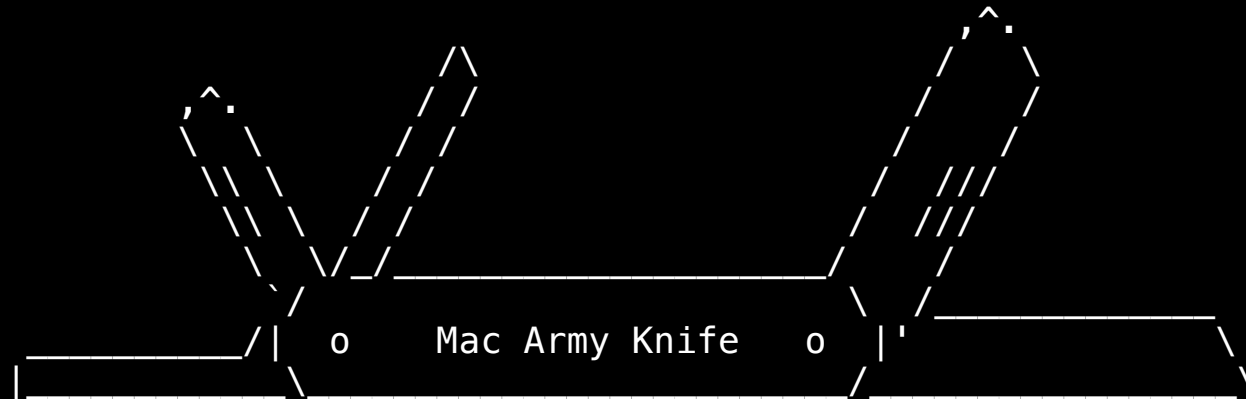
jamf

setOFP	Sets the Open Firmware mode and password
startSSH	Starts the ssh server
uninstall	Uninstalls a package
unmountServer	Unmounts a file server
updatePrebindings	Updates the prebindings on a volume
version	Prints the version of this application

- So I thought, why don't I do the same with my scripts?
- Thus, mak.py is born
 - Stands for Mac Army Knife
 - Sounds like "magpie"



mak.py features



- Replicates some System Preferences panes
- Replicates some application Preferences (Finder, Safari)
- Creates launch daemons
- More..

mak.py goals

- Do what `/usr/local/bin/jamf` doesn't do
- Include as much as possible as one-off commands
 - Free my brain so I can forget it all
 - Quit having to look it up on the web
- Easily support different versions of OS X/macOS
 - Make debugging new OS'es easier
- Community project

Commands

- `ard` - enables users to use ARD
- `disable_touristd`
- `hack_jamf_hooks` - delays login until network shows up
- `help`
- `launchdaemon` - creates LaunchDaemons
- `locatedb` - activates

Commands

- **networksetup** - Just a shortcut to /usr/sbin/networksetup
- **pref** - The bulk of the script (more slides on this)
- **set_volume**
- **set_zone_ntp** - sets timezone and ntp server
- **shell_paths** - adds shell paths (to /etc/paths.d/)
- **systemsetup** - Just a shortcut to /usr/sbin/system setup

The Prefs

`Generic.Computer=<domain>=<key>=<format>=<value>`

- Generic computer preference; 4 args; user domain

`Generic.User=<domain>=<key>=<format>=<value>`

- Generic user preference; 4 args; user domain

`Generic.User.ByHost=<domain>=<key>=<format>=<value>`

- Generic user byhost preference; 4 args; user domain

The Prefs

- Clock.User
- CrashReporter.User
- Dock.User
- Finder.User
- Gateway.Computer
- KeyAccess.Computer
- Mouse.User.Click
- Quicktime7

The Prefs

- Safari.User
- ScreenSaver.Computer
- ScreenSaver.User
- Screenshot.User
- SoftwareUpdate.Computer
- SystemUIServer.User
- Tourist.User

Constructing Prefs

```
'Finder.User.AppleShowAllExtensions':{  
    'help':'bla bla bla',  
    'versions':{  
        '10.12':{  
            'defaults':[  
                { 'domain':'com.apple.finder', 'args':['AppleShowAllExtensions',  
                '-bool', '%ARG0%'], 'user':True, 'byhost':True, 'arg_count':1, },  
            ],  
        },  
    },  
},
```

mak.py plans

- Add more functionality to each command
 - ard restart
 - ntp re-sync
 - networksetup and systemsetup shortcuts
- Add logout and restart functions

mak.py possibilities

- Support displays
- Support profiles (bleck 🙄)
- Support launchservices (duti?)
- Switch defaults to Objective-C bridge (CFPreferences)
- Duplicate functionality of /usr/local/bin/jamf for people not using Jamf?