

Introduction to PyObjC

Greg Neagle, Disney Animation Studios

<https://github.com/gregneagle/macaduk2017>

This repository

Pull requests Issues Gist

Unwatch 1 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Pulse Graphs Settings

Example code for MacADUK 2017 presentation: "Introduction to PyObjC"

New Add topics

3 commits 1 branch 0 releases 1 contributor

Branch: master New pull request Create new file Upload files Find file Clone or download

Clone with HTTPS Use SSH
Use Git or checkout with SVN using the web URL.
<https://github.com/gregneagle/macaduk2>

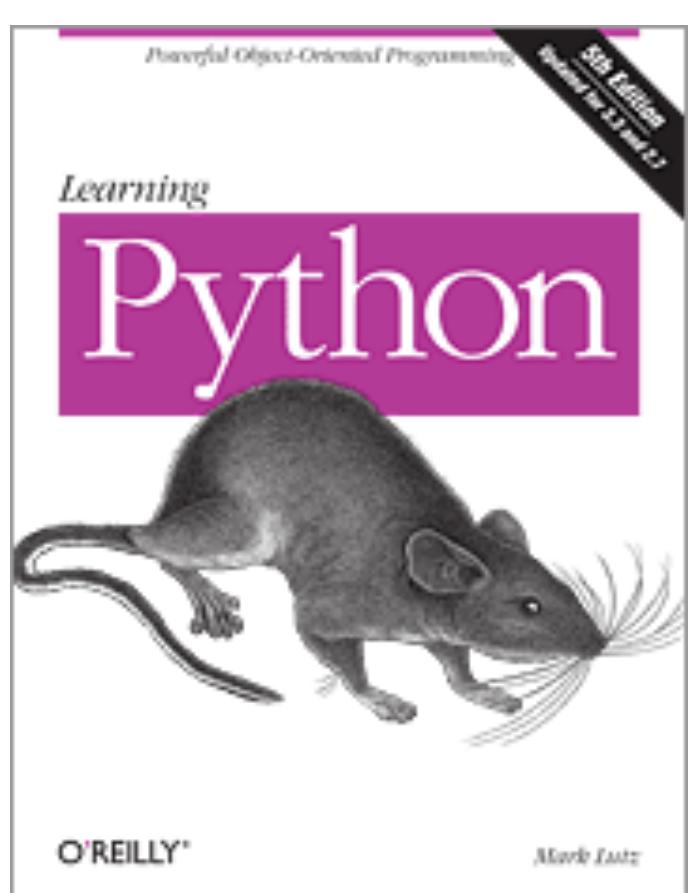
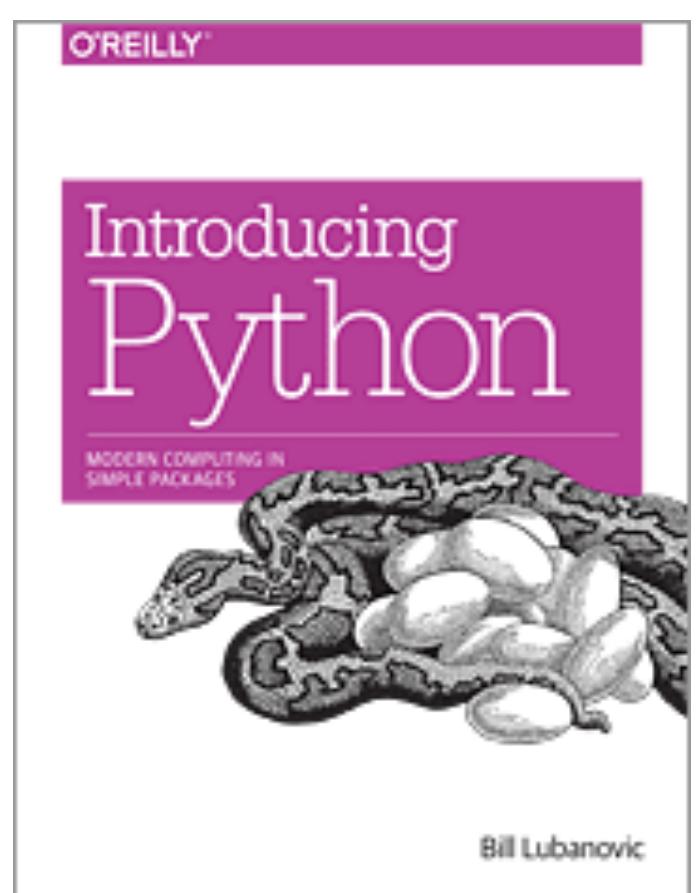
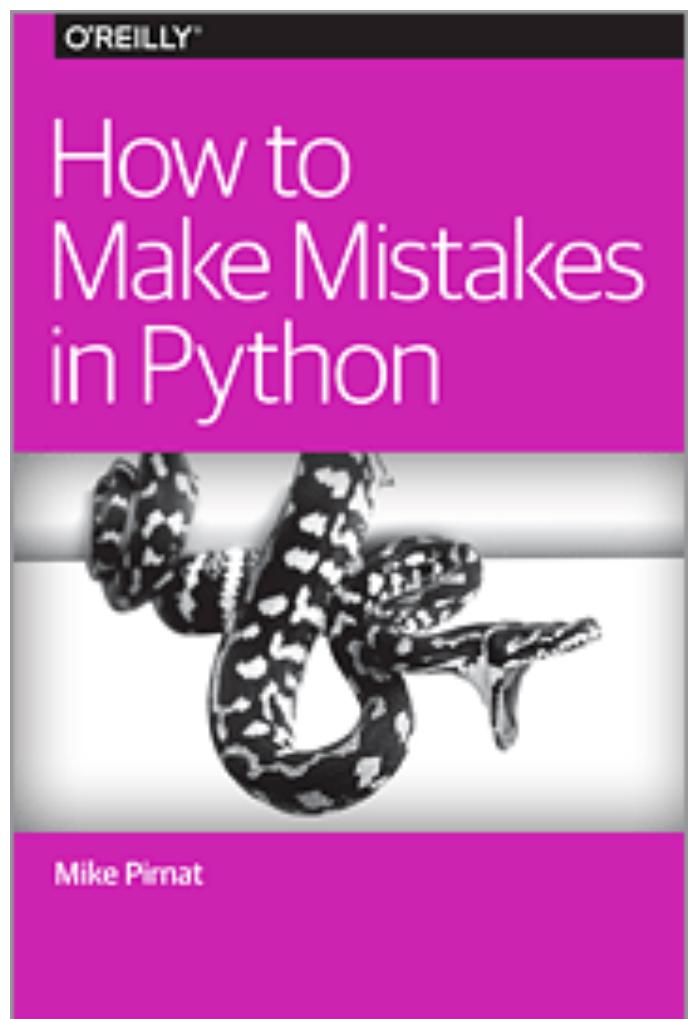
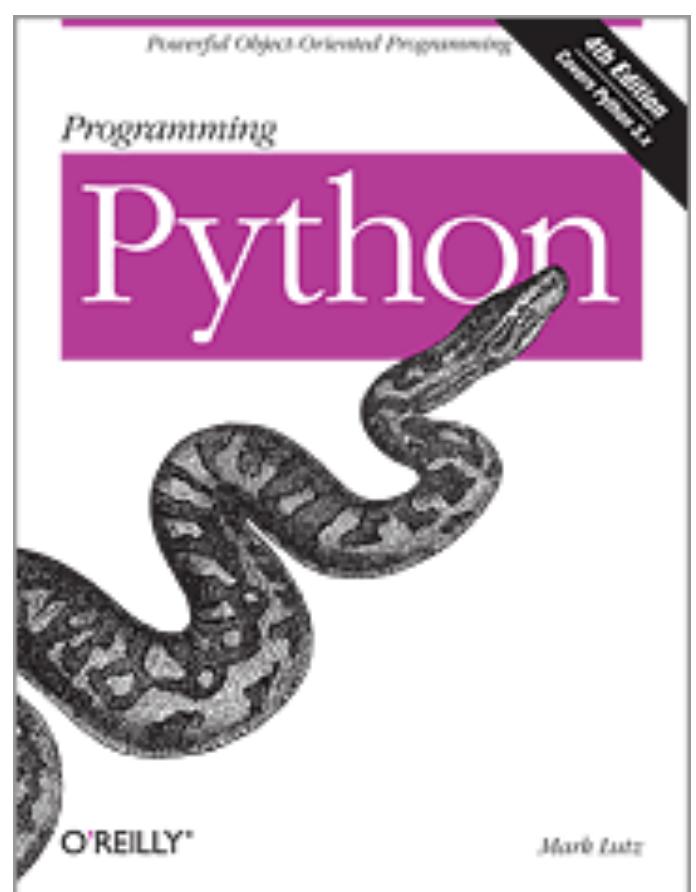
Open in Desktop Download ZIP

Download

<https://github.com/gregneagle/macaduk2017/archive/master.zip>

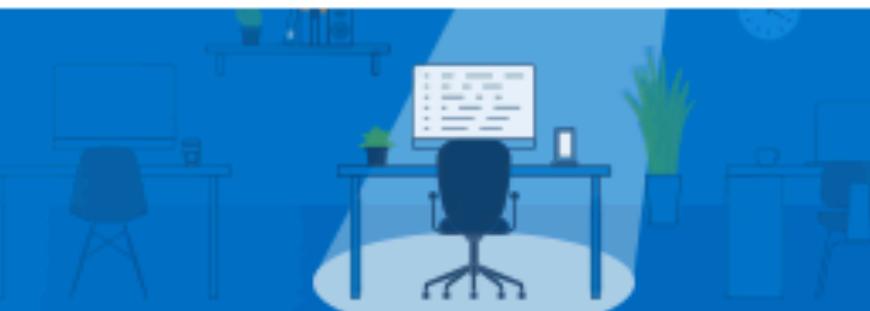
-or-

`git clone https://github.com/gregneagle/macaduk2017.git`



Why is the PyObjC documentation so bad? [closed]

Get personalized job matches now



stackoverflow
JOBS

Get started

asked 8 years ago

viewed 3777 times

active 3 years ago

16

For example, <http://developer.apple.com/cocoa/pyobjc.html> is still for OS X 10.4 Tiger, not 10.5 Leopard.. And that's the official Apple documentation for it..

The official PyObjC page is equally bad, <http://pyobjc.sourceforge.net/>

It's so bad it's baffling.. I'm considering learning Ruby primarily because the RubyCocoa stuff is so

BLOG

Stack Overflow Podcast #100 - Jeff Atwood Is Back! (For Today)

Developers without Borders: The

All I want to do is create fairly simple Python applications with Cocoa GUI's..
Can anyone shed light on the horrible documentation, or point me at some tutorials that don't just give you huge blocks of code and assume you know what
`NSThread.detachNewThreadSelector_toTarget_withObject_("queryController", self, None)` does..?

python osx cocoa pyobjc

share improve this question

edited Aug 20 '13 at 12:56



Will

98.4k • 41 • 245 • 342

asked Aug 18 '08 at 10:23



dbr

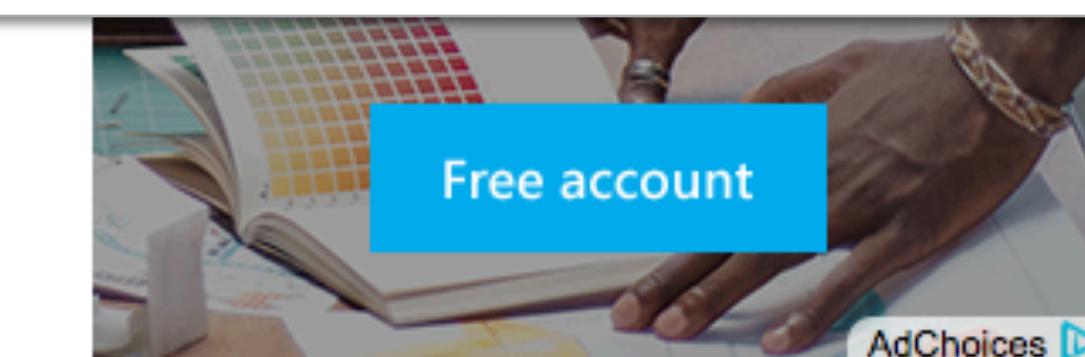
96.4k • 48 • 229 • 298

closed as primarily opinion-based by [Will](#), [ale](#), [SheetJS](#), [Jave](#), [smerny](#) Aug 20 '13 at 14:46

Many good questions generate some degree of opinion based on expert experience, but answers to this question will tend to be almost entirely based on opinions, rather than facts, references, or specific expertise.

If this question can be reworded to fit the rules in the [help center](#), please [edit the question](#).

The first PyObjC tutorial you link to is an article by Jonathan Rentzsch (you can hear his voice in the videos, and you may recognize it from the C4 videos). Even though it's on Apple's site without a by-line, it's not "Apple



Looking for a job?

Python Developer - Build the smartest way to buy a car (Telecommute - full time)

Unhaggle Inc. • No office location

REMOTE

django python

Backend Engineer

Farmers Insurance • Los Angeles, CA

Why is the PyObjC documentation so bad? [closed]

asked 8 years ago

I agree that that tutorial is flawed, throwing random, unexplained code right in front of your eyes. It introduces concepts such as the autorelease pool and user defaults without explaining why you would want them ("Autorelease pool for memory management" is hardly an explanation).
20

That said...

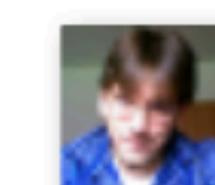
basically all I want to do is write Cocoa applications without having to learn ObjC.

I'm afraid that for the time being, you *will* need a basic grasp of ObjC in order to benefit from any language that uses Cocoa. PyObjC, RubyCocoa, Nu and others are niches at best, and all of them were developed by people intimately familiar with the ins and outs of ObjC *and* Cocoa.

For now, you will benefit the most if you realistically see those bridges as useful where scripting languages truly shine, rather than trying to build a whole application with them. While this *has* been done (with LimeChat, I'm using a RubyCocoa-written app right now), it is rare and likely will be for a while.

share improve this answer

answered Aug 18 '08 at 12:18



Sören Kuklau

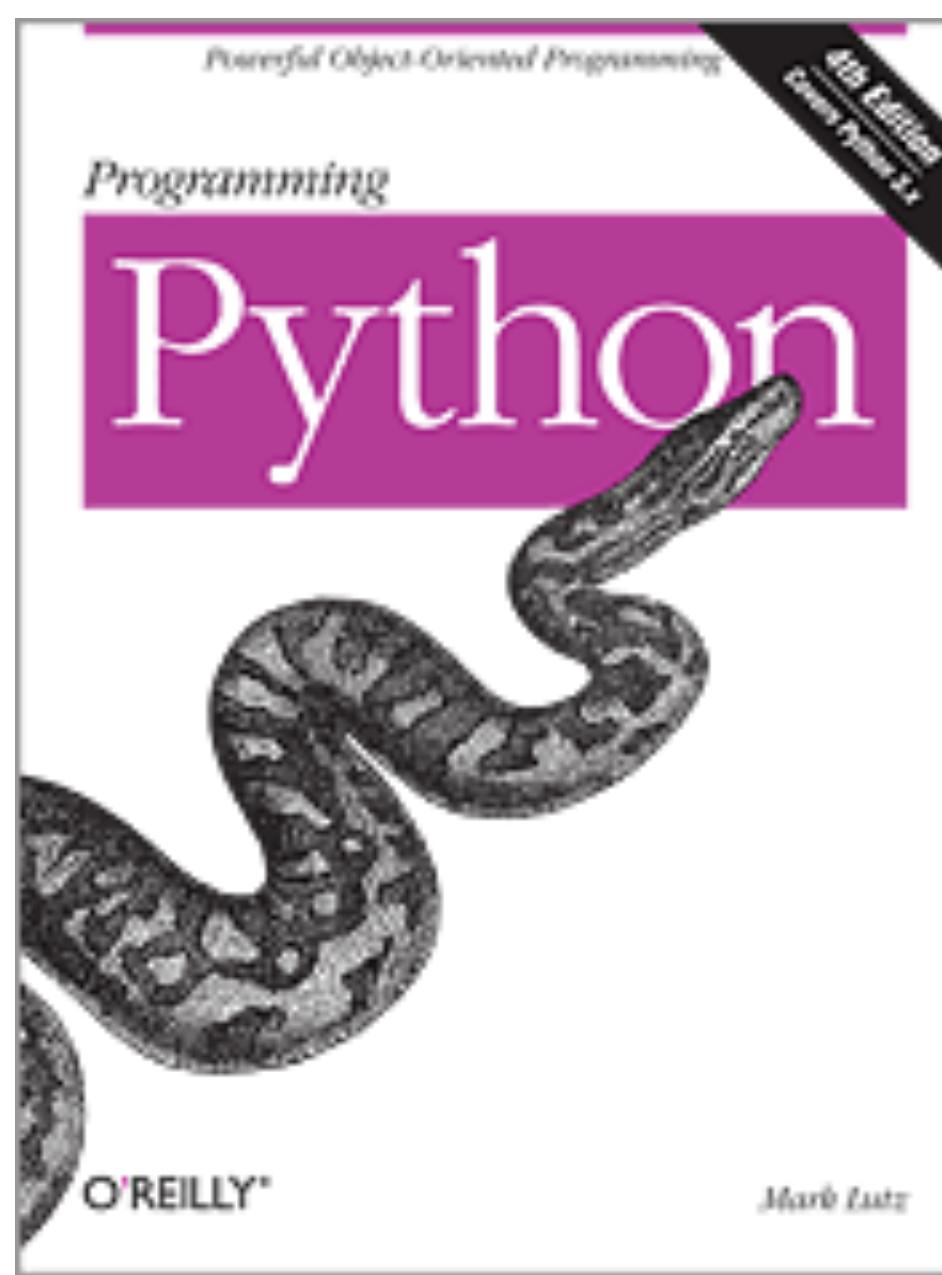
12.4k • 3 • 33 • 69

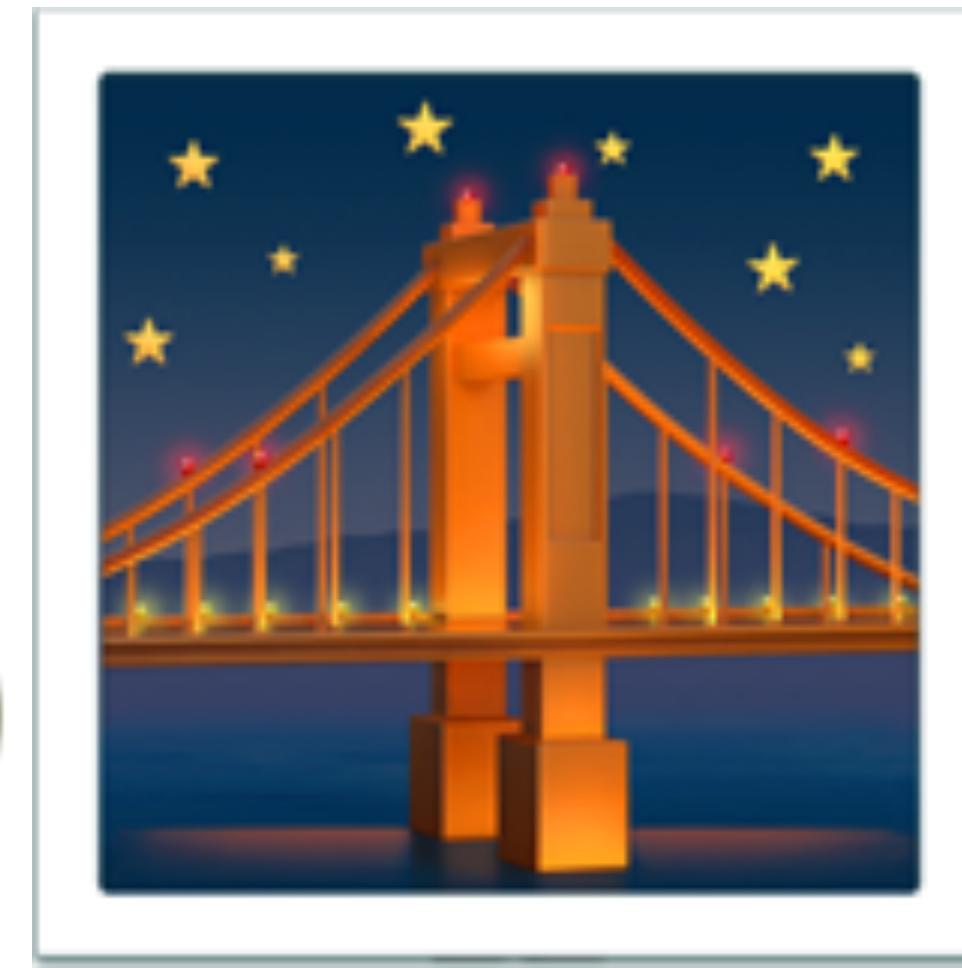
django python

The first PyObjC tutorial you link to is an article by Jonathan Rentzsch (you can hear his voice in the videos, and you may recognize it from the C4 videos). Even though it's on Apple's site without a by-line, it's not "Apple

