py2app Documentation

Release 0.28.6

Ronald Oussoren, Bob Ippolito

Jun 05, 2023

Contents

py2app is a Python setuptools command which will allow you to make standalone application bundles and plugins from Python scripts. py2app is similar in purpose and design to py2exe for Windows.

CHAPTER 1

Contents

1.1 Installation

1.1.1 Installing with pip

To install py2app using pip, or to upgrade to the latest released version of py2app:

```
$ pip3 install -U py2app
```

1.1.2 Installing from source

To install py2app from source, simply use the normal procedure for installing any Python package. Since py2app uses setuptools, all dependencies (including setuptools itself) will be automatically acquired and installed for you as appropriate:

```
$ python setup.py install
```

1.1.3 Upgrade Notes

The `setup.py` template has changed slightly in py2app 0.3 in order to accommodate the enhancements brought on by setuptools. Old `setup.py` scripts look like this:

```
from distutils.core import setup
import py2app

setup(
    app=["myscript.py"],
)
```

New py2app scripts should look like this:

```
from setuptools import setup
setup(
    app=["myscript.py"],
    setup_requires=["py2app"],
)
```

1.2 Tutorial

Converting your scripts to Mac OS X applications is easy with py2app.

1.2.1 Create a setup.py file

The first step is to create a `setup.py` file for your script. `setup.py` is the "project file" that tells setuptools everything it needs to know to build your application. We'll use the *py2applet* script to do that:

```
$ py2applet --make-setup MyApplication.py
Wrote setup.py
```

If your application has an icon (in `.icns` format) or data files that it requires, you should also specify them as arguments to *py2applet*.

1.2.2 Clean up your build directories

Before starting development or switching development modes it's usually a good idea to ensure that your `build` and `dist` directories are cleaned out:

```
$ rm -rf build dist
```

1.2.3 Development with alias mode

Alias mode (the `-A` or `--alias` option) instructs py2app to build an application bundle that uses your source and data files in-place. It does not create standalone applications, and the applications built in alias mode are not portable to other machines. This mode is similar to the setuptools `develop` command, or Xcode's zero-link feature.

To build the application in alias mode, execute `setup.py` with the `py2app` command and specify the `-A` option (or `--alias`):

```
$ python setup.py py2app -A
```

After this, py2app will spit out a bunch of messages to your terminal and you'll end up with new `build` and `dist` folders. The `build` folder contains build sludge that you'll never need to touch, and the `dist` folder contains your application bundle. The application bundle will be named after your script; if your script was named `MyApplication.py`, then your application bundle will be named `MyApplication.app`. Note that Finder displays application bundles without the `.app` extension.

You only need to run this command again when you add data files or change options. Changes to your source code won't require rebuilding!

1.2.4 Running your application

During development, it's often useful to have your application attached to the Terminal. This allows you to better debug it, e.g. by inserting `import pdb; pdb.set_trace()` into your code to inspect it interactively at runtime.

To run your application directly from the Terminal:

```
$ ./dist/MyApplication.app/Contents/MacOS/MyApplication
```

To start your application normally with LaunchServices, you can use the `open` tool:

```
$ open -a dist/MyApplication.app
```

If you want to specify "open document" events, to simulate dropping files on your application, just specify them as additional arguments to `open`.

You may of course also double-click your application from Finder.

When run normally, your application's stdout and stderr output will go to the Console logs. To see them, open the Console application:

```
$ open -a Console
```

1.2.5 Building for deployment

After you've got your application working smoothly in alias mode, it's time to start building a redistributable version. Since we're switching from alias mode to normal mode, you should remove your `build` and `dist` folders as above.

Building a redistributable application consists of simply running the `py2app` command:

```
$ python setup.py py2app
```

This will assemble your application as `dist/MyApplication.app`. Since this application is self-contained, you will have to run the `py2app` command again any time you change any source code, data files, options, etc.

The easiest way to wrap your application up for distribution at this point is simply to right-click the application from Finder and choose "Create Archive".

1.3 Debugging application building

The py2app builder won't always generate a working application out of the box for various reasons. An incomplete build generally results in an application that won't launch, most of the time with a generic error dialog from py2app.

The easiest way to debug build problems is to start the application directly in the Terminal.

Given an application "MyApp.app" you can launch the application as follows:

```
$ dist/MyApp.app/Contents/MacOS/MyApp
```

This will start the application as a normal shell command, with output from the application (both stdout and stderr) shown in the Terminal window.

Some common problems are:

- An import statement fails due to a missing module or package

This generally happens when the dependency cannot be found by the source code analyzer, either due to dynamic imports (using `__import__()` or `importlib` to load a module), or due to imports in a C extension.

In both cases use `--includes` or `--packages` to add the missing module to the application.

If this is needed for a project on PyPI: Please file a bug on GitHub, that way we can teach py2app to do the right thing.

- C library cannot find resources

  This might happen when a C library looks for resources in a fixed location instead to looking relative to the library itself. There are often APIs to tell the library which location it should use for resources.

  If this needed for a project on PyPI: Please file a bug on GitHub, including the workaround, that way we can teach py2app to the the right thing.

1.4 Dependencies

Note that these dependencies should automatically be satisfied by the installation procedure and do not need to be acquired separately.

setuptools: setuptools provides enhancements to distutils.

macholib: macholib reads and writes the Mach-O object file format. Used by py2app to build a dependency graph of dyld and framework dependencies for your application, and then to copy them into your application and rewrite their load commands to be `@executable_path` relative. The end result is that your application is going to be completely standalone beyond a default install of Mac OS X. You no longer have to worry about linking all of your dependencies statically, using install_name_tool, etc. It's all taken care of!

modulegraph: modulegraph is a replacement for the Python standard library modulefinder. Stores the module dependency tree in a graph data structure and allows for advanced filtering and analysis capabilities, such as GraphViz dot output.

altgraph: altgraph is a fork of Istvan Albert's graphlib, and it used internally by both macholib and modulegraph. It contains several small feature and performance enhancements over the original graphlib.

