Swiftly? Moving on from Python

Greg Neagle, October 2024


#!/bin/sh

```
#!/bin/sh

# configure Screen Sharing
# October 2009, Greg Neagle, Disney Animation

# make sure ARD is turned off
/bin/rm -f "/Library/Application Support/Apple/Remote Desktop/RemoteMa

# set up com.apple.RemoteManagement preferences:
/usr/bin/defaults delete /Library/Preferences/com.apple.RemoteManageme
/usr/bin/defaults write /Library/Preferences/com.apple.RemoteManagemer
/usr/bin/defaults write /Library/Preferences/com.apple.RemoteManagemer
/usr/bin/defaults write /Library/Preferences/com.apple.RemoteManagemer
/usr/bin/defaults write /Library/Preferences/com.apple.RemoteManagemer
/usr/bin/defaults write /Library/Preferences/com.apple.RemoteManagemer
/usr/bin/defaults write /Library/Preferences/com.apple.RemoteManagemer
```

```
tell application "System Events"
  display alert "Please restart immediately!" as critical
end tell
```

DevOps

DevOps

32 languages

Article Talk Read Edit View history Tools

From Wikipedia, the free encyclopedia

Checked

DevOps is a methodology in the software development and IT industry. Used as a set of practices and tools, DevOps integrates and automates the work of software development (*Dev*) and IT operations (*Ops*) as a means for improving and shortening the systems development life cycle.[1] DevOps is complementary to agile software development; several DevOps aspects came from the *agile* way of working.

Automation is an important part of DevOps. Software programmers and architects should use "fitness functions" to keep their software in check.[2]

- Version control (Subversion, Mercurial, Git)
- Code linters and formatters
- Continuous integration (Hudson, Jenkins, etc)
- Test-driven development
- Software-style project management (Agile/Scrum/etc)

Puppet

CHEF

Ansible

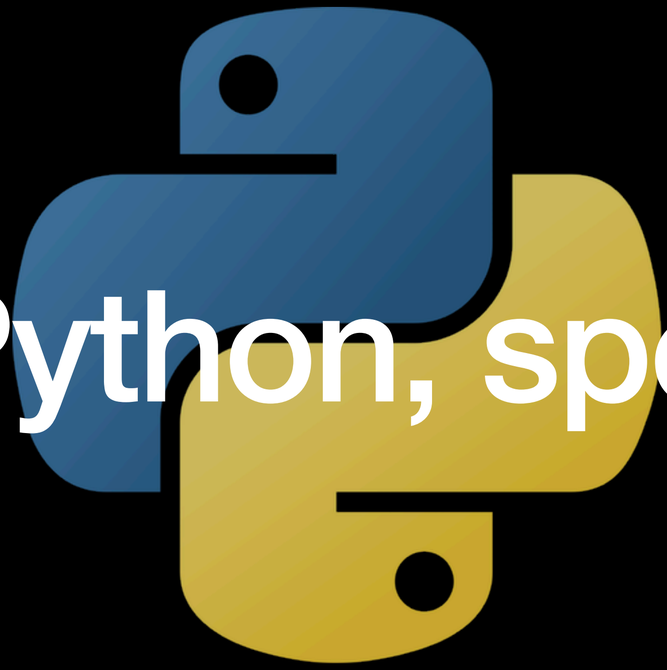
Perl

Python

Ruby

- Complex data structures (dictionaries, classes/objects)
- Libraries (add-on modules)
- Exceptions (robust error handling)
- Debuggers/formatters/linters

So why Python, specifically?



pymacadmin

A collection of Python utilities for Mac OS X system administration

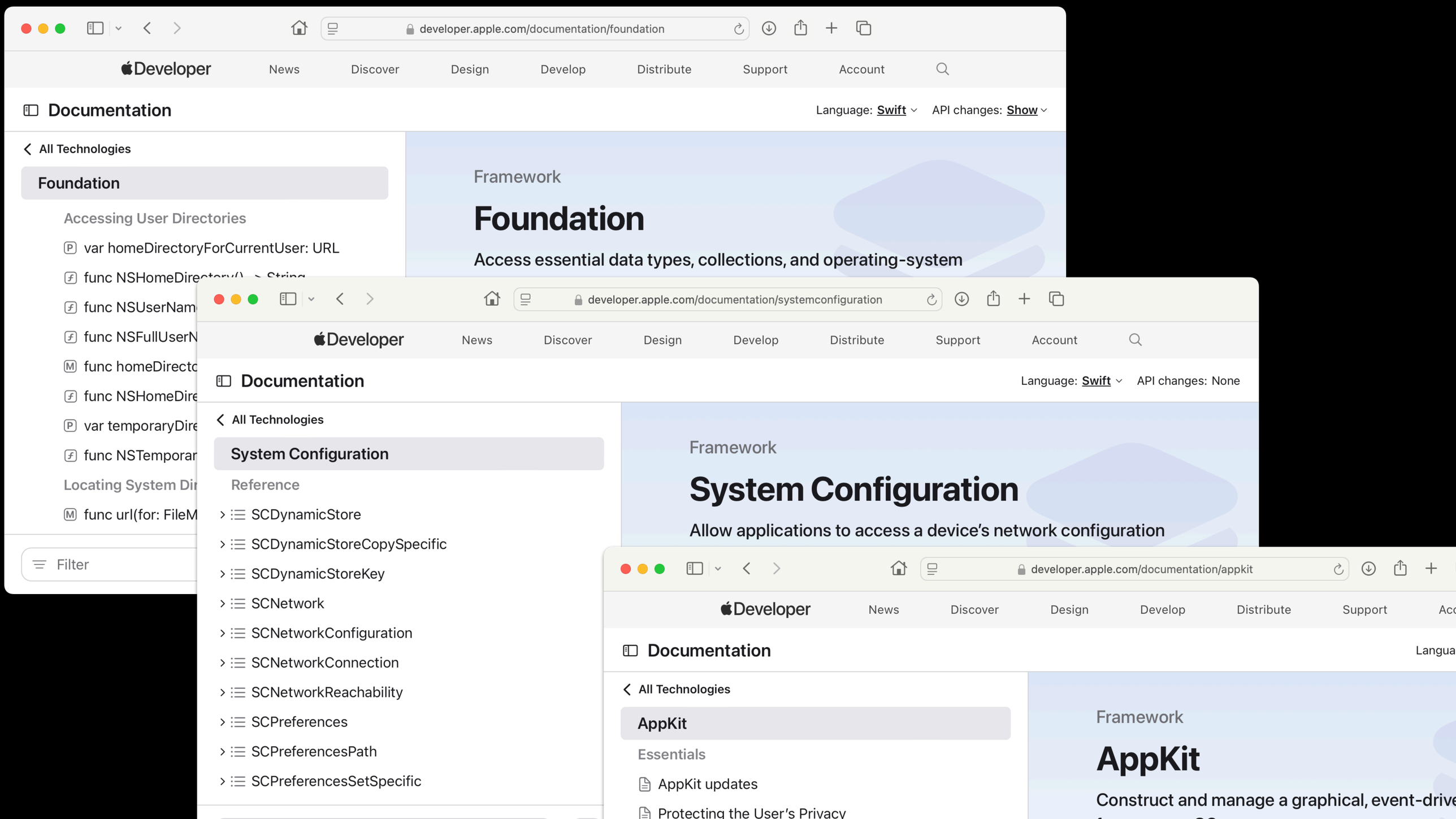
Source path: hg/ **Branch:** default

Directories
hg
bin
examples
lib
pymacds-dist
utilities

Filename	Size	Rev	Date
airport-update.py	1.0 KB	r15f74f5a4495	Jan 8, 2009
crankd.py	21.6 KB	r8e1d8cc1ddbd	Oct 29, 2009
create-location.py	3.9 KB	rce07a075971d	Mar 10, 2009
delete-certificate.py	1.2 KB	rfba505276256	Mar 14, 2009
keychain-delete.py	1.4 KB	re007acca5619	Mar 20, 2009
proxy-setenv.py	615 bytes	rd656bed3b39e	Sep 15, 2008
set-proxy.py	2.2 KB	r7e67eee6c27d	Mar 10, 2009
usb-notifier.c	23.7 KB	rd656bed3b39e	Sep 15, 2008

```
#!/usr/bin/env python
# encoding: utf-8
"""
Usage: crankd

Monitor system event notifications

Configuration:

The configuration file is divided into sections for each class of
events. Each section is a dictionary using the event condition as the
key ("NSWorkspaceDidWakeNotification", "State:/Network/Global/IPv4",
etc). Each event must have one of the following properties:

command:      a shell command
function:     the name of a python function
class:        the name of a python class which will be instantiated on
              and have methods called as events occur.
```

PyObjC


developer.apple.com/documentation/foundation
Developer
News
Discover
Design
Develop
Distribute
Support
Account
Documentation
Language: Swift
API changes: Show
All Technologies
Foundation
Accessing User Directories
var homeDirectoryForCurrentUser: URL
Framework
Foundation
Access essential data types, collections, and operating-system
developer.apple.com/documentation/systemconfiguration
Developer
News
Discover
Design
Develop
Distribute
Support
Account
Documentation
Language: Swift
API changes: None
All Technologies
System Configuration
Reference
SCDynamicStore
SCDynamicStoreCopySpecific
SCDynamicStoreKey
SCNetwork
SCNetworkConfiguration
SCNetworkConnection
SCNetworkReachability
SCPreferences
SCPreferencesPath
SCPreferencesSetSpecific
Framework
System Configuration
Allow applications to access a device's network configuration
developer.apple.com/documentation/appkit
Developer
News
Discover
Design
Develop
Distribute
Support
Documentation
All Technologies
AppKit
Essentials
AppKit updates
Protecting the User's Privacy
Framework
AppKit
Construct and manage a graphical, event-drive

```
$ python

Python 2.7.16 (default, Mar 25 2021, 03:11:28)
[GCC 4.2.1 Compatible Apple LLVM 11.0.3 (clang-1103.0.29.20) (-macos10.
Type "help", "copyright", "credits" or "license" for more information.

>>> import Foundation
>>> print Foundation.NSUserName()
gneagle
```

Search

Follow The Mule On Twitter

My Tweets

Muleish Banter

- Josh on How To: Get the currently logged in user, in a more Apple approved way

The Mule's Musings Categorised

- Application (1)
- Errors (30)
- Extension Attribute (3)
- How to (47)
- Musing (6)
- OSX Server (2)

19/11/2014 · 22:19

↓ Jump to Comments

How To: Get the currently logged in user, in a more Apple approved way

When running scripts via the JSS, you sometimes have a challenge to get the username of the currently logged in user.

If you're running the script at login or via Self Service, then $3 can be used to grab the username. However, you'll

```
loggedInUser=$(/usr/bin/python -c 'from SystemConfiguration import SCDynamicStoreCopyConsoleUser; import sys; username = (SCDynamicStoreCopyConsoleUser(None, None, None) or [None])[0]; username = [username,""][username in [u"loginwindow", None, u""]]; sys.stdout.write(username + "\n");')
```

On seeing an example of the above, Frogor over at the ##OSX-Server IRC mentioned that this method was both a little "odd" & not Apple's recommended way. Then, being the guy he is.. He not only pointed us in IRC to a link about Apple's recommended method, but also gave an example of how to call it within bash. That example is below:

```
loggedInUser=`python -c 'from SystemConfiguration import SCDynamicStoreCopyConsoleUser; impor
```

contents.sh view raw

You'll start to see the above soon in my new posts, & if you do use Fast User Switching then this may work better for you where other methods have failed.

```
$ python
Python 2.7.16 (default, Mar 25 2021, 03:11:28)
[GCC 4.2.1 Compatible Apple LLVM 11.0.3 (clang-1103.0.29.20) (-
macos10.15-objc- on darwin
Type "help", "copyright", "credits" or "license" for more information
>>> from SystemConfiguration import SCDynamicStoreCopyConsoleUser
>>> cfuser = SCDynamicStoreCopyConsoleUser(None, None, None)
>>> print cfuser
(u'gneagle', 4389, 100)
>>> username = cfuser[0]
>>> print username
gneagle
```

All Technologies

System Configuration

Reference

DHCPClientPreferences
SCDynamicStore
SCDynamicStoreCopyDHCPInfo
SCDynamicStoreCopySpecific

Group

SCDynamicStoreCopyComputerName
SCDynamicStoreCopyConsoleUser
SCDynamicStoreCopyLocalHostName
SCDynamicStoreCopyLocation
SCDynamicStoreCopyProxies

SCDynamicStoreKey
SCNetwork
SCNetworkConfiguration
SCNetworkConnection
SCNetworkReachability
SCPreferences
SCPreferencesPath

Filter /

System Configuration / SCDynamicStoreCopySpecific / SCDynamicStoreCopyConsoleUser

Function

SCDynamicStoreCopyConsoleUser

Returns information about the user currently logged into the system.

macOS 10.1+

```
CFStringRef SCDynamicStoreCopyConsoleUser(SCDynamicStoreRef store, uid_t *uid, gid
```

Parameters

`store`
The dynamic store session that should be used for communication with the server. Pass `NULL` to use a temporary session.

`uid`
A pointer to memory that, on output, is filled with the user ID of the currently logged-in user. If `NULL`, this value is not returned.

`gid`
A pointer to memory that, on output, is filled with the group ID of the currently logged-in user. If `NULL`, this value is not returned.

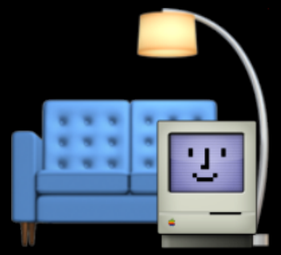
Return Value

Success leads to success

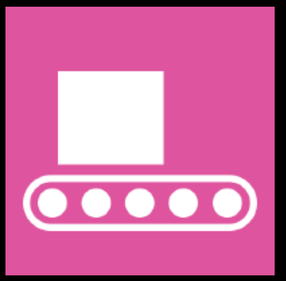
Mac Admin tools in Python

- crankd
- Recipe Robot
- Reposado
- Outset
- offset
- aamporter
- Commandment
- installinstallmacos.py
- munki-facts
- recache
- Nudge

- instaUp2Date
- createOSXinstallPkg
- pycreateuserpkg
- Imagr
- bsdpy
- Crypt
- AutoDMG
- AutoNBI
- Munki
- erase-install
- SOFA



- Zentral
- munkipkg
- dockutil
- quickpkg
- mcxToProfile
- Sal
- AutoPkg
- vfuse
- auto_logout
- installapplications
- munki-rebrand



Why move on?

#1: Python no longer included

Scripting Language Runtimes

Deprecations

- Scripting language runtimes such as Python, Ruby, and Perl are included in macOS for compatibility with legacy software. Future versions of macOS won't include scripting language runtimes by default, and might require you to install additional packages. If your software depends on scripting languages, it's recommended that you bundle the runtime within the app. (49764202)
- Use of Python 2.7 isn't recommended as this version is included in macOS for compatibility with legacy software. Future versions of macOS won't include Python 2.7. Instead, it's recommended that you run `python3` from within Terminal. (51097165)

Dec 9, 2019

gregneagle

v4.0.0

9ddc5a0

Compare

Munki 4.0 Official Release

This is the official release of Munki 4.0, a major architectural change to the Munki tools.

Munki 4 removes the dependency on Apple's Python, and includes its own copy of Python 3.7.4.
Functionality and features are intended to be identical to Munki 3.6.4.

See Introduction to Munki 4 for more information on the architectural changes.

Changes
The package and client code in this release is identical to Munki 4.0 Release Candidate 1.
The only changes in the repo between v4.0.0RC1 and v4.0.0 are to the code/tools/make_munki_mpkg.sh
code/tools/make_munki_mpkg_DEP.sh package build scripts.

AutoPkg users -- IMPORTANT NOTE:
If you use AutoPkg, do not use the munkitools3.munki recipe to import this release, as it will not import
embedded Python package and any clients "upgraded" with the results will be broken.
A new munkitools4.munki recipe is available in the AutoPkg recipes repo.

▸ **Assets** 3

relocatable-python

Public

Pin Unwatch 8 Fork 41 Star 155

main 1 Branch 0 Tags Go to file t Add file Code

WardsParadox Additional fixes around error codes (#21) 67648ff · last year 45 Commits

locallibs	Additional fixes around error codes (#21)	last year
.gitignore	First code commit	6 years ago
LICENSE	Initial commit	6 years ago
README.txt	Add Universal Python testing script (#26)	last year
make_relocatable_python_framework.py	Change shebang to /usr/bin/python3. Since the tool requir...	2 years ago
python_universal_tester.sh	Add Universal Python testing script (#26)	last year
requirements_python2_recommended.txt	Fixes for building on Apple silicon; shebang fix for some ...	3 years ago
requirements_python3_recommended.txt	Fixes for building on Apple silicon; shebang fix for some ...	3 years ago
research_notes.txt	First code commit	6 years ago

README Apache-2.0 license

```
This is a tool to make a relocatable Python framework containing PyObjC.

NOTE: while the resulting frameworks (and interpreters) have been successfully used in several projects (among
```

About

A tool for building standalone relocatable Python.framework bundles

Readme
Apache-2.0 license
Activity
155 stars
8 watching
41 forks

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

Contributors 7

Update Python to Version 3.12 #1204

Closed **skargbo** opened this issue on Feb 19 · 19 comments

skargbo commented on Feb 19

It appears that the current Python version used in the MSC project is vulnerable to a number of high severity security issues that have been disclosed to the public. To address these vulnerabilities, I propose bumping the Python version 3.12, which includes patches for these CVEs.

The following CVEs have been identified in the current version 3.10 of the Python framework used by MSC

- CVE-2018-25032
- CVE-2019-12900
- CVE-2020-10735
- CVE-2021-28861
- CVE-2022-45061
- CVE-2023-24329

Assignees

No one—assign you

Labels

None yet

Projects

None yet

Milestone

No milestone

Development

Create a branch fo

All Technologies

macOS Release Notes

Python

Deprecations

- Python 2.7 was removed from macOS in this update. Developers should use Python 3 or an alternative language instead. (39795874)

StoreKit

New Features

- SKTestSession has three new methods to simulate a subscription requiring price increase consent, simulate consenting to a pending price increase, and simulate declining a price increase in automated tests. (84556183)

- SKTestSession has two new Boolean properties to simulate billing retry and grace period in automated tests. You can identify and simulate the resolution of billing retry

Releases / v3.12.1.80742

Python 3.12.1.80742 Latest

github-actions released this Mar 14 v3.12.1.80742 5e5340a

Notes

Python 3.12.1 Framework

Changes

- Upgraded Python to 3.12.1
  Note: Some of these updates may have breaking changes. Always test your code before deploying to production!
- Uses `--no-rcs` in preinstall shebang

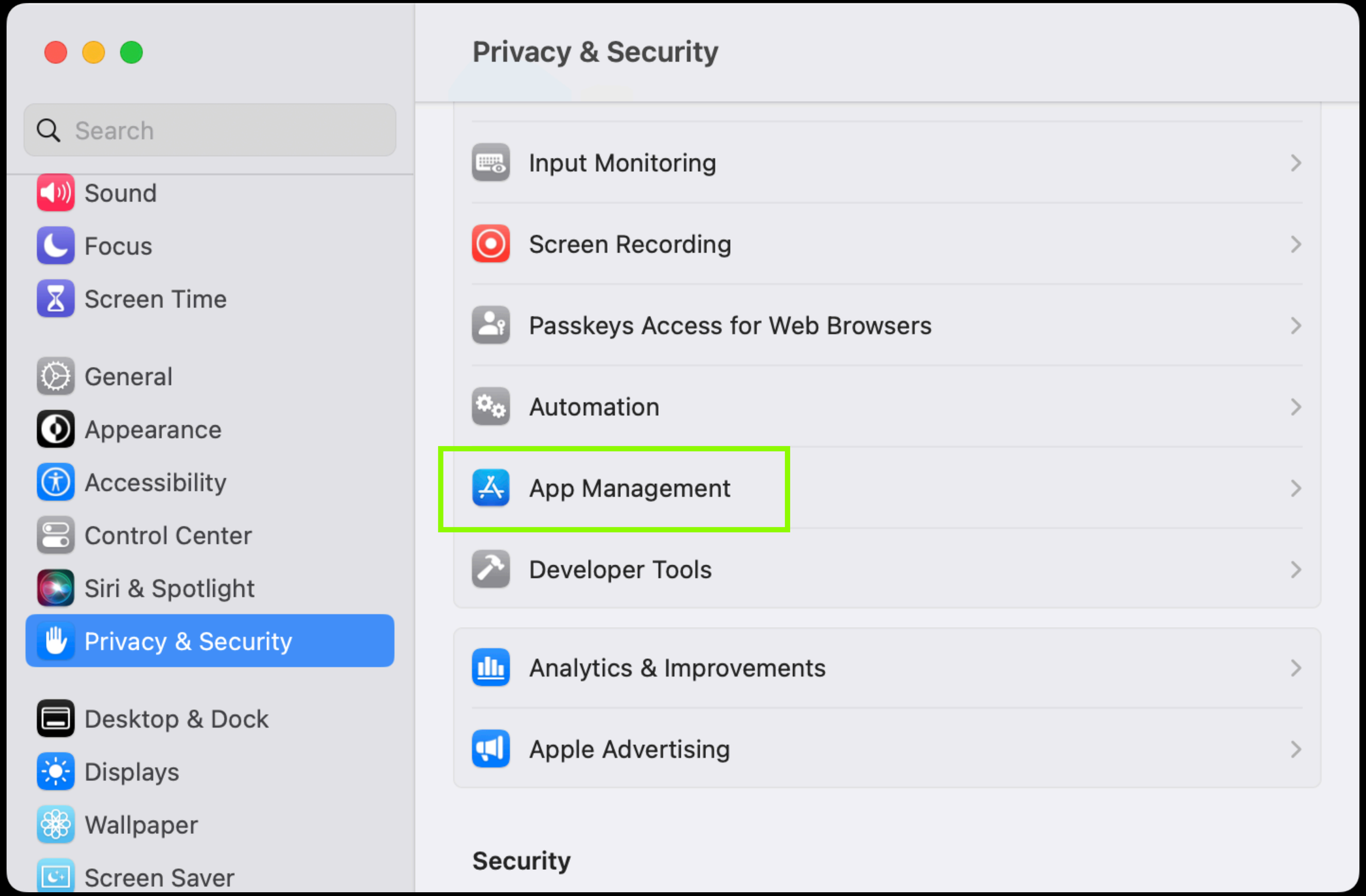
Please see the `requirements_recommended.txt` for the current libraries being used.