Backend Engineer

Farmers Insurance • Los Angeles, CA

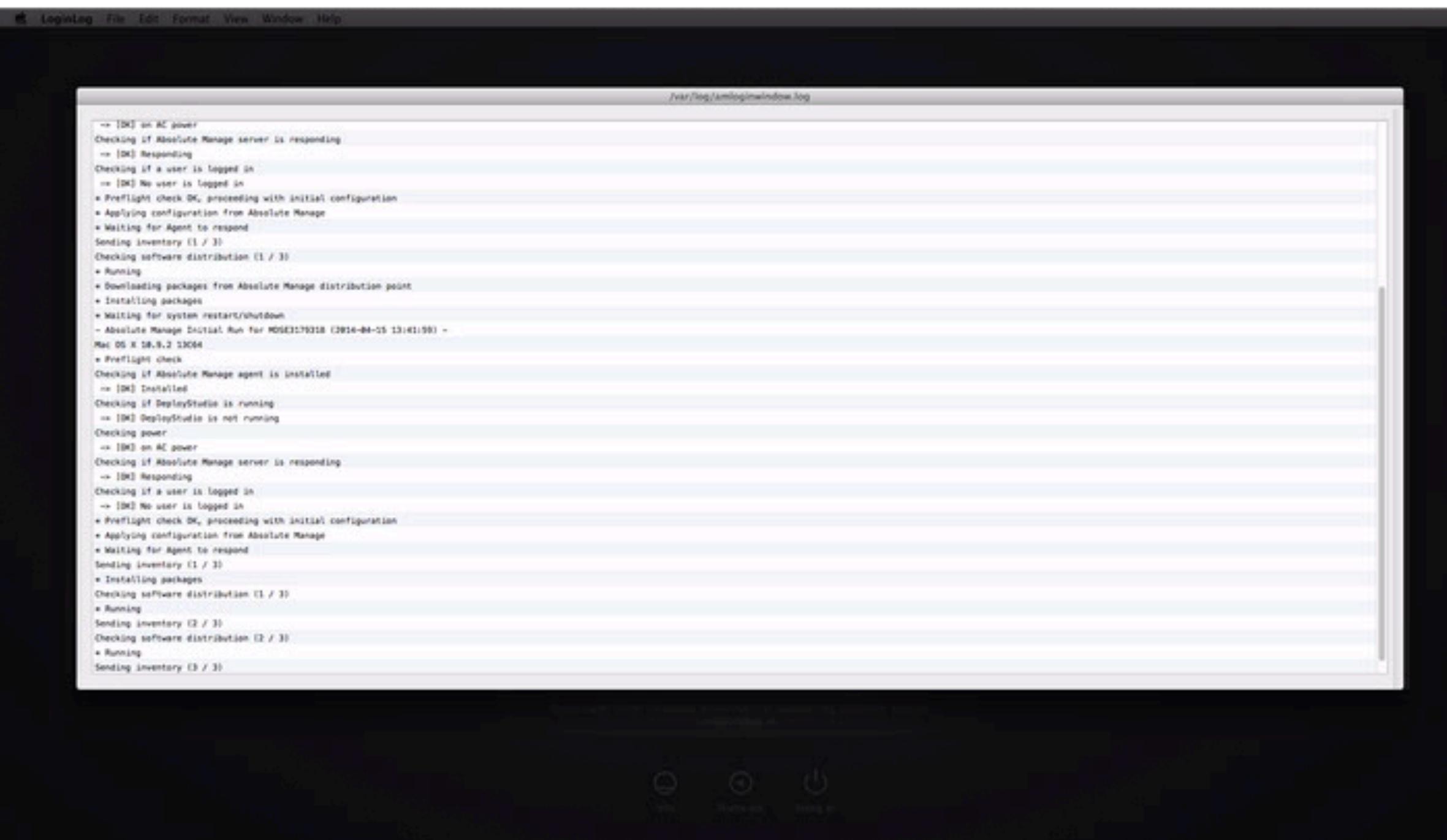




LoginLog

<https://github.com/MagerValp/LoginLog>

Dim the login window and display a log file of your choice over it.



How to use it

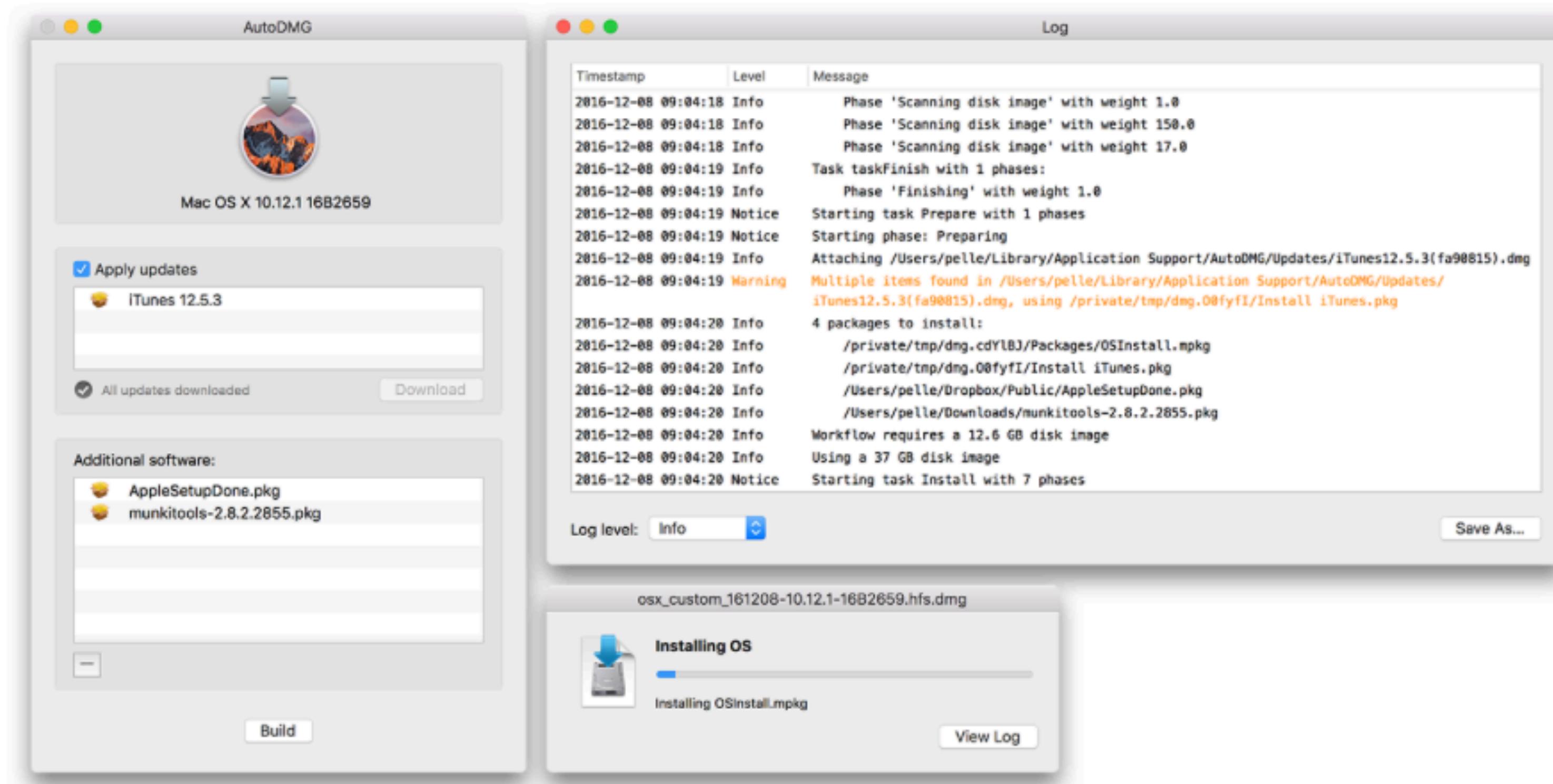
- Build LoginLog.app.
- Copy it to a test machine as `/Library/PrivilegedHelperTools/LoginLog.app`
- Configure the logfile argument in `se.gu.it.LoginLog.plist` and copy it to `/Library/LaunchAgents` on the test machine.
- `launchctl load -S loginwindow /Library/LaunchAgents/se.gu.it.LoginLog.plist`
- `launchctl unload -S loginwindow /Library/LaunchAgents/se.gu.it.LoginLog.plist`

Credits

AutoDMG

<https://github.com/MagerValp/AutoDMG>

The award winning AutoDMG takes a macOS installer (10.9 or newer) and builds a system image suitable for deployment with Imagr, DeployStudio, LANrev, Jamf Pro, and other asr-based imaging tools.



Documentation

Documentation and help is in the [AutoDMG wiki](#).

Presentation

For a great overview of modular imaging and a demonstration of some of AutoDMG's features, watch Anthony Reimer's presentation from Penn State MacAdmins 2014:

GitHub, Inc. github.com/grahamgilbert/imagr

https://github.com/grahamgilbert/imagr

grahamgilbert / imagr

Code Issues Pull requests Projects Wiki Pulse Graphs

http://imagr.io

455 commits 1 branch 31 releases 20 contributors

Branch: master New pull request Create new file Upload files Find file Clone or download

grahamgilbert committed on GitHub Merge pull request #171 from Oxedeb/master ... Latest commit a3b39a6 11 days ago

.github Update ISSUE_TEMPLATE.md 8 months ago

Imagr.xcodeproj Make sure loginlog shows all the time 4 months ago

Imagr Fixed call to urlopen 12 days ago

img update wiki images for NBICreator 8 months ago

.gitignore Ignore rc.netboot 8 months ago

LICENSE Resolves issue #20 2 years ago

Makefile Show old error if scripts don't output anything 2 months ago

Readme.md Potential fix for #119 a year ago

get_locale Add back omitted localize.sh and get_locale. 9 months ago

validateplist Movable restart actions 2 months ago

Readme.md





Software



Categories



My Items



Updates

Search

Managed Software Center

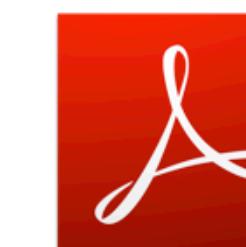
Your source for software for your Mac.



All items



Adobe Acrobat Reader DC
Uncategorized

[INSTALL](#)

Adobe Reader
Productivity - Adobe

[INSTALL](#)

Aspera Connect
Utilities - Aspera

[INSTALL](#)

Atom
Text Editors - GitHub, Inc.

[INSTALL](#)

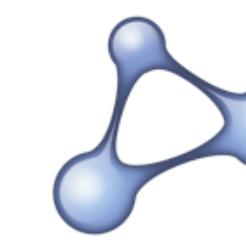
Bluejeans Application
Communications - Bluejeans

[INSTALL](#)

Box Sync
Cloud Storage - Box

[INSTALL](#)

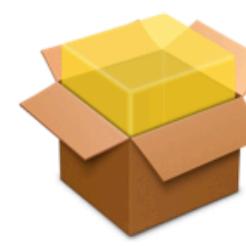
bugreport-2.0.5.0
Utilities - Disney Animation

[INSTALL](#)

cineSync
Uncategorized

[INSTALL](#)

Citrix Receiver
Virtualization - Citrix Systems, Inc.

[INSTALL](#)

Command Line Tools (macOS Si...
Developer - Apple
Installed

[REMOVE](#)

Docker for Mac
Virtualization - Docker Inc
Installed

[REMOVE](#)

EasyFind
Utilities - DEVONtechnologies

[INSTALL](#)

GLC_Player
Graphics - glc-player.net

[INSTALL](#)

Google Chrome
Internet - Google
Installed

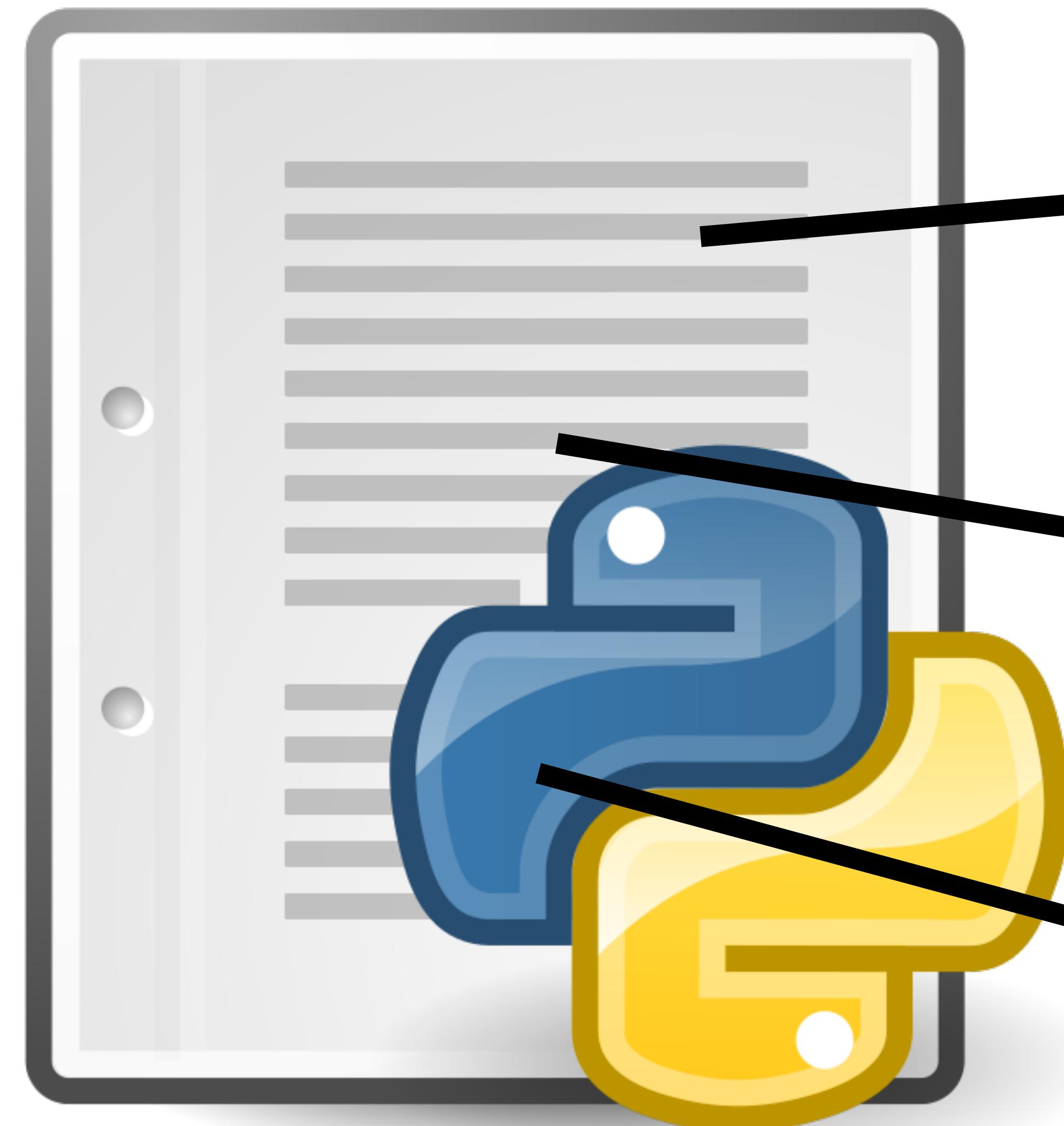
[REMOVE](#)

Google Earth
Internet - Google

[INSTALL](#)

Quick Links

[Help](#)[Support](#)[Contact Tech Support](#)[All Categories](#)[What's new in OS X](#)[Apple](#)[Disney Animation Web](#)[Change your passwords](#)



Foundation

AppKit

Quartz



Foundation

AppKit

Quartz

Basic conversion rules

aka I hope you like underscores

Cocoa

```
[someObject doSomething: arg1 withAdditionalData: arg2];
```

Cocoa

```
[someObject doSomething: arg1 withAdditionalData: arg2];
```

Cocoa

```
[someObject doSomething: arg1. withAdditionalData: arg2];
```

Cocoa

```
[someObject doSomething: arg1 withAdditionalData: arg2];
```

Python+PyObjC

```
someObject.
```

Cocoa

```
[someObject doSomething: arg1 withAdditionalData: arg2];
```

Python+PyObjC

```
someObject.doSomething_withAdditionalData_( )
```

Cocoa

```
[someObject doSomething: arg1 withAdditionalData: arg2];
```

Python+PyObjC

```
someObject.doSomething_withAdditionalData_(arg1, arg2)
```

Cocoa

```
[sharedWorkspace openFile:@"/Users/Shared/README" withApplication:@"TextEdit"];
```

Cocoa

```
[sharedWorkspace openFile:@"/Users/Shared/README" withApplication:@"TextEdit"] ;
```

Cocoa

```
[sharedWorkspace openFile:@"/Users/Shared/README" withApplication:@"TextEdit"];
```

Python+PyObjC

```
sharedWorkspace.openFile_withApplication_( )
```

Cocoa

```
[sharedWorkspace openFile:@"/Users/Shared/README" withApplication:@"TextEdit"];
```

Python+PyObjC

```
sharedWorkspace.openFile_withApplication_( "/Users/Shared/README", "TextEdit" )
```

Class

NSWorkspace

An `NSWorkspace` object can launch other apps and perform a variety of file-handling services.

Language

Swift | Objective-C

SDK

macOS 10.0+

Overview

There is one shared `NSWorkspace` object per app. You use the class method [sharedWorkspace](#) to access it. For example, the following statement uses an `NSWorkspace` object to request that a file be opened in the `TextEdit` app:

```
[[NSWorkspace sharedWorkspace] openFile:@"/Myfiles/README"  
withApplication:@"TextEdit"];
```

On This Page

[Overview](#)[Symbols](#)[Relationships](#)

You can use the workspace object to:

- Open, manipulate, and get information about files and devices
- Track changes to the file system, devices, and the user database
- Get and set Finder information for files
- Launch apps

```
[[NSWorkspace sharedWorkspace] openFile:@"/Myfiles/README" withApplication:@"TextEdit"];
```

```
[ [NSWorkspace sharedWorkspace] openFile:@"~/Myfiles/README" withApplication:@"TextEdit" ] ;
```

```
[ [NSWorkspace sharedWorkspace] openFile:@"~/Myfiles/README" withApplication:@"TextEdit" ];
```

```
[NSWorkspace sharedWorkspace]
```

```
[ [NSWorkspace sharedWorkspace] openFile:@"~/Myfiles/README" withApplication:@"TextEdit"];
```

```
NSWorkspace.sharedWorkspace()
```

```
[ [NSWorkspace sharedWorkspace] openFile:@"~/Myfiles/README" withApplication:@"TextEdit"];
```

```
NSWorkspace.sharedWorkspace().openFile_withApplication(
```

```
[ [NSWorkspace sharedWorkspace] openFile:@"~/Myfiles/README" withApplication:@"TextEdit"];  
  
NSWorkspace.sharedWorkspace().openFile_withApplication_("~/Myfiles/README", "TextEdit")
```

Cocoa

```
[sharedWorkspace launchApplication:@"TextEdit"];
```

Cocoa

```
[sharedWorkspace launchApplication:@"TextEdit"];
```

Python+PyObjC

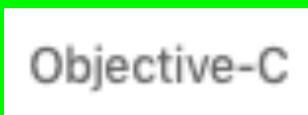
```
sharedWorkspace.launchApplication_("TextEdit")
```

Another example

Type Method

fileURLWithPath:

Initializes and returns a newly created NSURL object as a file URL with a specified path.