1.5 Environment in launched applications

1.5.1 Environment variables added by py2app

- `RESOURCEPATH`

  Filesystem path for the "Resources" folder inside the application bundle

1.5.2 System environment

When the application is launched normally (double clicking in the Finder, using the `open(1)` command) the application will be launched with a minimal shell environment, which does not pick up changes to the environment in the user's shell profile.

The "emulate_shell_environment" option will run a login shell in the background to fetch exported environment variables and inject them into your application.

It is also possible to inject extra variables into the environment by using the `LSEnvironment` key in the Info.plist file, for example like so:

```
setup(
    name='BasicApp',
    app=['main.py'],
    options=dict(py2app=dict(
        plist=dict(
            LSEnvironment=dict(
                LANG='nl_NL.latin1',
                LC_CTYPE='nl_NL.UTF-8',
                EXTRA_VAR='hello world',
                KNIGHT='ni!',
            )
           )
    )),
)
```

1.6 Frequently Asked Questions

- "Mach-O header may be too large to relocate"

  Py2app will fail with a relocation error when it cannot rewrite the load commands in shared libraries and binaries copied into the application or plugin bundle.

  This error can be avoided by rebuilding binaries with enough space in the Mach-O headers, either by using the linker flag "-headerpad_max_install_names" or by installing shared libraries in a deeply nested location (the path for the install root needs to be at least 30 characters long).

- M1 Macs and libraries not available for arm64

  A lot of libraries are not yet available as arm64 or universal2 libraries.

  For applications using those libraries you can create an x86_64 (Intel) application instead:

  1. Create a new virtual environment and activate this
  2. Use `arch -x86_64 python -mpip install ...` to install libraries.

     The `arch` command is necessary here to ensure that pip selects variants that are compatible with the x86_64 architecture instead of arm64.
  3. Use `arch -x86_64 python setup.py py2app --arch x86_64` to build

  This results in an application bundle where the launcher is an x86_64 only binary, and where included C extensions and libraries are compatible with that architecture as well.

- Using Cython with py2app

  Cython generates C extensions. Because of that the dependency walker in py2app cannot find import statements in ".pyx" files".

  To create working applications you have to ensure that dependencies are made visible to py2app, either by adding import statements to a python file that is included in the application, or by using the "includes" option.

  See examples/PyQt/cython_app in the repository for an example of the latter.

- Dark mode support

  Note: As of py2app 0.26 the stub executables are compiled with a modern SDK, with an automatic fallback to the older binaries for old builds of Tkinter.

The stub executables from py2app were compiled on an old version of macOS and therefore the system assumes that applications build with py2app do not support Dark Mode unless you're building a "Universal 2" or "Apple Silicon" application.

To enable Dark Mode support for other builds of Python you need to add a key to the Info.plist file. The easiest way to do this is using the following option in setup.py:

```
setup(
    ...
    options=dict(
      py2app=dict(
        plist=dict(
          NSRequiresAquaSystemAppearance=False
        )
      )
    )
)
```

1.7 Tweaking your Info.plist

It's often useful to make some tweaks to your Info.plist file to change how your application behaves and interacts with Mac OS X. The most complete reference for the keys available to you is in Apple's Runtime Configuration Guidelines.

1.7.1 Commonly customized keys

Here are some commonly customized property list keys relevant to py2app applications:

CFBundleDocumentTypes: An array of dictionaries describing document types supported by the bundle. Use this to associate your application with opening or editing document types, and/or to assign icons to document types.

CFBundleGetInfoString: The text shown by Finder's Get Info panel.

CFBundleIdentifier: The identifier string for your application (in reverse-domain syntax), e.g. `"org.pythonmac.py2app"`.

CFBundleURLTypes: An array of dictionaries describing URL schemes supported by the bundle.

LSBackgroundOnly: If `True`, the bundle will be a faceless background application.

LSUIElement: If `True`, the bundle will be an agent application. It will not appear in the Dock or Force Quit window, but still can come to the foreground and present a UI.

NSServices: An array of dictionaries specifying the services provided by the application.

1.7.2 Specifying customizations

There are three ways to specify `Info.plist` customizations to py2app.

You can specify an Info.plist XML file on the command-line with the `--plist` option, or as a string in your `setup.py`:

```
setup(
    app=['MyApplication.py'],
options=dict(py2app=dict(
```

(continues on next page)

(continued from previous page)

```
    plist='Info.plist',
)),
)
```

You may also specify the plist as a Python dict in the `setup.py`:

```
setup(
    app=['MyApplication.py'],
options=dict(py2app=dict(
    plist=dict(
        LSPrefersPPC=True,
    ),
)),
)
```

Or you may use a hybrid approach using the standard library plistlib module:

```
from plistlib import Plist
plist = Plist.fromFile('Info.plist')
plist.update(dict(
    LSPrefersPPC=True,
))
setup(
    app=['MyApplication.py'],
options=dict(py2app=dict(
    plist=plist,
)),
)
```

1.7.3 Universal Binaries

Note: the documentation about universal binaries is outdated!

py2app is currently fully compatible with Universal Binaries, however it does not try and detect which architectures your application will correctly run on.

If you are building your application with a version of Python that is not universal, or have extensions that are not universal, then you must set the `LSPrefersPPC` Info.plist key to `True`. This will force the application to run translated with Rosetta by default. This is necessary because the py2app bootstrap application is universal, so Finder will try and launch natively by default.

Alternatively, the `--prefer-ppc` option can be used as a shortcut to ensure that this Info.plist key is set.