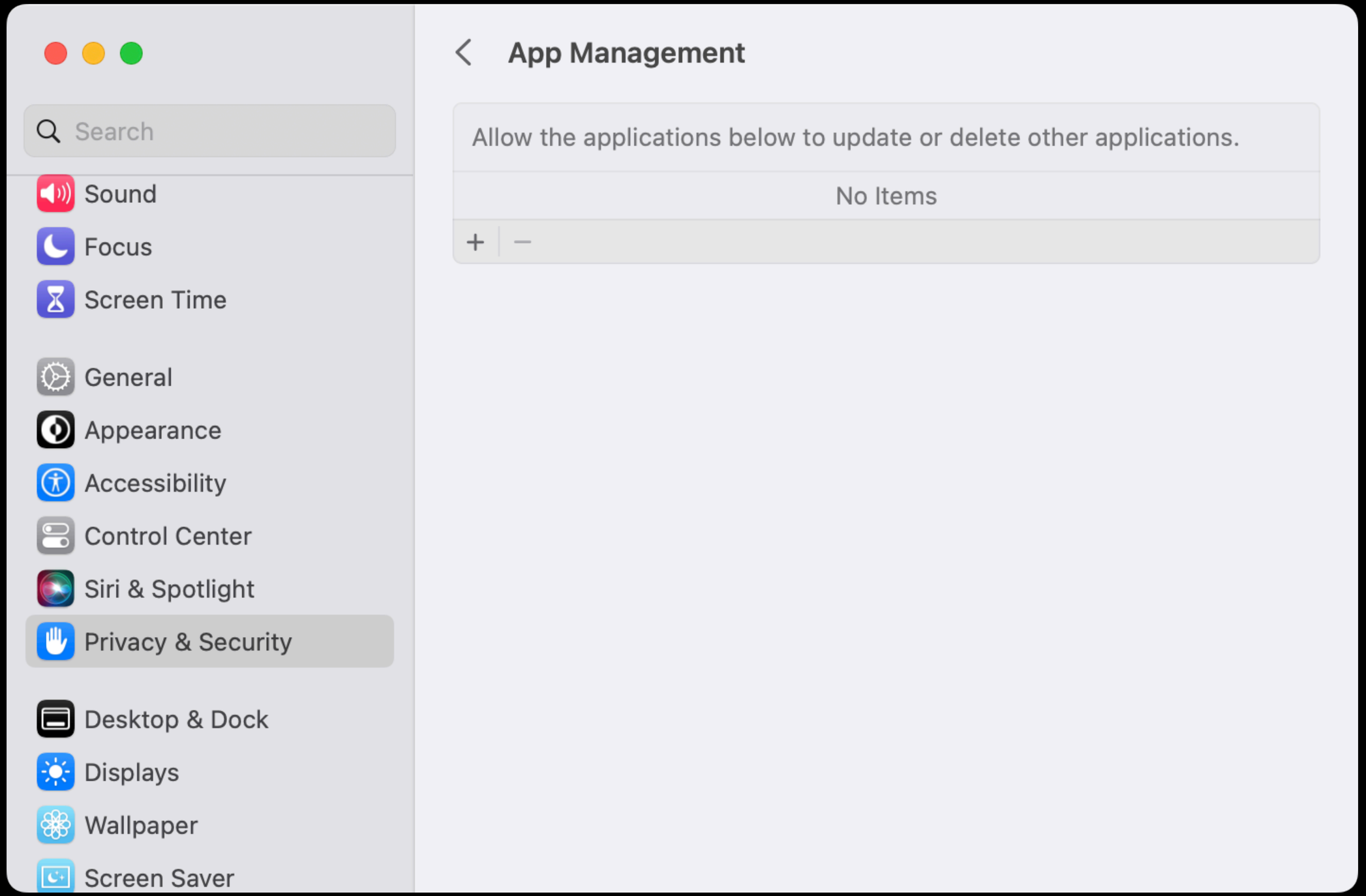
- 5e5340a - cleanup old scripts and move to --no-rcs shebang (#59)

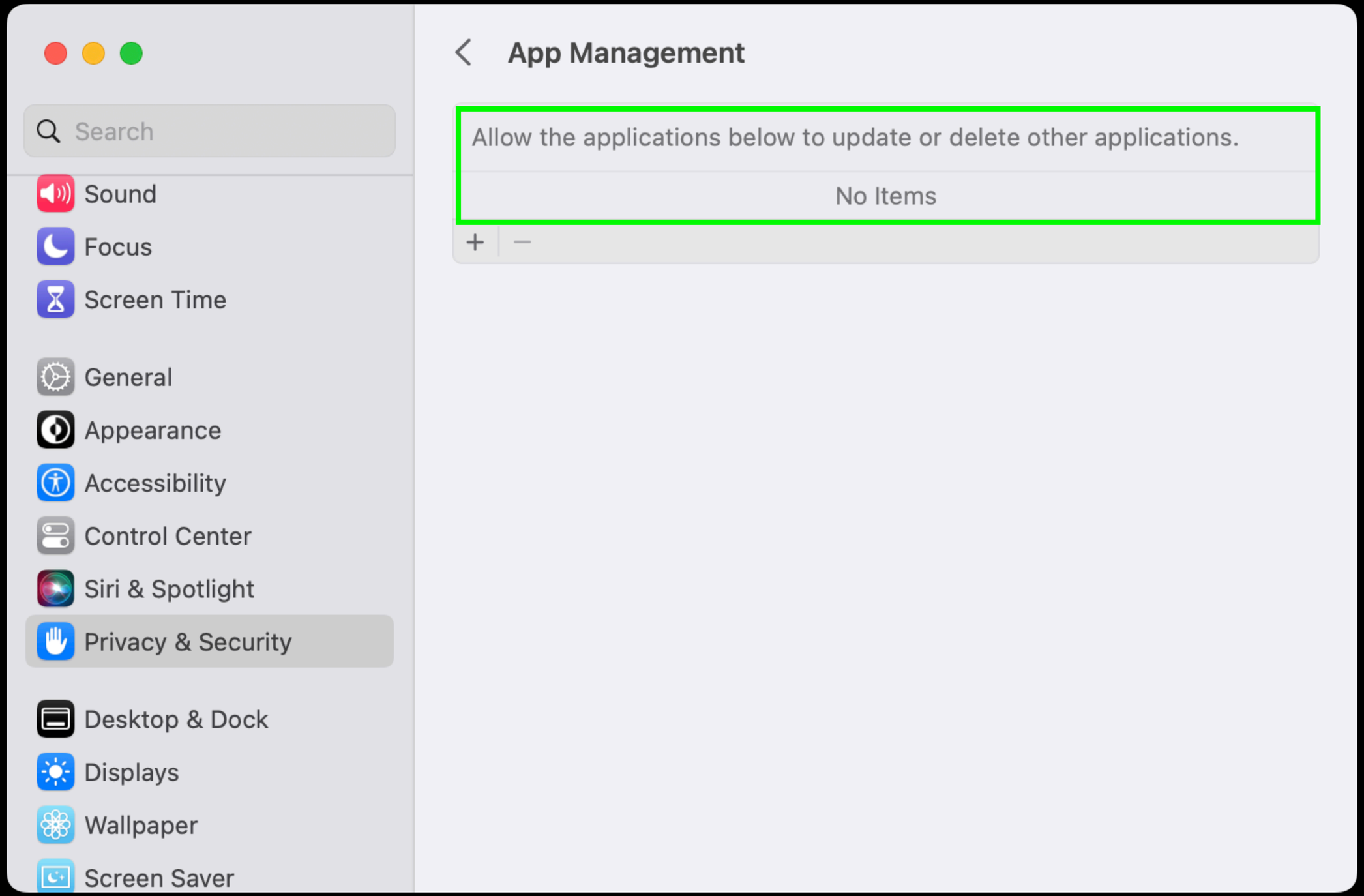

Pain

#2: Code signing

App Management TCC


Search
Sound
Focus
Screen Time
General
Appearance
Accessibility
Control Center
Siri & Spotlight
Privacy & Security
Desktop & Dock
Displays
Wallpaper
Screen Saver
Privacy & Security
Input Monitoring
Screen Recording
Passkeys Access for Web Browsers
Automation
App Management
Developer Tools
Analytics & Improvements
Apple Advertising
Security


Search
Sound
Focus
Screen Time
General
Appearance
Accessibility
Control Center
Siri & Spotlight
Privacy & Security
Desktop & Dock
Displays
Wallpaper
Screen Saver
App Management
Allow the applications below to update or delete other applications.
No Items


Search
Sound
Focus
Screen Time
General
Appearance
Accessibility
Control Center
Siri & Spotlight
Privacy & Security
Desktop & Dock
Displays
Wallpaper
Screen Saver
App Management
Allow the applications below to update or delete other applications.
No Items

Search

- Software
- Categories
- My Items
- Updates

Managed Software Center

Your source for software for your Mac.

Administration | Cloud Storage | Communication | Developer | Drivers | Editorial | Entertainment | File Sharing | Graphics | Internet | Media | Production | Productivity | Remote Access | Story | Text Editing | Text Editors | Uncategorized | Utilities | Virtualization | drivers

All items

Adobe Acrobat Reader DC
Productivity - Adobe
INSTALL

Amazon WorkSpaces
Remote Access - Amazon
Installed
REMOVE

Aspera Connect
Utilities - IBM Corp
INSTALL

Atom
Text Editors - GitHub, Inc.
INSTALL

Autodesk Maya 2018
Production - Autodesk
INSTALL

Autodesk Maya 2019
Production - Autodesk
INSTALL

Autodesk Meshmixer
Production - Autodesk
INSTALL

Blender
Production - The Blender Foun...
INSTALL

BlueJeans
Communication - Blue Jeans N...
Installed
REMOVE

```
$ file /usr/local/munki/managedsoftwareupdate
/usr/local/munki/managedsoftwareupdate: Mach-O universal binary with 2 architectures: [x86_64:Mach-O 64-bit executable x86_64] [arm64:Mach-O 64-bit executable arm64]
/usr/local/munki/managedsoftwareupdate (for architecture x86_64): Mach-O 64-bit executable x86_64
/usr/local/munki/managedsoftwareupdate (for architecture arm64):  Mach-O 64-bit executable arm64
```

Being Responsible

A tale of Transparency, Consent, and Control on macOS

Greg Neagle, June 2023

https://www.youtube.com/watch?v=DcrfCGqqjkA

Code signing

#3 The tide has turned

harder to find someone to help

Where are people going?


Back to shell

Shell advantages:

- Available by default! (sh, bash, zsh at least)
- Cross-platform
- Simpler syntax and learning curve
- Super easy to call other tools
- Low barrier of entry to new contributors/maintainers

Contributors

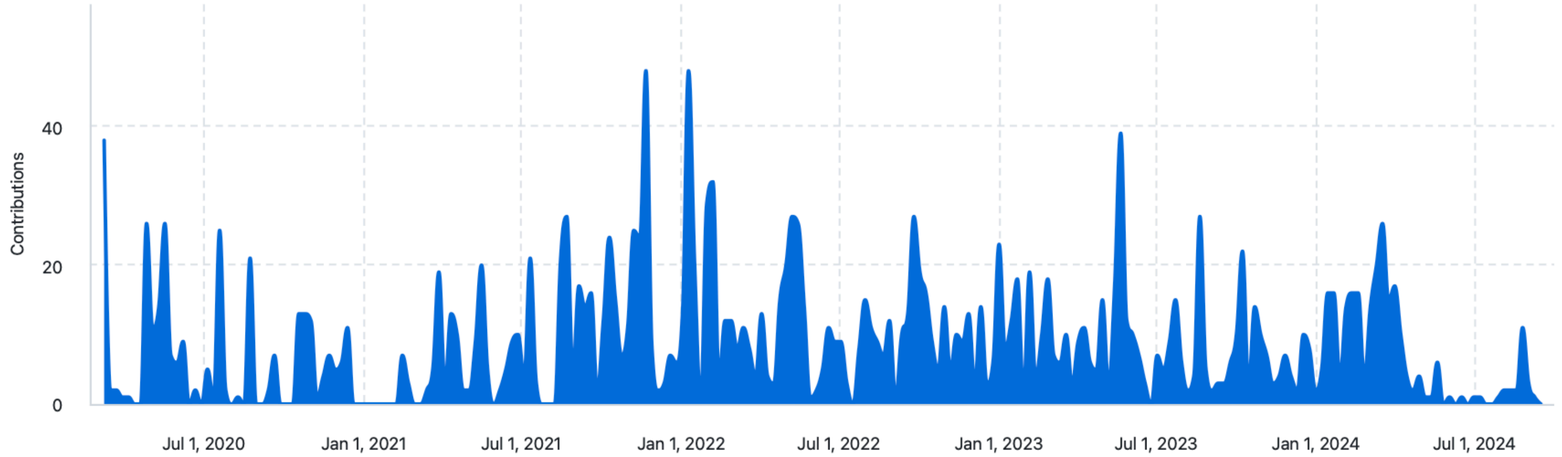
Beta Give feedback

Period: All | Contributions: Commits

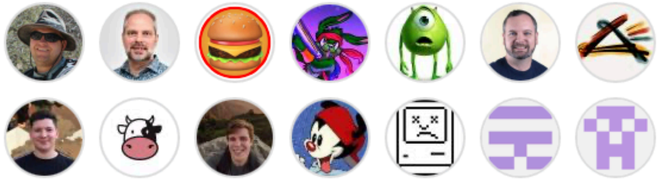
Contributions per week to main, excluding merge commits

Commits over time

From Mar 7, 2020 to Sep 14, 2024



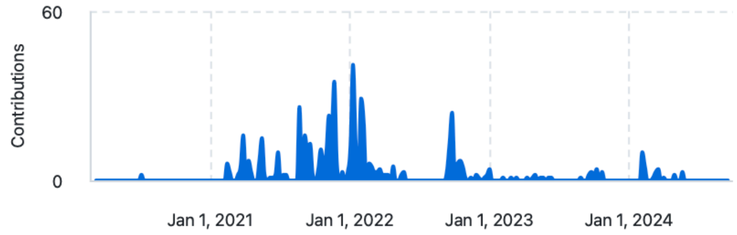
Contributors 172



+ 158 contributors

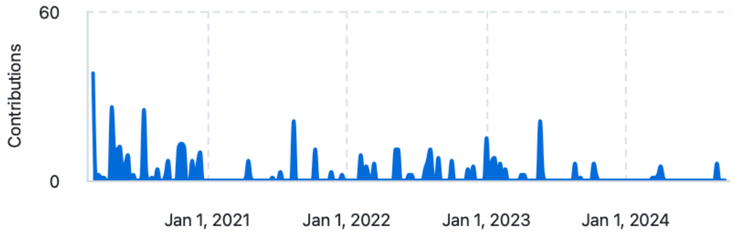
Theile #1

563 commits 22,933 ++ 13,208 --



scriptingosx #2

477 commits 31,511 ++ 22,155 --



Tool	shell	# of lines	URL
installomator	zsh	9000+	https://github.com/Installomator/Installomator
super (S.U.P.E.R.M.A.N)	bash	8000+	https://github.com/Macjutsu/super
Baseline	zsh	1700+	https://github.com/SecondSonConsulting/Baseline
Renew	zsh	850+	https://github.com/SecondSonConsulting/Renew

Perl? Ruby?

Scripting Language Runtimes

Deprecations

- Scripting language runtimes such as Python, Ruby, and Perl are included in macOS for compatibility with legacy software. Future versions of macOS won't include scripting language runtimes by default, and might require you to install additional packages. If your software depends on scripting languages, it's recommended that you bundle the runtime within the app. (49764202)
- Use of Python 2.7 isn't recommended as this version is included in macOS for compatibility with legacy software. Future versions of macOS won't include Python 2.7. Instead, it's recommended that you run `python3` from within Terminal. (51097165)

Compiled languages

• Go	MicroMDM, NanoMDM
• C#	SupportCompanion
• Objective-C	lots, though not many recent
• Swift	lots! (Nudge, DEPNotify, SwiftDialog, desktoppr, Outset...)

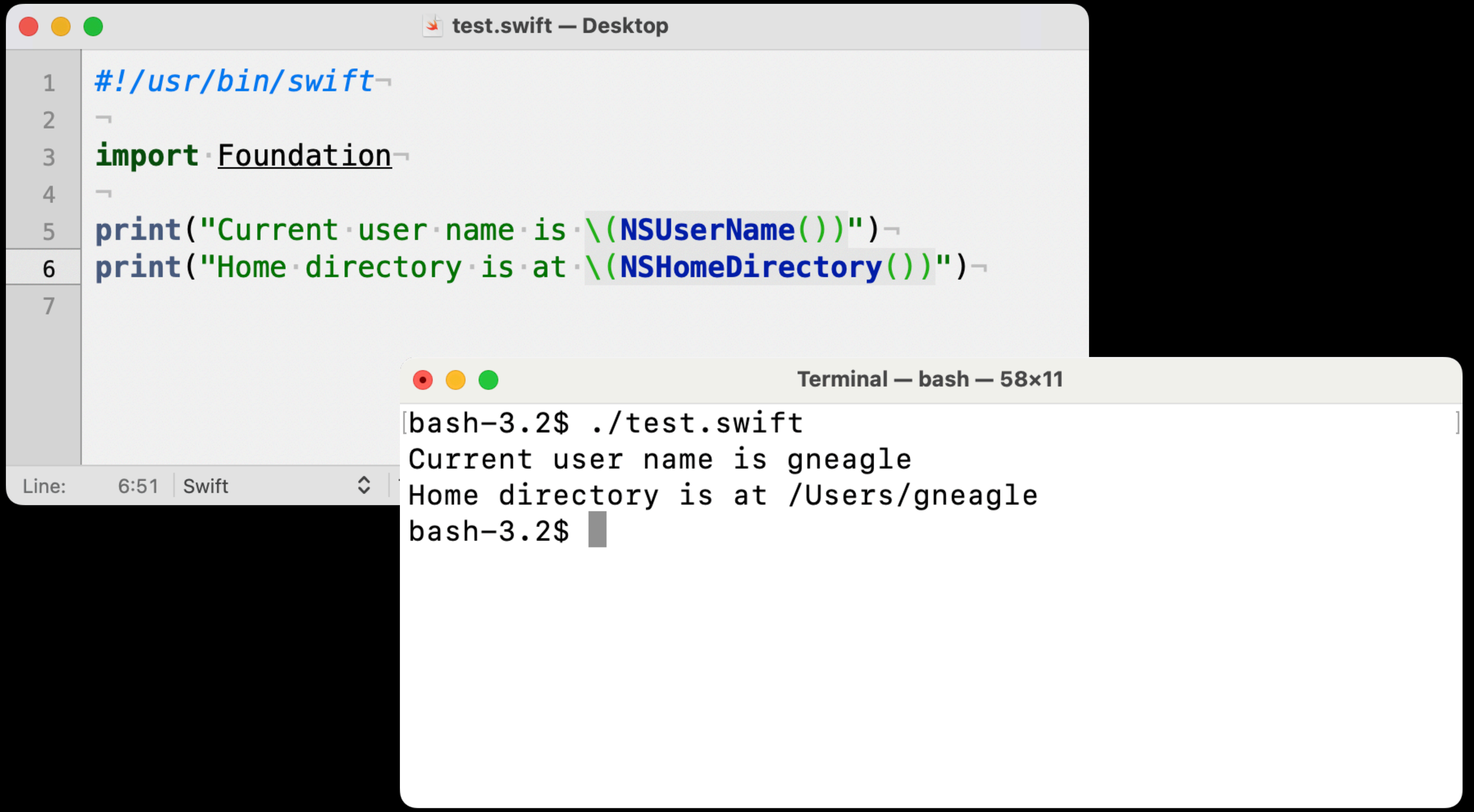
Y  ?

• Scripting language	✅	❌*
• Included with macOS	❌	✅
• Broad standard library	✅	✅
• Large number of add-on modules	✅	✅
• Lots of examples	✅	✅
• Access to Apple frameworks	✅	✅
• Write native-looking GUI apps	✅	✅

Scripting languages

```
bash-3.2$ munki-python
Python 3.12.2 (v3.12.2:6abddd9f6a, Feb  6 2024, 17:02:06) [Clang
13.0.0 (clang-1300.0.29.30)] on darwin
Type "help", "copyright", "credits" or "license" for more
information.
>>> import os
>>> os.uname()
posix.uname_result(sysname='Darwin', nodename='bagel',
release='23.6.0', version='Darwin Kernel Version 23.6.0: Wed Jul
31 20:48:52 PDT 2024; root:xnu-10063.141.1.700.5~1/
RELEASE_ARM64_T6020', machine='arm64')
>>> os.uname().machine
'arm64'
>>>
```

```
bash-3.2$ swift repl
Welcome to Apple Swift version 5.10 (swiftlang-5.10.0.13
clang-1500.3.9.4).
Type :help for assistance.
  1> import Foundation
  2> NSUserName()
$R0: String = "gneagle"
  3> NSHomeDirectory()
$R1: String = "/Users/gneagle"
  4>
```


test.swift — Desktop
#!/usr/bin/swift
import Foundation
print("Current user name is \(NSUserName())")
print("Home directory is at \(NSHomeDirectory())")
Line: 6:51 Swift
Terminal — bash — 58×11
bash-3.2$ ./test.swift
Current user name is gneagle
Home directory is at /Users/gneagle
bash-3.2$

But don't just take my word for it

Swift - For SysAdmins

- Compiled - so it can be easily signed
- Access to system APIs
- Not easily humanly-readable
- Some things are harder.... Other things are much easier than Bash


Swift
for Apple Admins
MAC · SYS · ADMIN
GOTHENBURG · 2018
CHALMERS
2018

Let's Swift Again

Build Swift Tools for MacAdmins

Armin Briegel
Scripting OSX

2021

Thursday | Friday

Swift - The Eras Tour

(Armin's Version)

13:00 - 13:55

Armin Briegel - Jamf/scriptingosx.com

It has been ten years since Apple introduced Swift. We will Speak Now about the Folklore and Reputation of the programming language, peek into *The Tortured Programmers Department* to see what it takes to build a tool or app, and fearlessly check out some great tools from the community made with Swift.

2024

You won't be alone,
and you don't have to go first


Your device requires a security update (Demo Mode)
Days Remaining To Update: 0
Deferred Count: 0
Update Device
More Info
Defer

LICENSE	LICENSE	5 years ago
README.md	Prepare the last version of Nudge-Python	3 years ago
build-info.json	Prepare the last version of Nudge-Python	3 years ago
example_config.json	prepare core files for python refactor	5 years ago

README | Apache-2.0 license

Nudge-Python (macadmin's Slack #nudge)

Nudge-Python is now considered End Of Life (EOL) as of February 11th, 2021. The last official version is 2.0.1 which enabled Big Sur support through the use of macadmins/python 3.9.1

If you are using a fork of Nudge-Python (a well known one is located here), support will not be offered unless the developer of the fork decides to continue support.

While Nudge-Python does support Big Sur, it is recommended to use the new version of Nudge, based on SwiftUI 5.2. You can find that project here.

On March 1st, 2021, this project will be marked as archived through GitHub.

Nudge functionality overview

Releases 5

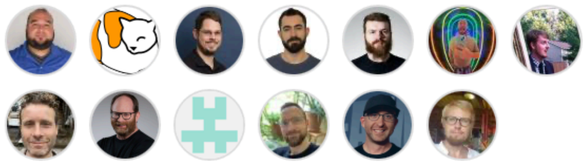
Nudge v2.0.1 - The Final Rele... Latest
on Mar 24, 2021

+ 4 releases

Packages

No packages published

Contributors 13



Languages

Python 96.7% Shell 3.3%

CHANGELOG.md	v2.0.12 - better handling of bad cached sofa feeds	5 days ago
LICENSE	Create LICENSE	3 years ago
Localizable.xcstrings	Update german (de) localization (#649)	5 days ago
README.md	Update README.md to mention 2.0	2 months ago
SECURITY.md	Fixed minor typo	7 months ago
build_nudge.zsh	redo build script for PRs	2 months ago

README | Apache-2.0 license | Security

Nudge (macadmin's Slack #nudge)

Nudge is a multi-linguistic application, offering custom user deferrals, which strongly encourages macOS updates. Written in Swift and SwiftUI.

Nudge will only work on macOS Big Sur 11 and later and is a replacement for the original Nudge, which was written in Python 2/3. If you need to enforce macOS updates for earlier versions, it is recommended to use nudge-python.

Contributors 45

+ 31 contributors

Languages

Swift 94.3% Shell 5.7%

Outset

README

Outset

> NOTE: This instance of Outset is decommissioned and you should start using the Swift-based Outset 4.0 available on the macadmins repo.

Outset is a script which automatically processes packages, profiles, and scripts during the boot sequence, user logins, or on demand.

Requirements

- macOS 10.15+
- python 3.7+

If you need to support 10.14 or lower, stick with the 2.x version.

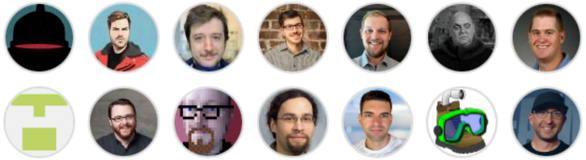
python3 can be installed from one of these sources:

- python.org
- MacAdmins
- Munki

If none of these are on disk, then fall back to Apple's system python3, which can be installed via the Command Line Tools.

No packages published

Contributors 14



Languages

build_outset.zsh	removed .sh extension from preinstall. Added preins...	last year

README | Apache-2.0 license

Outset

Outset is a utility application which automatically processes scripts and packages during the boot sequence, user logins, or on demand.

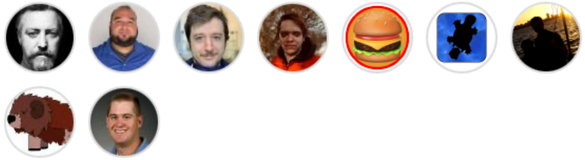
Check out the wiki for more information on how to use Outset or find out how it works.