Language
 

SDKs

iOS 2.0+

macOS 10.0+

tvOS 9.0+

watchOS 2.0+

On This Page

[Declaration](#) 

[Parameters](#) 

[Return Value](#) 

[Discussion](#) 

[See Also](#) 

Declaration

```
+ (NSURL *)fileURLWithPath:(NSString *)path;
```

Parameters

path

The path that the NSURL object will represent. path should be a valid system path, and must not be an empty path. If path begins with a tilde, it must first be expanded with [stringByExpandingTildeInPath](#). If path is a relative path, it is treated as being relative to the current working directory.

Return Value

An NSURL object initialized with path.

<https://developer.apple.com/reference/foundation/nsurl?language=objc>

Type Method

fileURLWithPath:

Initializes and returns a newly created NSURL object as a file URL with a specified path.

Language

Swift | Objective-C

SDKs

iOS 2.0+

macOS 10.0+

tvOS 9.0+

watchOS 2.0+

On This Page

Declaration ↴

Parameters ↴

Return Value ↴

Discussion ↴

See Also ↴

Declaration

```
+ (NSURL *)fileURLWithPath:(NSString *)path;
```

Parameters

path

The path that the NSURL object will represent. path should be a valid system path, and must not be an empty path. If path begins with a tilde, it must first be expanded with [stringByExpandingTildeInPath](#). If path is a relative path, it is treated as being relative to the current working directory.

Return Value

An NSURL object initialized with path.

<https://developer.apple.com/reference/foundation/nsurl?language=objc>

Type Method

fileURLWithPath:

Initializes and returns a newly created NSURL object as a file URL with a specified path.

Language

Swift | Objective-C

SDKs

iOS 2.0+

macOS 10.0+

tvOS 9.0+

watchOS 2.0+

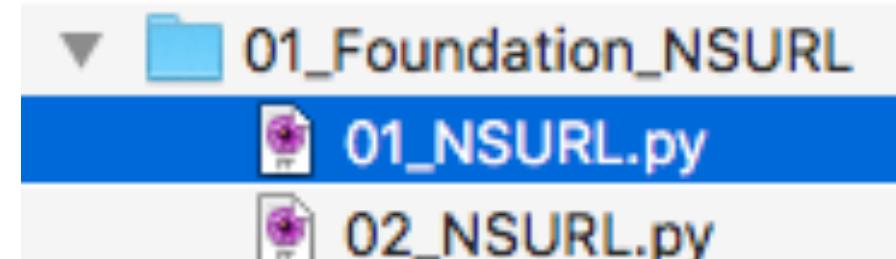
On This Page

Declaration 

Parameters 

```
+ (NSURL *)fileURLWithPath:(NSString *)path;
```

```
from Foundation import NSURL
```



Type Method

fileURLWithPath:

Initializes and returns a newly created NSURL object as a file URL with a specified path.

Language

Swift | Objective-C

SDKs

iOS 2.0+

macOS 10.0+

tvOS 9.0+

watchOS 2.0+

On This Page

Declaration ⓘ

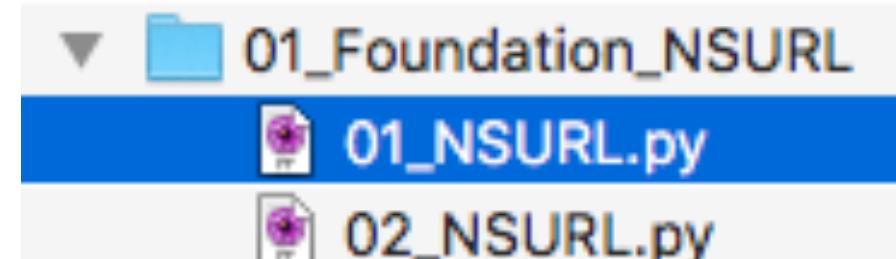
Parameters ⓘ

Declaration

```
+ (NSURL *)fileURLWithPath:(NSString *)path;
```

```
from Foundation import NSURL

print NSURL.fileURLWithPath_('~/Users/Shared/foo')
```



Type Method

fileURLWithPath:

Initializes and returns a newly created NSURL object as a file URL with a specified path.

Language

Swift | Objective-C

SDKs

iOS 2.0+

macOS 10.0+

tvOS 9.0+

watchOS 2.0+

On This Page

Declaration ⓘ

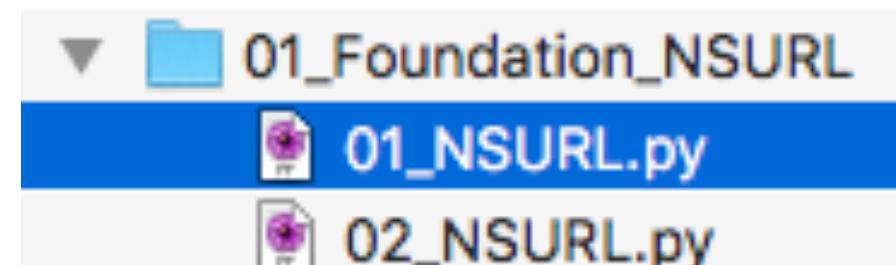
Parameters ⓘ

Declaration

```
+ (NSURL *)fileURLWithPath:(NSString *)path;
```

```
from Foundation import NSURL  
  
print NSURL.fileURLWithPath_('~/Users/Shared/foo')
```

```
file:///Users/Shared/foo
```



Type Method

fileURLWithPath:isDirectory:

Initializes and returns a newly created NSURL object as a file URL with a specified path.

Language

[Swift](#) | [Objective-C](#)

SDKs

iOS 2.0+

macOS 10.5+

tvOS 2.0+

watchOS 2.0+

On This Page

[Declaration](#) ⓘ

[Parameters](#) ⓘ

[Return Value](#) ⓘ

[See Also](#) ⓘ

Declaration

```
+ (NSURL *)fileURLWithPath:(NSString *)path  
                      isDirectory:(BOOL)isDir;
```

Parameters

path

The path that the NSURL object will represent. path should be a valid system path, and must not be an empty path. If path begins with a tilde, it must first be expanded with [stringByExpandingTildeInPath](#). If path is a relative path, it is treated as being relative to the current working directory.

isDir

A Boolean value that specifies whether path is treated as a directory path when resolving against relative path components. Pass YES if the path indicates a directory, NO otherwise.

Type Method

fileURLWithPath:isDirectory:

Initializes and returns a newly created NSURL object as a file URL with a specified path.

Language

Swift | Objective-C

SDKs

iOS 2.0+

macOS 10.5+

tvOS 2.0+

watchOS 2.0+

On This Page

Declaration ⓘ

Parameters ⓘ

Return Value ⓘ

Declaration

```
+ (NSURL *)fileURLWithPath:(NSString *)path  
                      isDirectory:(BOOL)isDir;
```

```
from Foundation import NSURL
```

Type Method

fileURLWithPath:isDirectory:

Initializes and returns a newly created NSURL object as a file URL with a specified path.

Language

Swift | Objective-C

SDKs

iOS 2.0+

macOS 10.5+

tvOS 2.0+

watchOS 2.0+

On This Page

Declaration ⓘ

Parameters ⓘ

Return Value ⓘ

Declaration

```
+ (NSURL *)fileURLWithPath:(NSString *)path  
                      isDirectory:(BOOL)isDir;
```

```
from Foundation import NSURL
```

```
print NSURL.fileURLWithPath_isDirectory_('~/Users/Shared/foo', True)
```

Type Method

fileURLWithPath:isDirectory:

Initializes and returns a newly created NSURL object as a file URL with a specified path.

Language

Swift | Objective-C

SDKs

iOS 2.0+

macOS 10.5+

tvOS 2.0+

watchOS 2.0+

On This Page

Declaration ⓘ

Parameters ⓘ

Return Value ⓘ

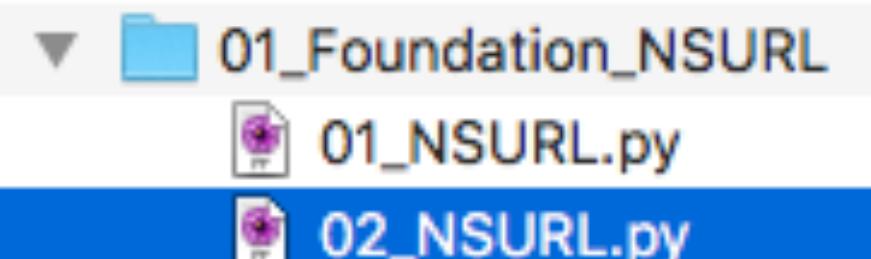
Declaration

```
+ (NSURL *)fileURLWithPath:(NSString *)path  
                      isDirectory:(BOOL)isDir;
```

```
from Foundation import NSURL
```

```
print NSURL.fileURLWithPath_isDirectory_('~/Users/Shared/foo', True)
```

```
file:///Users/Shared/foo/
```



Advanced conversion rules

out and in/out parameters

Technical Q&A QA1133

Determining console user login status

Q: How do I tell whether a console user is currently logged in?

A: How do I tell whether a console user is currently logged in?

It really depends on what you mean by "currently logged in". With the introduction of fast user switching in Mac OS X 10.3, multiple users can be logged into the system at the same time. So, you have to split the question in two:

1. How do I get a list of GUI login sessions?
2. Which GUI login session, if any, is currently using the console?

These are clear questions with clear answers (see below). However, it's likely that, if you're asking these questions, you need to rethink your architecture.

One of the fundamental design principles of the Mac OS X is that high-level services can depend on low-level services, but not the other way around. For example, it's fine for an application to depend on the kernel, but it's a problem if the kernel depends on an application.

The question of determining the current console user typically arises when you break this design principle. Only application-level services should care whether a user's session is currently active on the console, and it's easy for such services to [determine that](#). On the other hand, a daemon should not be concerned about whether a user is logged in on the console, and thus it's tricky to get this information from that context.

This issue is discussed in great depth in [Technical Note TN2083, 'Daemons and Agents'](#) and, specifically, in the "Design Considerations" section of that technote. So, before you read the rest of this document, I strongly recommend that you read that technote and think about how you can align your architecture with the overall Mac OS X architecture.

https://developer.apple.com/library/content/qa/qa1133/_index.html

Determining console user login status

'UTXplorer'.

Prior to Mac OS X 10.5 there was no supported way to get this information directly. [Technical Note TN2083, 'Daemons and Agents'](#) suggests an indirect way to do it, based a central daemon with a set of cooperating agents. This design will work on all versions of Mac OS X and is generally aligned with the overall Mac OS X design principle described earlier.

2. Which GUI login session, if any, is currently using the console? — This is answered below.

You can get the user name of the current console user by calling the `SCDynamicStoreCopyConsoleUser` function (from the System Configuration framework). Listing 1 shows how to do this.

Listing 1: Getting the current console user

```
static CFStringRef CopyCurrentConsoleUsername(SCDynamicStoreRef store)
    // Returns the name of the current console user, or NULL if there is
    // none. store may be NULL, in which case a transient dynamic store
    // session is used.
{
    CFStringRef result;

    result = SCDynamicStoreCopyConsoleUser(store, NULL, NULL);

    // If the current console user is "loginwindow", treat that as equivalent
    // to none.

    if ( (result != NULL) && CFEqual(result, CFSTR("loginwindow")) ) {
        CFRelease(result);
        result = NULL;
    }

    return result;
}
```

The System Configuration framework also makes it easy to be notified when this setting changes. Listing 2 shows how to do that.

https://developer.apple.com/library/content/qa/qa1133/_index.html

Listing 2: Discovering when that setting changes

Determining console user login status

'UTXplorer'.

Prior to Mac OS X 10.5 there was no supported way to get this information directly. [Technical Note TN2083, 'Daemons and Agents'](#) suggests an indirect way to do it, based a central daemon with a set of cooperating agents. This design will work on all versions of Mac OS X and is generally aligned with the overall Mac OS X design principle described earlier.

2. Which GUI login session, if any, is currently using the console? — This is answered below.

You can get the user name of the current console user by calling the `SCDynamicStoreCopyConsoleUser` function (from the System Configuration framework). Listing 1 shows how to do this.

Listing 1: Getting the current console user

```
static CFStringRef CopyCurrentConsoleUsername(SCDynamicStoreRef store)
    // Returns the name of the current console user, or NULL if there is
    // none. store may be NULL, in which case a transient dynamic store
    // session is used.
{
    CFStringRef result;

    result = SCDynamicStoreCopyConsoleUser(store, NULL, NULL);

    // If the current console user is "loginwindow", treat that as equivalent
    // to none.

    if ( (result != NULL) && CFEqual(result, CFSTR("loginwindow")) ) {
        CFRelease(result);
        result = NULL;
    }

    return result;
}
```

The System Configuration framework also makes it easy to be notified when this setting changes. Listing 2 shows how to do that.

Listing 2: Discovering when that setting changes

```
static CFStringRef CopyCurrentConsoleUsername(SCDynamicStoreRef store)
    // Returns the name of the current console user, or NULL if there is
    // none. store may be NULL, in which case a transient dynamic store
    // session is used.
{
    CFStringRef result;

    result = SCDynamicStoreCopyConsoleUser(store, NULL, NULL);

    // If the current console user is "loginwindow", treat that as equivalent
    // to none.

    if ( (result != NULL) && CFEqual(result, CFSTR("loginwindow")) ) {
        CFRelease(result);
        result = NULL;
    }

    return result;
}
```

Function

SCDynamicStoreCopyConsoleUser

Returns information about the user currently logged into the system.

Language

Swift | Objective-C

SDK

macOS 10.1+

On This Page

[Declaration](#) ⓘ

[Parameters](#) ⓘ

[Return Value](#) ⓘ

[Discussion](#) ⓘ

Declaration

```
CFStringRef SCDynamicStoreCopyConsoleUser(SCDynamicStoreRef store, uid_t
```

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

<https://developer.apple.com/reference/systemconfiguration/1517123-scdynamicstorecopyconsoleuser?language=objc>
If NULL, this value is not returned.

SDK

macOS 10.1+

On This Page

[Declaration](#) ⓘ

[Parameters](#) ⓘ

[Return Value](#) ⓘ

[Discussion](#) ⓘ

Declaration

```
CFStringRef SCDynamicStoreCopyConsoleUser(SCDynamicStoreRef store, uid_t
```

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

Returns the name, user ID, and group ID of the user currently logged into the system, or NULL if no user is logged in or if an error occurred. You must release the returned values.

SDK

macOS 10.1+

[On This Page](#)[Declaration](#)[Parameters](#)[Return Value](#)[Discussion](#)

Declaration

```
CFStringRef SCDynamicStoreCopyConsoleUser(SCDynamicStoreRef store, uid_t
```

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

Returns the name, user ID, and group ID of the user currently logged into the system, or `NULL` if no user is logged in or if an error occurred. You must release the returned values.

SDK

macOS 10.1+

[On This Page](#)[Declaration](#)[Parameters](#)[Return Value](#)[Discussion](#)

Declaration

```
CFStringRef SCDynamicStoreCopyConsoleUser(SCDynamicStoreRef store, uid_t
```

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

Returns the name, user ID, and group ID of the user currently logged into the system, or NULL if no user is logged in or if an error occurred. You must release the returned values.

SDK

macOS 10.1+

On This Page

[Declaration](#)[Parameters](#)[Return Value](#)[Discussion](#)

Declaration

```
CFStringRef SCDynamicStoreCopyConsoleUser(SCDynamicStoreRef store, uid_t
```

```
from SystemConfiguration import SCDynamicStoreCopyConsoleUser
```

SDK

macOS 10.1+

On This Page

[Declaration](#)[Parameters](#)[Return Value](#)[Discussion](#)

Declaration

```
CFStringRef SCDynamicStoreCopyConsoleUser(SCDynamicStoreRef store, uid_t
```

```
from SystemConfiguration import SCDynamicStoreCopyConsoleUser  
  
print SCDynamicStoreCopyConsoleUser(None, None, None)
```

SDK

macOS 10.1+

On This Page

[Declaration](#)[Parameters](#)[Return Value](#)[Discussion](#)

Declaration

```
CFStringRef SCDynamicStoreCopyConsoleUser(SCDynamicStoreRef store, uid_t
```

```
from SystemConfiguration import SCDynamicStoreCopyConsoleUser  
  
print SCDynamicStoreCopyConsoleUser(None, None, None)  
  
(u'gneagle', 1234, 5678)
```

SDK

macOS 10.1+

[On This Page](#)[Declaration](#)[Parameters](#)[Return Value](#)[Discussion](#)

Declaration

```
CFStringRef SCDynamicStoreCopyConsoleUser(SCDynamicStoreRef store, uid_t
```

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

Returns the name, user ID, and group ID of the user currently logged into the system, or NULL if no user is logged in or if an error occurred. You must release the returned values.

SDK

macOS 10.1+

Declaration

```
CFStringRef SCDynamicStoreCopyConsoleUser(SCDynamicStoreRef store, uid_t
```

[On This Page](#)[Declaration](#)[Parameters](#)[Return Value](#)[Discussion](#)

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

Returns the name, user ID, and group ID of the user currently logged into the system, or NULL if no user is logged in or if an error occurred. You must release the returned values.

SDK

macOS 10.1+

Declaration

```
CFStringRef SCDynamicStoreCopyConsoleUser(SCDynamicStoreRef store, uid_t
```

[On This Page](#)[Declaration](#)[Parameters](#)[Return Value](#)[Discussion](#)

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

Returns the name, user ID, and group ID of the user currently logged into the system, or NULL if no user is logged in or if an error occurred. You must release the returned values.

SDK

macOS 10.1+

Declaration

```
CFStringRef SCDynamicStoreCopyConsoleUser(SCDynamicStoreRef store, uid_t
```

[On This Page](#)[Declaration](#)[Parameters](#)[Return Value](#)[Discussion](#)

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

Returns the name, user ID, and group ID of the user currently logged into the system, or NULL if no user is logged in or if an error occurred. You must release the returned values.