1.8 Example setup.py templates

1.8.1 Basic

The simplest possible `setup.py` script to build a py2app application looks like the following:

```
"""
py2app build script for MyApplication

Usage:
    python setup.py py2app
"""
from setuptools import setup
setup(
    app=["MyApplication.py"],
setup_requires=["py2app"],
)
```

The *py2applet* script can create `setup.py` files of this variety for you automatically:

```
$ py2applet --make-setup MyApplication.py
```

1.8.2 Cross-platform

Cross-platform applications can share a `setup.py` script for both py2exe and py2app. Here is an example `setup.py` that will build an application on Windows or Mac OS X:

```
"""
py2app/py2exe build script for MyApplication.

Will automatically ensure that all build prerequisites are available
via ez_setup

Usage (Mac OS X):
    python setup.py py2app

Usage (Windows):
    python setup.py py2exe
"""
import sys
from setuptools import setup

mainscript = 'MyApplication.py'

if sys.platform == 'darwin':
    extra_options = dict(
        setup_requires=['py2app'],
        app=[mainscript],
        # Cross-platform applications generally expect sys.argv to
        # be used for opening files.
        # Don't use this with GUI toolkits, the argv
        # emulator causes problems and toolkits generally have
        # hooks for responding to file-open events.
        options=dict(py2app=dict(argv_emulation=True)),
    )
elif sys.platform == 'win32':
    extra_options = dict(
        setup_requires=['py2exe'],
        app=[mainscript],
    )
else:
    extra_options = dict(
```

(continues on next page)

(continued from previous page)

```
        # Normally unix-like platforms will use "setup.py install"
        # and install the main script as such
        scripts=[mainscript],
    )

setup(
    name="MyApplication",
    **extra_options
)
```

1.9 py2app Options

Options can be specified to py2app to influence the build procedure in three different ways:

At the command line:

```
$ python setup.py py2app --includes=os,platform
```

In your `setup.py`:

```
setup(
    app=['MyApplication.py'],
    options=dict(py2app=dict(
        includes=["os", "platform"]
    )),
)
```

In a `setup.cfg` file:

```
[py2app]
includes=os,platform
```

Note that when translating command-line options for use in `setup.py`, you must replace hyphens (`-`) with underscores (_). `setup.cfg` files may use either hyphens or underscores, but command-line options must always use the hyphens.

Lists of values are a comma seperated sequence of names on the command-line and in setup.cfg, and regular python lists in setup.py (as shown in the earlier example).

1.9.1 Option Reference

To enumerate the options that py2app supports, use the following command:

```
$ python setup.py py2app --help
```

Options for 'py2app' command:

Table 1: Options

Command-line	Setup.py	Value	Description
`--optimize`	optimize	level (integer)	Specifies the optimization level for the Pytho interpreter level 0 to disable, level 1 for `python -O`, and level 2 for `python -OO`. Defaults to the optimation level of the process running py2app.
`--includes`	includes	list of module names	A list of Python modules to include even if they are not detected by dependency checker. Packages in this list are ignored.
`--packages`	packages	list of package names	A list of Python packages to include even if they are not detected by dependency checker. The whole package will be included.
`--excludes`	excludes	list of module or package names	A list of Python modules or packages to exclude even if they are detected by dependency checker.
`--matplotlib-backends`	matplotlib_backends	list of matplotlib backend names	The matplotlib backends that will be included when matplotlib is one of the included libraries. The default is to include all of matplotlib. Use '*' to include all backends, and "-" to only include backends that are explicitly included.
`--qt-plugins`	qt_plugins	List of Qt plugins	Specifies plugins to include in an application using PyQt4.
`--dylib-excludes`	dylib_excludes	A list of shared libraries or frameworks	The specified libraries and frameworks will not be included in the output.
`--frameworks`	frameworks	A list of shared libraries or frameworks	The specified libraries and frameworks will be included in the output.
`--iconfile`	iconfile	Path the the icon file	Specify the icon to use for the application, the ".icns" suffix may be left off. The default is to use a generic icon.
`--plist`	plist	Path to a plist template, or (in setup.py) a Python dictionary.	Specify the contents of the Info.plist. Py2app will add some information to the file when it is copied into the output.
`--datamodels`	datamodels	List of xcdatamodels	The specified xcdatamodel files will be compiled and included into the bundle Resources
`--mappingmodels`	mappingmodels	List of xcmappingmodels	The specified xcmappingmodel files will be compiled and included into the bundle Resources
`--resources`	resources	List of files and folders	Specifies additional files and folders to include in the bundle Resource. Do not use this to copy additional code.
`--extension`	extensionn	file extension, includding the dot	The extension to use of the output, defaults to ".app" for applications and ".plugin" for plugins. Commonly only used for plugins.
`--arch`	arch	"intel", "fat", "universal", "universal2", "i386", "x86_64", "ppc"	The (set of) architecture(s) to use for the main executable in the output. This should be a subset of the architectures supported by the python interpreter.
`--no-strip`	no_strip	None (use `True` in setup.py)	Don't strip debug information and local symbols from the output. Default is to strip.

Continued on next page

Table 1 – continued from previous page

Command-line	Setup.py	Value	Description
`--semi-standalone`	semi_standalone	None (use `True` in setup.py)	Create output that depends on an existing installation of Python, but does contain all code and dependencies.
`--alias`	alias	None (use `True` in setup.py)	Create output that depends on an existing installation of Python and uses the sources outside of the bundle. This is only useful during development, you can update source files and relaunche the application without re-building the bundle. **Do not use for distribution**
`--graph`	•	None	Emit a ".dot" file with the module dependency graph after the build. The output will be stored next to the regular output.
`--xref`	xref	None	Emit a module cross reference as HTML. The output will be stored next to the regular output.
`--report-missing-from-imports`	•	None (use `True` in setup.py)	Include a list of missing names for `from module import name` in the output at the end of the py2app run.
`--no-report-missing-conditional-import`	•	None	Do not include missing modules that might be conditionally imported in the output at the end of the py2app run.
`--use-faulthandler`	use_faulthandler	None (use `True` in setup.py)	Enable the Python faulthandler, requires Python 3.3 or later.
`--no-chdir`	no_chdir	None	Don't change the working directory to the bundle Resource directory. This option is always enabled in plugins.
`--argv-emulation`	argv_emulation	None (use `True` in setup.py)	Fill `sys.argv` during program launch. The argv emulator runs a small event loop during program launch to intercept file-open and url-open events. The to-be-opened resources will be added to `sys.argv` **WARNING**: Do no use this option when the program uses a GUI toolkit. The emulator tends to confuse GUI toolkits, and most GUI toolkits have APIs to react to these events at runtime (for example to open a file when your program is already running). This option cannot be enabled for plugins.
`--emulate-shell-environment`	emulate_shell_environment	None (use `True` in setup.py)	Set up environment variables as if the program was launched from a fresh Terminal window. Don't use this with plugins. By default applications inherit the environment from the application launcher (when double clicking the application in the Finder), which is does not include environment variables set in the users shell profile. Only use this when the application needs to access environment varialbes set in the Terminal. This option is not meant for general use.
`--use-pythonpath`	use_pythonpath	None (use `True` in setup.py)	Allow the PYTHONPATH environment varialble to affect the interpreter's search path. This is generally not useful, PYTHONPATH is not included in the minimal shell environment used by the application launcher.