Requirements

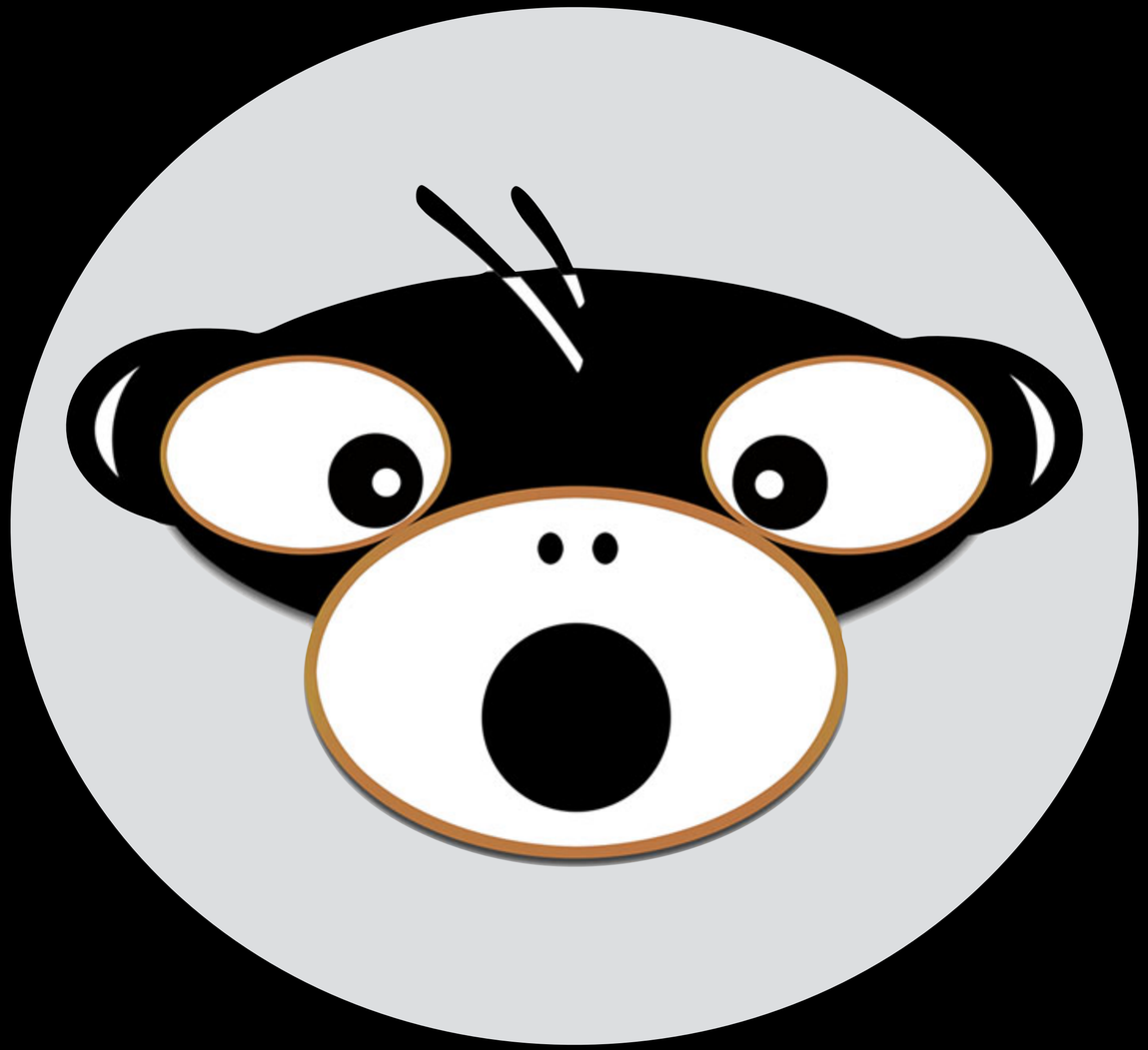
- macOS 10.15+

Usage

Contributors 9

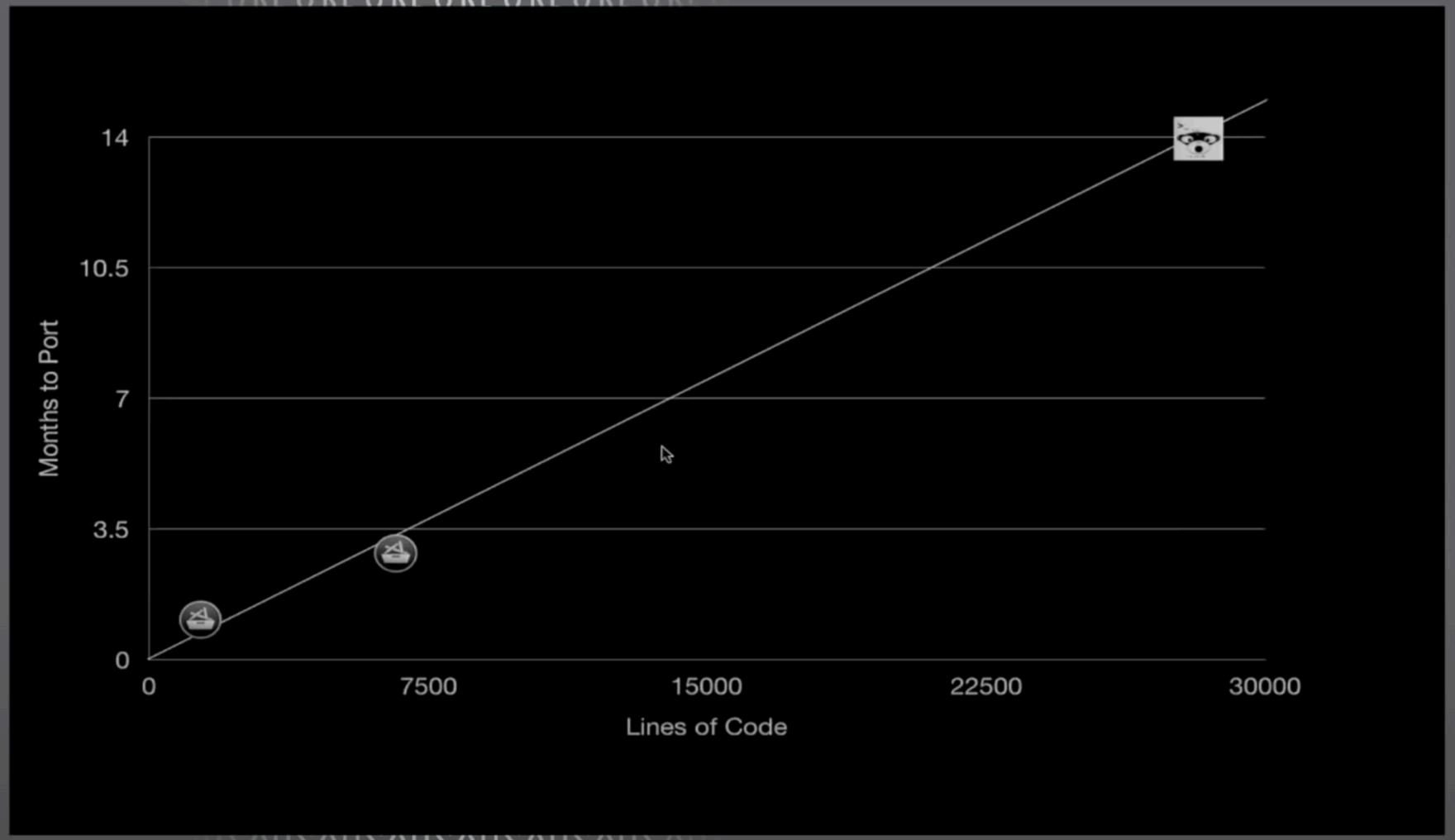


Languages



MAC
DEV
OPS
YVR

June 12-14, 2019
Vancouver, Canada
MDOYVR.com



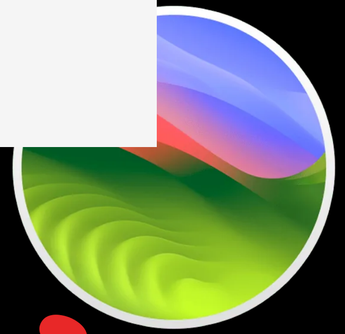
MAC DEV OPS YVR

June 12-14, 2019
Vancouver, Canada
MDOYVR.com

python™ Sunsetting Python 2

Apple Removing Python 2.7: What Admins Need to Know and Do

Kandji Team
Feb 25, 2022
9 min read



zoom

2020

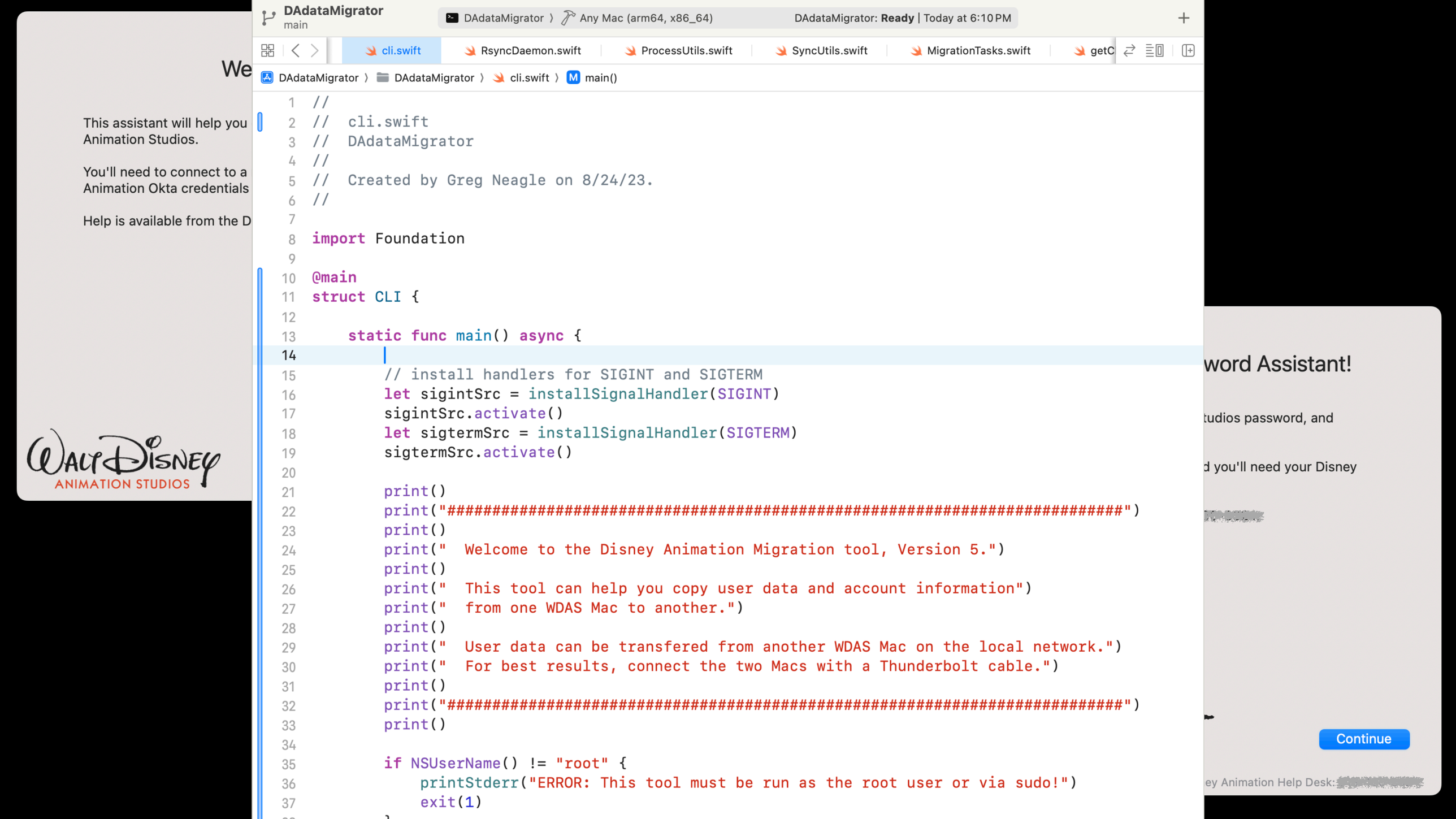
PRESS RELEASE
June 22, 2020

Apple announces Mac transition to Apple silicon

BUSINESS • COVID-19

The Coronavirus Outbreak Has Become the World's Largest Work-From-Home Experiment

7 MINUTE READ


DAdataMigrator
main
DAdataMigrator 〉 Any Mac (arm64, x86_64)
DAdataMigrator: Ready | Today at 6:10 PM
cli.swift
RsyncDaemon.swift
ProcessUtils.swift
SyncUtils.swift
MigrationTasks.swift
DAdataMigrator 〉 DAdataMigrator 〉 cli.swift 〉 main()
1 //
2 // cli.swift
3 // DAdataMigrator
4 //
5 // Created by Greg Neagle on 8/24/23.
6 //
7
8 import Foundation
9
10 @main
11 struct CLI {
12
13 static func main() async {
14
15 // install handlers for SIGINT and SIGTERM
16 let sigintSrc = installSignalHandler(SIGINT)
17 sigintSrc.activate()
18 let sigtermSrc = installSignalHandler(SIGTERM)
19 sigtermSrc.activate()
20
21 print()
22 print("##")
23 print()
24 print(" Welcome to the Disney Animation Migration tool, Version 5.")
25 print()
26 print(" This tool can help you copy user data and account information")
27 print(" from one WDAS Mac to another.")
28 print()
29 print(" User data can be transfered from another WDAS Mac on the local network.")
30 print(" For best results, connect the two Macs with a Thunderbolt cable.")
31 print()
32 print("##")
33 print()
34
35 if NSUserName() != "root" {
36 printStderr("ERROR: This tool must be run as the root user or via sudo!")
37 exit(1)
This assistant will help you
Animation Studios.
You'll need to connect to a
Animation Okta credentials
Help is available from the D
word Assistant!
udios password, and
d you'll need your Disney
Continue
ey Animation Help Desk:

MAC
DEV
OPS
YVR

managedsof...areupdate.py | installhelper | selfservice.py | dmgutils.py

```python
def mountdmg(dmgpath, use_shadow=False, use_existing_mounts=False,
             random_mountpoint=True, skip_verification=False):
    """
    Attempts to mount the dmg at dmgpath
    and returns a list of mountpoints
    If use_shadow is true, mount image with shadow file
    If random_mountpoint, mount at random dir under /tmp
    """
    mountpoints = []
    dmgname = os.path.basename(dmgpath)

    if use_existing_mounts:
        # Check if this dmg is already mounted
        # and if so, bail out and return the mountpoints
        if diskImageIsMounted(dmgpath):
            mountpoints = mount_points_for_disk_image(dmgpath)
            return mountpoints

    # Attempt to mount the dmg
    stdin = b''
    if dmg_has_sla(dmgpath):
        stdin = b'Y\n'
        display.display_detail(
            'NOTE: %s has embedded Software License Agreement' % dmgna
    cmd = ['/usr/bin/hdiutil', 'attach', dmgpath, '-nobrowse', '-plist
    if random_mountpoint:
        cmd.extend(['-mountRandom', '/tmp'])
    if use_shadow:
        cmd.append('-shadow')
    if skip_verification:
        cmd.append('-noverify')
    proc = subprocess.Popen(cmd,
                            bufsize=-1, stdout=subprocess.PIPE,
                            stderr=subprocess.PIPE, stdin=subprocess.P
    (out, err) = proc.communicate(stdin)
    if proc.returncode:
        display.display_error(
            u'Error: "%s" while mounting %s.'
            % (err.decode('UTF-8').rstrip(), dmgname))
    (pliststr, out) = utils.getFirstPlist(out)
    if pliststr:
        try:
            plist = readPlistFromString(pliststr)
            for entity in plist.get('system-entities', []):
                if 'mount-point' in entity:
                    mountpoints.append(entity['mount-point'])
        except PlistReadError as err:
            display.display_error("%s" % err)
            display.display_error(
                'Bad plist string returned when mounting diskimage %s:
                % (dmgname, pliststr))
    return mountpoints
```

Line: 251:18-2... | Python | Soft Tabs: 4 | Symbols

munki
swift-cli

Any Mac (arm | Build **Succeeded** | 9/20/24 at 11:42 AM

dmgutils.swift

GitFileRepo.swift | RepoFactory.swift | prefs.swift | dmgutils.swift

munki ⟩ shared ⟩ utils ⟩ dmgutils.swift ⟩ pathIsVolumeMountPoint(_:)

- shared
  - headers
  - network
  - updatecheck
  - utils
    - cliutils.swift
    - dmgutils.swift
    - fileutils.swift
    - iconutils.swift
    - osutils.swift
    - pkgutils.swift
    - plistutils.swift
    - scriptutils.swift
    - utils.swift
    - versionutils.swift
    - ODutils.swift
  - socket
  - installer
  - munkirepo
  - admin
  - prefs.swift
  - constants.swift
  - display.swift
  - munkilog.swift
  - reports.swift
  - munkihash.swift
  - osinstaller.swift
  - version.swift
  - errors.swift
  - sqlite3.swift
  - launchd.swift
  - appusage.swift
  - xattr.swift
  - processes.swift
  - munkistatus.swift
  - powermanager.swift
  - stoprequested.swift

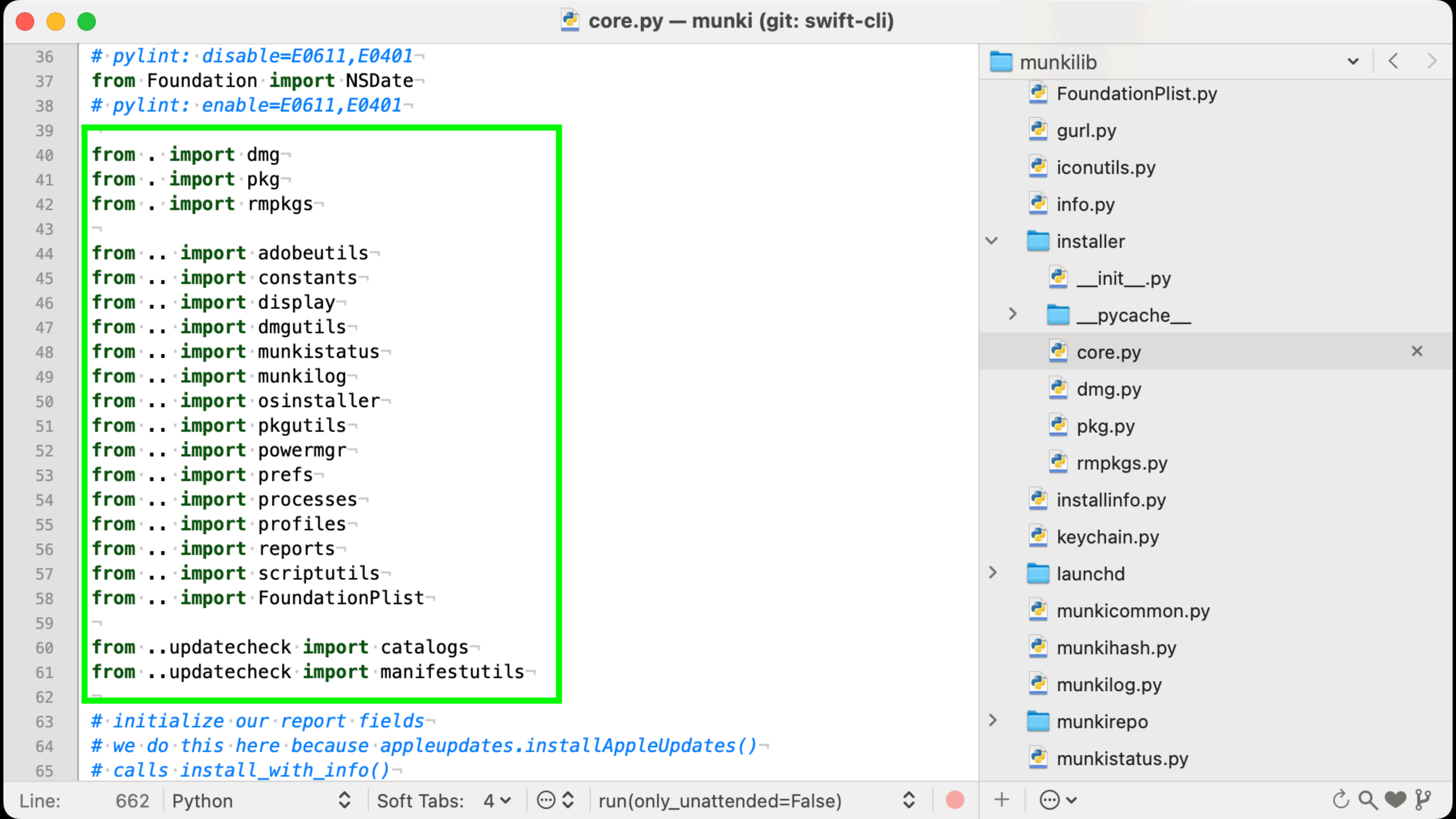
Filter

```swift
/// Attempts to mount the dmg at dmgpath
/// and returns the first item in the list of mountpoints
/// If use_shadow is true, mount image with shadow file
/// If random_mountpoint, mount at random dir under /tmp
func mountdmg(_ dmgPath: String,
              useShadow: Bool = false,
              useExistingMounts: Bool = false,
              randomMountpoint: Bool = true,
              skipVerification: Bool = false) throws -> String
{
    let dmgName = (dmgPath as NSString).lastPathComponent

    if useExistingMounts, let currentMountPoint =
        mountPointForDiskImage(dmgPath) {
        return currentMountPoint
    }

    // attempt to mount the dmg
    var stdIn = ""
    if dmgHasSLA(dmgPath) {
        stdIn = "Y\n"
        displayDetail("NOTE: \(dmgName) has embedded Software
            License Agreement")
    }
    var arguments = ["attach", dmgPath, "-nobrowse"]
    if randomMountpoint {
        arguments += ["-mountRandom", "/tmp"]
    }
    if useShadow {
        arguments.append("-shadow")
    }
    if skipVerification {
        arguments.append("-noverify")
    }
    let plistData = try hdiutilData(arguments: arguments, stdIn:
        stdIn)
    if let systemEntities = plistData["system-entities"] as?
        [PlistDict] {
        for entity in systemEntities {
```

38 characters

Things I've Learned

Code organization

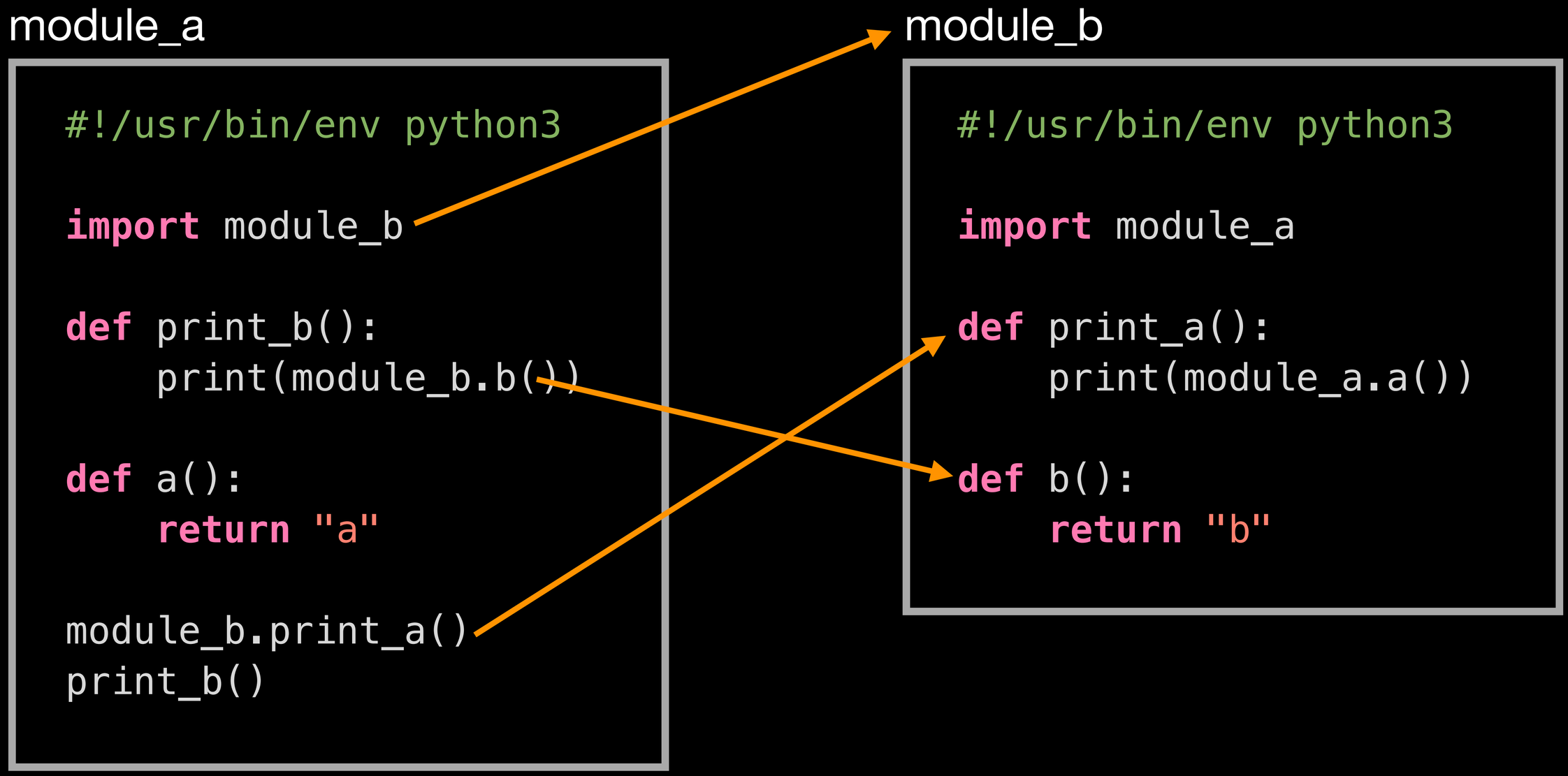

core.py — munki (git: swift-cli)
# pylint: disable=E0611,E0401
from Foundation import NSDate
# pylint: enable=E0611,E0401
from . import dmg
from . import pkg
from . import rmpkgs
from .. import adobeutils
from .. import constants
from .. import display
from .. import dmgutils
from .. import munkistatus
from .. import munkilog
from .. import osinstaller
from .. import pkgutils
from .. import powermgr
from .. import prefs
from .. import processes
from .. import profiles
from .. import reports
from .. import scriptutils
from .. import FoundationPlist
from ..updatecheck import catalogs
from ..updatecheck import manifestutils
# initialize our report fields
# we do this here because appleupdates.installAppleUpdates()
# calls install_with_info()
munkilib
FoundationPlist.py
gurl.py
iconutils.py
info.py
installer
__init__.py
__pycache__
core.py
dmg.py
pkg.py
rmpkgs.py
installinfo.py
keychain.py
launchd
munkicommon.py
munkihash.py
munkilog.py
munkirepo
munkistatus.py
Line: 662
Python
Soft Tabs: 4
run(only_unattended=False)

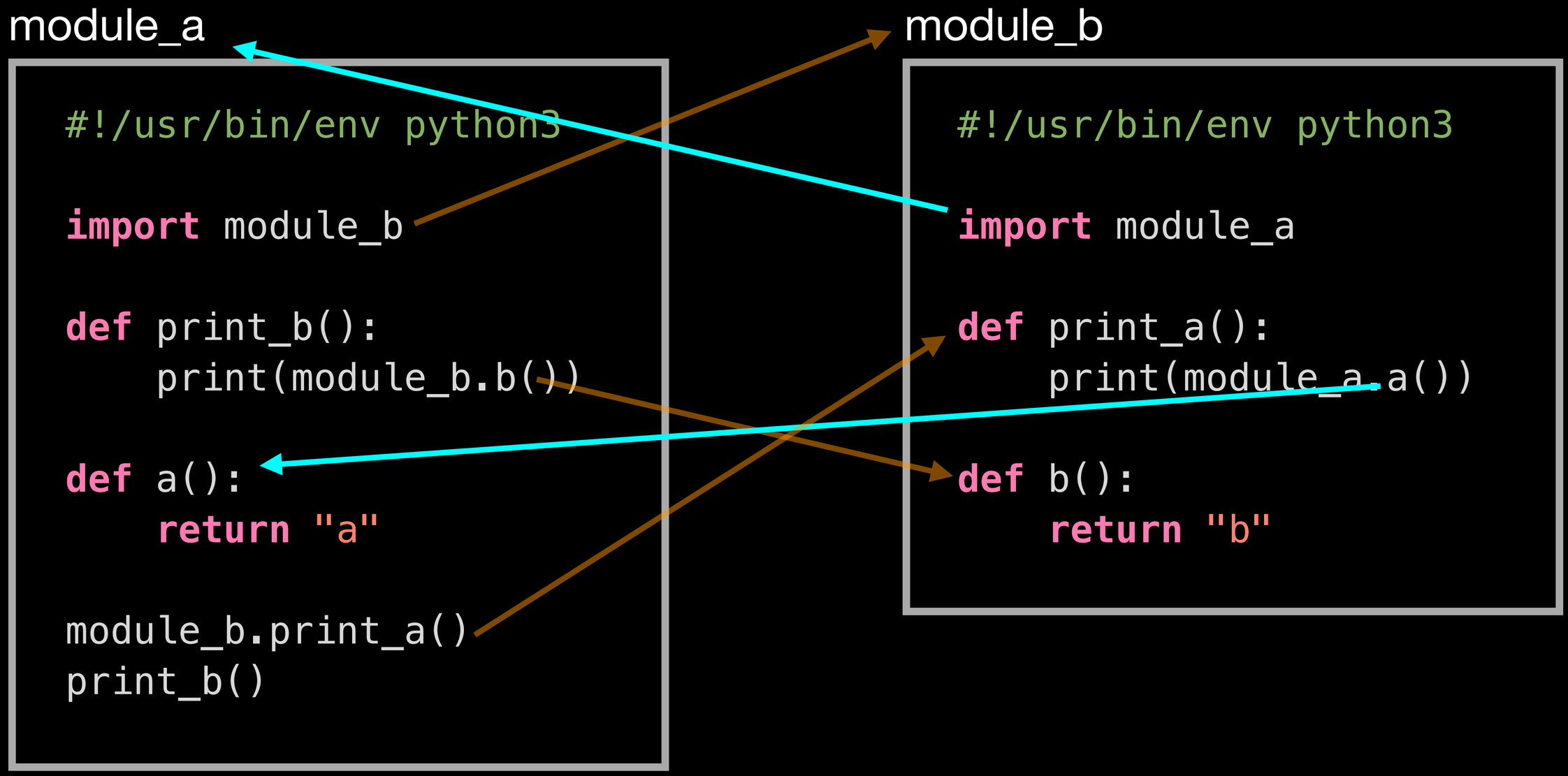
module_a

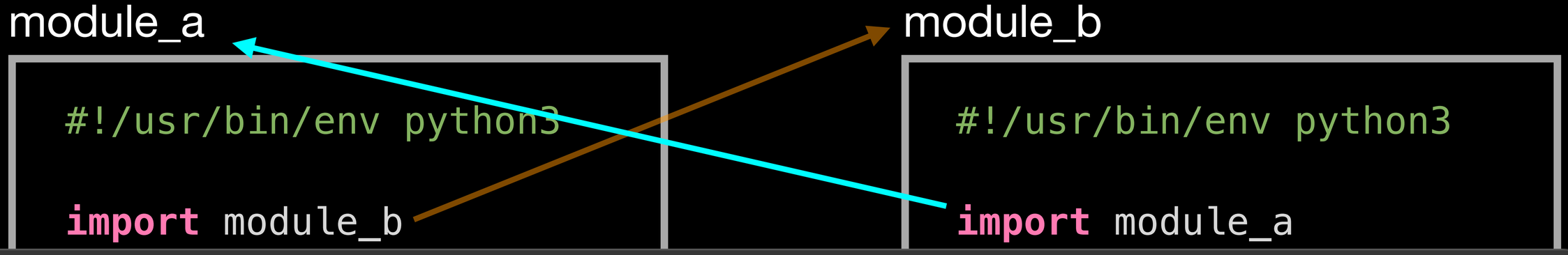
```
#!/usr/bin/env python3

import module_b

def print_b():
    print(module_b.b())

def a():
    return "a"

module_b.print_a()
print_b()
```

module_b

```
#!/usr/bin/env python3

import module_a

def print_a():
    print(module_a.a())

def b():
    return "b"
```