C/Objective-C

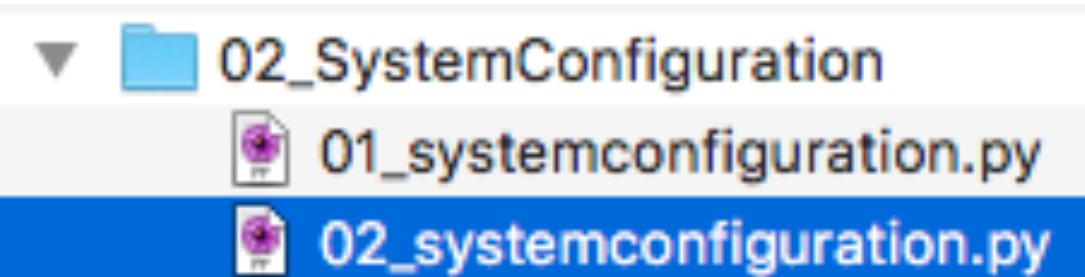
```
name = SCDynamicStoreCopyConsoleUser(store, &uid, &gid);
```

Python+PyObjC

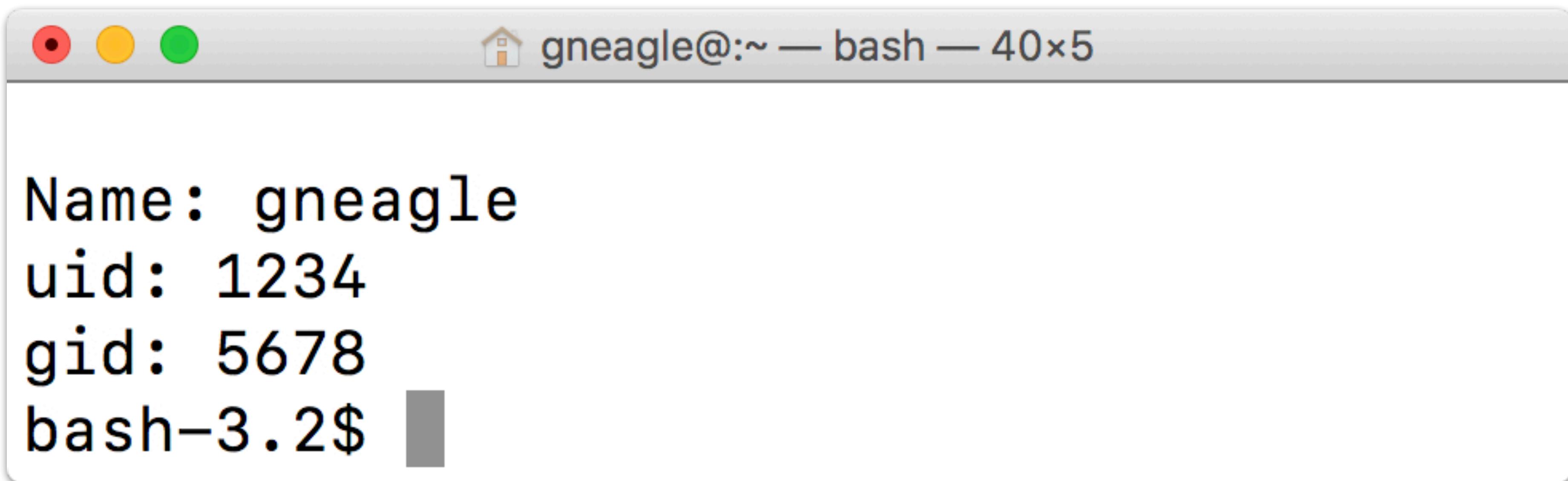
```
name, uid, gid = SCDynamicStoreCopyConsoleUser(store, None, None);
```

```
(u'gneagle', 1234, 5678)
```

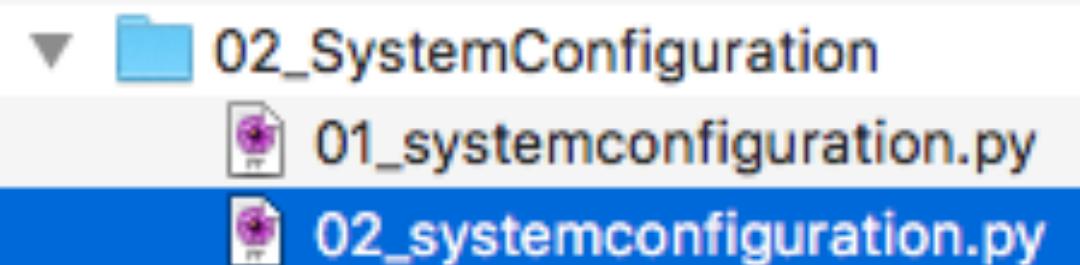
```
from SystemConfiguration import SCDynamicStoreCopyConsoleUser  
  
name, uid, gid = SCDynamicStoreCopyConsoleUser(None, None, None)  
  
print 'Name: %s' % name  
print 'uid: %s' % uid  
print 'gid: %s' % gid
```



```
from SystemConfiguration import SCDynamicStoreCopyConsoleUser  
  
name, uid, gid = SCDynamicStoreCopyConsoleUser(None, None, None)  
  
print 'Name: %s' % name  
print 'uid: %s' % uid  
print 'gid: %s' % gid
```



```
Name: gneagle  
uid: 1234  
gid: 5678  
bash-3.2$
```



WebKit

CoreFoundation

SystemConfiguration

AppKit

Frameworks

OpenDirectory

Cocoa

Foundation

Quartz

LaunchServices

`/System/Library/Frameworks/Python.framework/Versions/Current/Extras/lib/python/PyObjC`

AVFoundation

Accounts

AddressBook

AppKit

AppleScriptKit

AppleScriptObjC

Automator

CFNetwork

CFOpenDirectory

Cocoa

Collaboration

CoreData

CoreFoundation

CoreLocation

CoreText

DictionaryServices

EventKit

ExceptionHandling

FSEvents

Foundation

InputMethodKit

InstallerPlugins

InstantMessage

JavaScriptCore

LatentSemanticMapping

LaunchServices

Message

OpenDirectory

PreferencePanes

PubSub

PyObjCTools

QTKit

Quartz

ScreenSaver

ScriptingBridge

SearchKit

ServiceManagement

Social

SyncServices

SystemConfiguration

WebKit

Foundation types

and their Python mappings

Foundation type	Python type
NSNumber	int, float, long
NSString	unicode, str
NSDictionary/NSMutableDictionary	dict
NSArray/NSMutableArray	list, tuple
NSData	buffer
NULL	None

Useful(?) examples

Foundation

Logging Output

[NSLog](#)

Logs an error message to the Apple System Log facility.

[NSLogv](#)

Logs an error message to the Apple System Log facility.

Managing File Paths

[NSFullUserName](#)

Returns a string containing the full name of the current user.

[NSHomeDirectory](#)

Returns the path to either the user's or application's home directory, depending on the platform.

[NSHomeDirectoryForUser](#)

Returns the path to a given user's home directory.

[NSOpenStepRootDirectory](#)

Returns the root directory of the user's system.

[NSSearchPathForDirectoriesInDomains](#)

Creates a list of directory search paths.

[NSTemporaryDirectory](#)

Returns the path of the temporary directory for the current user.

[NSUserName](#)

Return the logon name of the current user.

[https://developer.apple.com/reference/foundation/1613024-foundation_functions?
language=objc](https://developer.apple.com/reference/foundation/1613024-foundation_functions?language=objc)

```
import Foundation

print Foundation.NSUserName()
print Foundation.NSFullUserName()
print Foundation.NSHomeDirectory()
print Foundation.NSHomeDirectoryForUser('root')
print Foundation.NSTemporaryDirectory()
```

[https://developer.apple.com/reference/foundation/1613024-foundation_functions?
language=objc](https://developer.apple.com/reference/foundation/1613024-foundation_functions?language=objc)

```
import Foundation
```

```
print Foundation.NSUserName()
print Foundation.NSFullUserName()
print Foundation.NSHomeDirectory()
print Foundation.NSHomeDirectoryForUser('root')
print Foundation.NSTemporaryDirectory()
```

```
gneagle
Greg Neagle
/Users/gneagle
/var/root
/var/folders/tc/sd4_mtvj14jdy7cg21m2gmcw000495/T/
```



CoreFoundation

(CPreferences)

Symbols

Getting Preference Values

[CFPreferencesCopyAppValue](#)

Obtains a preference value for the specified key and application.

[CFPreferencesCopyKeyList](#)

Constructs and returns the list of all keys set in the specified domain.

[CFPreferencesCopyMultiple](#)

Returns a dictionary containing preference values for multiple keys.

[CFPreferencesCopyValue](#)

Returns a preference value for a given domain.

[CFPreferencesGetAppBooleanValue](#)

Convenience function that directly obtains a boolean preference value for the specified key.

[CFPreferencesGetAppIntegerValue](#)

Convenience function that directly obtains an integer preference value for the specified key.

Setting Preference Values

[CFPreferencesSetAppValue](#)

Adds, modifies, or removes a preference.

[CFPreferencesSetMultiple](#)

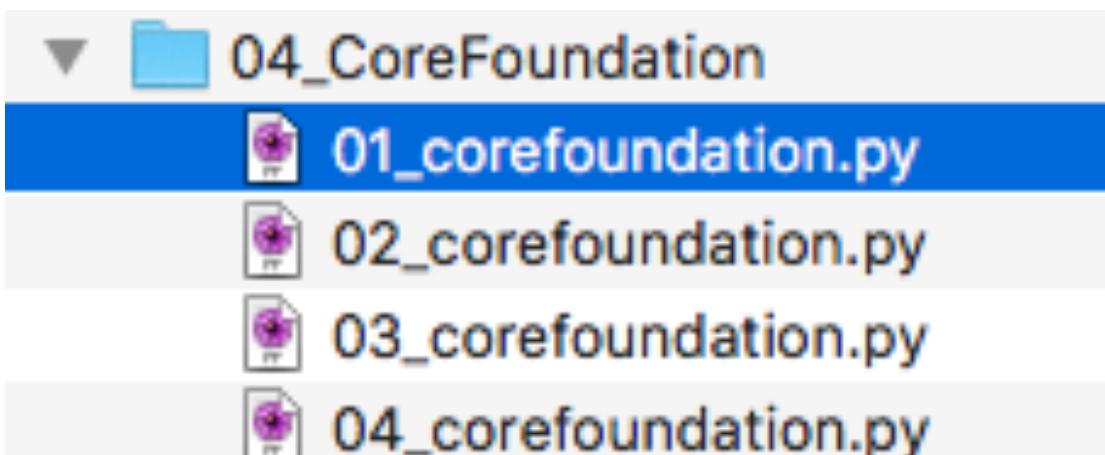
Convenience function that allows you to set and remove multiple preference values.

https://developer.apple.com/reference/corefoundation/1666648-preferences_utilities?language=objc

```
import CoreFoundation

# similar to `defaults read com.apple.Finder ShowHardDrivesOnDesktop`
print CoreFoundation.CFPreferencesCopyAppValue("ShowHardDrivesOnDesktop", "com.apple.Finder")
```

True



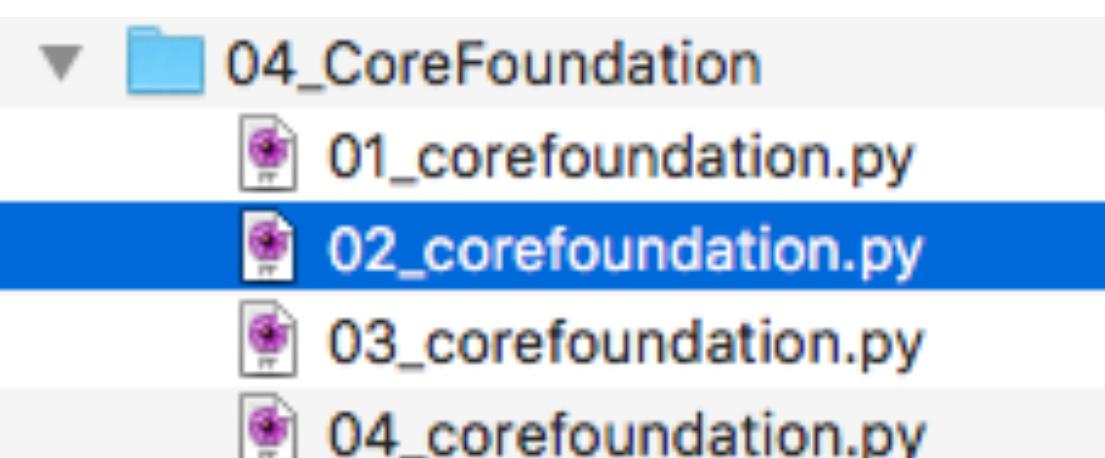
```
defaults read com.apple.Finder ShowHardDrivesOnDesktop
```

1

```
import subprocess
import CoreFoundation

# get the value using defaults
value = subprocess.check_output(
    ['/usr/bin/defaults', 'read', 'com.apple.Finder', 'ShowHardDrivesOnDesktop']
)
print 'Value is %s and is type %s' % (value, type(value))

# get the value using CFPrefrences
value = CoreFoundation.CFPreferencesCopyAppValue('ShowHardDrivesOnDesktop', 'com.apple.Finder')
print 'Value is %s and is type %s' % (value, type(value))
```

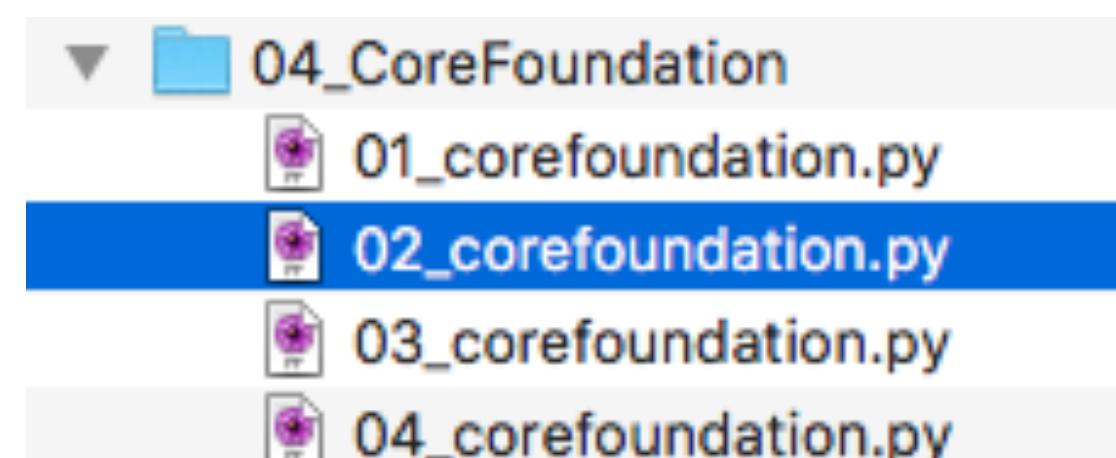


```
import subprocess
import CoreFoundation

# get the value using defaults
value = subprocess.check_output(
    ['/usr/bin/defaults', 'read', 'com.apple.Finder', 'ShowHardDrivesOnDesktop']
)
print 'Value is %s and is type %s' % (value, type(value))
```

```
# get the value using CFPrefences
value = CoreFoundation.CFPrefsCopyAppValue('ShowHardDrivesOnDesktop', 'com.apple.Finder')
print 'Value is %s and is type %s' % (value, type(value))
```

```
Value is 1
and is type <type 'str'>
Value is True and is type <type 'bool'>
```

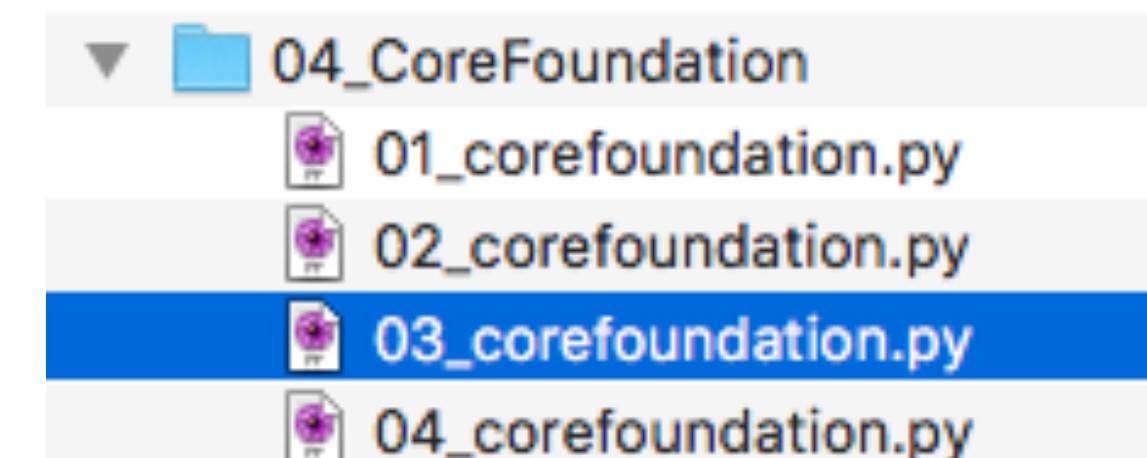


```
import subprocess
import CoreFoundation

# get the value using defaults
print 'Running defaults...'
value = subprocess.check_output(
    ['/usr/bin/defaults', 'read', 'com.apple.Finder', 'StandardViewSettings']
)
print 'Type returned from defaults is %s' % type(value)
print value
```

```
# get the value using CFPrefrences
print 'Calling CFPrefrencesCopyAppValue...'
value = CoreFoundation.CFPrefrencesCopyAppValue('StandardViewSettings', 'com.apple.Finder')
print 'Type returned from CFPrefrencesCopyAppValue is %s' % type(value)

print ("StandardViewSettings['ExtendedListViewSettings']['calculateAllSizes'] " "is: %s" % value['ExtendedListViewSettings']['calculateAllSizes'])
```

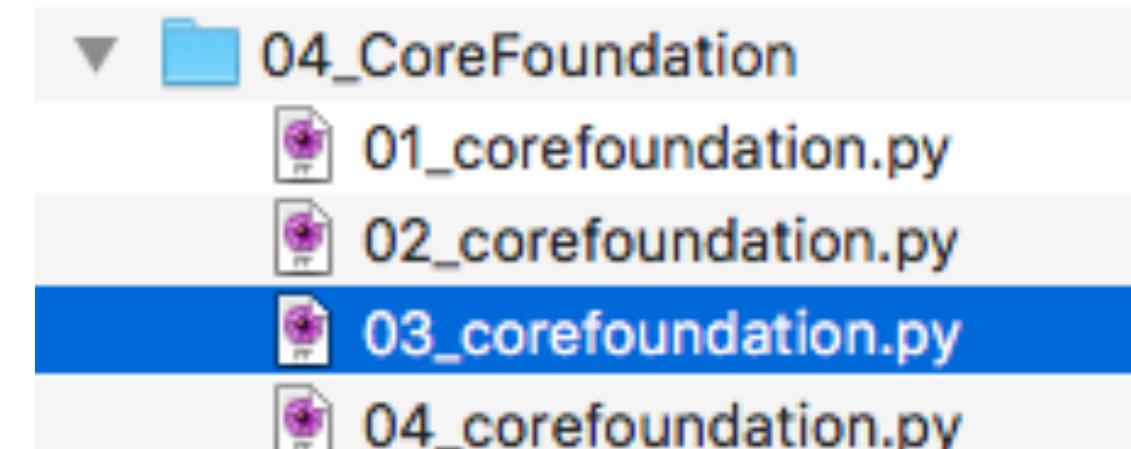


```
import subprocess
import CoreFoundation

# get the value using defaults
print 'Running defaults...'
value = subprocess.check_output(
    ['/usr/bin/defaults', 'read', 'com.apple.Finder', 'StandardViewSettings']
)
print 'Type returned from defaults is %s' % type(value)
print value
```

```
# get the value using CFPrefrences
print 'Calling CFPrefrencesCopyAppValue...'
value = CoreFoundation.CFPrefrencesCopyAppValue('StandardViewSettings', 'com.apple.Finder')
print 'Type returned from CFPrefrencesCopyAppValue is %s' % type(value)
```

```
print ("StandardViewSettings['ExtendedListViewSettings']['calculateAllSizes'] " "is: %s" % value['ExtendedListViewSettings']['calculateAllSizes'])
```

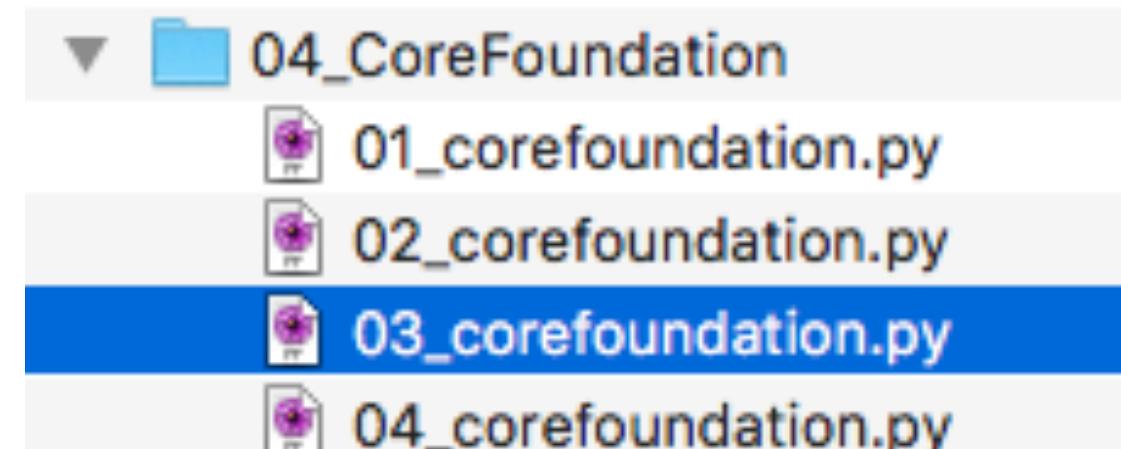


```
import subprocess
import CoreFoundation

# get the value using defaults
print 'Running defaults...'
value = subprocess.check_output(
    ['/usr/bin/defaults', 'read', 'com.apple.Finder', 'StandardViewSettings']
)
print 'Type returned from defaults is %s' % type(value)
print value

# get the value using CFPrefences
print 'Calling CFPrefencesCopyAppValue...'
value = CoreFoundation.CFPrefencesCopyAppValue('StandardViewSettings', 'com.apple.Finder')
print 'Type returned from CFPrefencesCopyAppValue is %s' % type(value)

print ("StandardViewSettings['ExtendedListViewSettings']['calculateAllSizes'] " "is: %s" % value['ExtendedListViewSettings']['calculateAllSizes'])
```



Running defaults...