Continued on next page

Table 1 – continued from previous page

Command-line	Setup.py	Value	Description
`--site-packages`	site_packages	None (use `True` in setup.py)	Include the system and user site-packages in `sys.path` Note that this makes the bundle less standalone, packages installed on a users's system may affect the bundle.
`--extra-scripts`	extra_scripts	List of file names for scripts	The mentioned scripts will be included in the `Contents/MacOS`. For Python scripts the file in `Contents/MacOS` will be a binary that launches the script using the Python interpreter and environment from the bundle.
`--argv-inject`	argv_inject	values to inject, a single string will be split using `shlex.split`	The values will be inserted in to `sys.argv` after `argv[0]`.
`--bdist-base`	bdist_base	directory name	base directory for build library (default is build)
`--dist-dir`	dist_dir	directory name	directory to put the final built distributions in (default is dist)
`--include-plugins`	include_plugins	List of plugin bundles	The plugin bundles will be copied into the application bundle at the expected location for the type of plugin
`--redirect-stdout-to-asl`	redirect_stdout_to_asl	None (use `True` in setup.py)	Forward the stdout/stderr streams to Console.app using ASL
`--force-system-tk`	force_system_tk	None (use `True` in setup.py)	Ensures that Tkinter will be linked to the system copy of Tcl and Tk. This makes the bundle smaller, but the system version of Tcl/Tk is ancient an buggy. Don't use this option. **This is a legacy option that will be dropped in a future version**
`--prefer-ppc`	prefer_ppc	None (use `True` in setup.py)	Force the application to run translated on i386 **This is a legacy option that will be dropped in a future version**
`--debug-modulegraph`	debug_modulegraph	None (use `True` in setup.py)	Drop into the pdb debugger after building the module graph *This is an development option*
`--debug-skip-macholib`	debug_skip_macholib	None (use `True` in setup.py)	Don't run macholib. The output will not be standalone. *This is an development option*

Options to specify which objects to include or exclude (the first part of the table above) are used to finetune the behaviour of py2app and should generally not be necessary. Please file an issue on the py2app tracker if a package on PyPI requires one of these options, which allows me to change py2app to do the right thing automatically.

1.10 Recipes

py2app includes a mechanism for working around package incompatibilities, and stripping unwanted dependencies automatically. These are called recipes.

A future version of py2app will support packaging of Python Eggs. Once this is established, recipes will be obsolete since eggs contain all of the metadata needed to build a working standalone application.

1.10.1 Common causes for incompatibility

Some Python packages are written in such a way that they aren't compatible with being packaged. There are two main causes of this:

- Using `__import__` or otherwise importing code without usage of the `import` statement.
- Requiring in-package data files

1.10.2 Built-in recipes

`cjkcodecs`: All codecs in the package are imported.

`docutils`: Several of its internal components are automatically imported (`languages`, `parsers`, `readers`, `writers`, `parsers.rst.directives`, `parsers.rst.langauges`).

`matplotlib`: A dependency on `pytz.zoneinfo.UTC` is implied, and the `matplotlib` package is included in its entirety out of the zip file.

`numpy`: The `numpy` package is included in its entirety out of the zip file.

`PIL`: Locates and includes all image plugins (Python modules that end with `ImagePlugin.py`), removes unwanted dependencies on `Tkinter`.

`pydoc`: The implicit references on the several modules are removed (`Tkinter`, `tty`, `BaseHTTPServer`, `mimetools`, `select`, `threading`, `ic`, `getopt`, `nturl2path`).

`pygame`: Several data files that are included in the zip file where `pygame` can find them (`freesansbold.ttf`, `pygame_icon.tiff`, `pygame_icon.icns`).

`PyOpenGL`: If the installed version of PyOpenGL reads a `version` file to determine its version, then the `OpenGL` package is included in its entirety out of the zip file.

`scipy`: The `scipy` and `numpy` packages are included in their entirety out of the zip file.

`sip`: If `sip` is detected, then all sip-using packages are included (e.g. PyQt).

1.10.3 Developing Recipes

py2app currently searches for recipes only in the `py2app.recipes` module. A recipe is an object that implements a `check(py2app_cmd, modulegraph)` method.

`py2app_cmd`: The py2app command instance (a subclass of `setuptools.Command`). See the source for `py2app.build_app` for reference.

`modulegraph`: The `modulegraph.modulegraph.ModuleGraph` instance.

A recipe should return either `None` or a `dict` instance.

If a recipe returns `None` it should not have performed any actions with side-effects, and it may be called again zero or more times.

If a recipe returns a `dict` instance, it will not be called again. The returned `dict` may have any of these optional string keys:

`filters`: A list of filter functions to be called with every module in the modulegraph during flattening. If the filter returns False, the module and any of its dependencies will not be included in the output. This is similar in purpose to the `excludes` option, but can be any predicate (e.g. to exclude all modules in a given path).

loader_files: Used to include data files inside the `site-packages.zip`. This is a list of 2-tuples: `[(subdir, files), ...]`. `subdir` is the path within `site-packages.zip` and `files` is the list of files to include in that directory.

packages: A list of package names to be included in their entirety outside of the `site-packages.zip`.

prescripts: A list of additional Python scripts to run before initializing the main script. This is often used to monkey-patch included modules so that they work in a frozen environment. The prescripts may be module names, file names, or file-like objects containing Python code (e.g. StringIO). Note that if a file-like object is used, it will not currently be scanned for additional dependencies.