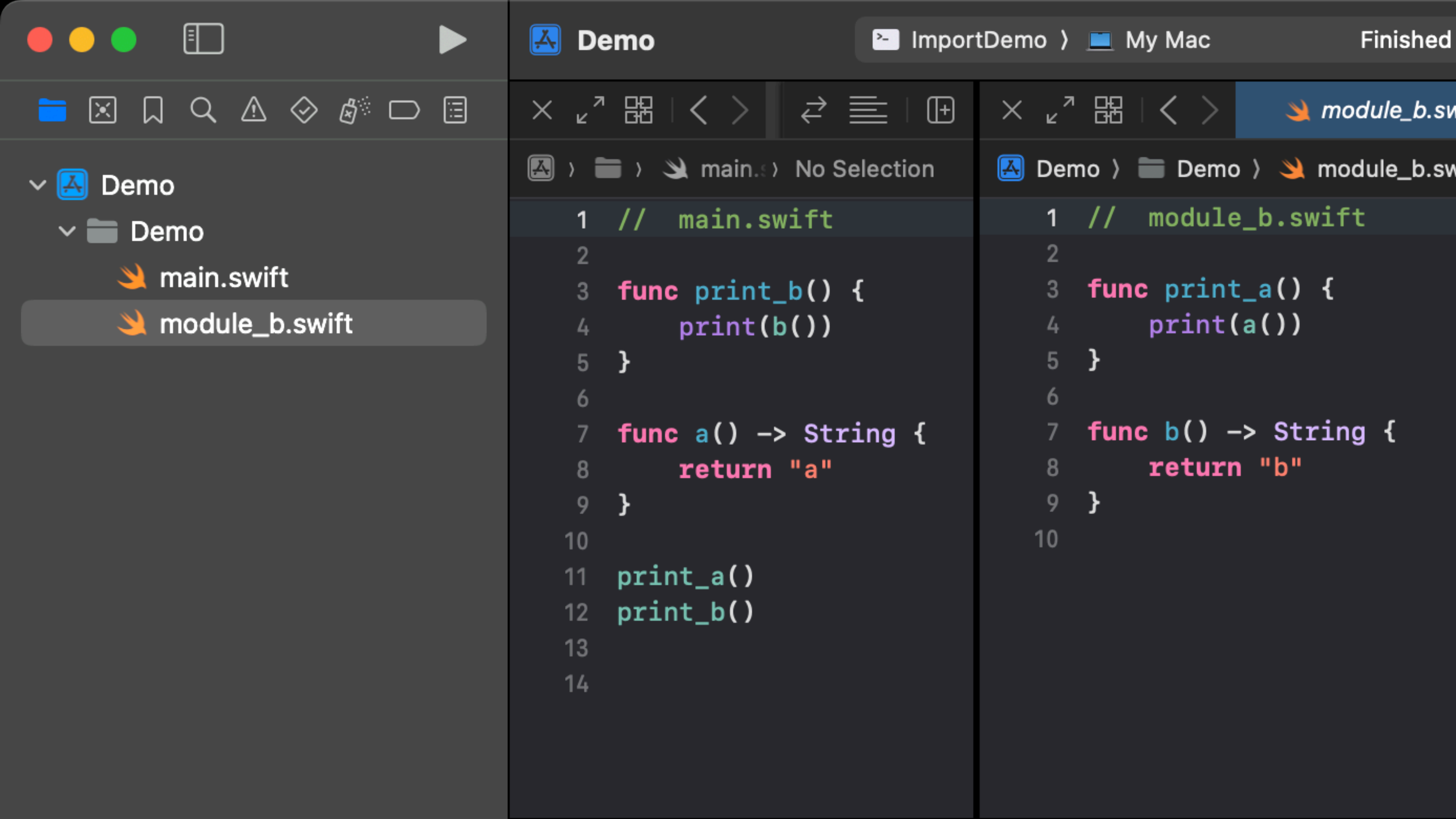

module_a
#!/usr/bin/env python3
import module_b
def print_b():
print(module_b.b())
def a():
return "a"
module_b.print_a()
print_b()
module_b
#!/usr/bin/env python3
import module_a
def print_a():
print(module_a.a())
def b():
return "b"


module_a
#!/usr/bin/env python3
import module_b
def print_b():
print(module_b.b())
def a():
return "a"
module_b.print_a()
print_b()
module_b
#!/usr/bin/env python3
import module_a
def print_a():
print(module_a.a())
def b():
return "b"



Terminal — bash — 72×17

```
bash-3.2$ ./module_a.py
Traceback (most recent call last):
  File "/Users/gneagle/Dropbox/MacSysAdmin2024/scripts/circular imports/
./module_a.py", line 3, in <module>
    import module_b
  File "/Users/gneagle/Dropbox/MacSysAdmin2024/scripts/circular imports/
module_b.py", line 3, in <module>
    import module_a
  File "/Users/gneagle/Dropbox/MacSysAdmin2024/scripts/circular imports/
module_a.py", line 11, in <module>
    module_b.print_a()
AttributeError: partially initialized module 'module_b' has no attribute
 'print_a' (most likely due to a circular import)
bash-3.2$
```


Demo
ImportDemo
My Mac
Finished
Demo
Demo
main.swift
module_b.swift
main.swift
No Selection
// main.swift
func print_b() {
print(b())
}
func a() -> String {
return "a"
}
print_a()
print_b()
module_b.swift
Demo
Demo
module_b.swift
// module_b.swift
func print_a() {
print(a())
}
func b() -> String {
return "b"
}

main.swift

```swift
//  main.swift

func print_b() {
    print(b())
}

func a() -> String {
    return "a"
}

print_a()
print_b()
```

module_b.swift

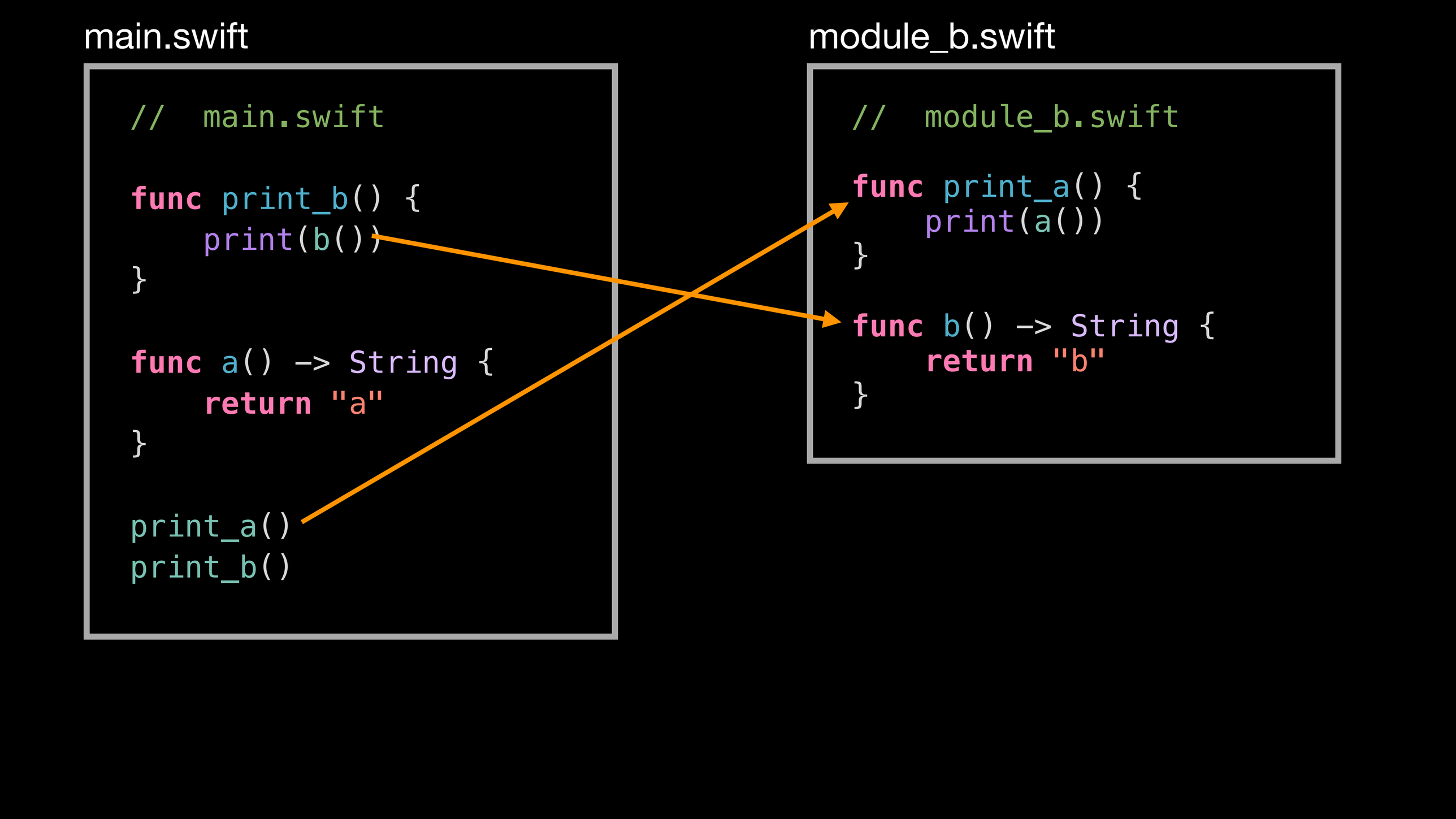
```swift
//  module_b.swift

func print_a() {
    print(a())
}

func b() -> String {
    return "b"
}
```

main.swift

```swift
//  main.swift

func print_b() {
    print(b())
}

func a() -> String {
    return "a"
}

print_a()
print_b()
```

module_b.swift

```swift
//  module_b.swift

func print_a() {
    print(a())
}

func b() -> String {
    return "b"
}
```

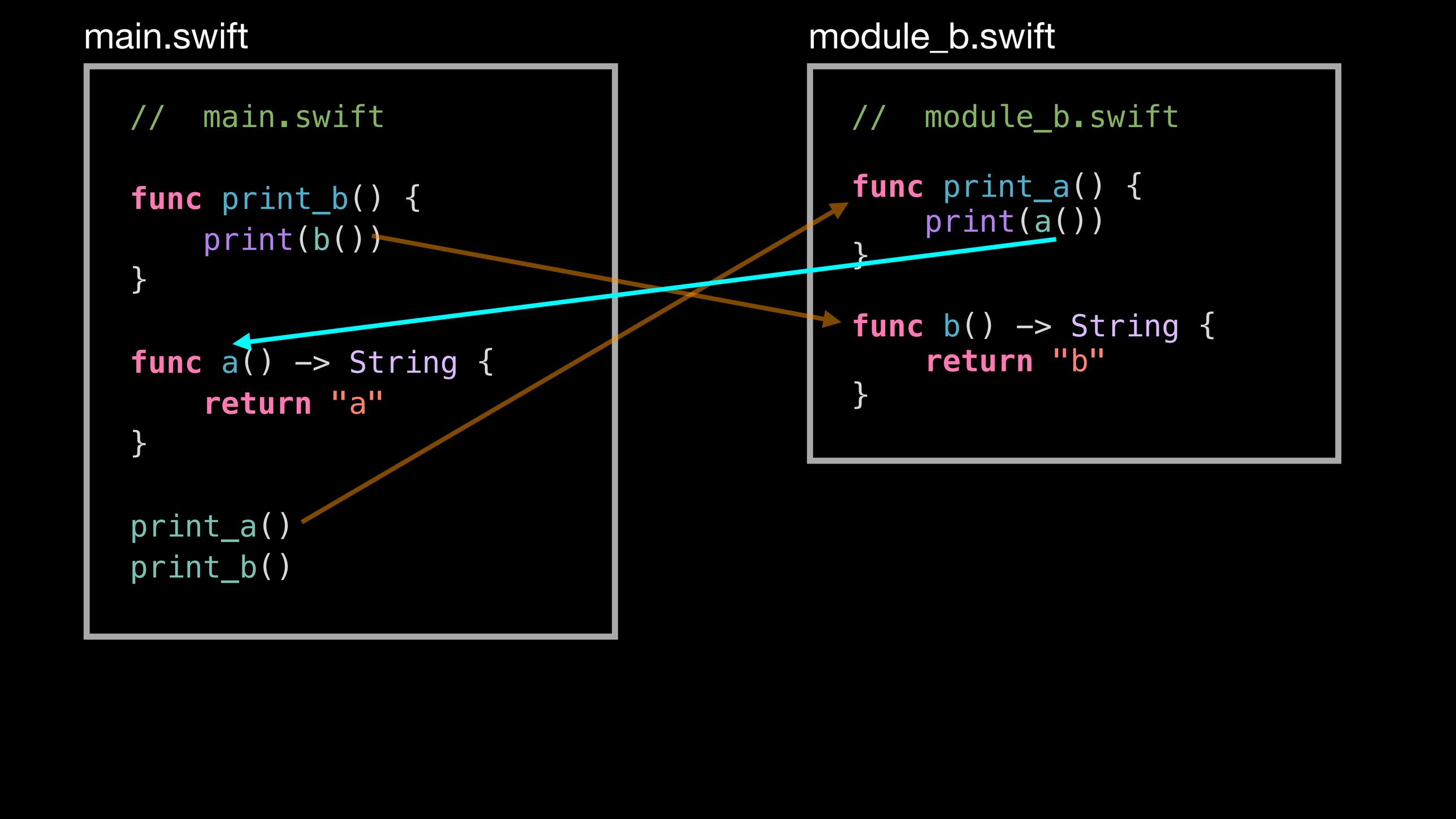


main.swift

```swift
//  main.swift

func print_b() {
    print(b())
}

func a() -> String {
    return "a"
}

print_a()
print_b()
```

module_b.swift

```swift
//  module_b.swift

func print_a() {
    print(a())
}

func b() -> String {
    return "b"
}
```

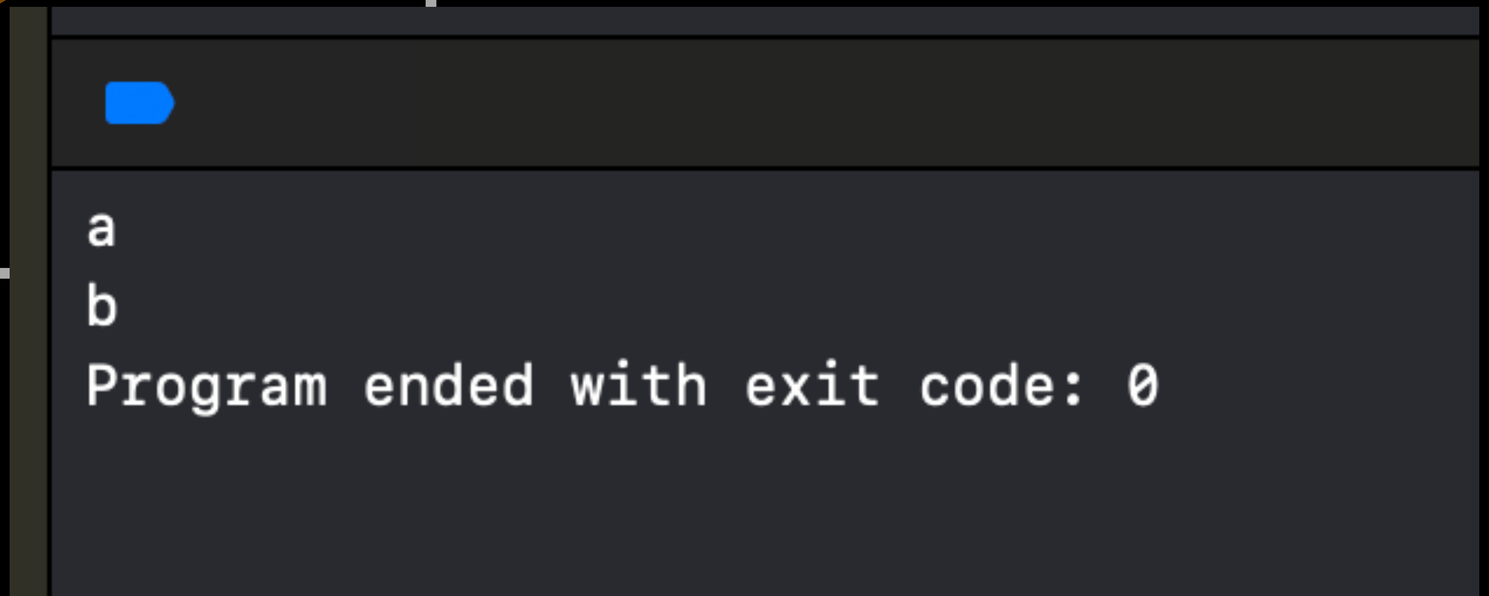


main.swift

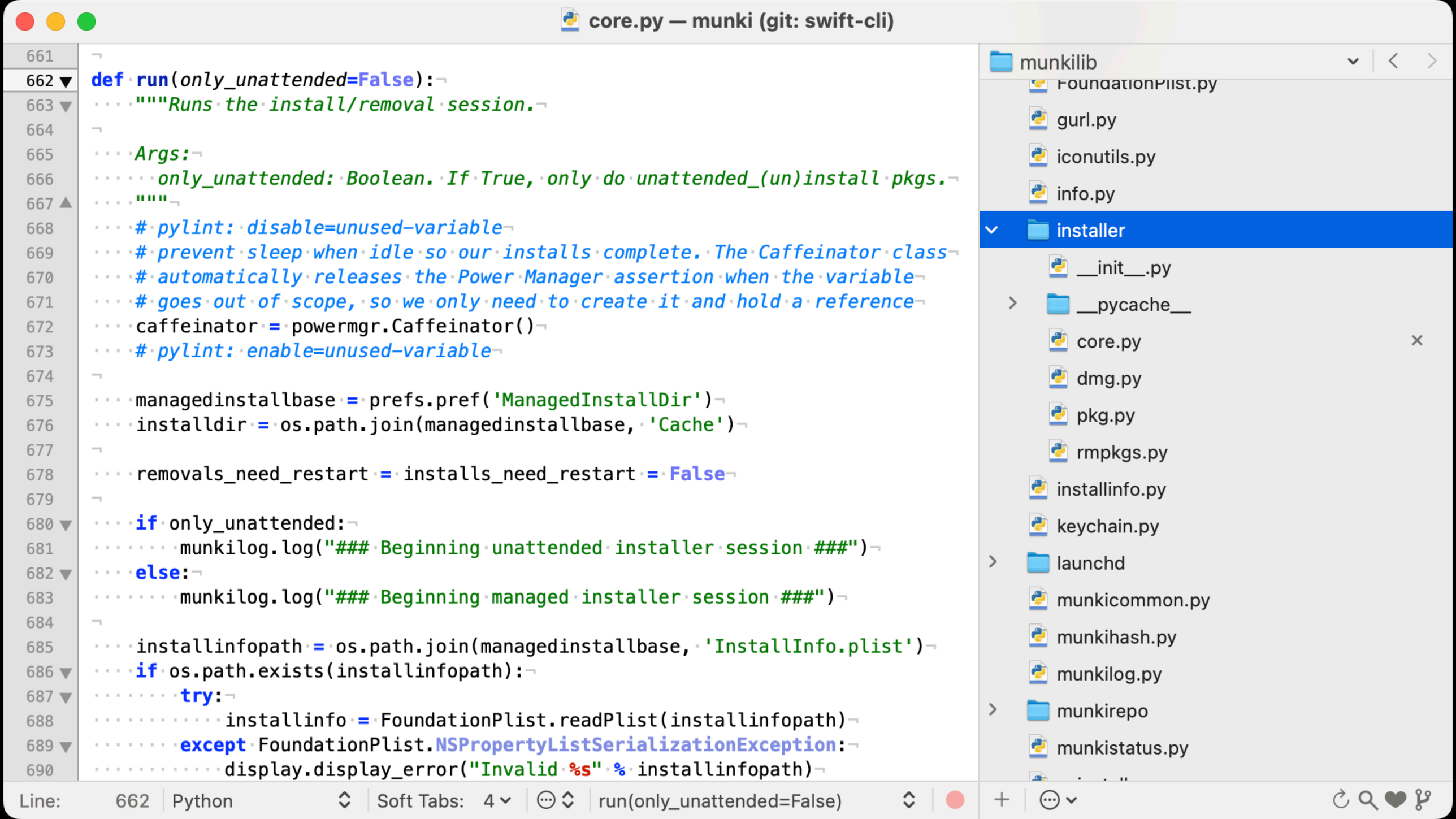
```swift
//  main.swift

func print_b() {
    print(b())
}

func a() -> String {
    return "a"
}

print_a()
print_b()
```