Type returned from defaults is <type 'str'>

```
{  
    ExtendedListViewSettings = {  
        calculateAllSizes = 0;  
        columns = (  
            {  
                ascending = 1;  
                identifier = name;  
                visible = 1;  
                width = 300;  
            },  
            <snip>  
            textSize = 12;  
            useRelativeDates = 1;  
            viewOptionsVersion = 0;  
        );  
        SettingsType = StandardViewSettings;  
    }  
}
```

```
Running defaults...
Type returned from defaults is <type 'str'>
{
    ExtendedListViewSettings =      {
        calculateAllSizes = 0;
        columns =          (
            {
                ascending = 1;
                identifier = name;
                visible = 1;
                width = 300;
            },
<snip>

        textSize = 12;
        useRelativeDates = 1;
        viewOptionsVersion = 0;
    };
    SettingsType = StandardViewSettings;
}
```

Calling CFPreferencesCopyAppValue...
Type returned from CFPreferencesCopyAppValue is <objective-c class __NSCFDictionary at 0x7fff9c3262f0>
StandardViewSettings['ExtendedListViewSettings']['calculateAllSizes'] is: False

```
Running defaults...
```

```
Type returned from defaults is <type 'str'>
```

```
{  
    ExtendedListViewSettings = {  
        calculateAllSizes = 0;  
        columns = (  
            {  
                ascending = 1;  
                identifier = name;  
                visible = 1;  
                width = 300;  
            },  
            {  
                textSize = 12;  
                useRelativeDates = 1;  
                viewOptionsVersion = 0;  
            };  
        SettingsType = StandardViewSettings;  
    }
```

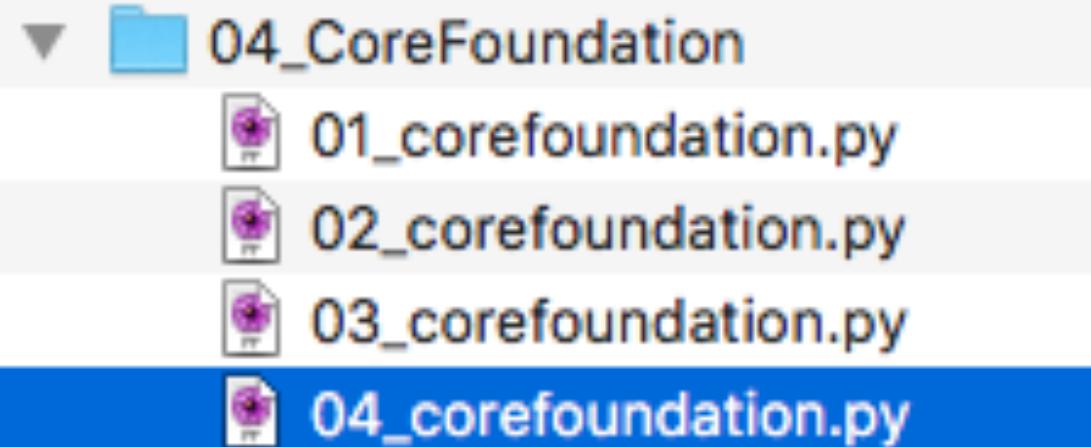
```
Calling CFPreferencesCopyAppValue...
```

```
Type returned from CFPreferencesCopyAppValue is <objective-c class __NSCSDictionary at 0x7fff9c3262f0>
```

```
StandardViewSettings['ExtendedListViewSettings']['calculateAllSizes'] is: False
```

```
import CoreFoundation

# similar to `defaults write com.apple.Finder ShowHardDrivesOnDesktop -bool NO`
CoreFoundation.CFPreferencesSetAppValue("ShowHardDrivesOnDesktop", False, "com.apple.Finder")
```





AppKit

(well a tiny part)

Class

NSWorkspace

An NSWorkspace object can launch other apps and perform a variety of file-handling services.

Language

Swift | Objective-C

SDK

macOS 10.0+

On This Page

[Overview](#) ⓘ[Symbols](#) ⓘ[Relationships](#) ⓘ

Overview

There is one shared NSWorkspace object per app. You use the class method `sharedWorkspace` to access it. For example, the following statement uses an NSWorkspace object to request that a file be opened in theTextEdit app:

```
[[NSWorkspace sharedWorkspace] openFile:@"/Myfiles/README"
    withApplication:@"TextEdit"];
```

You can use the workspace object to:

- Open, manipulate, and get information about files and devices
- Track changes to the file system, devices, and the user database
- Get and set Finder information for files
- Launch apps

<https://developer.apple.com/reference/appkit/nsworkspace?language=objc>

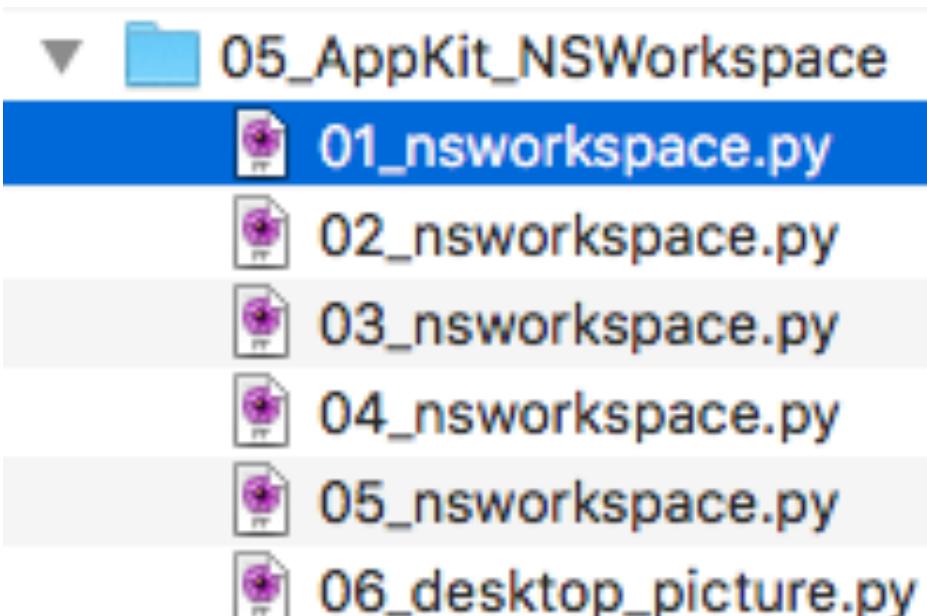
```
from AppKit import NSWorkspace

# get the shared workspace
workspace = NSWorkspace.sharedWorkspace()

app_name = 'Microsoft Word'

# find the full path for an application
print workspace fullPathForApplication_(app_name)

# launch an application
workspace.launchApplication_(app_name)
```



<https://developer.apple.com/reference/appkit/nsworkspace?language=objc>

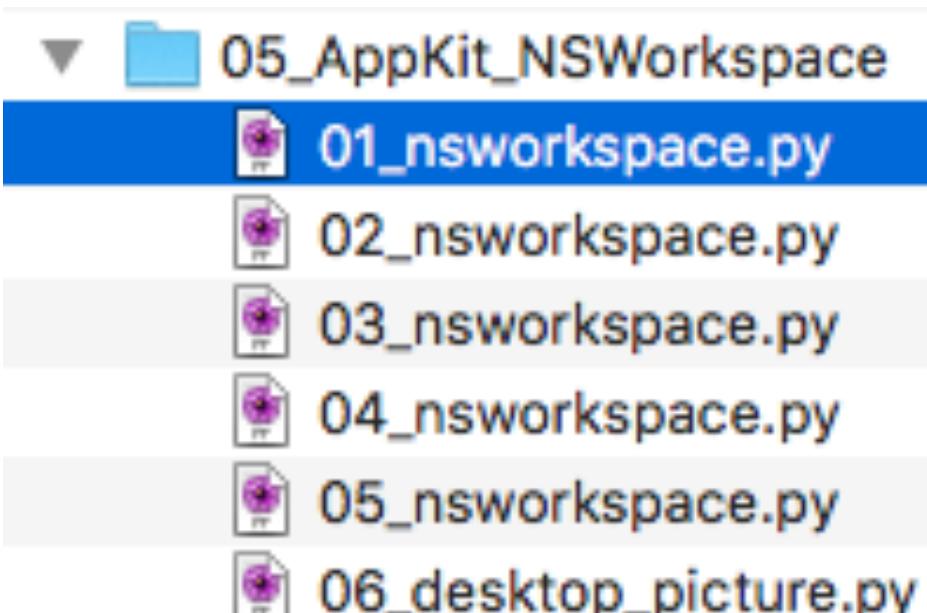
```
from AppKit import NSWorkspace
```

```
# get the shared workspace  
workspace = NSWorkspace.sharedWorkspace()
```

```
app_name = 'Microsoft Word'
```

```
# find the full path for an application  
print workspace.fullPathForApplication_(app_name)
```

```
# launch an application  
workspace.launchApplication_(app_name)
```



<https://developer.apple.com/reference/appkit/nsworkspace?language=objc>

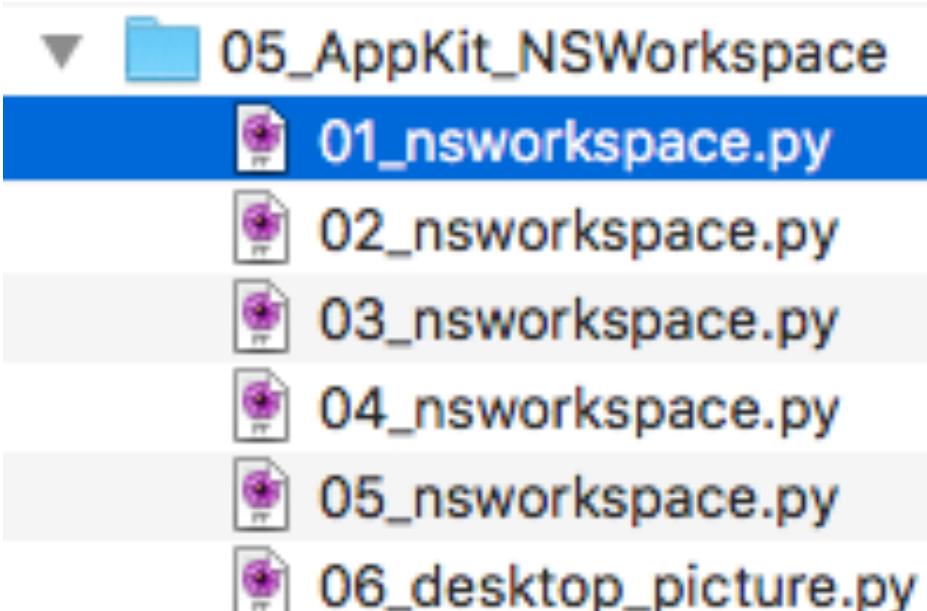
```
from AppKit import NSWorkspace
```

```
# get the shared workspace  
workspace = NSWorkspace.sharedWorkspace()
```

```
app_name = 'Microsoft Word'
```

```
# find the full path for an application  
print workspace fullPathForApplication_(app_name)
```

```
# launch an application  
workspace.launchApplication_(app_name)
```



<https://developer.apple.com/reference/appkit/nsworkspace?language=objc>

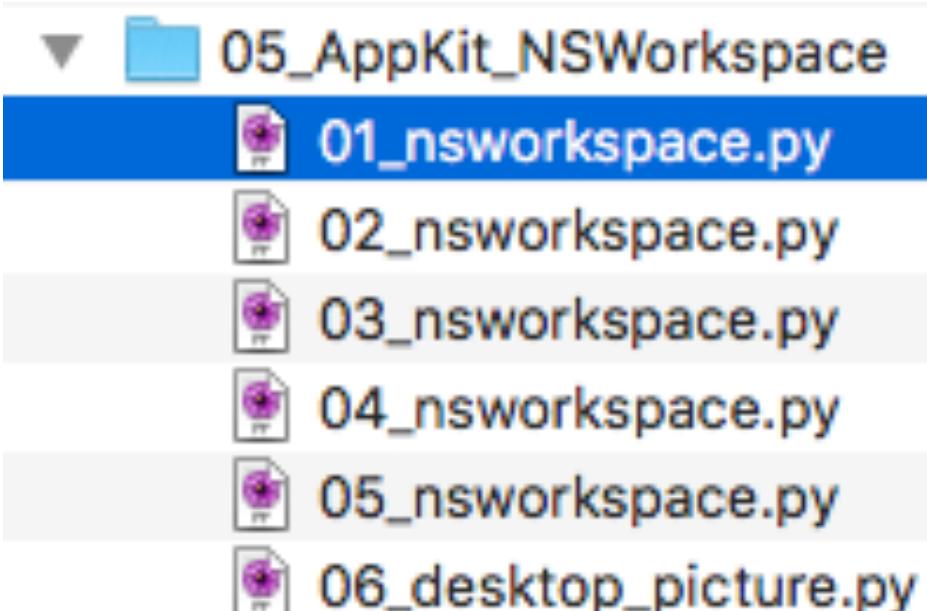
```
from AppKit import NSWorkspace

# get the shared workspace
workspace = NSWorkspace.sharedWorkspace()
```

```
app_name = 'Microsoft Word'

# find the full path for an application
print workspace fullPathForApplication_(app_name)
```

```
# launch an application
workspace.launchApplication_(app_name)
```



<https://developer.apple.com/reference/appkit/nsworkspace?language=objc>

```
from AppKit import NSWorkspace

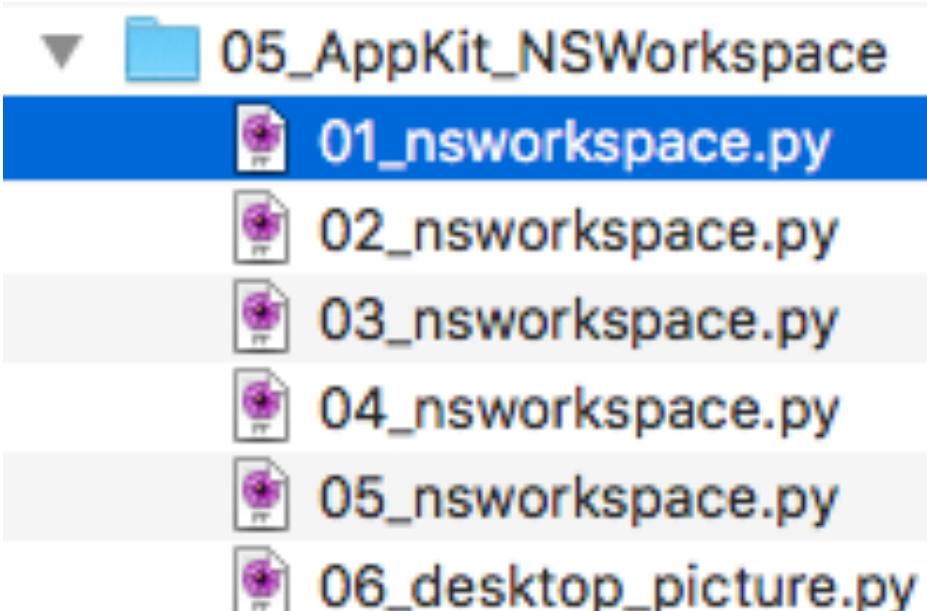
# get the shared workspace
workspace = NSWorkspace.sharedWorkspace()

app_name = 'Microsoft Word'

# find the full path for an application
print workspace fullPathForApplication_(app_name)

# launch an application
workspace.launchApplication_(app_name)
```