1.11 Implementation Details

For those interested in the implementation of py2app, here's a quick rundown of what happens.

1.11.1 Argument Parsing

When `setup.py` is run, the normal setuptools / distutils `sys.argv` parsing takes place.

1.11.2 Run build command

The `build` command is run to ensure that any extensions specified in the `setup.py` will be built prior to the `py2app` command. The build directory will be added to `sys.path` so that `modulegraph` will find the extensions built during this command.

1.11.3 Dependency resolution via modulegraph

The main script is compiled to Python bytecode and analyzed by modulegraph for `import` bytecode. It uses this to build a dependency graph of all involved Python modules.

The dependency graph is primed with any `--includes`, `--excludes`, or `--packages` options.

1.11.4 Apply recipes

All of the *Recipes* will be run in order to find library-specific tweaks necessary to build the application properly.

1.11.5 Apply filters

All filters specified in recipes or otherwise added to the py2app Command object will be run to filter out the dependency graph.

The built-in filter `not_system_filter` will always be run for every application built. This ensures that the contents of your Mac OS X installation (`/usr/`, `/System/`, excluding `/usr/local/`) will be excluded.

If the `--semi-standalone` option is used (forced if a vendor Python is being used), then the `not_stdlib_filter` will be automatically added to ensure that the Python standard library is not included.

1.11.6 Produce graphs

If the `--xref` or `--graph` option is used, then the `modulegraph` is output to HTML or GraphViz respectively. The `.html` or `.dot` file will be in the `dist` folder, and will share the application's name.

1.11.7 Create the .app bundle

An application bundle will be created with the name of your application.

The `Contents/Info.plist` will be created from the `dict` or filename given in the `plist` option. py2app will fill in any missing keys as necessary.

A `__boot__.py` script will be created in the `Contents/Resources/` folder of the application bundle. This script runs any prescripts used by the application and then your main script.

If the `--alias` option is being used, the build procedure is finished.

The main script of your application will be copied *as-is* to the `Contents/Resources/` folder of the application bundle. If you want to obfuscate anything (by having it as a `.pyc` in the zip), then you *must not* place it in the main script!

Packages that were explicitly included with the `packages` option, or by a recipe, will be placed in `Contents/Resources/lib/python2.X/`.

A zip file containing all Python dependencies is created at `Contents/Resources/Python/site-packages.zip`.

Extensions (which can't be included in the zip) are copied to the `Contents/Resources/lib/python2.X/lib-dynload/` folder.

1.11.8 Include Mach-O dependencies

macholib is used to ensure the application will run on other computers without the need to install additional components. All Mach-O files (executables, frameworks, bundles, extensions) used by the application are located and copied into the application bundle.

The Mach-O load commands for these Mach-O files are then rewritten to be `@executable_path/../Frameworks/` relative, so that dyld knows to find them inside the application bundle.

`Python.framework` is special-cased here so as to only include the bare minimum, otherwise the documentation, entire standard library, etc. would've been included. If the `--semi-standalone` option or a vendor Python is used, then the `Python.framework` is ignored. All other vendor files (those in `/usr/` or `/System/` excluding `/usr/local/`) are also excluded.

1.11.9 Strip the result

Unless the `--no-strip` option is specified, all Mach-O files in the application bundle are stripped using the `strip` tool. This removes debugging symbols to make your application smaller.

1.11.10 Copy Python configuration

This only occurs when not using a vendor Python or using the `--semi-standalone` option.

The Python configuration, which is used by `distutils` and `pkg_resources` is copied to `Contents/Resources/lib/python2.X/config/`. This is needed to acquire settings relevant to the way Python was built.

1.12 py2applet

The `py2applet` script can be used either to create an application quickly in-place, or to generate a `setup.py` file that does the same.

In normal usage, simply run `py2applet` with the options you would normally pass to the `py2app` command, plus the names of any scripts, packages, icons, plist files, or data files that you want to generate the application from.

The first `.py` file is the main script. The application's name will be derived from this main script.

The first `.icns` file, if any, will be used as the application's icon (equivalent to using the `--iconfile` option).

Any folder given that contains an `__init__.py` will be wholly included as out of the zip file (equivalent to using the `--packages` option).

Any other file or folder will be included in the `Contents/Resources/` directory of the application bundle (equivalent to the `--resources` option).

If `--make-setup` is passed as the first option to `py2applet`, it will generate a `setup.py` file that would do the above if run. This can be used to quickly generate a `setup.py` for a new project, or if you need to tweak a few complex options. The *Tutorial* demonstrates this functionality.

1.13 Release history

1.13.1 py2app 0.28.6

- Fix support for Python 2.7

  These are best-effort changes, I no longer have a setup where I can perform a good test run for Python 2.7.

- Introduce support for Python 3.12

1.13.2 py2app 0.28.5

- #476: Update black recipe

  The black recipe no longer worked with recent versions of black due to relying on a metadata file from the "egg" spec that's not included by black's current build tool.

  The recipe now scans the python code that's next to the mypyc compiled extension modules for dependencies and uses that to update the dependency graph. This should ensure that new dependencies of black will be automaticly detected in the future.

- Update wheel dependencies

1.13.3 py2app 0.28.4

- Fix incompatibility with Python 3.11

1.13.4 py2app 0.28.3

- #453: Fix crash in py2applet when specifying a directory to include in the application bundle.

1.13.5 py2app 0.28.2

- Fix incompatibility with recent setuptools

1.13.6 py2app 0.28.1

- #448: Fix typo in qt6 recipe
- #444: Fix issue where the standard output and standard error streams are set to non-blocking when using py2app.

  For some reason the "ibtool" command (part of Xcode) sets these streams to non-blocking when compiling NIB files. I've added a context manager that resets the non-blocking status of these streams.
- PR #446: Fix Qt5 recipe for newer versions of PyQt5

  PR by kangi.
- #447: Fix error when using `py2applet --help`

  Bug was introduced in the fix for #414

1.13.7 py2app 0.28

Note: This is the last version of py2app with compatibility with Python 2.7. Future versions will require Python 3.6 or later.