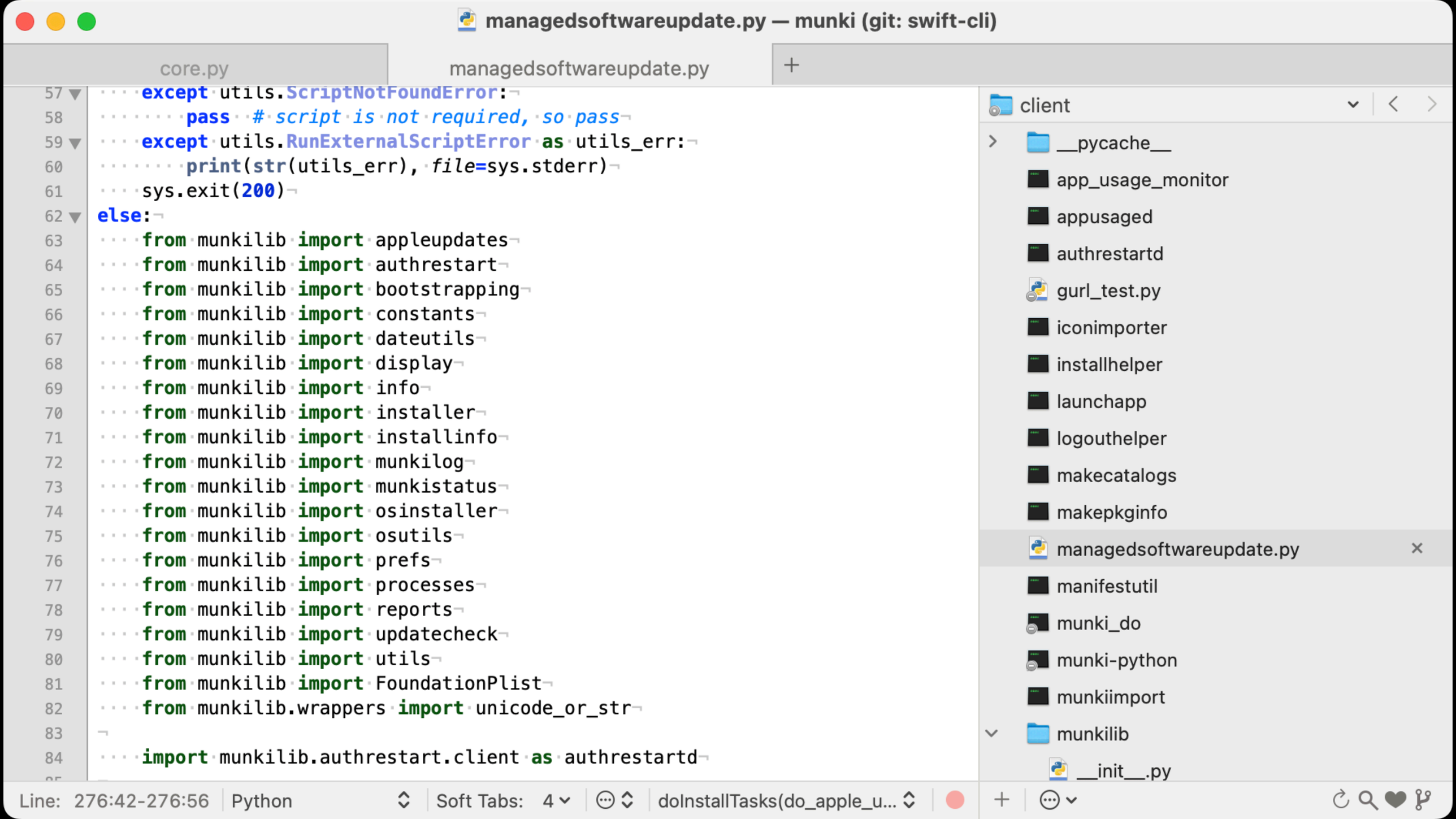
module_b.swift

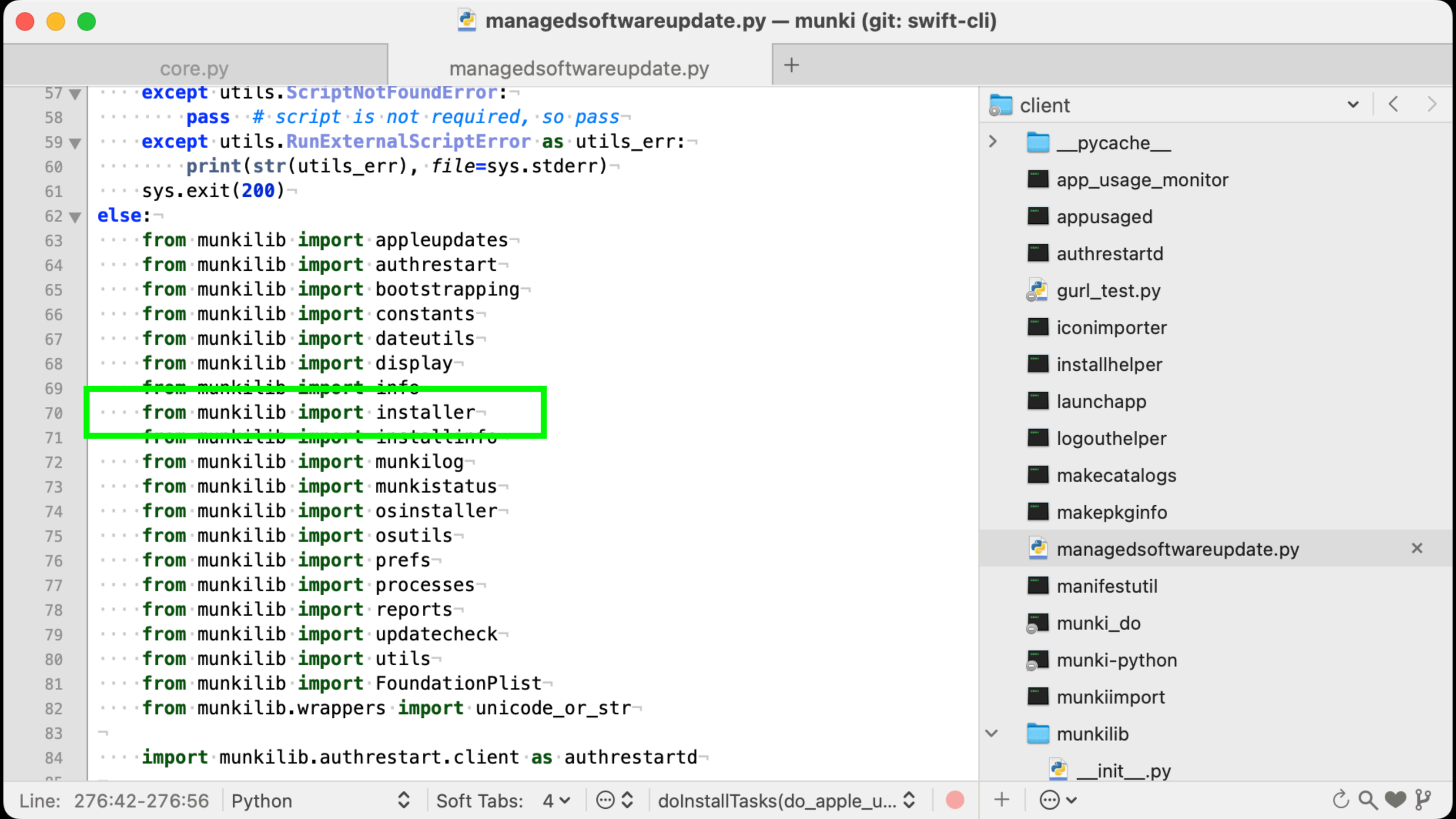
```swift
//  module_b.swift

func print_a() {
    print(a())
}

func b() -> String {
    return "b"
}
```

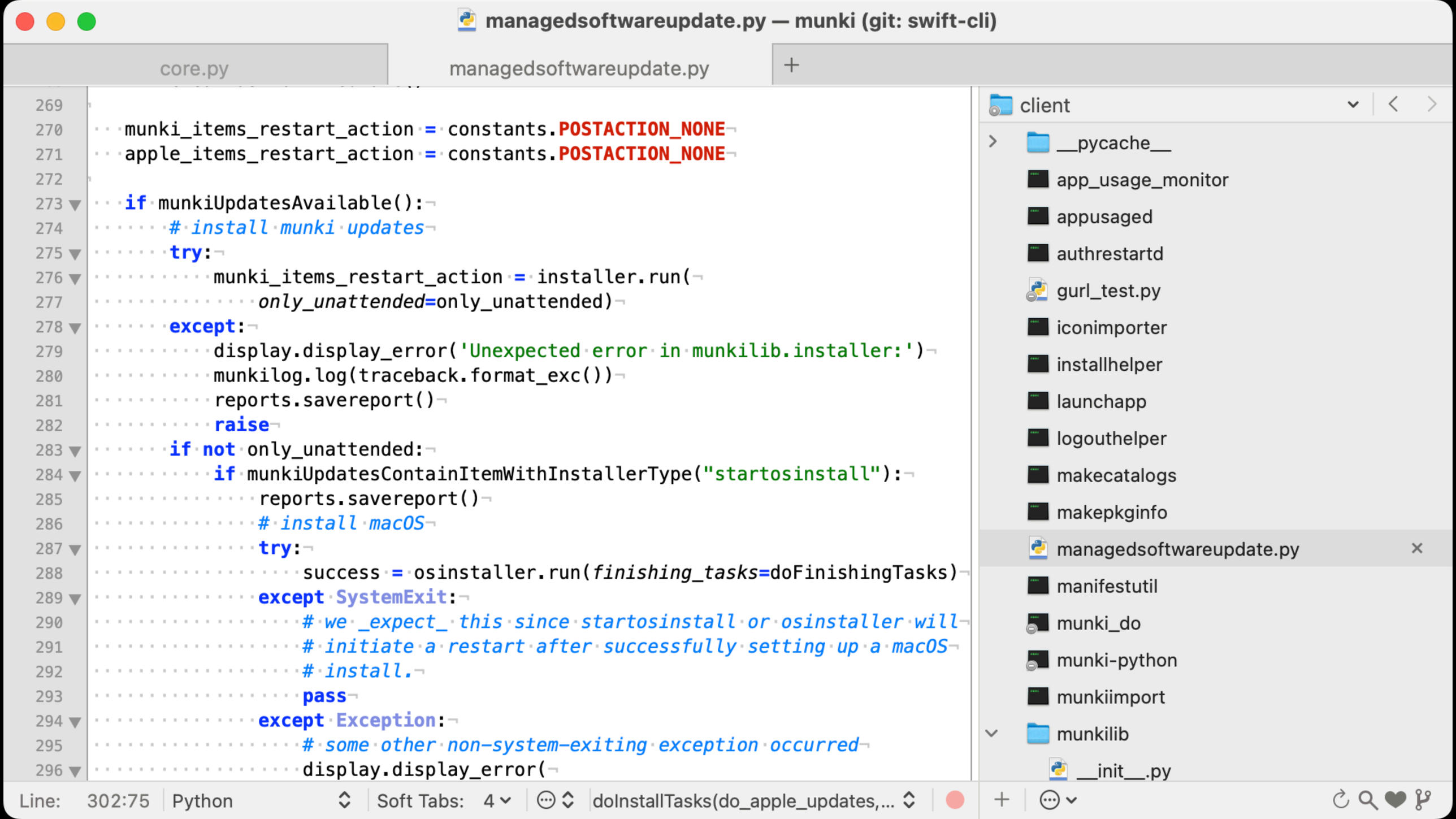


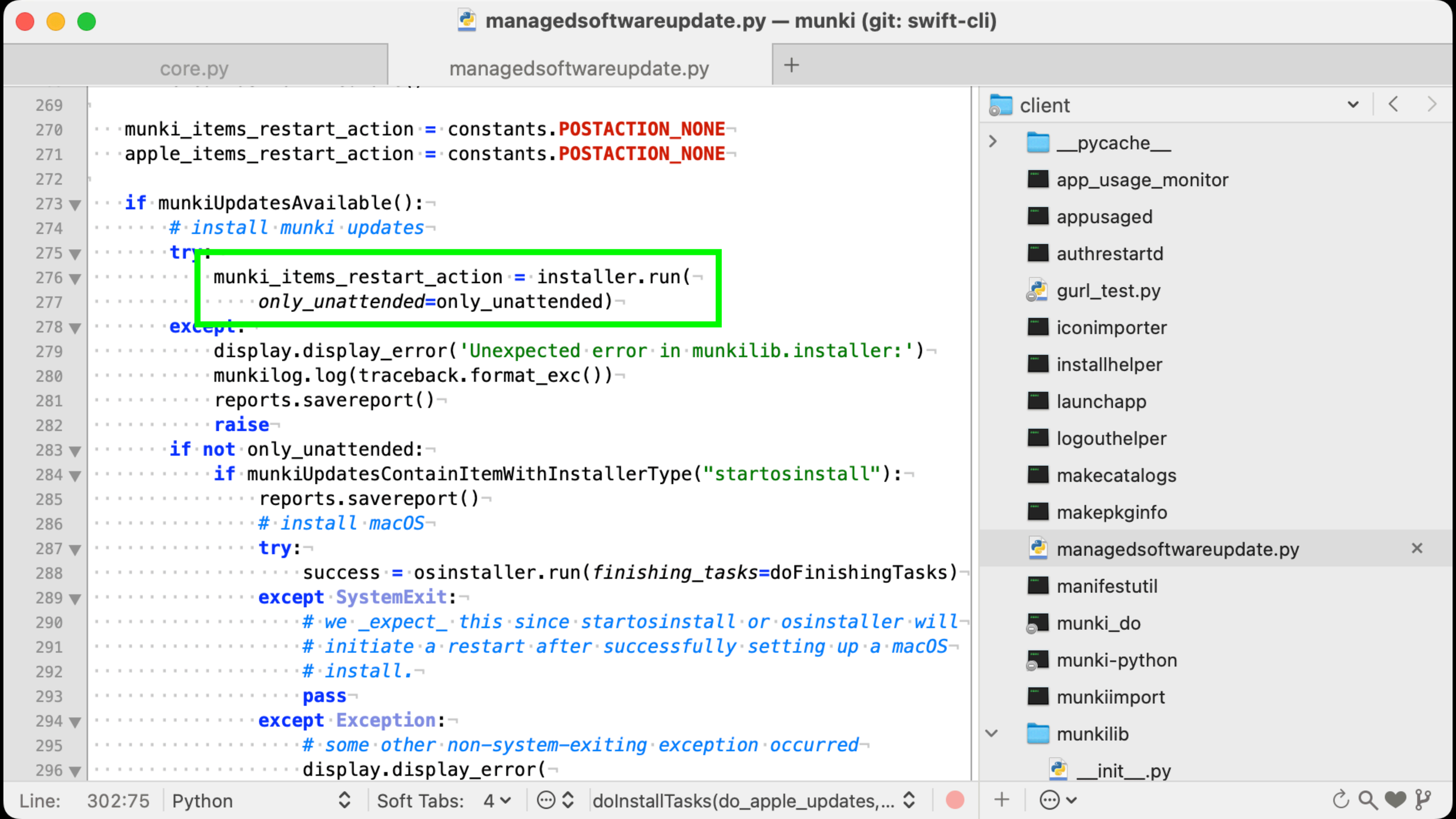
Naming

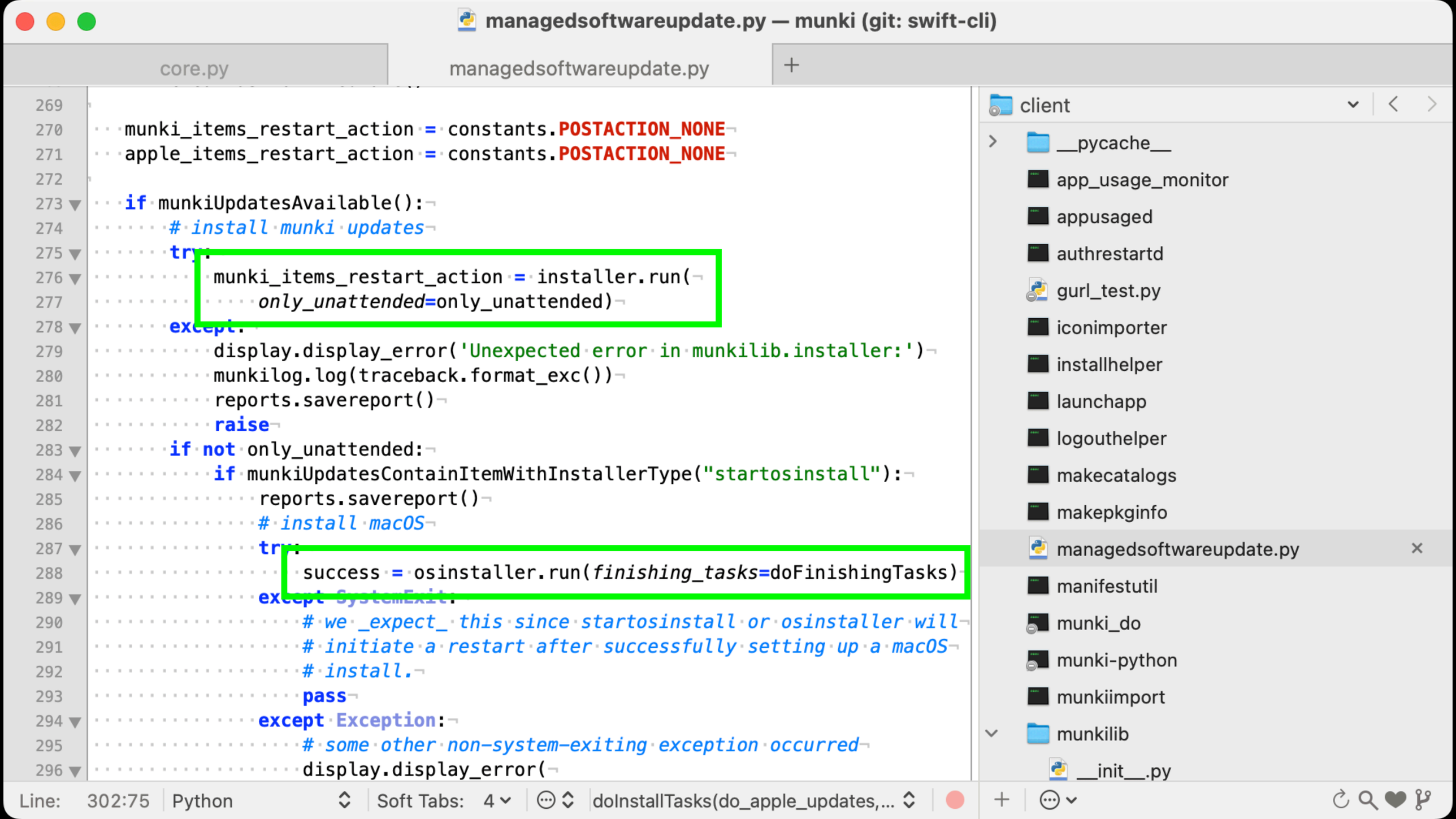

core.py — munki (git: swift-cli)
def run(only_unattended=False):
"""Runs the install/removal session.
Args:
only_unattended: Boolean. If True, only do unattended_(un)install pkgs.
"""
# pylint: disable=unused-variable
# prevent sleep when idle so our installs complete. The Caffeinator class
# automatically releases the Power Manager assertion when the variable
# goes out of scope, so we only need to create it and hold a reference
caffeinator = powermgr.Caffeinator()
# pylint: enable=unused-variable
managedinstallbase = prefs.pref('ManagedInstallDir')
installdir = os.path.join(managedinstallbase, 'Cache')
removals_need_restart = installs_need_restart = False
if only_unattended:
munkilog.log("### Beginning unattended installer session ###")
else:
munkilog.log("### Beginning managed installer session ###")
installinfopath = os.path.join(managedinstallbase, 'InstallInfo.plist')
if os.path.exists(installinfopath):
try:
installinfo = FoundationPlist.readPlist(installinfopath)
except FoundationPlist.NSPropertyListSerializationException:
display.display_error("Invalid %s" % installinfopath)
munkilib
FoundationPlist.py
gurl.py
iconutils.py
info.py
installer
__init__.py
__pycache__
core.py
dmg.py
pkg.py
rmpkgs.py
installinfo.py
keychain.py
launchd
munkicommon.py
munkihash.py
munkilog.py
munkirepo
munkistatus.py
Line: 662 Python Soft Tabs: 4 run(only_unattended=False)

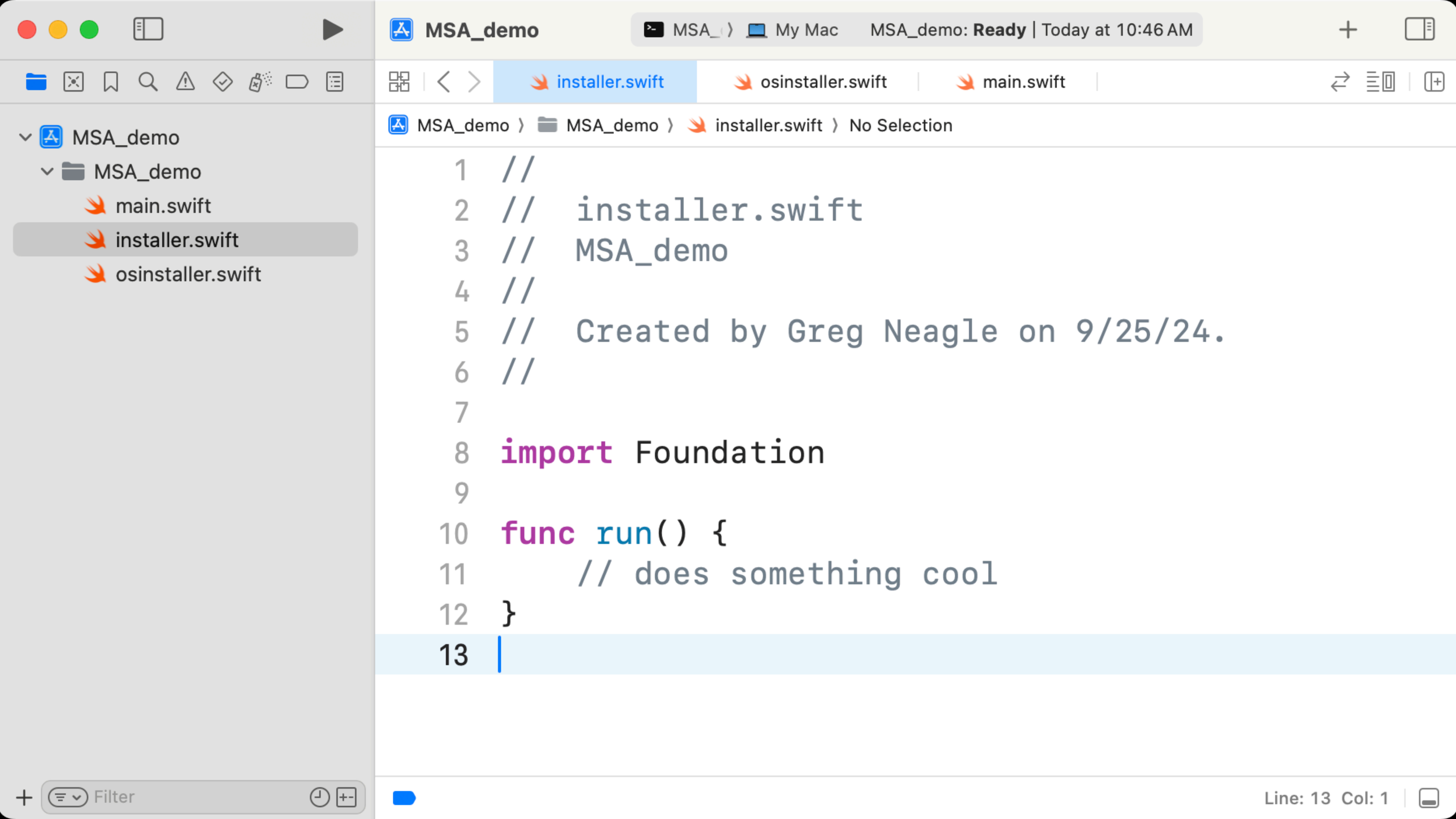

managedsoftwareupdate.py — munki (git: swift-cli)
core.py
managedsoftwareupdate.py
except utils.ScriptNotFoundError:
pass # script is not required, so pass
except utils.RunExternalScriptError as utils_err:
print(str(utils_err), file=sys.stderr)
sys.exit(200)
else:
from munkilib import appleupdates
from munkilib import authrestart
from munkilib import bootstrapping
from munkilib import constants
from munkilib import dateutils
from munkilib import display
from munkilib import info
from munkilib import installer
from munkilib import installinfo
from munkilib import munkilog
from munkilib import munkistatus
from munkilib import osinstaller
from munkilib import osutils
from munkilib import prefs
from munkilib import processes
from munkilib import reports
from munkilib import updatecheck
from munkilib import utils
from munkilib import FoundationPlist
from munkilib.wrappers import unicode_or_str
import munkilib.authrestart.client as authrestartd
client
__pycache__
app_usage_monitor
appusaged
authrestartd
gurl_test.py
iconimporter
installhelper
launchapp
logouthelper
makecatalogs
makepkginfo
managedsoftwareupdate.py
manifestutil
munki_do
munki-python
munkiimport
munkilib
__init__.py
Line: 276:42-276:56
Python
Soft Tabs: 4
doInstallTasks(do_apple_u...

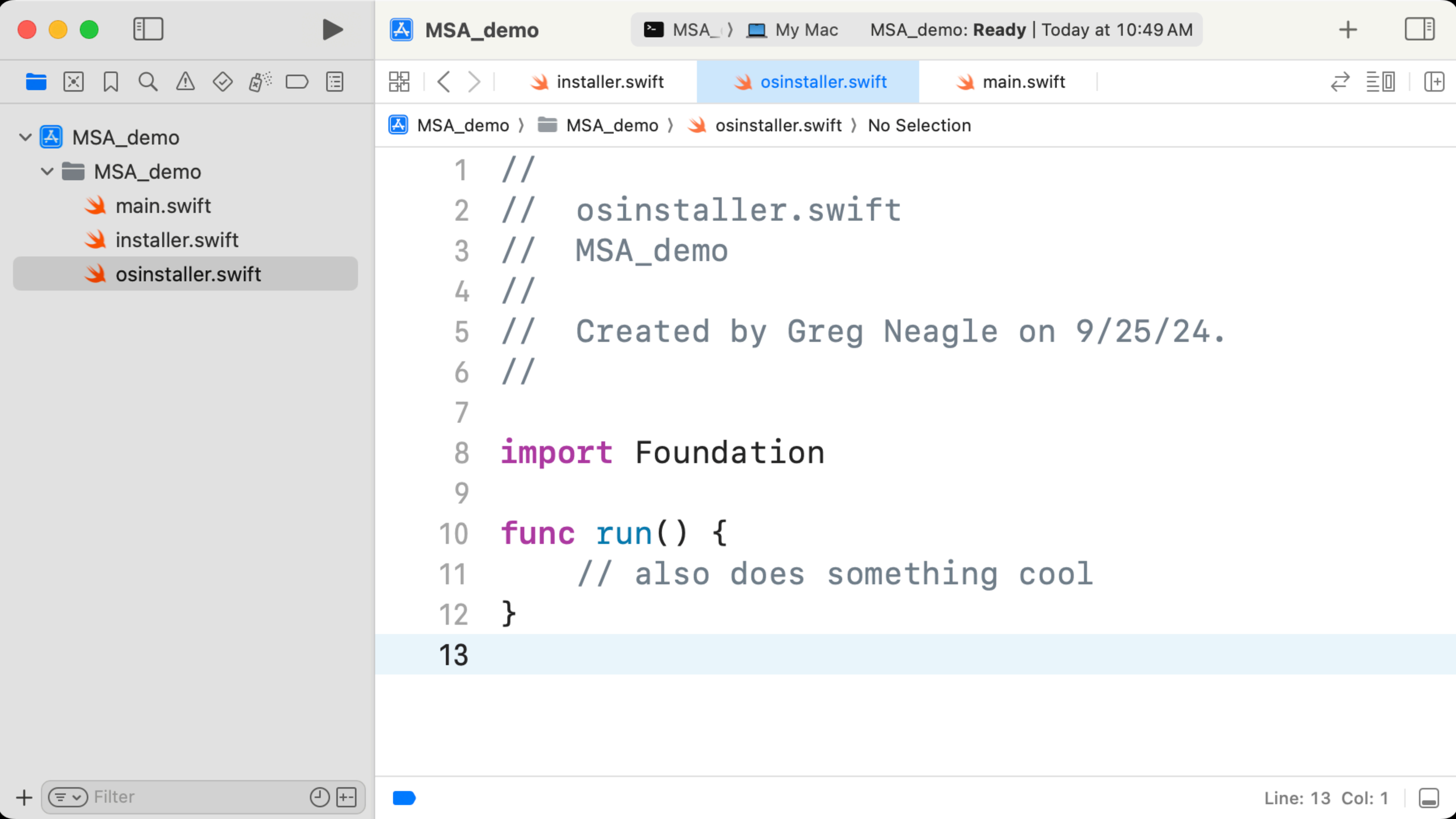

managedsoftwareupdate.py — munki (git: swift-cli)
core.py
managedsoftwareupdate.py
except utils.ScriptNotFoundError:
pass # script is not required, so pass
except utils.RunExternalScriptError as utils_err:
print(str(utils_err), file=sys.stderr)
sys.exit(200)
else:
from munkilib import appleupdates
from munkilib import authrestart
from munkilib import bootstrapping
from munkilib import constants
from munkilib import dateutils
from munkilib import display
from munkilib import installer
from munkilib import munkilog
from munkilib import munkistatus
from munkilib import osinstaller
from munkilib import osutils
from munkilib import prefs
from munkilib import processes
from munkilib import reports
from munkilib import updatecheck
from munkilib import utils
from munkilib import FoundationPlist
from munkilib.wrappers import unicode_or_str
import munkilib.authrestart.client as authrestartd
client
__pycache__
app_usage_monitor
appusaged
authrestartd
gurl_test.py
iconimporter
installhelper
launchapp
logouthelper
makecatalogs
makepkginfo
managedsoftwareupdate.py
manifestutil
munki_do
munki-python
munkiimport
munkilib
__init__.py
Line: 276:42-276:56
Python
Soft Tabs: 4
doInstallTasks(do_apple_u...