/Applications/Microsoft Word.app



https://

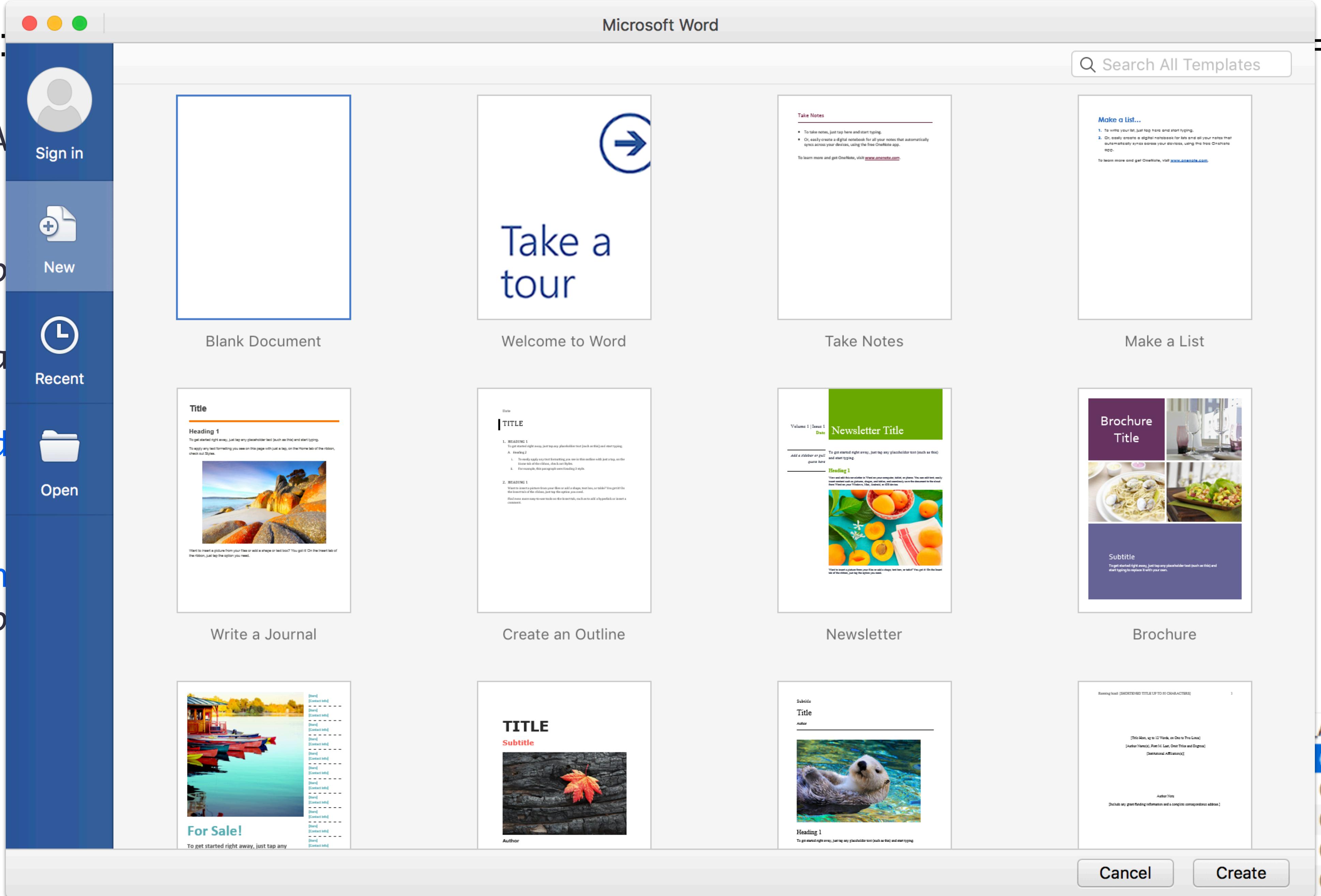
from A

get
workspace

app_na

find
print

laun
workspace



=objc

AppKit_NSWorkspace

01_nsworkspace.py

02_nsworkspace.py

03_nsworkspace.py

04_nsworkspace.py

05_nsworkspace.py

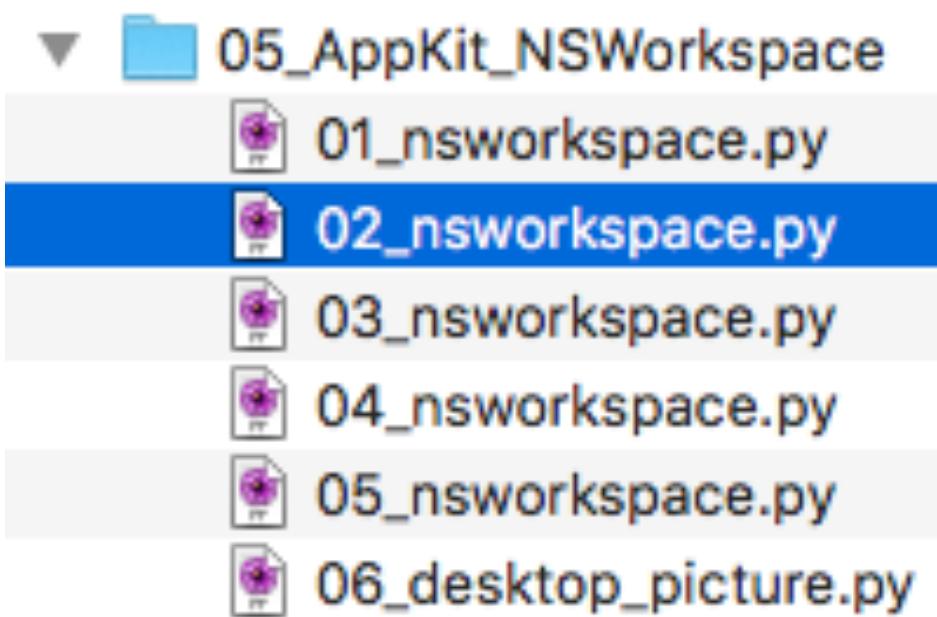
06_desktop_picture.py

<https://developer.apple.com/reference/appkit/nsworkspace?language=objc>

```
from AppKit import NSWorkspace  
  
# get the shared workspace  
workspace = NSWorkspace.sharedWorkspace()
```

```
# print a list of running applications  
print workspace.runningApplications()
```

```
"<NSRunningApplication: 0x7fc83f47e300 (com.apple.loginwindow - 37803)>","  
"<NSRunningApplication: 0x7fc83f48e760 ((null) - 37864)>","  
"<NSRunningApplication: 0x7fc83f461a00 (com.apple.touchbar.agent - 37865)>","  
"<NSRunningApplication: 0x7fc83f461b00 (com.apple.controlstrip - 37870)>","  
"<NSRunningApplication: 0x7fc83f483010 (com.apple.Terminal - 37872)>","  
"<NSRunningApplication: 0x7fc83f483110 (com.macromates.TextMate - 37880)>","  
"<NSRunningApplication: 0x7fc83f486be0 (com.apple.Safari - 37881)>","  
"<NSRunningApplication: 0x7fc83f486ce0 (com.apple.mail - 37882)>","  
"<NSRunningApplication: 0x7fc83f484da0 (com.apple.Preview - 37883)>","  
"<NSRunningApplication: 0x7fc83f484e30 (com.apple.iCal - 37885)>","  
"<NSRunningApplication: 0x7fc83f484f30 (com.apple.iChat - 37889)>","  
"<NSRunningApplication: 0x7fc83f488ad0 (com.apple.systemuiserver - 37894)>","  
"<NSRunningApplication: 0x7fc83f488bd0 (com.apple.dock - 37899)>"
```



Class

NSRunningApplication

NSRunningApplication is a class to manipulate and provide information for a single instance of an application. Only user applications are tracked; this does not provide information about every process on the system.

Language

[Swift](#) | [Objective-C](#)

SDK

macOS 10.6+

On This Page

[Overview](#) ▾[Symbols](#) ▾[Relationships](#) ▾

Overview

Some properties of an application are fixed, such as the bundle identifier. Other properties may vary over time, such as whether the app is hidden. Properties that vary can be observed with key-value observing, in which case the description comment for the method notes this capability.

Properties that vary over time are inherently race-prone. For example, a hidden app may unhide itself at any time. To ameliorate this, properties persist until the next turn of the main run loop in a common mode. For example, if you repeatedly poll an unhidden

app for its hidden property without allowing the run loop to run, it will continue to return NO, even if the app hides, until the next turn

Application Information[localizedName](#)

Indicates the localized name of the application.

[icon](#)

Returns the icon for the receiver's application.

[bundleIdentifier](#)

Indicates the CFBundleIdentifier of the application.

[bundleURL](#)

Indicates the URL to the application's bundle.

[executableArchitecture](#)

Indicates the executing processor architecture for the application.

[executableURL](#)

Indicates the URL to the application's executable.

[launchDate](#)

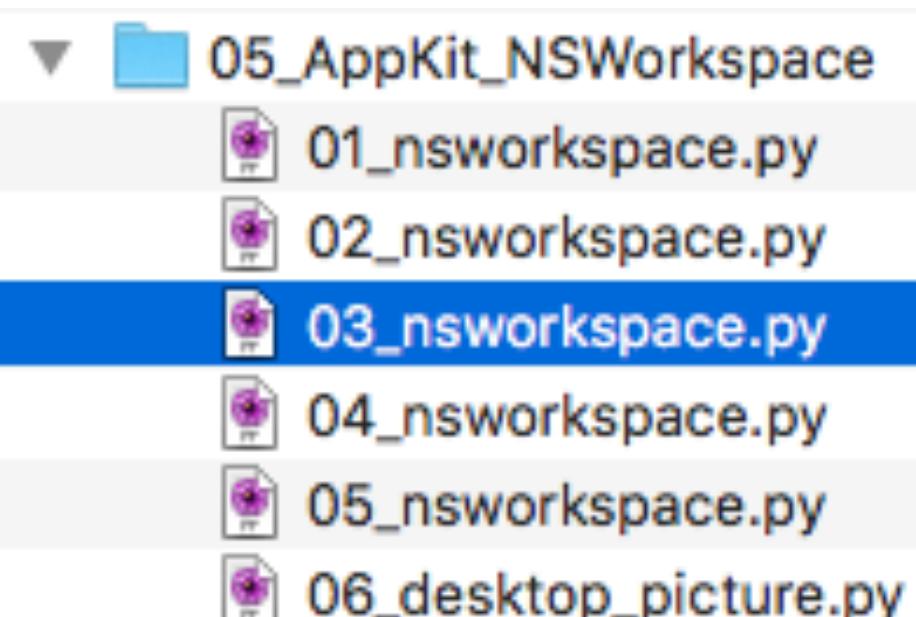
Indicates the date when the application was launched.

[finishedLaunching](#)

<https://developer.apple.com/reference/appkit/nsrunningapplication?language=objc>

```
from AppKit import NSWorkspace  
  
workspace = NSWorkspace.sharedWorkspace()  
  
running_apps = workspace.runningApplications()  
  
for app in running_apps:  
    print app.localizedDescription()
```

loginwindow
universalaccessd
Touch Bar agent
Control Strip
Terminal
TextMate
Safari
Mail
Preview



<https://developer.apple.com/reference/appkit/nsrunningapplication?language=objc>

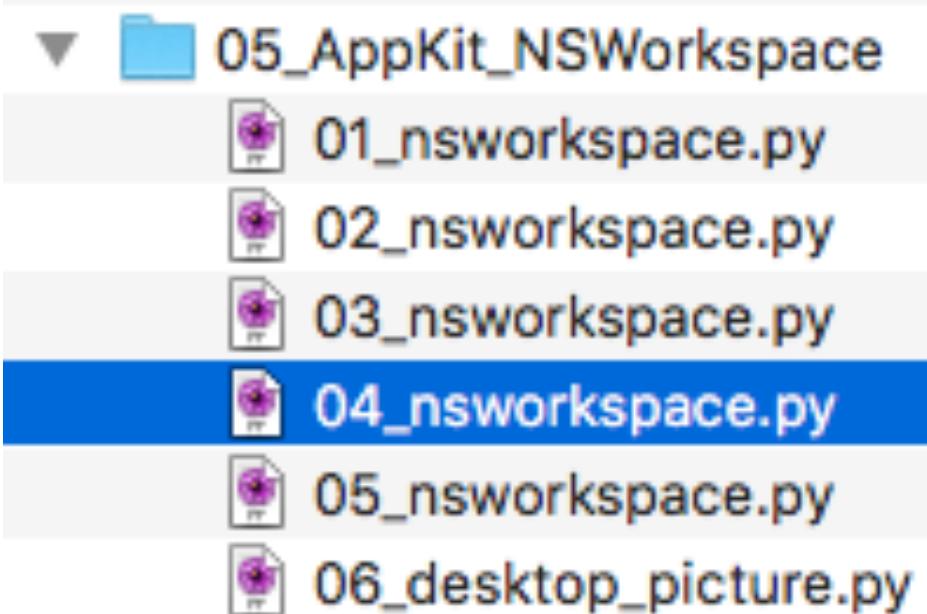
```
from AppKit import NSWorkspace

workspace = NSWorkspace.sharedWorkspace()

running_apps = workspace.runningApplications()

# iterate through the list looking for bundleIdentifier of com.apple.Terminal
for app in running_apps:
    if app.bundleIdentifier() == 'com.apple.Terminal':
        print 'Localized name: %s' % app.localizedName()
        print 'Bundle URL: %s' % app.bundleURL()
        print 'Launch date: %s' % app.launchDate()
        print 'Currently hidden: %s' % app.isHidden()
```

Localized name: Terminal
Bundle URL: file:///Applications/Utilities/Terminal.app
Launch date: 2017-02-06 16:13:30 +0000
Currently hidden: False



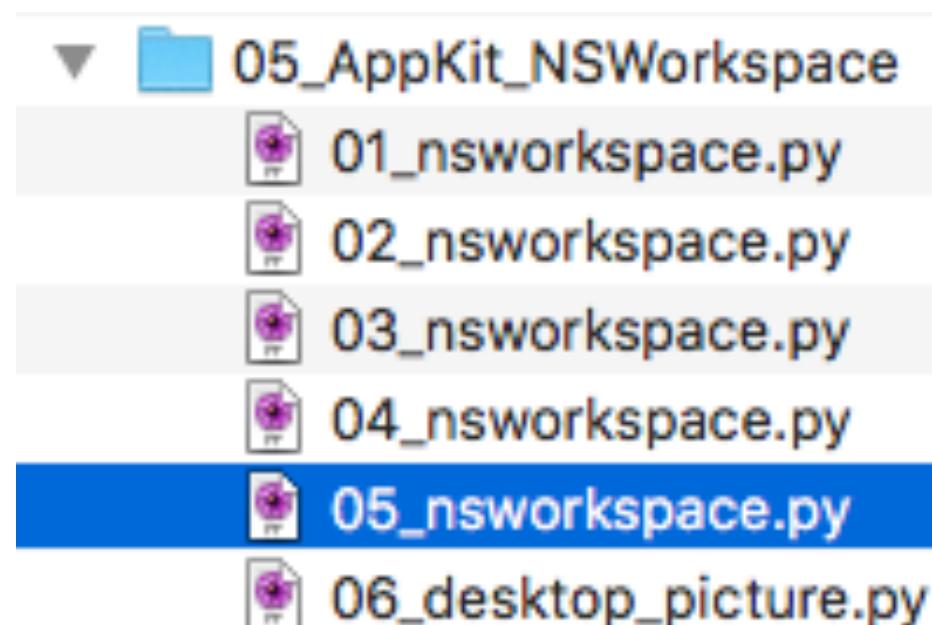
[https://developer.apple.com/reference/appkit/nsapplicationactivationoptions?
language=objc](https://developer.apple.com/reference/appkit/nsapplicationactivationoptions?language=objc)

```
from AppKit import NSWorkspace, NSApplicationActivateIgnoringOtherApps

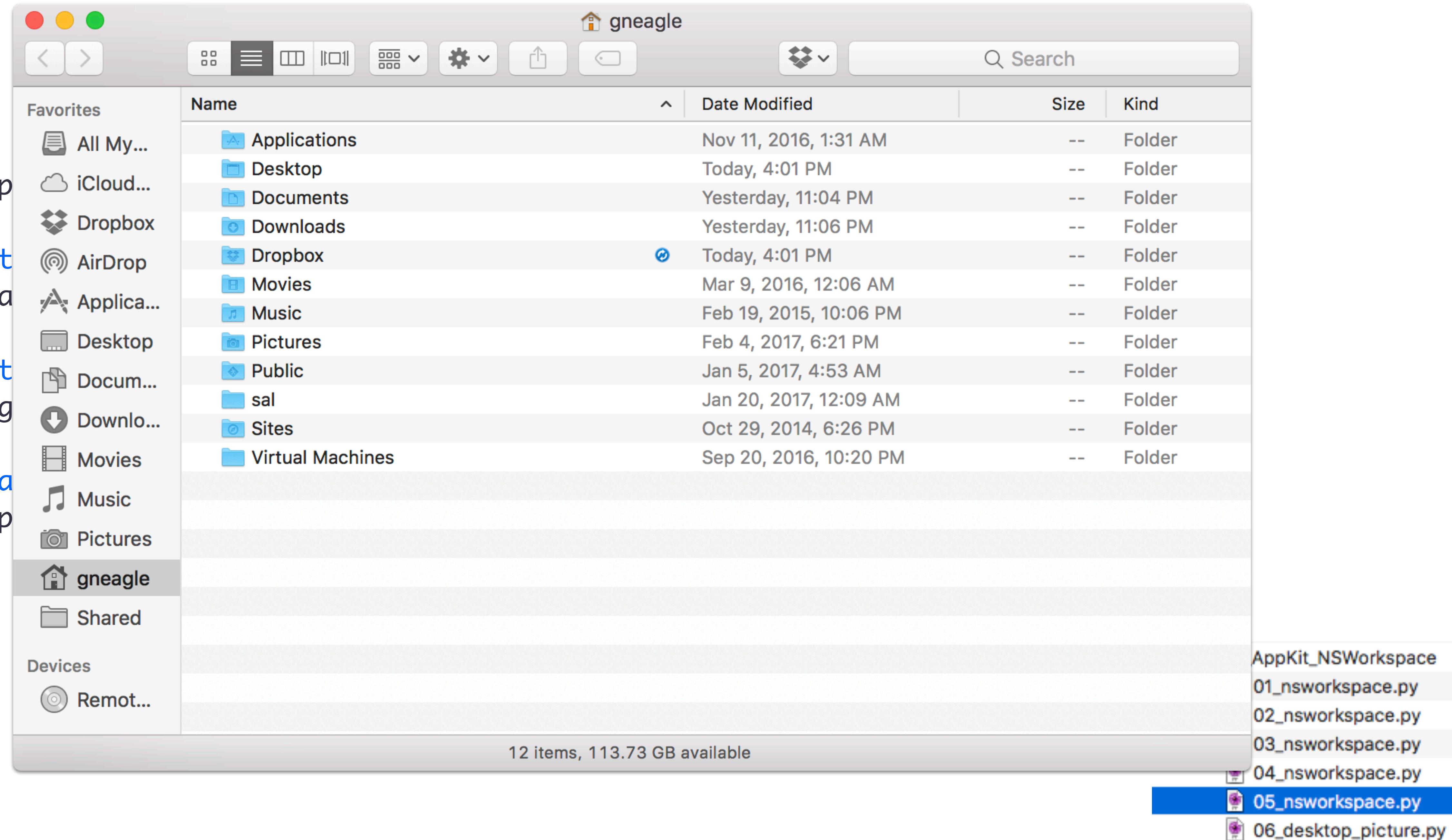
# get the shared workspace object
workspace = NSWorkspace.sharedWorkspace()

# get the list of running applications
running_apps = workspace.runningApplications()

# iterate through the list looking for bundleIdentifier of com.apple.Finder
for app in running_apps:
    if app.bundleIdentifier() == 'com.apple.Finder':
        app.activateWithOptions_(NSApplicationActivateIgnoringOtherApps)
```



<https://developer.apple.com/reference/appkit/nsapplicationactivationoptions?>



```
import glob
import random

from AppKit import NSWorkspace, NSScreen
from Foundation import NSURL

# pick a picture at random from our folder of pictures
pictures_glob = glob.glob("/Library/Desktop Pictures/*.jpg")
picture_path = random.choice(pictures_glob)

# convert the desktop picture path to an NSURL object
# https://developer.apple.com/reference/foundation/nsurl?language=objc
file_url = NSURL.fileURLWithPath_(picture_path)

# get shared workspace
# https://developer.apple.com/reference/appkit/nsworkspace?language=objc
workspace = NSWorkspace.sharedWorkspace()

# iterate over all screens
# https://developer.apple.com/reference/appkit/nsscreen?language=objc
for screen in NSScreen.screens():
    # tell the workspace to set the desktop picture
    (result, error) = workspace.setDesktopImageURL_forScreen_options_error_(
        file_url, screen, None, None)
```

▼	05_AppKit_NSWorkspace
	01_nsworkspace.py
	02_nsworkspace.py
	03_nsworkspace.py
	04_nsworkspace.py
	05_nsworkspace.py
	06_desktop_picture.py

```
import glob
import random

from AppKit import NSWorkspace, NSScreen
from Foundation import NSURL

# pick a picture at random from our folder of pictures
pictures_glob = glob.glob("/Library/Desktop Pictures/*.jpg")
picture_path = random.choice(pictures_glob)

# convert the desktop picture path to an NSURL object
# https://developer.apple.com/reference/foundation/nsurl?language=objc
file_url = NSURL.fileURLWithPath_(picture_path)

# get shared workspace
# https://developer.apple.com/reference/appkit/nsworkspace?language=objc
workspace = NSWorkspace.sharedWorkspace()

# iterate over all screens
# https://developer.apple.com/reference/appkit/nsscreen?language=objc
for screen in NSScreen.screens():
    # tell the workspace to set the desktop picture
    (result, error) = workspace.setDesktopImageURL_forScreen_options_error_(
        file_url, screen, None, None)
```