- PR #410: Fix typo in NamedTemporyFile call

  PR by MAKOMO
- #414 Workaround for autodiscovery in setuptools 61.0

  Setuptools 61.0 introduces autodiscovery of distribution attributes, and that broke py2app. This version introduces a `setuptools.finalize_distribution_options` entrypoint in py2app that will set the distributions's *name* and *py_modules* attributes in a way that is compatible with the main code of py2app when they are not yet set (before autodiscovery kicks in).

  In older versions of py2app buildin an app can fail in two ways with setuptools 61.0 or later:

  - The name of the generated application is not based on the script name, but some other value.
  - Calling `python setup.py py2app` results in an error mentioning `Multiple top-level modules discovered`.
- PR #418: Add recipe for black

  PR by mrclary
- #417: Also include package dist-info for editable installs
- The qt5 and qt6 recipes used dodge logic to detect if the Qt library itself is inside the python package, resulting in duplicate copies of Qt.
- #406: Fix incompatibility with python 2.7

  py2app 0.24 accidently broke compatibility with Python 2.7, and this release fixes this.

  This is the last release with Python 2.7 support, the next release will contain package metadata that ensures it can only be installed on Python 3.

- #413: Find dist-info in included pythonXX.zip

  By default the `working_set` of pkg_resources does not contain distribution information from packages included in zip files, such as the zipped-up stdlib + site-pakckages in py2app bundles.

  Add some monkey patching to apps using `pkg_resources` to fix this.

- Fix hard crash in "rtree" recipe when the package contents doesn't match the recipe expectations.
- #408: Add definition of `site.PREFIXES`
- #412: Fix incompatibility with setuptools 60.8.1

  The setuptools recipe did not recoginize all vendored dependencies in `pkg_resources` and that breaks app bundles that use `pkg_resoures`.

- PR #388: Add builtin definitions for 'quit' and 'exit' in site.py

  PR by mcclary

- PR #388: Set "ENABLE_USER_SITE=False" in site.py

  PR by mcclary

- PR #396: Update pygame recipe to remove missing icon

  PR by glyph

1.13.8 py2app 0.27

- #377: The qt5 and qt6 recipes caused a py2app crash when the PyQt5 or PyQt6 is not installed.
- #401: Fix incompatibility with setuptools 60.7 and later
- #391: Drop usage of tempfile.mktemp
- #387: Add `site.ENABLE_USER_SITE` in the site.py file for applications (value is always `False`).

1.13.9 py2app 0.26.1

- #374: Actually ship the "old" stub executables introduced in version 0.26

1.13.10 py2app 0.26

- Stub executables were recompiled on macOS 11

  This means support for light mode/dark mode should now work out of the box.

  The old stub executables are still used when detecting that Tkinter is used with an old build of Tk.

- #1: Include ".egg-info" and ".dist-info" information in the bundled application

  This fixes any python package that uses `pkg_resources` to look for specific distributions.

- `py2app.filters.not_stdlib_filter` now knows about Python's "venv"
- #368: Add recipe "detect_dunder_file"

  This recipe will ensure that a Python package is stored outside of site-packages.zip when a module in that package uses the `__file__` variable.

  This variable is most commonly used to load resources stored in the package (instead of the newer `importlib.resources` and `pkg_resources` libraries).

- #339: Add recipe for pydantic

  The recipe is needed because pydantic uses Cython to compile all sources (including the package `__init__`) and therefore hides imports from the dependency analyzer.

- #338: Add "imageio_ffmpeg" to autopackages
- PR367: Add recipes for pandas, pylsp, and zmq
- PR367: Add docutils and pylint to autopackages

  PR by Ryan Clary (mrclary on GitHub)

- #344: Invocation of codesign on the whole bundle sometimes fails

  Py2app will now try this a number of times before giving up. This is at best a workaround for and doesn't completely fix the problem.

- #370: py2app now works with Python 3.10

  Python 3.10 no longer exports a (private) symbol used by the py2app stub executable. Switched to a public API to accomplish the same task where available.

- #110: Add recipe for SQLAlchemy

  The recipe includes all dialects and connectors, including implicit dependencies, because SQLAlchemy uses `__import__` to load dependencies.

- #328: Add recipe for gcloud
- #195: Add `USER_BASE`, `getuserbase()` and `getusersitepackages()` to py2app's version of `site.py`.
- #184: Add recipe for 'ssl'

  This recipe is only used for Python 3.4 or later and ensures that the CA bundle used by Python's ssl module is included in the app bundle and OpenSSL is configured to look for that bundle in the application bundle.

- #371: change default error message on launch problems

  The default error message shown when the application cannot be launched is now slightly more useful and refers the py2app debug page.

- #345, #169: Adjust qt5 and qt6 recipes for non-PyPI installations

  The qt5 and qt6 recipes now should work when the Qt installation prefix is outside of the PyQt package, for example when PyQt was installed through homebrew.

  I've tested this for PyQt5 and made the same change to the PyQt6 recipe, although I haven't tested that change.

1.13.11 py2app 0.25

- #358: Add recipe for multiprocessing
- PR363: Add recipe for platformdirs

  PR by Ryan Clary (mrclary on GitHub)

- PR353: Add recipe for sphinx

  PR by Ryan Clary (mrclary on GitHub)

- PR352: Fix for using ipython

  PR by Ryan Clary (mrclary on GitHub)

- PR351: Tweak the matplotlib recipe

  PR by Ryan Clary (mrclary on GitHub)

- PR348: Fix for checking for dead symlinks links in py2app

  PR by Oliver Cordes (ocordes on GitHub)

- #354: Fix buggy "autopackages" and "automissing" recipes
- #350: Add sentencepiece to the autopackages list
- #359: Add recipe for PyQt6
- #349: Add recipe for OpenCV (opencv-python, `import cv2`)
- PR365: Add RTree recipe

  PR by Ryan Clary (mrclary on GitHub)

1.13.12 py2app 0.24

- Consolidate recipes that just include a package as is into a single recipe to reduce code complexity.
- Consolidate recipes that just mark imports as expected missing into a single recipe to reduce code complexity.
- #334: Include binary stubs for Universal 2 and arm64 binaries in the archives

  The files were in the repository, but were excluded from the source and wheel archives.