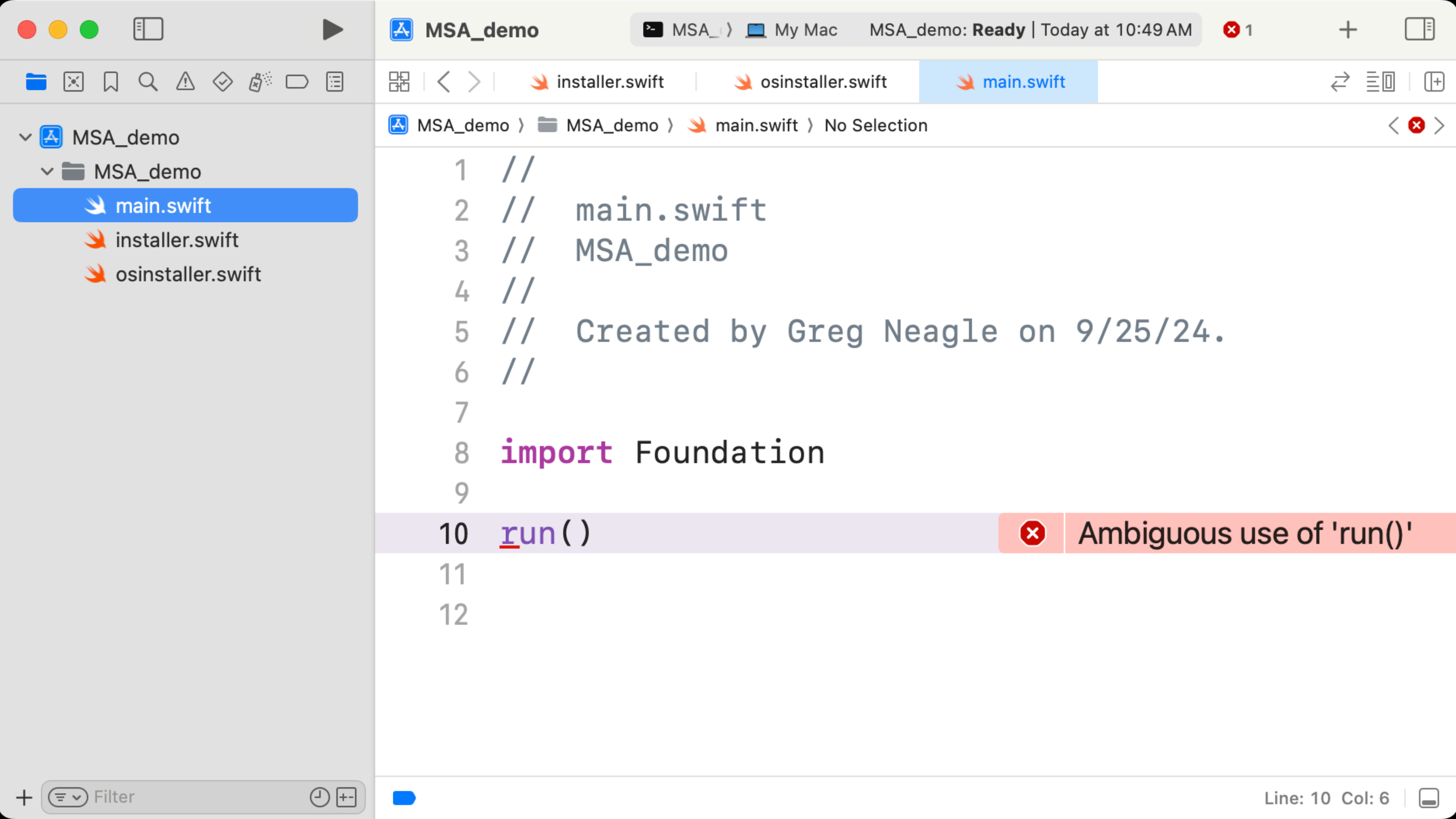

managedsoftwareupdate.py — munki (git: swift-cli)
core.py
managedsoftwareupdate.py
client
__pycache__
app_usage_monitor
appusaged
authrestartd
gurl_test.py
iconimporter
installhelper
launchapp
logouthelper
makecatalogs
makepkginfo
managedsoftwareupdate.py
manifestutil
munki_do
munki-python
munkiimport
munkilib
__init__.py
munki_items_restart_action = constants.POSTACTION_NONE
apple_items_restart_action = constants.POSTACTION_NONE
if munkiUpdatesAvailable():
# install munki updates
try:
munki_items_restart_action = installer.run(
only_unattended=only_unattended)
except:
display.display_error('Unexpected error in munkilib.installer:')
munkilog.log(traceback.format_exc())
reports.savereport()
raise
if not only_unattended:
if munkiUpdatesContainItemWithInstallerType("startosinstall"):
reports.savereport()
# install macOS
try:
success = osinstaller.run(finishing_tasks=doFinishingTasks)
except SystemExit:
we _expect_ this since startosinstall or osinstaller will
initiate a restart after successfully setting up a macOS
install.
pass
except Exception:
some other non-system-exiting exception occurred
display.display_error(
Line: 302:75
Python
Soft Tabs: 4
doInstallTasks(do_apple_updates,...

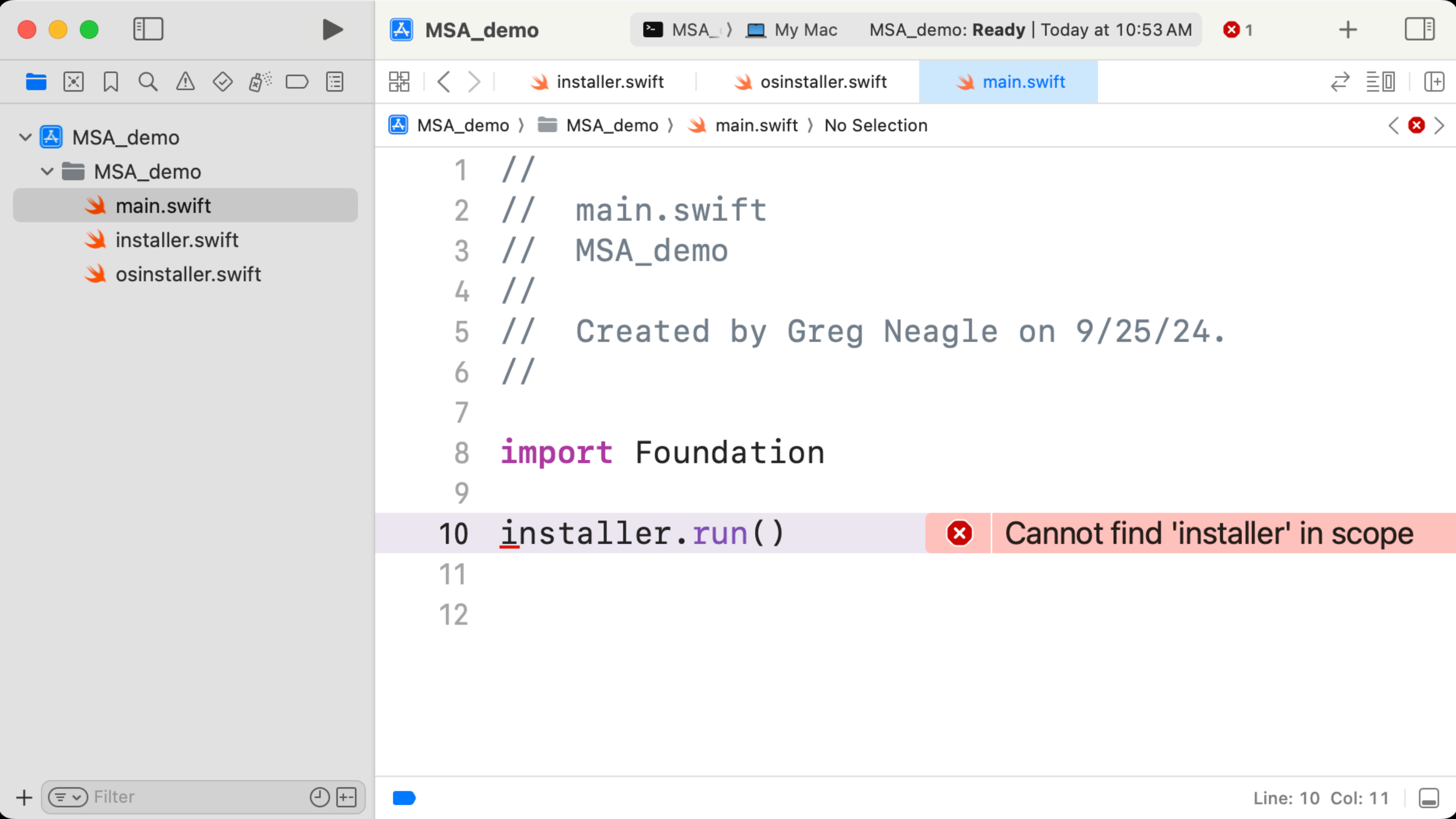

managedsoftwareupdate.py — munki (git: swift-cli)
core.py
managedsoftwareupdate.py
client
__pycache__
app_usage_monitor
appusaged
authrestartd
gurl_test.py
iconimporter
installhelper
launchapp
logouthelper
makecatalogs
makepkginfo
managedsoftwareupdate.py
manifestutil
munki_do
munki-python
munkiimport
munkilib
__init__.py
munki_items_restart_action = constants.POSTACTION_NONE
apple_items_restart_action = constants.POSTACTION_NONE
if munkiUpdatesAvailable():
# install munki updates
try:
munki_items_restart_action = installer.run(
only_unattended=only_unattended)
except:
display.display_error('Unexpected error in munkilib.installer:')
munkilog.log(traceback.format_exc())
reports.savereport()
raise
if not only_unattended:
if munkiUpdatesContainItemWithInstallerType("startosinstall"):
reports.savereport()
# install macOS
try:
success = osinstaller.run(finishing_tasks=doFinishingTasks)
except SystemExit:
we _expect_ this since startosinstall or osinstaller will
initiate a restart after successfully setting up a macOS
install.
pass
except Exception:
some other non-system-exiting exception occurred
display.display_error(
Line: 302:75
Python
Soft Tabs: 4
doInstallTasks(do_apple_updates,...


managedsoftwareupdate.py — munki (git: swift-cli)
core.py
managedsoftwareupdate.py
client
__pycache__
app_usage_monitor
appusaged
authrestartd
gurl_test.py
iconimporter
installhelper
launchapp
logouthelper
makecatalogs
makepkginfo
managedsoftwareupdate.py
manifestutil
munki_do
munki-python
munkiimport
munkilib
__init__.py
munki_items_restart_action = constants.POSTACTION_NONE
apple_items_restart_action = constants.POSTACTION_NONE
if munkiUpdatesAvailable():
# install munki updates
try:
munki_items_restart_action = installer.run(
only_unattended=only_unattended)
except:
display.display_error('Unexpected error in munkilib.installer:')
munkilog.log(traceback.format_exc())
reports.savereport()
raise
if not only_unattended:
if munkiUpdatesContainItemWithInstallerType("startosinstall"):
reports.savereport()
# install macOS
try:
success = osinstaller.run(finishing_tasks=doFinishingTasks)
except SystemExit:
we _expect_ this since startosinstall or osinstaller will
initiate a restart after successfully setting up a macOS
install.
pass
except Exception:
some other non-system-exiting exception occurred
display.display_error(
Line: 302:75
Python
Soft Tabs: 4
doInstallTasks(do_apple_updates,...


MSA_demo
MSA_ › My Mac
MSA_demo: Ready | Today at 10:46 AM
installer.swift
osinstaller.swift
main.swift
MSA_demo › MSA_demo › installer.swift › No Selection
MSA_demo
MSA_demo
main.swift
installer.swift
osinstaller.swift
Filter
1 //
2 // installer.swift
3 // MSA_demo
4 //
5 // Created by Greg Neagle on 9/25/24.
6 //
7
8 import Foundation
9
10 func run() {
11 // does something cool
12 }
13
Line: 13 Col: 1


MSA_demo
MSA_ › My Mac
MSA_demo: Ready | Today at 10:49 AM
installer.swift
osinstaller.swift
main.swift
MSA_demo › MSA_demo › osinstaller.swift › No Selection
MSA_demo
MSA_demo
main.swift
installer.swift
osinstaller.swift
Filter
1 //
2 // osinstaller.swift
3 // MSA_demo
4 //
5 // Created by Greg Neagle on 9/25/24.
6 //
7
8 import Foundation
9
10 func run() {
11 // also does something cool
12 }
13
Line: 13 Col: 1


MSA_demo
MSA_ › My Mac
MSA_demo: Ready | Today at 10:49 AM
1
installer.swift
osinstaller.swift
main.swift
MSA_demo › MSA_demo › main.swift › No Selection
MSA_demo
MSA_demo
main.swift
installer.swift
osinstaller.swift
1 //
2 // main.swift
3 // MSA_demo
4 //
5 // Created by Greg Neagle on 9/25/24.
6 //
7
8 import Foundation
9
10 run()
11
12
Ambiguous use of 'run()'
Filter
Line: 10 Col: 6


MSA_demo
MSA_ › My Mac
MSA_demo: Ready | Today at 10:53 AM
1
installer.swift
osinstaller.swift
main.swift
MSA_demo › MSA_demo › main.swift › No Selection
MSA_demo
MSA_demo
main.swift
installer.swift
osinstaller.swift
1 //
2 // main.swift
3 // MSA_demo
4 //
5 // Created by Greg Neagle on 9/25/24.
6 //
7
8 import Foundation
9
10 installer.run()
Cannot find 'installer' in scope
11
12
Filter
Line: 10 Col: 11

managedsoftwareupdate.py | core.py

```
            'display_name': display_name,
            'name': item['name'],
            'status': retcode,
            'time': NSDate.new(),
            'unattended': only_unattended,
        }
        reports.report['RemovalResults'].append(removal_result)

    return (restart_flag, skipped_removals)


def run(only_unattended=False):
    """Runs the install/removal session.

    Args:
      only_unattended: Boolean. If True, only do unattended_(un)
    """
    # pylint: disable=unused-variable
    # prevent sleep when idle so our installs complete. The Caff
    # automatically releases the Power Manager assertion when th
    # goes out of scope, so we only need to create it and hold a
    caffeinator = powermgr.Caffeinator()
    # pylint: enable=unused-variable

    managedinstallbase = prefs.pref('ManagedInstallDir')
    installdir = os.path.join(managedinstallbase, 'Cache')

    removals_need_restart = installs_need_restart = False

    if only_unattended:
        munkilog.log("### Beginning unattended installer session
    else:
        munkilog.log("### Beginning managed installer session ##

    installinfopath = os.path.join(managedinstallbase, 'InstallI
    if os.path.exists(installinfopath):
        try:
            installinfo = FoundationPlist.readPlist(installinfop
        except FoundationPlist.NSPropertyListSerializationExcept
            display.display_error("Invalid %s" % installinfopath
            return -1

        if prefs.pref('SuppressStopButtonOnInstall'):
            munkistatus.hideStopButton()
```

Xcode — munki — swift-cli — Any Mac (arm — Build Succeeded | 9/20/24 at 11:42 AM

installer.swift

munkilog.swift | makecatalogs.swift | removepackages.swift | repou

munki ⟩ shared ⟩ installer ⟩ installer.swift ⟩ No Selection

- munkilog.swift
- reports.swift
- munkihash.swift
- osinstaller.swift
- version.swift
- errors.swift
- sqlite3.swift
- launchd.swift
- appusage.swift
- xattr.swift
- processes.swift
- munkistatus.swift
- powermanager.swift
- stoprequested.swift
- info.swift
- appinventory.swift
- facts.swift
- Predicates.m
- installinfo.swift
- bootstrapping.swift
- UNIXProcessInfo.swift
- SignalHandler.swift
- appleupdates.swift
- managedsoftwareupdate
- munkitester
- makecatalogs
- makepkginfo
- munkiimport
- removepackages
- app_usage_monitor
- appusaged
- launchapp
- iconimporter

```
578
579 /// Runs the install/removal session.
580 ///
581 /// Args:
582 /// only_unattended: Boolean. If True, only do unattended_(un)install pkgs.
583 func doInstallsAndRemovals(onlyUnattended: Bool = false) async ->
        PostAction {
584     var removalsNeedRestart = false
585     var installsNeedRestart = false
586
587     if onlyUnattended {
588         munkiLog("### Beginning unattended installer session ###")
589     } else {
590         munkiLog("### Beginning managed installer session ###")
591     }
592
593     // no sleep assertion
594     let caffeinator = Caffeinator(
595         reason: "managedsoftwareupdate is installing software")
596
597     let installInfoPath = managedInstallsDir(subpath:
            "InstallInfo.plist")
598     if pathExists(installInfoPath),
599        let installInfo = try? readPlist(fromFile: installInfoPath) as?
               PlistDict
600     {
601         var updatedInstallInfo = installInfo
602         if pref("SuppressStopButtonOnInstall") as? Bool ?? false {
603             munkiStatusHideStopButton()
604         }
605         // process removals
606         if let removals = installInfo["removals"] as? [PlistDict] {
607             // filter list to items that need to be removed
608             let removalList = removals.filter {
609                 $0["installed"] as? Bool ?? false
610             }
611             Report.shared.record(removalList, to: "ItemsToRemove")
```

managedsoftwareupdate.py | core.py

```
            'display_name': display_name,
            'name': item['name'],
            'status': retcode,
            'time': NSDate.new(),
            'unattended': only_unattended,
        }
        reports.report['RemovalResults'].append(removal_result)

    return (restart_flag, skipped_removals)


def run(only_unattended=False):
    """Runs the install/removal session.

    Args:
      only_unattended: Boolean. If True, only do unattended_(un)
    """
    # pylint: disable=unused-variable
    # prevent sleep when idle so our installs complete. The Caff
    # automatically releases the Power Manager assertion when th
    # goes out of scope, so we only need to create it and hold a
    caffeinator = powermgr.Caffeinator()
    # pylint: enable=unused-variable

    managedinstallbase = prefs.pref('ManagedInstallDir')
    installdir = os.path.join(managedinstallbase, 'Cache')

    removals_need_restart = installs_need_restart = False

    if only_unattended:
        munkilog.log("### Beginning unattended installer session
    else:
        munkilog.log("### Beginning managed installer session ##

    installinfopath = os.path.join(managedinstallbase, 'InstallI
    if os.path.exists(installinfopath):
        try:
            installinfo = FoundationPlist.readPlist(installinfop
        except FoundationPlist.NSPropertyListSerializationExcept
            display.display_error("Invalid %s" % installinfopath
            return -1

        if prefs.pref('SuppressStopButtonOnInstall'):
            munkistatus.hideStopButton()
```

Any Mac (arm | Build Succeeded | 9/20/24 at 11:42 AM

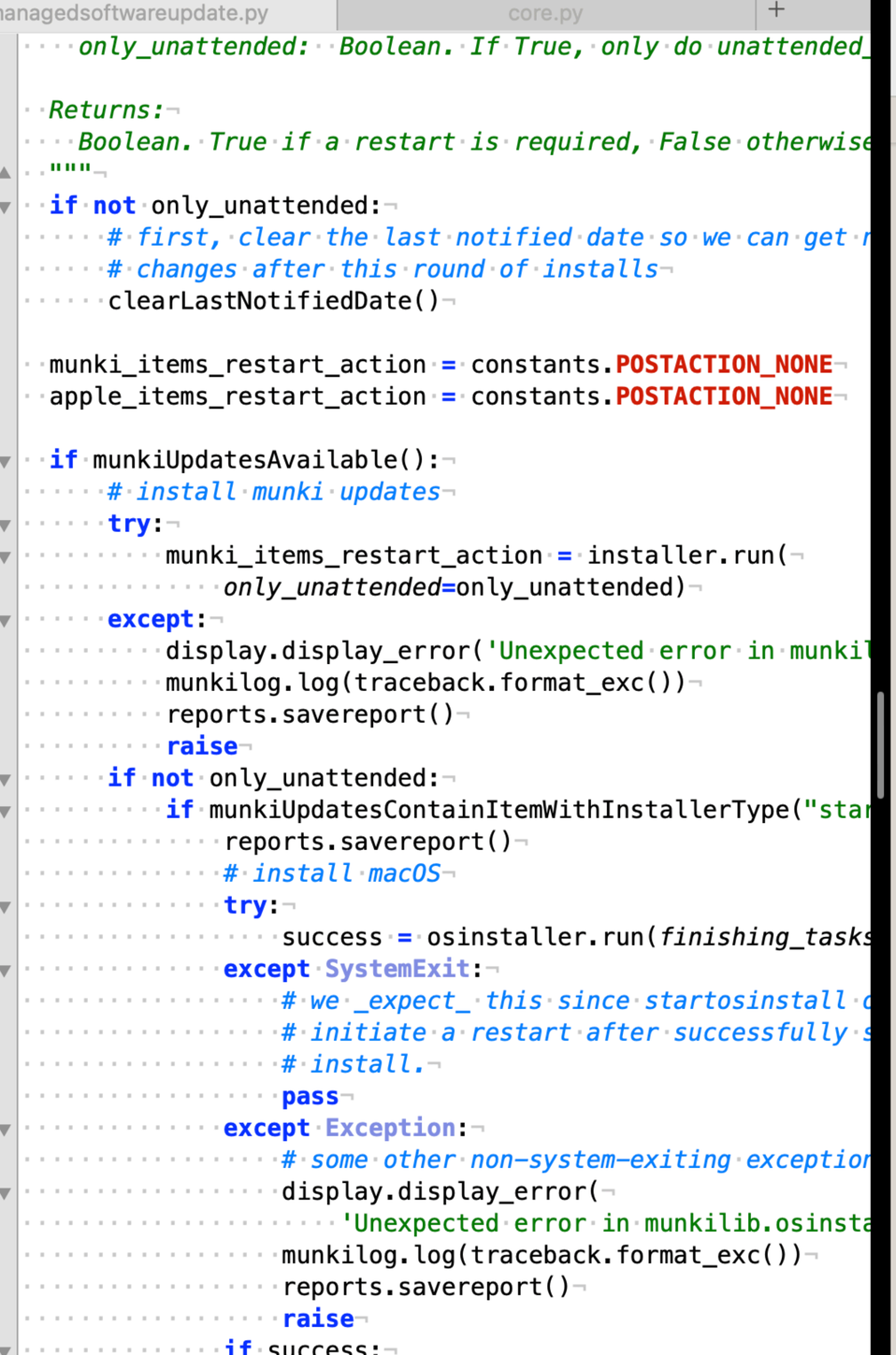
installer.swift

munkilog.swift | makecatalogs.swift | removepackages.swift | repou

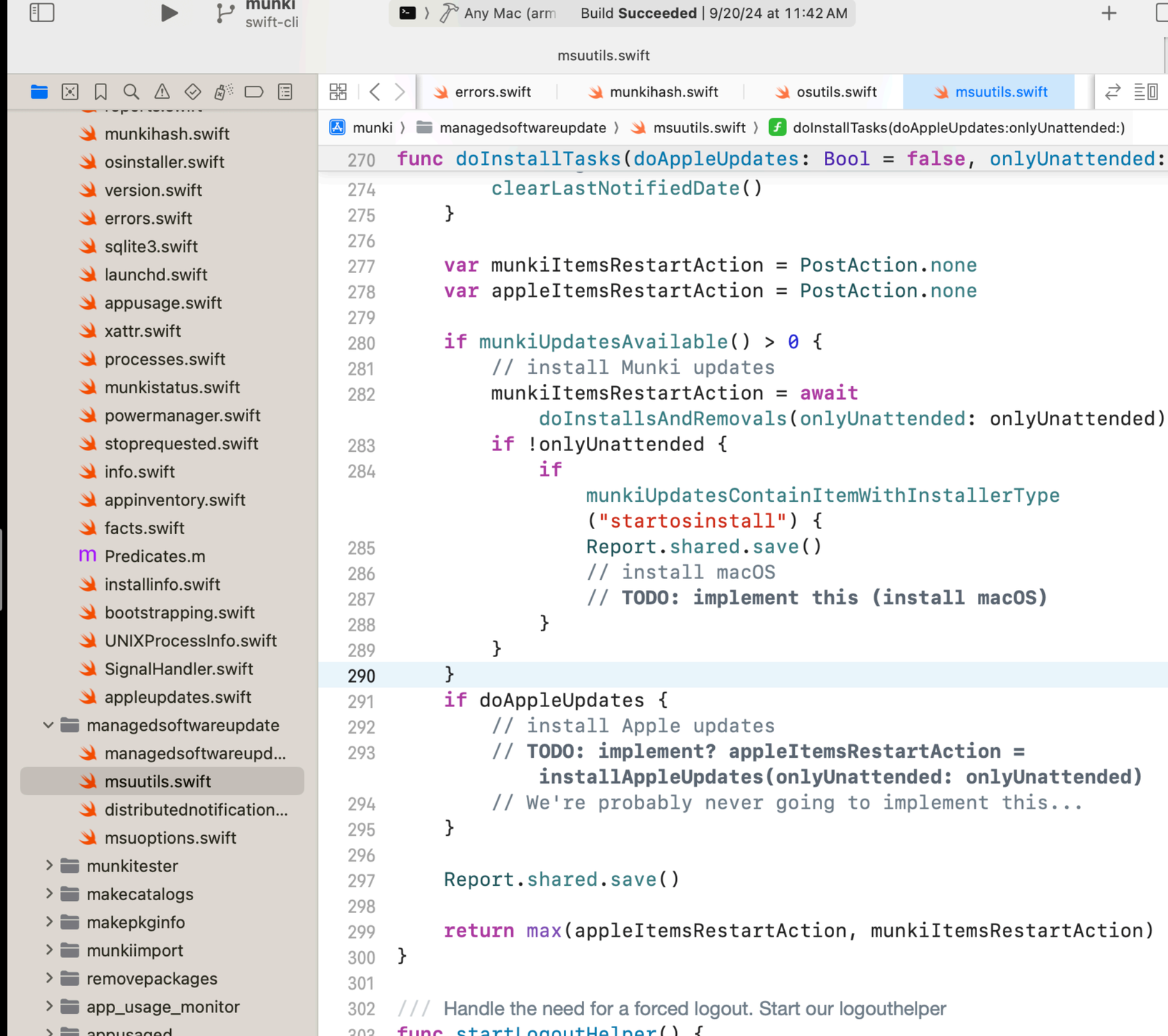
munki ⟩ shared ⟩ installer ⟩ installer.swift ⟩ No Selection