▼	05_AppKit_NSWorkspace
	01_nsworkspace.py
	02_nsworkspace.py
	03_nsworkspace.py
	04_nsworkspace.py
	05_nsworkspace.py
	06_desktop_picture.py

```
import glob
import random

from AppKit import NSWorkspace, NSScreen
from Foundation import NSURL

# pick a picture at random from our folder of pictures
pictures_glob = glob.glob("/Library/Desktop Pictures/*.jpg")
picture_path = random.choice(pictures_glob)

# convert the desktop picture path to an NSURL object
# https://developer.apple.com/reference/foundation/nsurl?language=objc
file_url = NSURL.fileURLWithPath_(picture_path)

# get shared workspace
# https://developer.apple.com/reference/appkit/nsworkspace?language=objc
workspace = NSWorkspace.sharedWorkspace()

# iterate over all screens
# https://developer.apple.com/reference/appkit/nsscreen?language=objc
for screen in NSScreen.screens():
    # tell the workspace to set the desktop picture
    (result, error) = workspace.setDesktopImageURL_forScreen_options_error_(
        file_url, screen, None, None)
```

▼	05_AppKit_NSWorkspace
	01_nsworkspace.py
	02_nsworkspace.py
	03_nsworkspace.py
	04_nsworkspace.py
	05_nsworkspace.py
	06_desktop_picture.py

```
import glob
import random

from AppKit import NSWorkspace, NSScreen
from Foundation import NSURL

# pick a picture at random from our folder of pictures
pictures_glob = glob.glob("/Library/Desktop Pictures/*.jpg")
picture_path = random.choice(pictures_glob)

# convert the desktop picture path to an NSURL object
# https://developer.apple.com/reference/foundation/nsurl?language=objc
file_url = NSURL.fileURLWithPath_(picture_path)

# get shared workspace
# https://developer.apple.com/reference/appkit/nsworkspace?language=objc
workspace = NSWorkspace.sharedWorkspace()

# iterate over all screens
# https://developer.apple.com/reference/appkit/nsscreen?language=objc
for screen in NSScreen.screens():
    # tell the workspace to set the desktop picture
    (result, error) = workspace.setDesktopImageURL_forScreen_options_error_(
        file_url, screen, None, None)
```

▼	05_AppKit_NSWorkspace
	01_nsworkspace.py
	02_nsworkspace.py
	03_nsworkspace.py
	04_nsworkspace.py
	05_nsworkspace.py
	06_desktop_picture.py

```
import glob
import random

from AppKit import NSWorkspace, NSScreen
from Foundation import NSURL

# pick a picture at random from our folder of pictures
pictures_glob = glob.glob("/Library/Desktop Pictures/*.jpg")
picture_path = random.choice(pictures_glob)

# convert the desktop picture path to an NSURL object
# https://developer.apple.com/reference/foundation/nsurl?language=objc
file_url = NSURL.fileURLWithPath_(picture_path)

# get shared workspace
# https://developer.apple.com/reference/appkit/nsworkspace?language=objc
workspace = NSWorkspace.sharedWorkspace()

# iterate over all screens
# https://developer.apple.com/reference/appkit/nsscreen?language=objc
for screen in NSScreen.screens():
    # tell the workspace to set the desktop picture
    (result, error) = workspace.setDesktopImageURL_forScreen_options_error(
        file_url, screen, None, None)
```

▼	05_AppKit_NSWorkspace
	01_nsworkspace.py
	02_nsworkspace.py
	03_nsworkspace.py
	04_nsworkspace.py
	05_nsworkspace.py
	06_desktop_picture.py

Demo



A 3D rendering of a white smartphone lying diagonally. A large, semi-transparent red circle is positioned at the top right corner of the screen, representing a notification badge. The word "Notifications" is written in a bold, black, sans-serif font across the center of the phone's face.

Notifications

Class

NSUserNotificationCenter

The `NSUserNotificationCenter` class delivers user notifications to the user from applications or helper applications.

Language

Swift | Objective-C

SDK

macOS 10.8+

Overview

When a user notification's delivery date has been reached, or it is manually delivered, the notification center may display the notification to the user. The user notification center reserves the right to decide if a delivered user notification is presented to the user. For example, it may suppress the notification if the application is already frontmost (the delegate can override this action). The application can check the result of this decision by examining the `presented` property of a delivered user notification.

`NSUserNotification` instances the `NSUserNotificationCenter` are tracking will be in one of two states: scheduled or delivered. A scheduled user notification has a `deliveryDate`. On that delivery date, the notification will move from being scheduled to being delivered. Note that the user notification may be displayed later than the delivery date depending on a number of factors.

A delivered user notification has an `actualDeliveryDate`. That is the date when it moved from being scheduled to delivered, or when it was manually delivered using the `deliverNotification:` method.

The application and the user notification center are both ultimately subject to the user's preferences. If the user decides to hide all alerts from your application, the `presented` property will still behave as above, but the user will not see any animation or hear any sound.

On This Page

[Overview](#) ⓘ[Symbols](#) ⓘ[Relationships](#) ⓘ

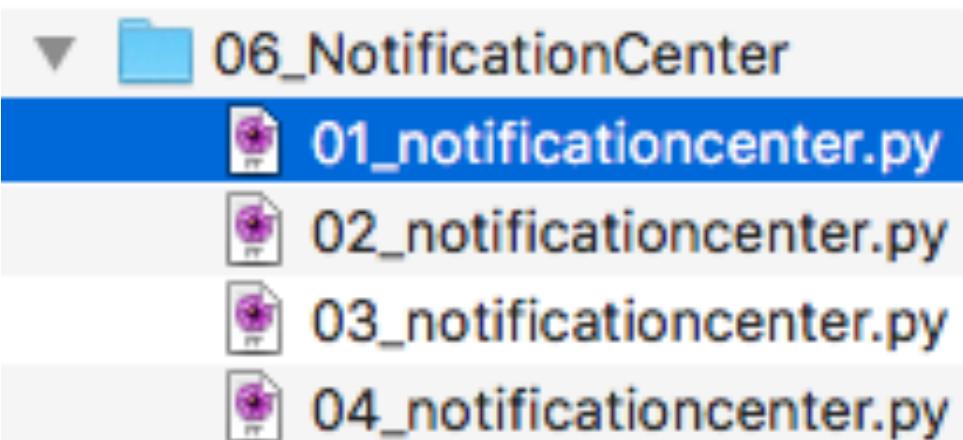
```
from Foundation import NSUserNotificationCenter
from Foundation import NSUserNotification

def notify(title, subtitle, text):
    # create a user notification
    notification = NSUserNotification.alloc().init()
    notification.setTitle_(title)
    notification.setSubtitle_(subtitle)
    notification.setInformativeText_(text)

    # get the default User Notification Center
    nc = NSUserNotificationCenter.defaultUserNotificationCenter()

    # tell the notification center to deliver the user notification
    nc.deliverNotification_(notification)

notify(u'Updates available', u'', u'Software updates are available.')
```



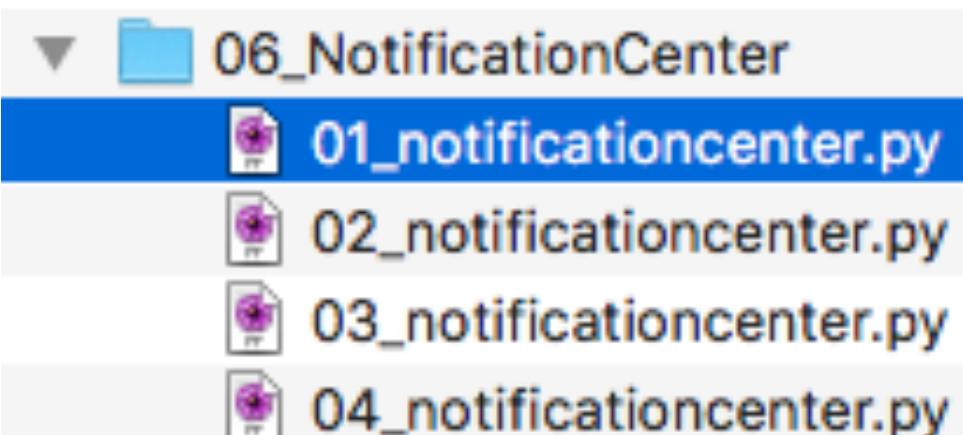
```
from Foundation import NSUserNotificationCenter
from Foundation import NSUserNotification

def notify(title, subtitle, text):
    # create a user notification
    notification = NSUserNotification.alloc().init()
    notification.setTitle_(title)
    notification.setSubtitle_(subtitle)
    notification.setInformativeText_(text)

    # get the default User Notification Center
    nc = NSUserNotificationCenter.defaultUserNotificationCenter()

    # tell the notification center to deliver the user notification
    nc.deliverNotification_(notification)

notify(u'Updates available', u'', u'Software updates are available.')
```



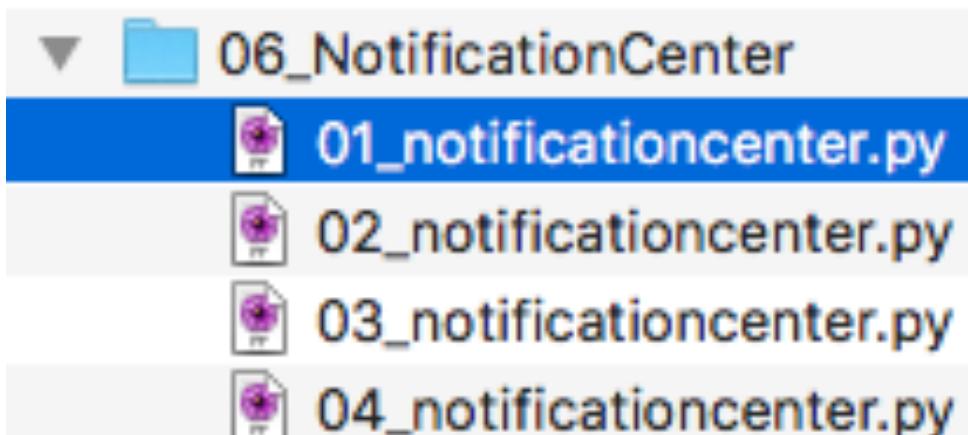
```
from Foundation import NSUserNotificationCenter
from Foundation import NSUserNotification

def notify(title, subtitle, text):
    # create a user notification
    notification = NSUserNotification.alloc().init()
    notification.setTitle_(title)
    notification.setSubtitle_(subtitle)
    notification.setInformativeText_(text)

    # get the default User Notification Center
    nc = NSUserNotificationCenter.defaultUserNotificationCenter()

    # tell the notification center to deliver the user notification
    nc.deliverNotification_(notification)

notify(u'Updates available', u'', u'Software updates are available.')
```



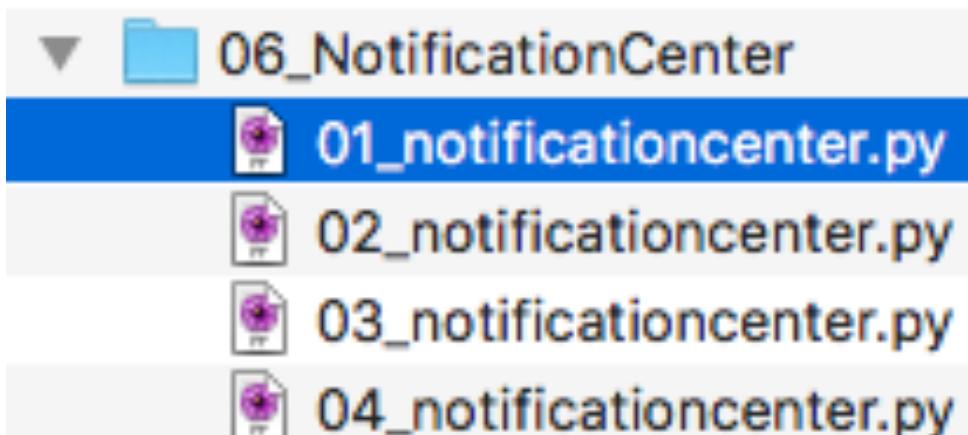
```
from Foundation import NSUserNotificationCenter
from Foundation import NSUserNotification

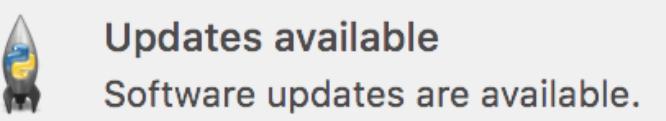
def notify(title, subtitle, text):
    # create a user notification
    notification = NSUserNotification.alloc().init()
    notification.setTitle_(title)
    notification.setSubtitle_(subtitle)
    notification.setInformativeText_(text)

    # get the default User Notification Center
    nc = NSUserNotificationCenter.defaultUserNotificationCenter()

    # tell the notification center to deliver the user notification
    nc.deliverNotification_(notification)

notify(u'Updates available', u'', u'Software updates are available.')
```





Updates available

Software updates are available.

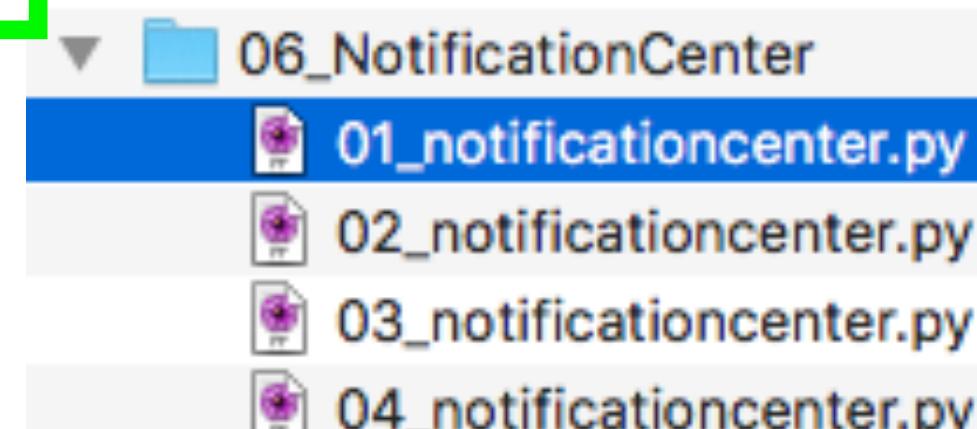
```
from Foundation import NSUserNotificationCenter
from Foundation import NSUserNotification

def notify(title, subtitle, text):
    # create a user notification
    notification = NSUserNotification.alloc().init()
    notification.setTitle_(title)
    notification.setSubtitle_(subtitle)
    notification.setInformativeText_(text)

    # get the default User Notification Center
    nc = NSUserNotificationCenter.defaultUserNotificationCenter()

    # tell the notification center to deliver the user notification
    nc.deliverNotification_(notification)
```

```
notify(u'Updates available', u'', u'Software updates are available.')
```



```

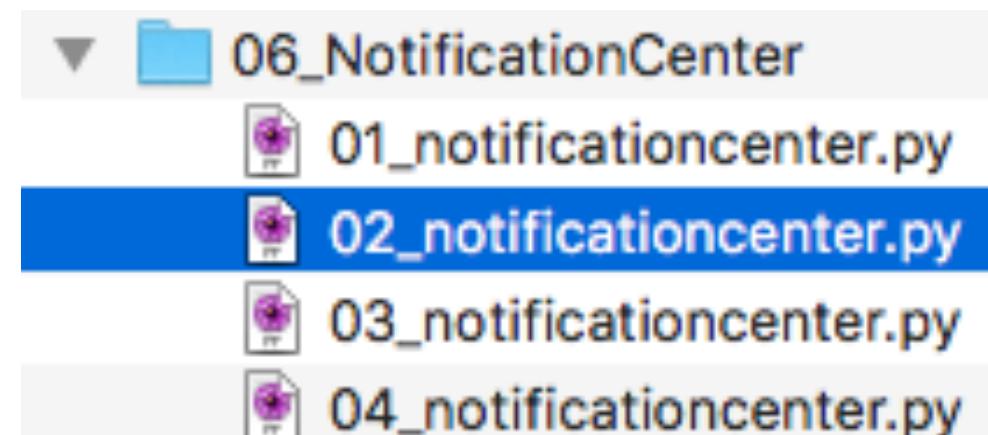
from Foundation import NSBundle
from Foundation import NSUserNotificationCenter
from Foundation import NSUserNotification

def set_fake_bundleid(bundleid):
    # disturbing hack warning! Thank you, @frogor
    bundle = NSBundle mainBundle()
    info = bundle.localizedInfoDictionary() or bundle.infoDictionary()
    # override the bundleid with the one we want
    info['CFBundleIdentifier'] = bundleid

def notify(title, subtitle, text, bundleid=None):
    if bundleid:
        # fake our bundleid
        set_fake_bundleid(bundleid)
    # create a new user notification
    notification = NSUserNotification.alloc().init()
    notification.setTitle_(title)
    notification.setSubtitle_(subtitle)
    notification.setInformativeText_(text)
    # get the default User Notification Center
    nc = NSUserNotificationCenter.defaultUserNotificationCenter()
    # deliver the notification
    nc.deliverNotification_(notification)

notify(u'Updates available', u'', u'Software updates are available.',
      bundleid='com.googlecode.munki.ManagedSoftwareCenter')

```



```

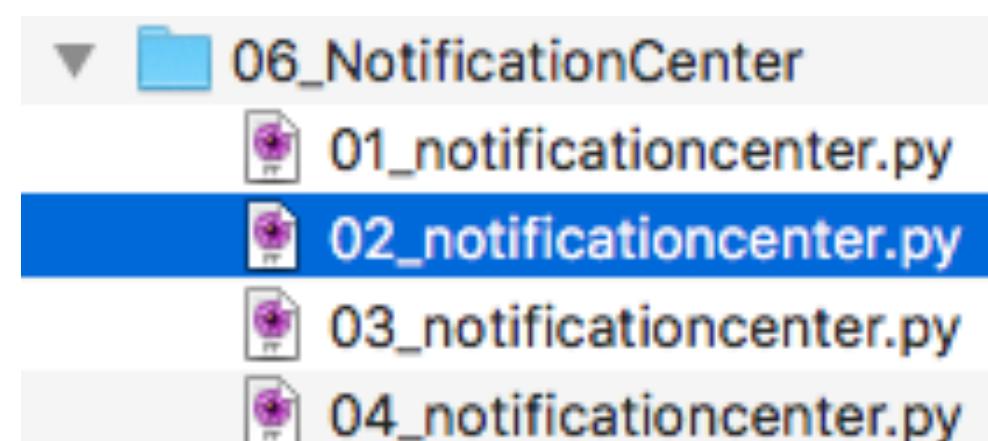
from Foundation import NSBundle
from Foundation import NSUserNotificationCenter
from Foundation import NSUserNotification

def set_fake_bundleid(bundleid):
    # disturbing hack warning! Thank you, @frogor
    bundle = NSBundle mainBundle()
    info = bundle.localizedInfoDictionary() or bundle.infoDictionary()
    # override the bundleid with the one we want
    info['CFBundleIdentifier'] = bundleid

def notify(title, subtitle, text, bundleid=None):
    if bundleid:
        # fake our bundleid
        set_fake_bundleid(bundleid)
    # create a new user notification
    notification = NSUserNotification.alloc().init()
    notification.setTitle_(title)
    notification.setSubtitle_(subtitle)
    notification.setInformativeText_(text)
    # get the default User Notification Center
    nc = NSUserNotificationCenter.defaultUserNotificationCenter()
    # deliver the notification
    nc.deliverNotification_(notification)

notify(u'Updates available', u'', u'Software updates are available.',
      bundleid='com.googlecode.munki.ManagedSoftwareCenter')