1.13.13 py2app 0.23

- #315: Stub executables have an LC_RPATH that points to the Frameworks folder

  PR by Aleksandar Topuzović (atopuzov)

- #322: Port wxPython examples to 4.0

  PR by Hamish Mcintyre-Bhatty (hamishmb)

- #314: Don't use Image.DEBUG in the PIL recipy, that attribute is not longer valid

  PR by Aleksandar Topuzović

- #320: Process "@loader_path" in load commands

  A popular pattern in C extensions with bindins to C library on PyPI is to copy those C libraries into the wheel and reference those using an "@loader_path" linker command in the C extension. Until this release py2app could not process those linker commands correctly.

- #298: Add recipe for pycryptodome
- #282: Add recipe for h5py
- #283: Add recipe for tensorflow

  The recipe just includes the entire package into the generated app bundle, I haven't checked yet if there is a way to reduce the size of this package (which is rather huge).

1.13.14 py2app 0.22

- #319: Add ad-hoc signature for application bundles

  ARM64 binaries on macOS 11 must be signed, even if it is only an ad-hoc signature. py2app will now add an ad-hoc code signature.

- #300: Add support for ARM64 and Universal 2 binaries

  Note: Support is highly experimental, these stubs have not been tested yet.

- #299: Fix build error when building with the copy of Python 3 shipped with Xcode.
- #281: Generated bundle doesn't work on macOS 10.9 and 10.10.

1.13.15 py2app 0.21

- PR 277 (Christian Clauss): Fix some Python 3 issues
- #276: Rebuilt the binary stubs on a 10.12 machine to fix launching

1.13.16 py2app 0.20

- Migrate to GitHub
- #274: Fix an issue in the PyQt5 recipe
- Fix issue with emulate-shell-environment option on macOS 10.15 (Catalina)
- #269: Py2app didn't work with Python 3.8

1.13.17 py2app 0.19

- #251: Add recipe for "botocore"
- #253: "python setup.py py2app -A" creates invalid bundle from "venv" virtual environments
- Updated recipe for PySide2 and new recipe for Shiboken2

  Patch by Alberto Sottile.

1.13.18 py2app 0.18

- #250: Add recipe for "six.moves", which also works when the six library is vendored by other packages

1.13.19 py2app 0.17

- #247: The new tkinter recipe didn't work properly for installations that do use a framework install of Tcl/Tk.

1.13.20 py2app 0.16

- #244: Copy the Tcl/Tk support libraries into the application bundle for Python builds using a classic unix install of Tcl/Tk instead of a framework build.

  This results in working app bundles when a Python.org installation that includes Tcl/Tk (such as Python 3.7).

- Don't copy numpy into application just because the application uses Pillow.
- Add recipe for Pyside

  Patch by Alberto Sottile

1.13.21 py2app 0.15

- Fixed issues for Python 3.7, in particular changes in the plistlib library (Issue #242, #239)
- Updated dependencies on macholib, altgraph and modulegraph

Due to a bug in CPython 3.7.0 using -O does not work with that version of CPython

1.13.22 py2app 0.14.1

- Updated dependencies
- Updated PyPI metadata

1.13.23 py2app 0.14

Features:

- Started using flake8 to improve coding style

Bug fixes:

- Issue #222: The fix for issue #179 broke the argv emulator
- Issue #226: Py2app could fail while reporting on possibly missing modules
- Issue #228: The python executable included in the app bundle as `sys.exectuable` was not executable

1.13.24 py2app 0.13

Bug fixes:

- Issue 185 in PyObjC's tracker: sysconfig using `__import__` in Python 3.6 or later, which confuses modulegraph.
- Pull request #17: Location of site-packages in the "–user" location has changed

  Patch by Matt Mukerjee

Features:

- (None yet)

1.13.25 py2app 0.12

- Pull request #15 by Armin Samii: Safer symlink and file copying
- Update recipes: a number of recipe names conflicted with toplevel modules imported by recipes. This causes problems on Python 2.7 (without absolute imports)

1.13.26 py2app 0.11

- Make sure the stdout/stderr streams of the main binary of the application are unbuffered.

  See issue #177 in PyObjC's repository for more information.

- Fix issue #201: py2app is not compatible with pyvenv virtualenvs

  With additional fix by Oskari Timperi.

- Fix issue #179: the stdout/stderr streams are no longer forwarded to console.app using ASL (by default), use "–redirect-stdout-to-asl" to enable the redirection functionality.

  Note that for unclear reasons the redirection doesn't work on OSX 10.12 at the moment.

- Fix issue #188: Troubles with lxml.isoschematron

  The package 'lxml.isoschematron' is not zip-safe and tries to load resources using the normal filesystem APIs, which doesn't work when the package is part of a zipfile.

- py2applet now longer uses "argv_emulation" by default, that results in too many problems.
- Issue #174: clean up the summary about missing modules by removing warnings about things that aren't modules.

  Also notes when an module is likely an alias for some other module. These changes should remove a lot of false postive warnings from the output of py2app.

- Fix issue #161: opengl recipe uses "file" function that isn't present on Python 3
- Add "qt5" recipe that does the right thing for the PyQt5 wheel on PyPI (tested with PyQt5 5.6)
- Add support for "@loader_path" in the link commands of C extension.

  This makes it possible to use wheels that were processed by delocate-listdeps when building application bundles.

- Do not report imports that are expected to be missing

  Patch by Barry Scott.

1.13.27 py2app 0.10

- The recipe for virtualenv calls a modulegraph method that was made private in a recent release and hence no longer worked with py2app 0.9.

  Update the recipe to work around this.