- reports.swift
- munkihash.swift
- osinstaller.swift
- version.swift
- errors.swift
- sqlite3.swift
- launchd.swift
- appusage.swift
- xattr.swift
- processes.swift
- munkistatus.swift
- powermanager.swift
- stoprequested.swift
- info.swift
- appinventory.swift
- facts.swift
- Predicates.m
- installinfo.swift
- bootstrapping.swift
- UNIXProcessInfo.swift
- SignalHandler.swift
- appleupdates.swift
- managedsoftwareupdate
- munkitester
- makecatalogs
- makepkginfo
- munkiimport
- removepackages
- app_usage_monitor
- appusaged
- launchapp
- iconimporter

```
/// Runs the install/removal session.
///
/// Args:
/// only_unattended: Boolean. If True, only do unattended_(un)install pkgs.
func doInstallsAndRemovals(onlyUnattended: Bool = false) async ->
    var removalsNeedRestart = false
    var installsNeedRestart = false

    if onlyUnattended {
        munkiLog("### Beginning unattended installer session ###")
    } else {
        munkiLog("### Beginning managed installer session ###")
    }

    // no sleep assertion
    let caffeinator = Caffeinator(
        reason: "managedsoftwareupdate is installing software")

    let installInfoPath = managedInstallsDir(subpath:
        "InstallInfo.plist")
    if pathExists(installInfoPath),
       let installInfo = try? readPlist(fromFile: installInfoPath) as?
           PlistDict
    {
        var updatedInstallInfo = installInfo
        if pref("SuppressStopButtonOnInstall") as? Bool ?? false {
            munkiStatusHideStopButton()
        }
        // process removals
        if let removals = installInfo["removals"] as? [PlistDict] {
            // filter list to items that need to be removed
            let removalList = removals.filter {
                $0["installed"] as? Bool ?? false
            }
            Report.shared.record(removalList, to: "ItemsToRemove")
```



```
    only_unattended: Boolean. If True, only do unattended_

Returns:
    Boolean. True if a restart is required, False otherwise
"""
if not only_unattended:
    # first, clear the last notified date so we can get r
    # changes after this round of installs
    clearLastNotifiedDate()

munki_items_restart_action = constants.POSTACTION_NONE
apple_items_restart_action = constants.POSTACTION_NONE

if munkiUpdatesAvailable():
    # install munki updates
    try:
        munki_items_restart_action = installer.run(
            only_unattended=only_unattended)
    except:
        display.display_error('Unexpected error in munki
        munkilog.log(traceback.format_exc())
        reports.savereport()
        raise
    if not only_unattended:
        if munkiUpdatesContainItemWithInstallerType("star
            reports.savereport()
            # install macOS
            try:
                success = osinstaller.run(finishing_tasks
            except SystemExit:
                # we _expect_ this since startosinstall c
                # initiate a restart after successfully s
                # install.
                pass
            except Exception:
                # some other non-system-exiting exceptior
                display.display_error(
                    'Unexpected error in munkilib.osinsta
                munkilog.log(traceback.format_exc())
                reports.savereport()
                raise
            if success:
```



```
munkihash.swift
osinstaller.swift
version.swift
errors.swift
sqlite3.swift
launchd.swift
appusage.swift
xattr.swift
processes.swift
munkistatus.swift
powermanager.swift
stoprequested.swift
info.swift
appinventory.swift
facts.swift
Predicates.m
installinfo.swift
bootstrapping.swift
UNIXProcessInfo.swift
SignalHandler.swift
appleupdates.swift
managedsoftwareupdate
managedsoftwareupd...
msuutils.swift
distributednotification...
msuoptions.swift
munkitester
makecatalogs
makepkginfo
munkiimport
removepackages
app_usage_monitor
```

munki swift-cli

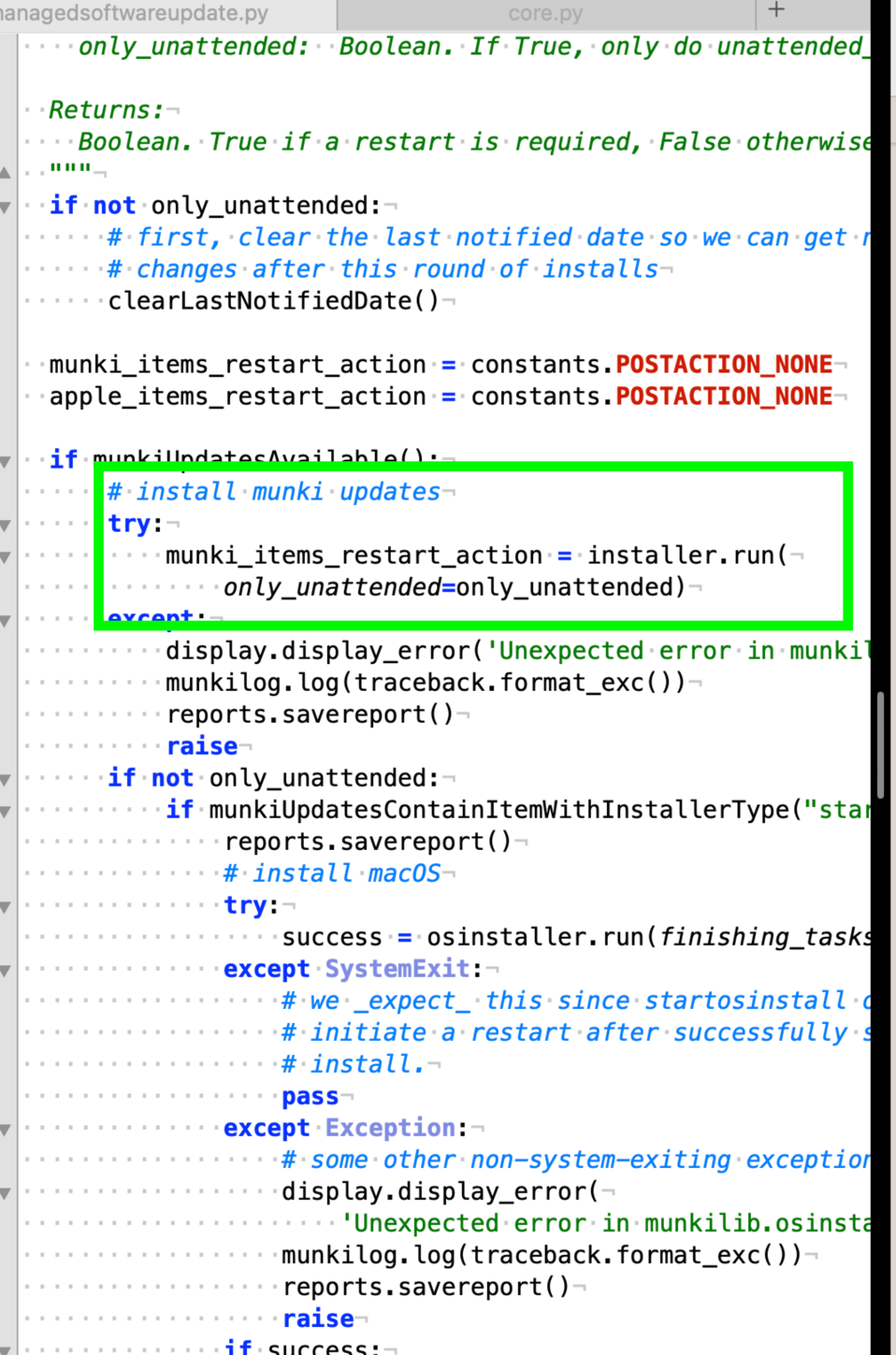
Any Mac (arm Build Succeeded | 9/20/24 at 11:42 AM

msuutils.swift

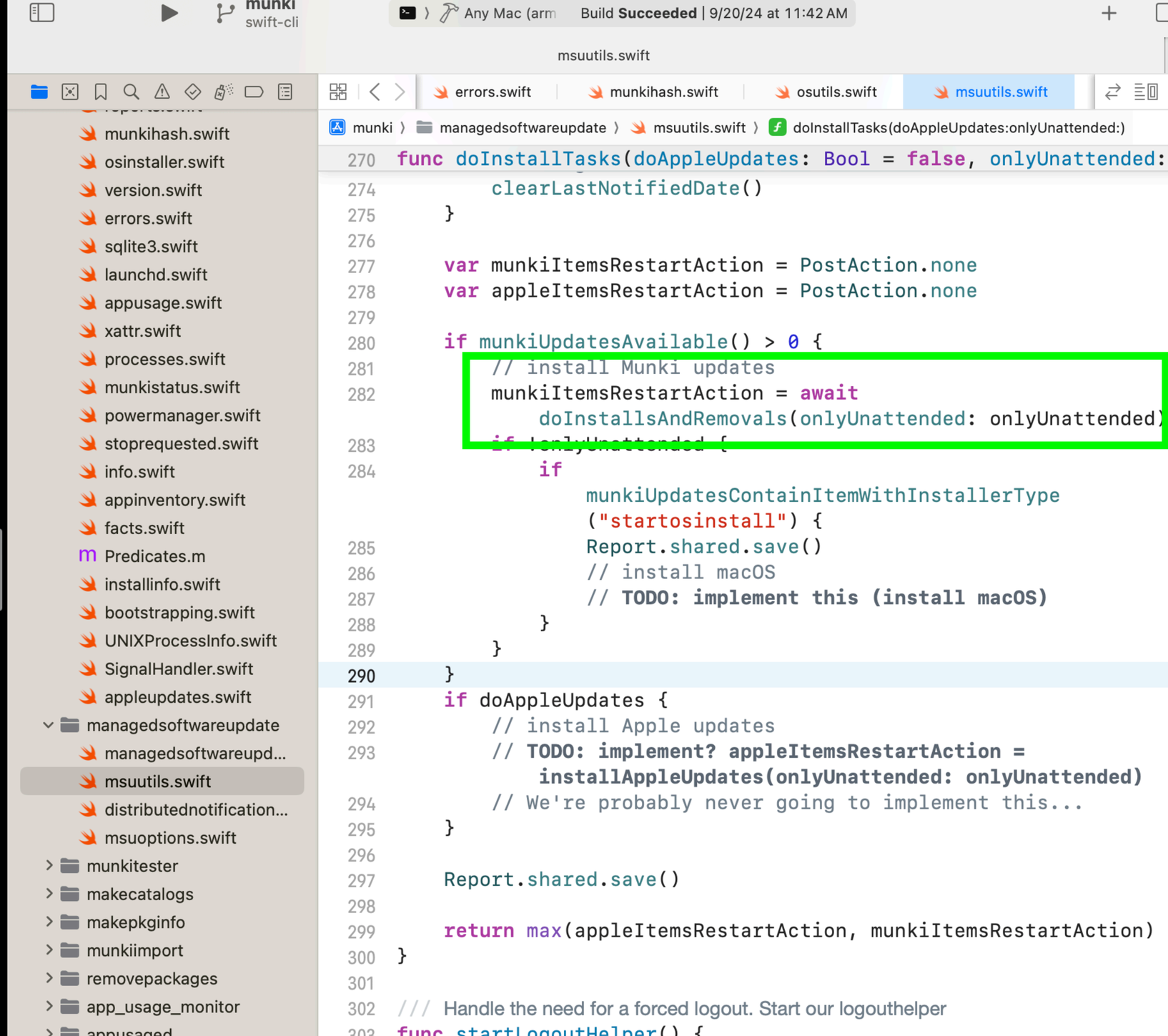
errors.swift | munkihash.swift | osutils.swift | msuutils.swift

munki ⟩ managedsoftwareupdate ⟩ msuutils.swift ⟩ doInstallTasks(doAppleUpdates:onlyUnattended:)

```
270 func doInstallTasks(doAppleUpdates: Bool = false, onlyUnattended:
274         clearLastNotifiedDate()
275     }
276
277     var munkiItemsRestartAction = PostAction.none
278     var appleItemsRestartAction = PostAction.none
279
280     if munkiUpdatesAvailable() > 0 {
281         // install Munki updates
282         munkiItemsRestartAction = await
                doInstallsAndRemovals(onlyUnattended: onlyUnattended)
283         if !onlyUnattended {
284             if
                    munkiUpdatesContainItemWithInstallerType
                    ("startosinstall") {
285                 Report.shared.save()
286                 // install macOS
287                 // TODO: implement this (install macOS)
288             }
289         }
290     }
291     if doAppleUpdates {
292         // install Apple updates
293         // TODO: implement? appleItemsRestartAction =
                installAppleUpdates(onlyUnattended: onlyUnattended)
294         // We're probably never going to implement this...
295     }
296
297     Report.shared.save()
298
299     return max(appleItemsRestartAction, munkiItemsRestartAction)
300 }
301
302 /// Handle the need for a forced logout. Start our logouthelper
```



```
    only_unattended: Boolean. If True, only do unattended_

Returns:
    Boolean. True if a restart is required, False otherwise
"""
if not only_unattended:
    # first, clear the last notified date so we can get r
    # changes after this round of installs
    clearLastNotifiedDate()

munki_items_restart_action = constants.POSTACTION_NONE
apple_items_restart_action = constants.POSTACTION_NONE

if munkiUpdatesAvailable():
    # install munki updates
    try:
        munki_items_restart_action = installer.run(
            only_unattended=only_unattended)
    except:
        display.display_error('Unexpected error in munki
        munkilog.log(traceback.format_exc())
        reports.savereport()
        raise
    if not only_unattended:
        if munkiUpdatesContainItemWithInstallerType("star
            reports.savereport()
            # install macOS
            try:
                success = osinstaller.run(finishing_tasks
            except SystemExit:
                # we _expect_ this since startosinstall c
                # initiate a restart after successfully s
                # install.
                pass
            except Exception:
                # some other non-system-exiting exceptior
                display.display_error(
                    'Unexpected error in munkilib.osinsta
                munkilog.log(traceback.format_exc())
                reports.savereport()
                raise
            if success:
```

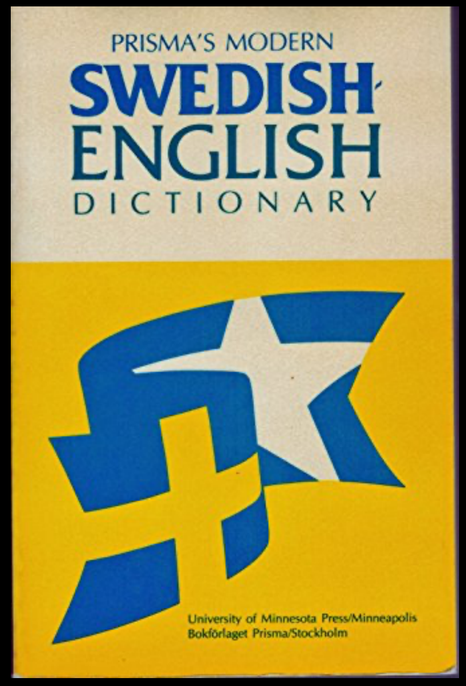


```
munkihash.swift
osinstaller.swift
version.swift
errors.swift
sqlite3.swift
launchd.swift
appusage.swift
xattr.swift
processes.swift
munkistatus.swift
powermanager.swift
stoprequested.swift
info.swift
appinventory.swift
facts.swift
Predicates.m
installinfo.swift
bootstrapping.swift
UNIXProcessInfo.swift
SignalHandler.swift
appleupdates.swift
managedsoftwareupdate
managedsoftwareupd...
msuutils.swift
distributednotification...
msuoptions.swift
munkitester
makecatalogs
makepkginfo
munkiimport
removepackages
app_usage_monitor
appusaged
```

```
munki swift-cli
Any Mac (arm  Build Succeeded | 9/20/24 at 11:42 AM
msuutils.swift
errors.swift  munkihash.swift  osutils.swift  msuutils.swift
munki > managedsoftwareupdate > msuutils.swift > doInstallTasks(doAppleUpdates:onlyUnattended:)
```

```
270 func doInstallTasks(doAppleUpdates: Bool = false, onlyUnattended:
274         clearLastNotifiedDate()
275     }
276
277     var munkiItemsRestartAction = PostAction.none
278     var appleItemsRestartAction = PostAction.none
279
280     if munkiUpdatesAvailable() > 0 {
281         // install Munki updates
282         munkiItemsRestartAction = await
                doInstallsAndRemovals(onlyUnattended: onlyUnattended)
283         if !onlyUnattended {
284             if
                    munkiUpdatesContainItemWithInstallerType
                    ("startosinstall") {
285                 Report.shared.save()
286                 // install macOS
287                 // TODO: implement this (install macOS)
288             }
289         }
290     }
291     if doAppleUpdates {
292         // install Apple updates
293         // TODO: implement? appleItemsRestartAction =
                installAppleUpdates(onlyUnattended: onlyUnattended)
294         // We're probably never going to implement this...
295     }
296
297     Report.shared.save()
298
299     return max(appleItemsRestartAction, munkiItemsRestartAction)
300 }
301
302 /// Handle the need for a forced logout. Start our logouthelper
```

Dealing with Dictionaries



```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/
PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
    <key>catalogs</key>
    <array>
        <string>production</string>
    </array>
    <key>description</key>
    <string>Google's web browser.</string>
    <key>installer_item_location</key>
    <string>apps/GoogleChrome-129.0.6668.71.dmg</string>
    <key>installer_item_size</key>
    <integer>210383</integer>
    <key>name</key>
    <string>GoogleChrome</string>
    <key>uninstallable</key>
    <true/>
    <key>version</key>
    <string>129.0.6668.71</string>
</dict>
</plist>
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/
PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
    <key>catalogs</key>
    <array>
        <string>production</string>
    </array>
    <key>description</key>
    <string>Google's web browser.</string>
    <key>installer_item_location</key>
    <string>apps/GoogleChrome-129.0.6668.71.dmg</string>
    <key>installer_item_size</key>
    <integer>210383</integer>
    <key>name</key>
    <string>GoogleChrome</string>
    <key>uninstallable</key>
    <true/>
    <key>version</key>
    <string>129.0.6668.71</string>
</dict>
</plist>
```

```json
{
    "catalogs": ["production"],
    "description": "Google's web browser.",
    "display_name": "Google Chrome",
    "installer_item_location": "apps/GoogleChrome-129.0.6668.71.dmg",
    "installer_item_size": 210383,
    "name": "GoogleChrome",
    "uninstallable": True,
    "version": "129.0.6668.71"
}
```

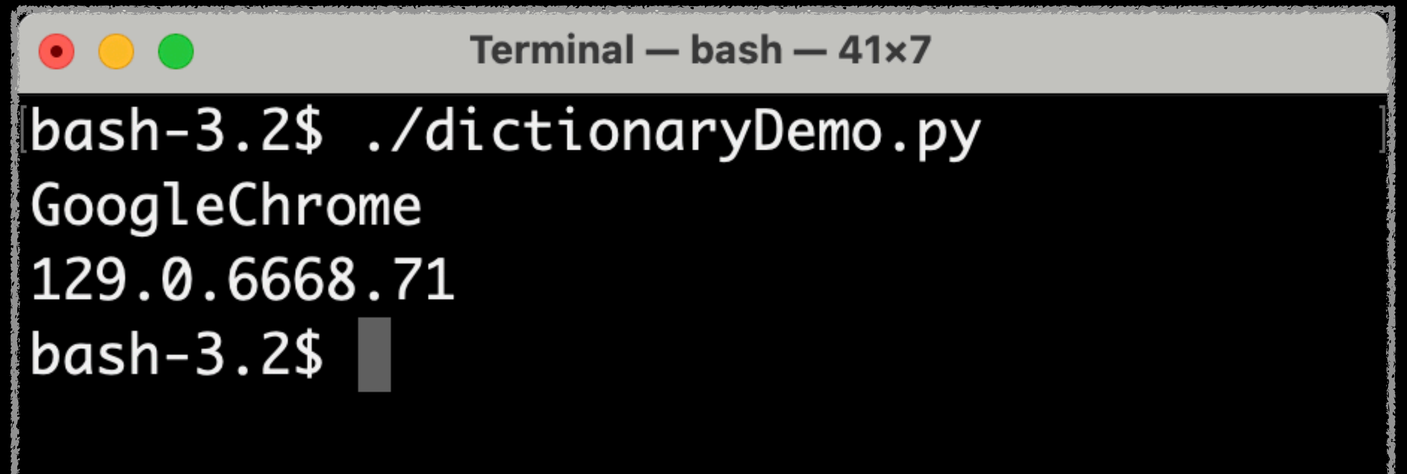
```
{
    "catalogs": ["production"],
    "description": "Google's web browser.",
    "display_name": "Google Chrome",
    "installer_item_location": "apps/GoogleChrome-129.0.6668.71.dmg",
    "installer_item_size": 210383,
    "name": "GoogleChrome",
    "uninstallable": True,
    "version": "129.0.6668.71"
}
```

```
#!/usr/bin/env python3

item = {
    "catalogs": ["production"],
    "description": "Google's web browser.",
    "display_name": "Google Chrome",
    "installer_item_location": "apps/GoogleChrome-129.0.6668.71.dmg",
    "installer_item_size": 210383,
    "name": "GoogleChrome",
    "uninstallable": True,
    "version": "129.0.6668.71"
}

print(item["name"])
print(item["version"])
```

```
#!/usr/bin/env python3

item = {
    "catalogs": ["production"],
    "description": "Google's web browser.",
    "display_name": "Google Chrome",
    "installer_item_location": "apps/GoogleChrome-129.0.6668.71.dmg",
    "installer_item_size": 210383,
    "name": "GoogleChrome",
    "uninstallable": True,
    "version": "129.0.6668.71"
}

print(item["name"])
print(item["version"])
```

```
#!/usr/bin/env python3

item = {
    "catalogs": ["production"],
    "description": "Google's web browser.",
    "display_name": "Google Chrome",
    "installer_item_location": "apps/GoogleChrome-129.0.6668.71.dmg",
    "installer_item_size": 210383,
    "name": "GoogleChrome",
    "uninstallable": True,
    "version": "129.0.6668.71"
}

print(item["name"])
print(item["version"])
```

```python
#!/usr/bin/env python3

item = {
    "catalogs": ["production"],
    "description": "Google's web browser.",
    "display_name": "Google Chrome",
    "installer_item_location": "apps/GoogleChrome-129.0.6668.71.dmg",
    "installer_item_size": 210383,
    "name": "GoogleChrome",
    "uninstallable": True,
    "version": "129.0.6668.71"
}

print(item["name"])
print(item["version"])
```



```
#!/usr/bin/swift

let item = [
    "catalogs": ["production"],
    "description": "Google's web browser.",
    "display_name": "Google Chrome",
    "installer_item_location": "apps/GoogleChrome-129.0.6668.71.dmg",
    "installer_item_size": 210383,
    "name": "GoogleChrome",
    "uninstallable": true,
    "version": "129.0.6668.71"
]

print(item["name"])
print(item["version"])
```

```
#!/usr/bin/swift

let item
    "cata
    "desc
    "disp
    "inst                                                    mg",
    "inst
    "name
    "unin
    "vers
]

print(ite
print(ite
```

```
Terminal — bash — 93×24
bash-3.2$ ./dictionaryDemo.swift
dictionaryDemo.swift:3:12: error: heterogeneous collection literal could only be inferred to
'[String : Any]'; add explicit type annotation if this is intentional
let item = [
           ^
dictionaryDemo.swift:14:7: warning: expression implicitly coerced from 'Any?' to 'Any'
print(item["name"])
      ^~~~~~~~~~~~
dictionaryDemo.swift:14:11:       provide a default value to avoid this warning
print(item["name"])
      ~~~~^~~~~~~~
                  ?? <#default value#>
dictionaryDemo.swift:14:11:       force-unwrap the value to avoid this warning
print(item["name"])
      ~~~~^~~~~~~~
                  !
dictionaryDemo.swift:14:11:       explicitly cast to 'Any' with 'as Any' to silence this warn
ing
print(item["name"])
      ~~~~^~~~~~~~
                  as Any
dictionaryDemo.swift:15:7: warning: expression implicitly coerced from 'Any?' to 'Any'
print(item["version"])
      ^~~~~~~~~~~~~~~
```

```
bash-3.2$ swift repl
Welcome to Apple Swift version 5.10 (swiftlang-5.10.0.13 clang-1500.3.9.4).
Type :help for assistance.
  1> let item = [
  2.     "catalogs": ["production"],
  3.     "description": "Google's web browser.",
  4.     "display_name": "Google Chrome",
  5.     "installer_item_location": "apps/GoogleChrome-129.0.6668.71.dmg",
  6.     "installer_item_size": 210383,
  7.     "name": "GoogleChrome",
  8.     "uninstallable": true,
  9.     "version": "129.0.6668.71"
 10. ]
error: repl.swift:1:12: error: heterogeneous collection literal could only be inferred to
'[String : Any]'; add explicit type annotation if this is intentional
let item = [
           ^
```

```
bash-3.2$ swift repl
Welcome to Apple Swift version 5.10 (swiftlang-5.10.0.13 clang-1500.3.9.4).
Type :help for assistance.
  1> let item = [
  2.     "catalogs": ["production"],
  3.     "description": "Google's web browser.",
  4.     "display_name": "Google Chrome",
  5.     "installer_item_location": "apps/GoogleChrome-129.0.6668.71.dmg",
  6.     "installer_item_size": 210383,
  7.     "name": "GoogleChrome",
  8.     "uninstallable": true,
  9.     "version": "129.0.6668.71"
 10. ]
error: repl.swift:1:12: err
'[String : Any]'; add expli
let item = [
           ^
```