```



```

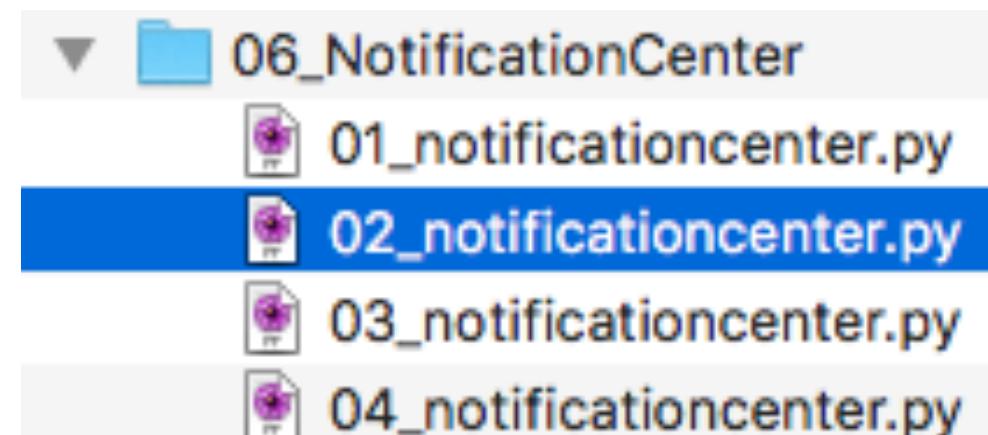
from Foundation import NSBundle
from Foundation import NSUserNotificationCenter
from Foundation import NSUserNotification

def set_fake_bundleid(bundleid):
    # disturbing hack warning! Thank you, @frogor
    bundle = NSBundle mainBundle()
    info = bundle.localizedInfoDictionary() or bundle.infoDictionary()
    # override the bundleid with the one we want
    info['CFBundleIdentifier'] = bundleid

def notify(title, subtitle, text, bundleid=None):
    if bundleid:
        # fake our bundleid
        set_fake_bundleid(bundleid)
    # create a new user notification
    notification = NSUserNotification.alloc().init()
    notification.setTitle_(title)
    notification.setSubtitle_(subtitle)
    notification.setInformativeText_(text)
    # get the default User Notification Center
    nc = NSUserNotificationCenter.defaultUserNotificationCenter()
    # deliver the notification
    nc.deliverNotification_(notification)

notify(u'Updates available', u'', u'Software updates are available.',
      bundleid='com.googlecode.munki.ManagedSoftwareCenter')

```

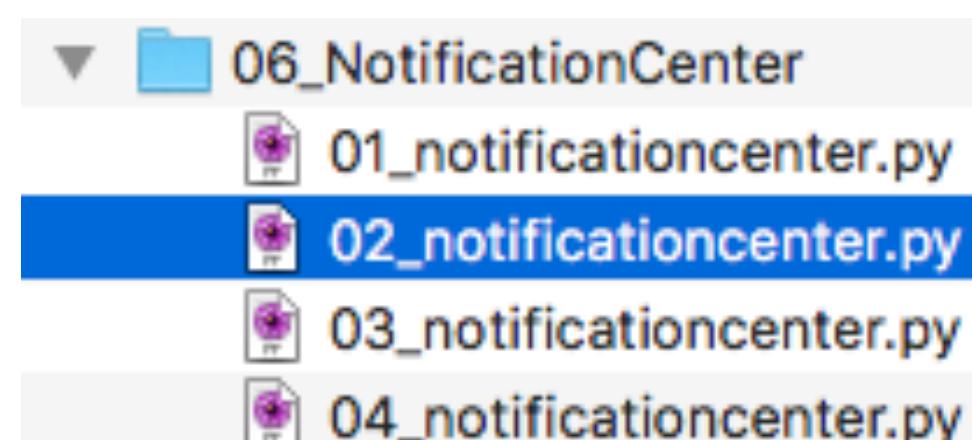
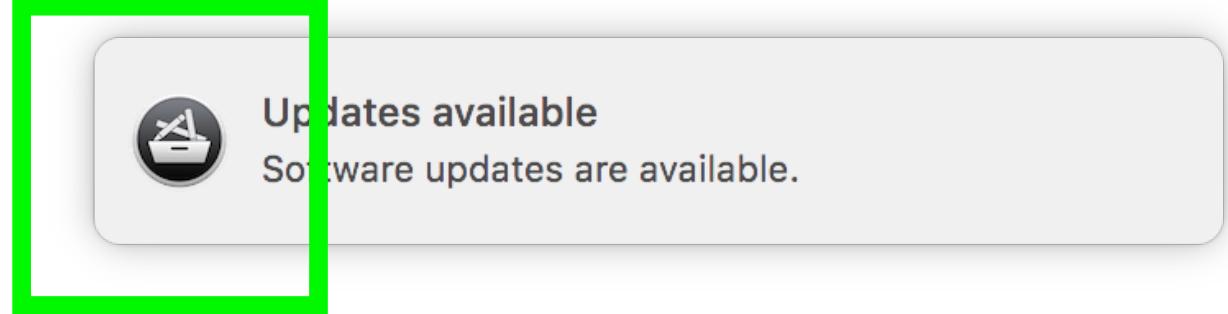


```
from Foundation import NSBundle
from Foundation import NSUserNotificationCenter
from Foundation import NSUserNotification

def set_fake_bundleid(bundleid):
    # disturbing hack warning! Thank you, @frogor
    bundle = NSBundle mainBundle()
    info = bundle.localizedInfoDictionary() or bundle.infoDictionary()
    # override the bundleid with the one we want
    info['CFBundleIdentifier'] = bundleid

def notify(title, subtitle, text, bundleid=None):
    if bundleid:
        # fake our bundleid
        set_fake_bundleid(bundleid)
    # create a new user notification
    notification = NSUserNotification.alloc().init()
    notification.setTitle_(title)
    notification.setSubtitle_(subtitle)
    notification.setInformativeText_(text)
    # get the default User Notification Center
    nc = NSUserNotificationCenter.defaultUserNotificationCenter()
    # deliver the notification
    nc.deliverNotification_(notification)

notify(u'Updates available', u'', u'Software updates are available.',
      bundleid='com.googlecode.munki.ManagedSoftwareCenter')
```



```
class NotificationCenterDelegate(NSObject):
    '''Class to implement required NSUserNotificationCenterDelegate
    methods'''

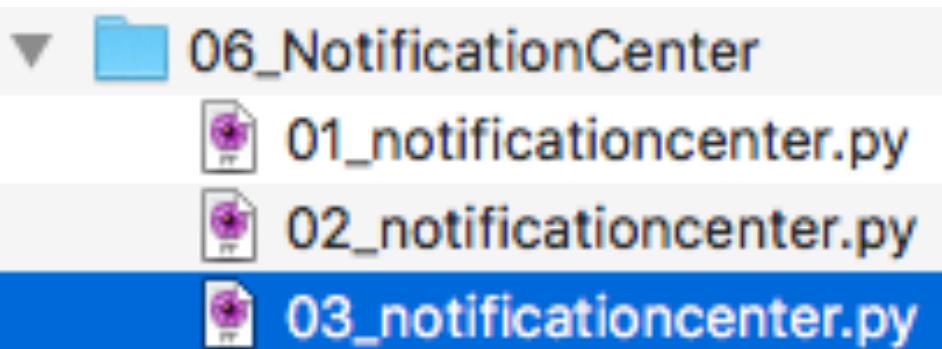
    # See https://developer.apple.com/reference/foundation/
    #           nsusernotificationcenterdelegate?language=objc

    keepRunning = True

    def userNotificationCenter_didActivateNotification_(
            self, center, userNotification):
        print "Got userNotificationCenter:didActivateNotification:"
        launch_app_by_bundleid(get_fake_bundleid())
        self.keepRunning = False

    def userNotificationCenter_shouldPresentNotification_(
            self, center, userNotification):
        print "Got userNotificationCenter:shouldPresentNotification:"
        return True

    def userNotificationCenter_didDeliverNotification_(
            self, center, notification):
        print "Got userNotificationCenter:didDeliverNotification:"
```



```
class NotificationCenterDelegate(NSObject):
    '''Class to implement required NSUserNotificationCenterDelegate
    methods'''

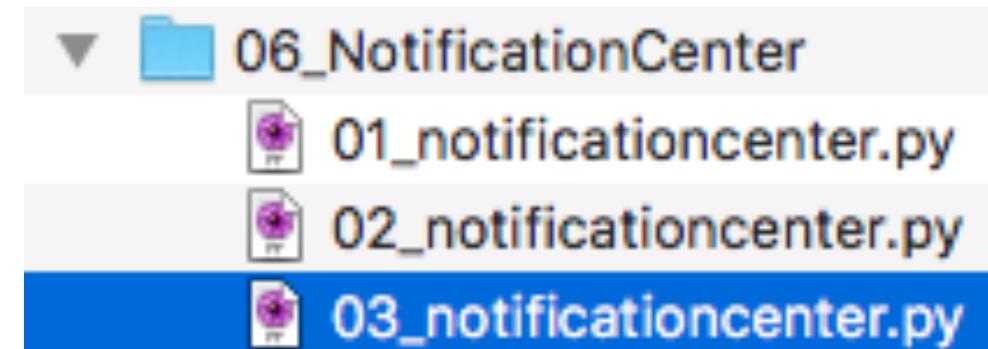
# See https://developer.apple.com/reference/foundation/
#           nsusernotificationcenterdelegate?language=objc
```

```
keepRunning = True
```

```
def userNotificationCenter_didActivateNotification_(
    self, center, userNotification):
    print "Got userNotificationCenter:didActivateNotification:"
    launch_app_by_bundleid(get_fake_bundleid())
    self.keepRunning = False
```

```
def userNotificationCenter_shouldPresentNotification_(
    self, center, userNotification):
    print "Got userNotificationCenter:shouldPresentNotification:"
    return True
```

```
def userNotificationCenter_didDeliverNotification_(
    self, center, notification):
    print "Got userNotificationCenter:didDeliverNotification:"
```



```
class NotificationCenterDelegate(NSObject):
    '''Class to implement required NSUserNotificationCenterDelegate
    methods'''

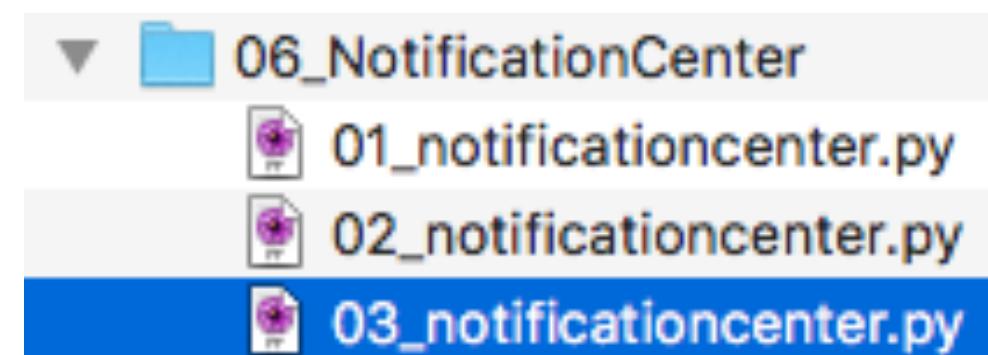
# See https://developer.apple.com/reference/foundation/
#           nsusernotificationcenterdelegate?language=objc

keepRunning = True

def userNotificationCenter_didActivateNotification_(
        self, center, userNotification):
    print "Got userNotificationCenter:didActivateNotification:"
    launch_app_by_bundleid(get_fake_bundleid())
    self.keepRunning = False

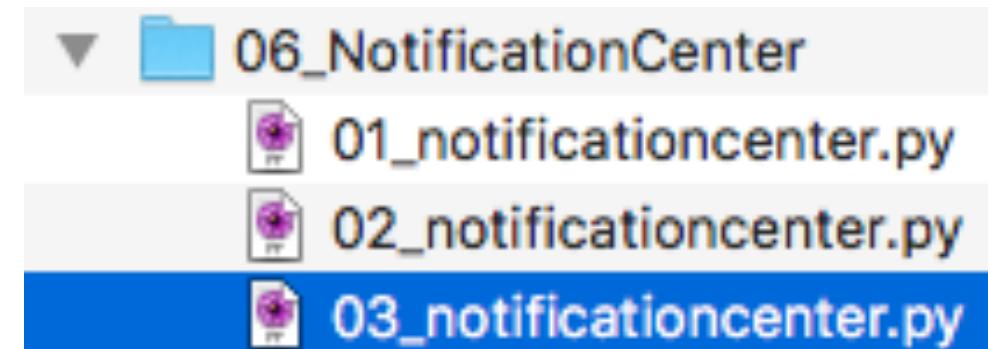
def userNotificationCenter_shouldPresentNotification_(
        self, center, userNotification):
    print "Got userNotificationCenter:shouldPresentNotification:"
    return True

def userNotificationCenter_didDeliverNotification_(
        self, center, notification):
    print "Got userNotificationCenter:didDeliverNotification:"
```



```
def get_fake_bundleid():
    # get the current bundleid (which might have been overridden)
    bundle = NSBundle mainBundle()
    info = bundle.localizedInfoDictionary() or bundle.infoDictionary()
    return info['CFBundleIdentifier']
```

```
def launch_app_by_bundleid(bundleid):
    # We really should use NSWorkspace methods but we'll be lazy
    subprocess.call(['/usr/bin/open', '-b', bundleid])
```



```

def notify(title, subtitle, text, bundleid=None):
    if bundleid:
        # fake our bundleid
        set_fake_bundleid(bundleid)

    # create a new user notification
    notification = NSUserNotification.alloc().init()
    notification.setTitle_(title)
    notification.setSubtitle_(subtitle)
    notification.setInformativeText_(text)

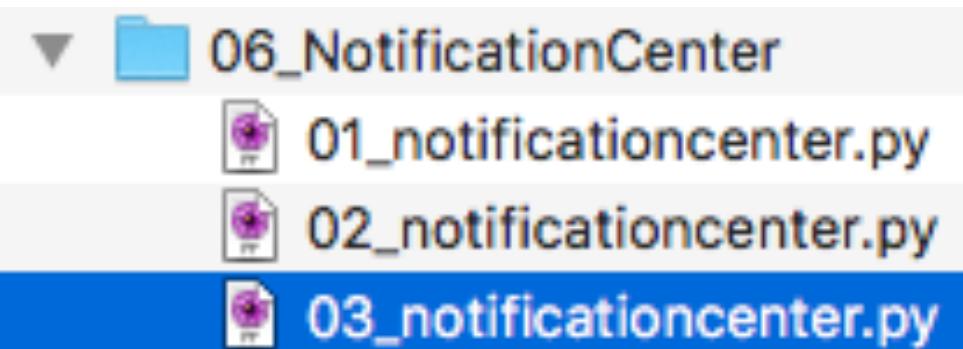
    # get the default User Notification Center
    nc = NSUserNotificationCenter.defaultUserNotificationCenter()

    # create a delegate object that implements our delegate methods
    my_delegate = NotificationCenterDelegate.alloc().init()
    nc.setDelegate_(my_delegate)

    # deliver the notification
    nc.deliverNotification_(notification)

    # keep running until the notification is activated
    while (nc_delegate.keepRunning):
        NSRunLoop.currentRunLoop().runUntilDate_
            (NSDate.dateWithTimeIntervalSinceNow_(0.1))

```



```
def notify(title, subtitle, text, bundleid=None):
    if bundleid:
        # fake our bundleid
        set_fake_bundleid(bundleid)

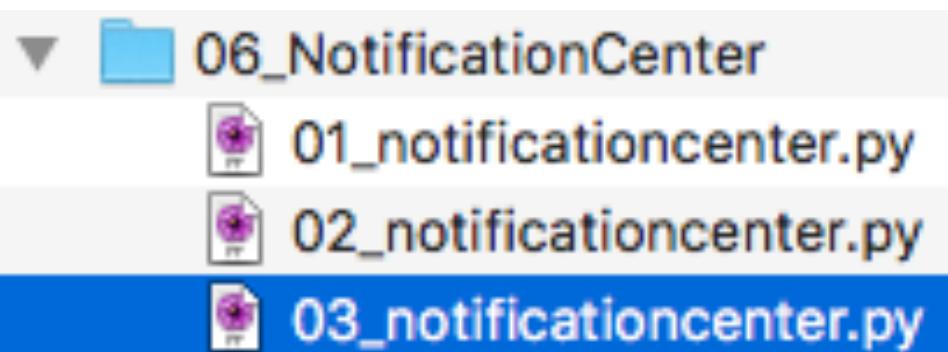
    # create a new user notification
    notification = NSUserNotification.alloc().init()
    notification.setTitle_(title)
    notification.setSubtitle_(subtitle)
    notification.setInformativeText_(text)

    # get the default User Notification Center
    nc = NSUserNotificationCenter.defaultUserNotificationCenter()

    # create a delegate object that implements our delegate methods
    my_delegate = NotificationCenterDelegate.alloc().init()
    nc.setDelegate_(my_delegate)

    # deliver the notification
    nc.deliverNotification_(notification)

    # keep running until the notification is activated
    while (nc_delegate.keepRunning):
        NSRunLoop.currentRunLoop().runUntilDate_
            (NSDate.dateWithTimeIntervalSinceNow_(0.1))
```



Demo



Links

<https://github.com/gregneagle/macaduk2017>

Official PyObjC documentation

<https://pythonhosted.org/pyobjc/>

The screenshot shows a web browser displaying the official PyObjC documentation at <https://pythonhosted.org/pyobjc/>. The page has a green header bar with the title "PyObjC – the Python ↔ Objective-C bridge »". On the left, there's a sidebar with "Table Of Contents" and "Resources" sections, and a "Donate" button. The main content area has three sections: "Introduction", "Release information", and "General documentation".

Table Of Contents

- Introduction
 - Release information
- General documentation
- API documentation
- PyObjC Development
- Indices and tables

Resources

- Examples
- Changelog
- API Notes
- Issues
- Repository

Support development

Donate

Quick search

Introduction

The PyObjC project aims to provide a bridge between the Python and Objective-C programming languages. The bridge is intended to be fully bidirectional, allowing the Python programmer to take full advantage of the power provided by various Objective-C based toolkits and the Objective-C programmer transparent access to Python based functionality.

The most important usage of this is writing Cocoa GUI applications on Mac OS X in pure Python. See our tutorial for an example of this.

Release information

PyObjC 3.2.1 was released on 2016-12-12. See the [changelog](#) for more information.

General documentation

- Installing PyObjC



<https://gist.github.com/pudquick>

aka @frogor on Slack

aka @Mikeymikey on Twitter

aka Michael Lynn

Battery status:

<https://gist.github.com/pudquick/134acb5f7423312effcc98ec56679136>

Timezone autodiscovery:

<https://gist.github.com/pudquick/ba235b7e90aafb9986158697a457a0d0>

Locking the screen:

<https://gist.github.com/pudquick/097f97f4cb67c554f04dba6b5a09a506>

A screenshot of a GitHub repository page for the project "pudquick / nibbler". The page shows basic repository statistics: 4 commits, 1 branch, 0 releases, 3 contributors, and an MIT license. It includes navigation links for Code, Issues (0), Pull requests (0), Projects (0), Wiki, Pulse, and Graphs. The "Code" tab is selected.

python pyobjc utility for displaying dialogs using .nib files

4 commits	1 branch	0 releases	3 contributors	MIT
Branch: master	New pull request	Create new file	Upload files	Find file
lazymutt committed with pudquick Documentation updates (#2) ...				Latest commit 35188f3 on Nov 29, 2016
examples Add documentation & examples				4 months ago
LICENSE Initial commit				5 months ago
README.md Documentation updates (#2)				2 months ago
nibbler.py Add documentation & examples				4 months ago

README.md

nibbler

<https://github.com/pudquick/nibbler>

Nibbler is a Python PyObjC utility for displaying dialogs using .nib files.

Examples

The best way to play with nibbler is to download this git repo and play with the two example files.

More

[http://www.mactech.com/articles/mactech/
Vol.25/25.08/2508MacEnterprise-SystemFrameworkScripting/index.html](http://www.mactech.com/articles/mactech/Vol.25/25.08/2508MacEnterprise-SystemFrameworkScripting/index.html)

[https://managingosx.wordpress.com/2015/02/02/
command-line-tools-via-python-and-cocoa/](https://managingosx.wordpress.com/2015/02/02/command-line-tools-via-python-and-cocoa/)

[https://managingosx.wordpress.com/2015/02/05/
accessing-more-frameworks-with-python-2/](https://managingosx.wordpress.com/2015/02/05/accessing-more-frameworks-with-python-2/)