heterogeneous

Overview | Usage examples | Similar and opposite words

adjective. **different in kind; unlike; incongruous**. composed of parts of different kinds; having widely dissimilar elements or constituents: The party was attended by a heterogeneous group of artists, politicians, and social climbers.

```
bash-3.2$ swift repl
Welcome to Apple Swift version 5.10 (swiftlang-5.10.0.13 clang-1500.3.9.4).
Type :help for assistance.
  1> let item = [
  2.     "catalogs": ["production"],
  3.     "description": "Google's web browser.",
  4.     "display_name": "Google Chrome",
  5.     "installer_item_location": "apps/GoogleChrome-129.0.6668.71.dmg",
  6.     "installer_item_size": 210383,
  7.     "name": "GoogleChrome",
  8.     "uninstallable": true,
  9.     "version": "129.0.6668.71"
 10. ]
error: repl.swift:1:12: error: heterogeneous collection literal could only be inferred to
'[String : Any]'; add explicit type annotation if this is intentional
let item = [
           ^
```

```
bash-3.2$ swift repl
Welcome to Apple Swift version 5.10 (swiftlang-5.10.0.13 clang-1500.3.9.4).
Type :help for assistance.
  1> let item = [
  2.     "catalogs": ["production"],
  3.     "description": "Google's web browser.",
  4.     "display_name": "Google Chrome",
  5.     "installer_item_location": "apps/GoogleChrome-129.0.6668.71.dmg",
  6.     "installer_item_size": 210383,
  7.     "name": "GoogleChrome",
  8.     "uninstallable": true,
  9.     "version": "129.0.6668.71"
 10. ]
error: repl.swift:1:12: error: heterogeneous collection literal could only be inferred to
 [String : Any] ; add explicit type annotation if this is intentional
let item = [
           ^
```

```
bash-3.2$ swift repl
Welcome to Apple Swift version 5.10 (swiftlang-5.10.0.13 clang-1500.3.9.4).
Type :help for assistance.
  1> let item = [
  2.     "catalogs": ["production"],
  3.     "description": "Google's web browser.",
  4.     "display_name": "Google Chrome",
  5.     "installer_item_location": "apps/GoogleChrome-129.0.6668.71.dmg",
  6.     "installer_item_size": 210383,
  7.     "name": "GoogleChrome",
  8.     "uninstallable": true,
  9.     "version": "129.0.6668.71"
 10. ]
error: repl.swift:1:12: error: heterogeneous collection literal could only be inferred to
 [String : Any] ; add explicit type annotation if this is intentional
let item = [
           ^
```

```
bash-3.2$ swift repl
Welcome to Apple Swift version 5.10 (swiftlang-5.10.0.13 clang-1500.3.9.4).
Type :help for assistance.
  1> let item: [String : Any] = [
  2.     "catalogs": ["production"],
  3.     "description": "Google's web browser.",
  4.     "display_name": "Google Chrome",
  5.     "installer_item_location": "apps/GoogleChrome-129.0.6668.71.dmg",
  6.     "installer_item_size": 210383,
  7.     "name": "GoogleChrome",
  8.     "uninstallable": true,
  9.     "version": "129.0.6668.71"
 10. ]
```

```
bash-3.2$ swift repl
Welcome to Apple Swift version 5.10 (swiftlang-5.10.0.13 clang-1500.3.9.4).
Type :help for assistance.
  1> let item: [String : Any] = [
  2.     "catalogs": ["production"],
  3.     "description": "Google's web browser.",
  4.     "display_name": "Google Chrome",
  5.     "installer_item_location": "apps/GoogleChrome-129.0.6668.71.dmg",
  6.     "installer_item_size": 210383,
  7.     "name": "GoogleChrome",
  8.     "uninstallable": true,
  9.     "version": "129.0.6668.71"
 10. ]
item: [String : Any] = 8 key/value pairs {
  [0] = {
    key = "version"
    value = "129.0.6668.71"
  }
  [1] = {
    key = "uninstallable"
    value = true
```

```
    value = "apps/GoogleChrome-129.0.6668.71.dmg"
  }
  [6] = {
    key = "installer_item_size"
    value = 210383
  }
  [7] = {
    key = "display_name"
    value = "Google Chrome"
  }
}
 11>
```

```
    value = "apps/GoogleChrome-129.0.6668.71.dmg"
  }
  [6] = {
    key = "installer_item_size"
    value = 210383
  }
  [7] = {
    key = "display_name"
    value = "Google Chrome"
  }
}
 11> print(item["name"])
Optional("GoogleChrome")
 12> print(item["uninstallable"])
Optional(true)
 13> print(item["NoSuchKey"])
nil
 14>
```

```
    value = "apps/GoogleChrome-129.0.6668.71.dmg"
  }
  [6] = {
    key = "installer_item_size"
    value = 210383
  }
  [7] = {
    key = "display_name"
    value = "Google Chrome"
  }
}
 11> print(item["name"])
Optional("GoogleChrome")
 12> print(item["uninstallable"])
Optional(true)
 13> print(item["NoSuchKey"])
nil
 14>
```

Optional

- Might be a value
- Might be nil (undefined)

Unwrapping an Optional

```
if let name = item["name"] {
    print(name)
} else {
    print("<UNKNOWN>")
}
if let version = item["version"] {
    print(version)
} else {
    print("<UNKNOWN>")
}
```

Unwrapping an Optional

```
if let name = item["name"] {
    print(name)
} else {
    print("<UNKNOWN>")
}
if let version = item["version"] {
    print(version)
} else {
    print("<UNKNOWN>")
}
```

Unwrapping an Optional

```
if let name = item["name"] {
    print(name)
} else {
    print("<UNKNOWN>")
}
if let version = item["version"] {
    print(version)
} else {
    print("<UNKNOWN>")
}
```

Unwrapping an Optional

```
if let name = item["name"] {
    print(name)
} else {
    print("<UNKNOWN>")
}
if let version = item["version"] {
    print(version)
} else {
    print("<UNKNOWN>")
}
```

Unwrapping an Optional

```
if let name = item["name"] {
    print(name)
} else {
    print("<UNKNOWN>")
}
if let version = item["version"] {
    print(version)
} else {
    print("<UNKNOWN>")
}
```

Unwrapping an Optional

```
if let name = item["name"] {
    print(name)
} else {
    print("<UNKNOWN>")
}
if let version = item["version"] {
    print(version)
} else {
    print("<UNKNOWN>")
}
```

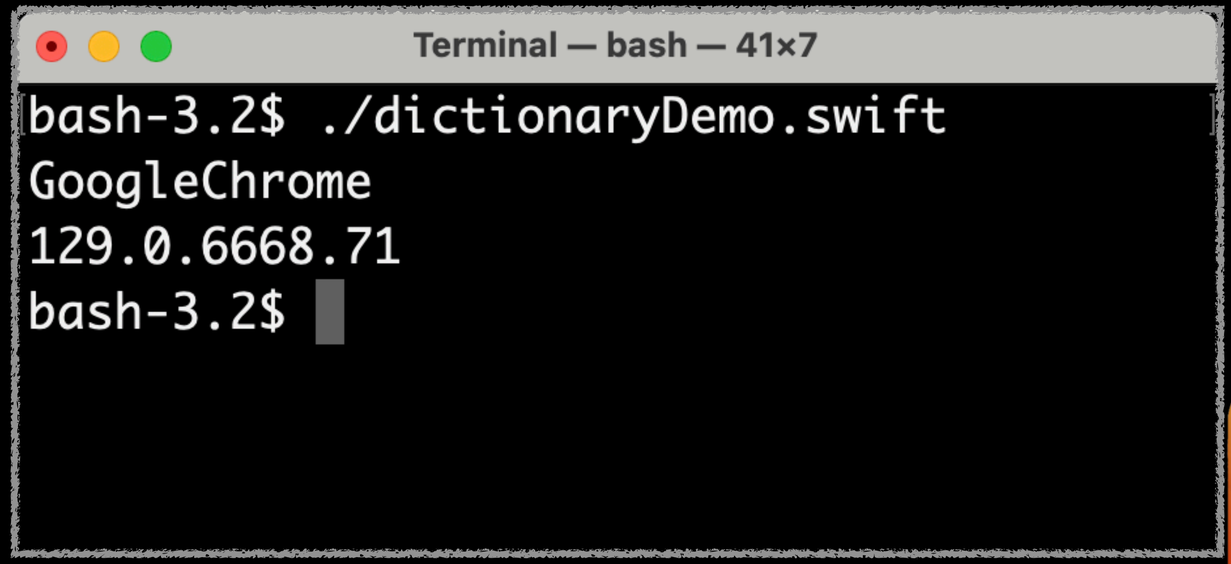
Unwrapping an Optional

```
print(item["name"] ?? "<UNKNOWN>")
print(item["version"] ?? "<UNKNOWN>")
```

Unwrapping an Optional

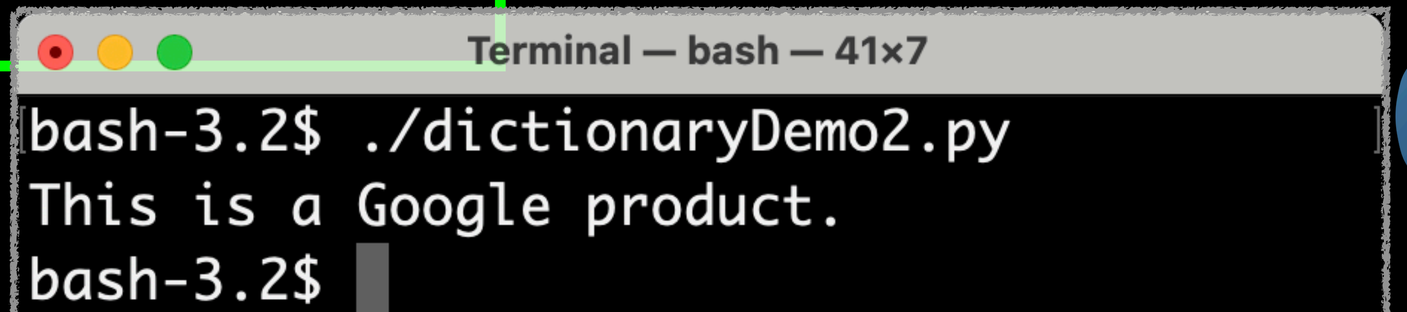
```
print(item["name"] ?? "<UNKNOWN>")
print(item["version"] ?? "<UNKNOWN>")
```

```
#!/usr/bin/swift

let item: [String : Any] = [
    "catalogs": ["production"],
    "description": "Google's web browser.",
    "display_name": "Google Chrome",
    "installer_item_location": "apps/GoogleChrome-129.0.6668.71.dmg",
    "installer_item_size": 210383,
    "name": "GoogleChrome",
    "uninstallable": true,
    "version": "129.0.6668.71"
]

print(item["name"] ?? "")
print(item["version"] ?? "")
```



But wait there's more

```python
#!/usr/bin/env python3

item = {
    "catalogs": ["production"],
    "description": "Google's web browser.",
    "display_name": "Google Chrome",
    "installer_item_location": "apps/GoogleChrome-129.0.6668.71.dmg",
    "installer_item_size": 210383,
    "name": "GoogleChrome",
    "uninstallable": True,
    "version": "129.0.6668.71"
}

if item["name"].startswith("Google"):
    print("This is a Google product.")
```



```
#!/usr/bin/swift

let item: [String : Any] = [
    "catalogs": ["production"],
    "description": "Google's web browser.",
    "display_name": "Google Chrome",
    "installer_item_location": "apps/GoogleChrome-129.0.6668.71.dmg",
    "installer_item_size": 210383,
    "name": "GoogleChrome",
    "uninstallable": true,
    "version": "129.0.6668.71"
]

let name = item["name"] ?? ""
if name.hasPrefix("Google") {
    print("This is a Google product.")
}
```

```
#!/usr/bin/swift

let item: [String : Any] = [
    "catalogs": ["production"],
    "description": "Google's web browser.",
    "display_name": "Google Chrome",
    "installer_item_location": "apps/GoogleChrome-129.0.6668.71.dmg",
    "installer_item_size": 210383,
    "name": "GoogleChrome",
    "uninstallable": true,
    "ve
]

let nam
if name
    pri
}
```

```
Terminal — bash — 80×25
bash-3.2$ ./dictionaryDemo2.swift
dictionaryDemo2.swift:15:9: error: value of type 'Any' has no member 'hasPrefix'
if name.hasPrefix("Google") {
   ~~~~ ^~~~~~~~~~
dictionaryDemo2.swift:15:9:     cast 'Any' to 'AnyObject' or use 'as!' to forc
e downcast to a more specific type to access members
if name.hasPrefix("Google") {
        ^
   (    as AnyObject)
bash-3.2$
```

```
#!/usr/bin/swift

let item: [String : Any] = [
    "catalogs": ["production"],
    "description": "Google's web browser.",
    "display_name": "Google Chrome",
    "installer_item_location": "apps/GoogleChrome-129.0.6668.71.dmg",
    "installer_item_size": 210383,
    "name": "GoogleChrome",
    "uninstallable": true,
    "ve
]

let nam
if name
    pri
}
```

```
Terminal — bash — 80×25
bash-3.2$ ./dictionaryDemo2.swift
dictionaryDemo2.swift:15:9: error: value of type 'Any' has no member 'hasPrefix'
if name.hasPrefix("Google") {
   ~~~~ ^~~~~~~~~
dictionaryDemo2.swift:15:9: note: cast 'Any' to 'AnyObject' or use 'as!' to forc
e downcast to a more specific type to access members
if name.hasPrefix("Google") {
        ^
   (    as AnyObject)
bash-3.2$
```

```
#!/usr/bin/swift

let item: [String : Any] = [
    "catalogs": ["production"],
    "description": "Google's web browser.",
    "display_name": "Google Chrome",
    "installer_item_location": "apps/GoogleChrome-129.0.6668.71.dmg",
    "installer_item_size": 210383,
    "name": "GoogleChrome",
    "uninstallable": true,
    "version": "129.0.6668.71"
]

let name = item["name"] ?? ""
if name.hasPrefix("Google") {
    print("This is a Google product.")
}
```

```
#!/usr/bin/swift

let item: [String : Any] = [
    "catalogs": ["production"],
    "description": "Google's web browser.",
    "display_name": "Google Chrome",
    "installer_item_location": "apps/GoogleChrome-129.0.6668.71.dmg",
    "installer_item_size": 210383,
    "name": "GoogleChrome",
    "uninstallable": true,
    "version": "129.0.6668.71"
]

let name = item["name"] ?? ""
if name.hasPrefix("Google") {
    print("This is a Google product.")
}
```

Array of strings

String

Integer

Boolean

```
#!/usr/bin/swift

let item: [String : Any] = [
    "catalogs": ["production"],
    "description": "Google's web browser.",
    "display_name": "Google Chrome",
    "installer_item_location": "apps/GoogleChrome-129.0.6668.71.dmg",
    "installer_item_size": 210383,
    "name": "GoogleChrome",
    "uninstallable": true,
    "version": "129.0.6668.71"
]

let name = item["name"] ?? ""
if name.hasPrefix("Google") {
    print("This is a Google product.")
}
```

Array of strings

String

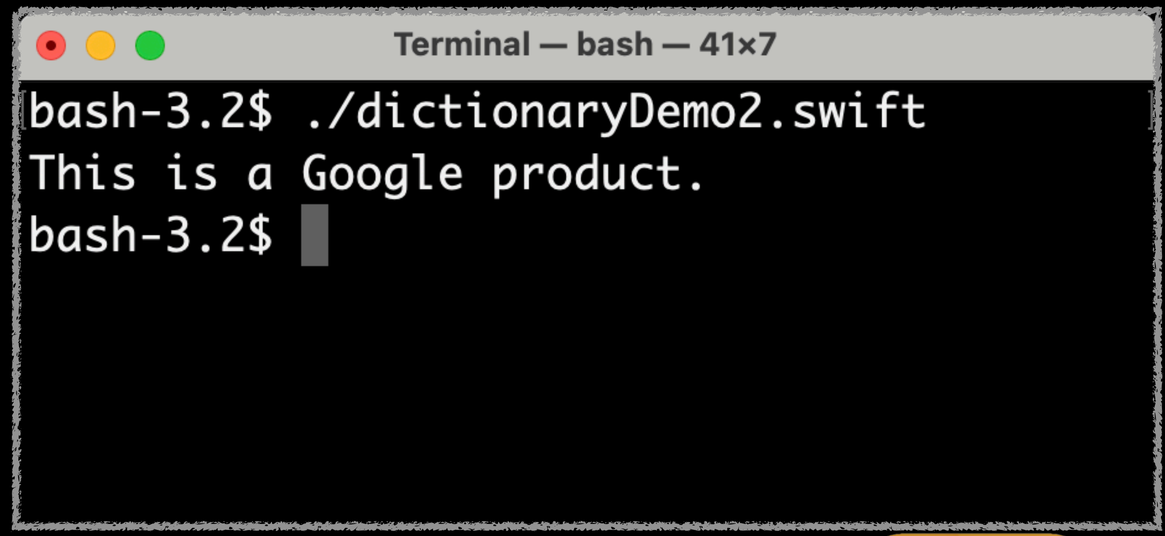
Integer

Boolean

```
#!/usr/bin/swift

let item: [String : Any] = [
    "catalogs": ["production"],
    "description": "Google's web browser.",
    "display_name": "Google Chrome",
    "installer_item_location": "apps/GoogleChrome-129.0.6668.71.dmg",
    "installer_item_size": 210383,
    "name": "GoogleChrome",
    "uninstallable": true,
    "version": "129.0.6668.71"
]

let name = item["name"] as? String ?? ""
if name.hasPrefix("Google") {
    print("This is a Google product.")
}
```

```
#!/usr/bin/swift

let item: [String : Any] = [
    "catalogs": ["production"],
    "description": "Google's web browser.",
    "display_name": "Google Chrome",
    "installer_item_location": "apps/GoogleChrome-129.0.6668.71.dmg",
    "installer_item_size": 210383,
    "name": "GoogleChrome",
    "uninstallable": true,
    "version": "129.0.6668.71"
]

let name = item["name"] as? String ?? ""
if name.hasPrefix("Google") {
    print("This is a Google product.")
}
```

```swift
#!/usr/bin/swift

let item: [String : Any] = [
    "catalogs": ["production"],
    "description": "Google's web browser.",
    "display_name": "Google Chrome",
    "installer_item_location": "apps/GoogleChrome-129.0.6668.71.dmg",
    "installer_item_size": 210383,
    "name": "GoogleChrome",
    "uninstallable": true,
    "version": "129.0.6668.71"
]

let name = item["name"] as? String ?? ""
if name.hasPrefix("Google") {
    print("This is a Google product.")
}
```



```swift
let itemName = item["name"] as? String ?? "<unknown>"
let installerType = item["installer_type"] as? String ?? "pkg_install"
let installerItem = item["installer_item"] as? String ?? ""
```

List Comprehensions

```python
#!/usr/bin/env python3

fruits = [
    "Apricot",
    "Apple",
    "Avocado",
    "Banana",
    "Blueberry",
    "Cantaloupe",
    "Cherry",
    "Cranberry"
]

big_fruits = [item.upper() for item in fruits]

print(big_fruits)
```

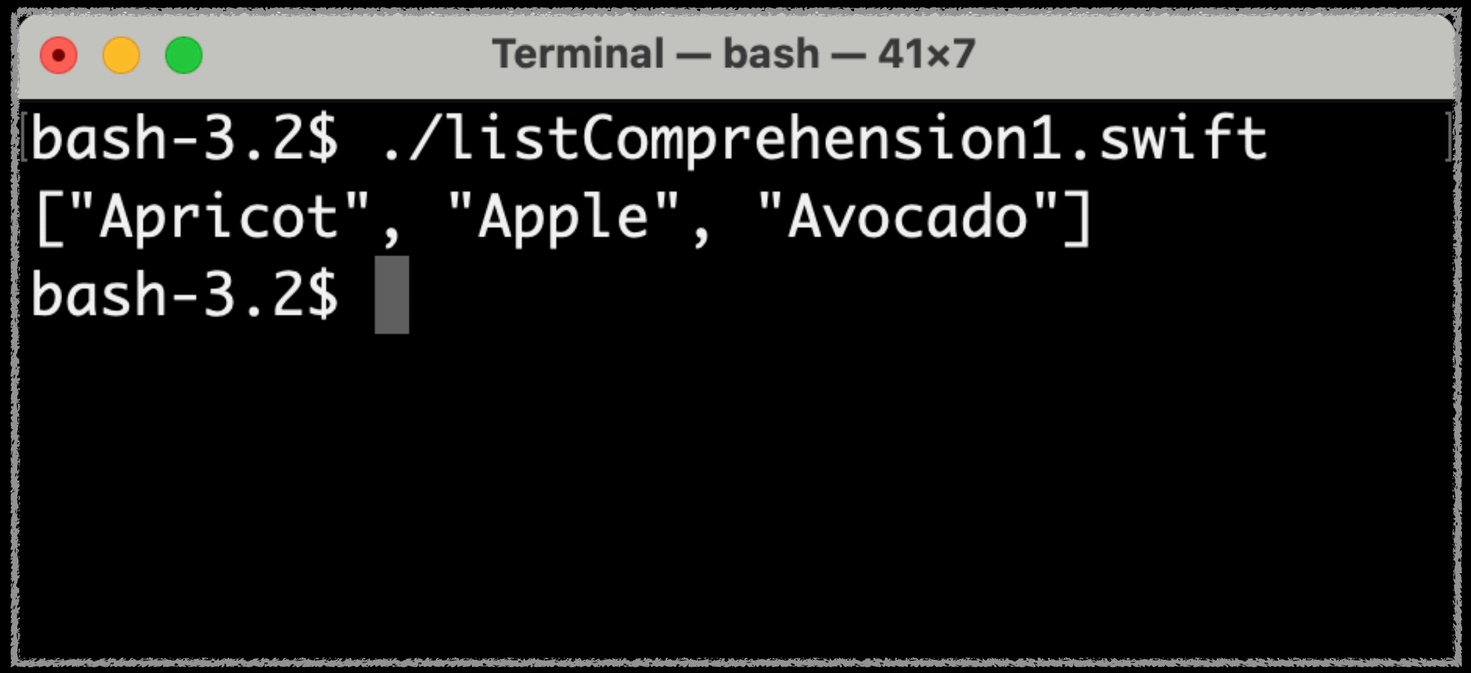
```
Terminal — bash — 41×7
bash-3.2$ ./listComprehension2.py
['APRICOT', 'APPLE', 'AVOCADO', 'BANANA',
 'BLUEBERRY', 'CANTALOUPE', 'CHERRY', 'CR
ANBERRY']
bash-3.2$
```

```
#!/usr/bin/swift

let fruits = [
    "Apricot",
    "Apple",
    "Avocado",
    "Banana",
    "Blueberry",
    "Cantaloupe",
    "Cherry",
    "Cranberry"
]

let msaFruit = fruits.map {
  $0.uppercased()
}

print(msaFruit)
```

```
Terminal — bash — 41×7
bash-3.2$ ./listComprehension2.swift
["APRICOT", "APPLE", "AVOCADO", "BANANA",
 "BLUEBERRY", "CANTALOUPE", "CHERRY", "CR
ANBERRY"]
bash-3.2$
```

```python
#!/usr/bin/env python3

fruits = [
    "Apricot",
    "Apple",
    "Avocado",
    "Banana",
    "Blueberry",
    "Cantaloupe",
    "Cherry",
    "Cranberry"
]

filtered_fruits = [fruit for fruit in fruits if fruit.startswith("A")]

print(filtered_fruits)
```

```swift
#!/usr/bin/swift

let fruits = [
    "Apricot",
    "Apple",
    "Avocado",
    "Banana",
    "Blueberry",
    "Cantaloupe",
    "Cherry",
    "Cranberry"
]

let filteredFruits = fruits.filter {
  $0.hasPrefix("A")
}

print(filteredFruits)
```



```python
#!/usr/bin/env python3

fruits = [
    "Apricot",
    "Apple",
    "Avocado",
    "Banana",
    "Blueberry",
    "Cantaloupe",
    "Cherry",
    "Cranberry"
]

msa_fruit = [item.upper() for item in fruits if item.endswith("a")]

print(msa_fruit)
```

```swift
#!/usr/bin/swift

let fruits = [
    "Apricot",
    "Apple",
    "Avocado",
    "Banana",
    "Blueberry",
    "Cantaloupe",
    "Cherry",
    "Cranberry"
]

let msaFruit = fruits.filter {
    $0.hasSuffix("a")
}.map {
    $0.uppercased()
}

print(msaFruit)
```

```
Terminal — bash — 41×7
bash-3.2$ ./listComprehension2.swift
["BANANA"]
bash-3.2$
```

Progress update

Lines of code

~30,000

~20,000 (so far)

Deprecations/Removals

Deprecations/Removals

- Configuration Profiles
- Special treatment of Adobe installers
- Apple update metadata
- `appdmg` installer type
- `suppress_bundle_relocation` pkginfo key
- Installation of Apple Software Updates

Maybe?

- Drop support for `startosinstall` (leaving support for `stage_os_installer`)

Demo

Tack!

Links

pymacadmin and crankd: https://github.com/MacSysadmin/pymacadmin

Ben Toms, Current user in Python/PyObjC: https://macmule.com/2014/11/19/how-to-get-the-currently-logged-in-user-in-a-more-apple-approved-way/

Munki: https://github.com/munki/munki

Relocatable Python: https://github.com/gregneagle/relocatable-python

MacAdmins Python: https://github.com/macadmins/python

MacDevOpsYVR 2023 "Being Responsible": https://www.youtube.com/watch?v=DcrfCGqqjkA

Installomator: https://github.com/installomator/installomator

Super: https://github.com/Macjutsu/super

Baseline: https://github.com/SecondSonConsulting/Baseline

Renew: https://github.com/SecondSonConsulting/Renew

Joel Rennich, From Shell to Swift (MacSysAdmin 2016): http://docs.macsysadmin.se/2016/video/Day1Session2.mp4

Armin Briegel, Swift for Apple Admins (MacSysAdmin 2018): http://docs.macsysadmin.se/2018/video/Day3Session5.mp4

Armin Briegel, Let's Swift Again (MacSysAdmin 2021): https://docs.macsysadmin.se/2021/video/Day1Session1.mp4

Nudge: https://github.com/macadmins/nudge

Outset: https://github.com/macadmins/outset