1.13.28 py2app 0.9

- issue #146, #147: The "python" binary in MyApp.app/Contents/MacOS was the small stub exetable from framework builds, instead of the actual command-line interpreter. The result is that you couldn't use `sys.executable` to start a new interpreter, which (amongst others) breaks multiprocessing.
- pull request #7: Add support for PyQt5 to the sip recipe. Patch by Mark Montague.

- pull request #4: Copying PySide plugins was broken due to bad indentation.
- pull request #5: py2app was broken for python versions that don't use _sysconfigdata.
- issue #135: Don't sleep for a second after compiling a XIB file
- issue #134: Remove target location before copying files into the bundle.
- issue #133: Ensure that the application's "Framework" folder is on the search path for `ctypes.util.find_library`.
- issue #132: Depend on modulegraph 0.12 to avoid build errors when the python code contains references to compatibility modules that contain SyntaxErrors for the current python version.
- Explicitly report modules that cannot be found at the end of the run (for non-alias builds)

  Note: This is just a warning, missing modules are not necessarily a problem because modulegraph can detect imports for modules that aren't used on OSX (for example)
- Report modules that contain syntax errors at the end of the run (for non-alias builds)

  Note: This is just a warning, syntax errors be valid when the dependency tree contains modules for the other major release of python (e.g a compat_py2 module that contains compatibility code for Python 2 and contains code that isn't valid Python 3)

1.13.29 py2app 0.8.1

- Loading scripts didn't work when –no-chdir was used

  Reported by Barry Scott in private mail.

1.13.30 py2app 0.8

py2app 0.8 is a feature release

- Fixed argv emulator on OSX 10.9, the way the code detected that the application was launched through the Finder didn't work on that OSX release.
- The launcher binary is now linked with Cocoa, that should avoid some problems with sandboxed applications (in particular: standard open panels don't seem to work properly in a sandboxed application when the main binary is not linked to AppKit)
- Don't copy Python's Makefile, Setup file and the like into a bundle when sysconfig and distutils.sysconfig don't need these files (basicly, when using any recent python version).
- Fix some issues with virtualenv support:
  - detection of system installs of Python didn't work properly when using a virtualenv. Because of this py2app did not create a "semi-standalone" bundle when using a virtualenv created with /usr/bin/python.
  - "semi-standalone" bundles created from a virtualenv included more files when they should (in particular bits of the stdlib)
- Issue #92: Add option '–force-system-tk' which ensures that the _tkinter extension (used by Tkinter) is linked against the Apple build of Tcl/Tk, even when it is linked to another framework in Python's std. library.

  This will cause a build error when tkinter is linked with a major version of Tcl/Tk that is not present in /System/Library/Frameworks.

- Issue #80: Add support for copying system plugins into the application bundle.

  Py2app now supports a new option *include_plugins*. The value of this is a list of paths to plugins that should be copied into the application bundle.

  Items in the list are either paths, or a tuple with the plugin type and the path:

```
include_plugins=[
  "MyPlugins/MyDocument.qlgenerator",
  ("SystemConfiguration", "MyPlugins/MyConfig.plugin"),
]
```

  Py2app currently knows about the following plugin suffixes: `.qlgenerator`, `.mdimporter`, `.xpc`, `.service`, `.prefPane`, `.iaplugin` and `.action`. These plugins can be added without specifying the plugin type.

- Issue #83: Setup.py now refuses to install when the current platform is not Mac OS X.

  This makes it clear that the package is only supported on OSX and avoids confusing errors later on.

- Issue #39: It is now possible to have subpackages on in the "packages" option of py2app.

- Issue #37: Add recipe for pyEnchant

  ..note:

```
The recipe only works for installations of pyEnchant
where pyEnchant is stored in the installation (such
as the binary eggs on PyPI), not for installations
that either use the "PYENCHANT_LIBRARY_PATH" environment
variable or MacPorts.
```

- Issue #90: Removed the 'email' recipe, but require a new enough version of modulegraph instead. Because of this py2app now requires modulegraph 0.11 or later.

1.13.31 py2app 0.7.4

- Issue #77: the stdout/stderr streams of application and plugin bundles did not end up in Console.app on OSX 10.8 (as they do on earlier releases of OSX). This is due to a change in OSX.

  With this version the application executable converts writes to the stdout and stderr streams to the ASL logging subsystem with the options needed to end up in the default view of Console.app.

  NOTE: The stdout and stderr streams of plugin bundles are not redirected, as it is rather bad form to change the global environment of the host application.

- The i386, x86_64 and intel stub binaries are now compiled with clang on OSX 10.8, instead of an older version of GCC. The other stub versions still are compiled on OSX 10.6.

- Issue #111: The site.py generated by py2app now contains a USER_SITE variable (with a default value of `None`) because some software tries to import the value.

- Py2app didn't preserve timestamps for files copied into application bundles, and this can cause a bytecompiled file to appear older than the corresponding source file (for packages copied in the bundle using the 'packages' option).

  Related to issue #101

- Py2app also didn't copy file permissions for files copied into application bundles, which isn't a problem in general but did cause binaries to lose there executable permissions (as noted on Stackoverflow)

- Issue #101: Set "PYTHONDONTWRITEBYTECODE" in the environment before calling Py_Initialize to ensure that the interpreter won't try to write bytecode files (which can cause problems when using sandboxed applications).
- Issue #105: py2app can now create app and plugin bundles when the main script has an encoding other than ASCII, in particular for Python 3.
- Issue #106: Ensure that the PIL recipe works on Python 3. PIL itself isn't ported yet, but Pillow does work with Python 3.
- "python setup.py install" now fails unless the machine is running Mac OS X.

  I've seen a number of reports of users that try to use py2app on Windows or Linux to build OSX applications. That doesn't work, py2app now fails during installation do make this clear.
- Disabled the 'email' recipe for python 3.x as it isn't needed there.
- Issue #91: Added a recipe for *lxml <http://lxml.de/>*, needed because lxml performs a number of imports from an extension and those cannot be detected automaticly by modulegraph.
- Issue #94: The site-packages zipfile in the application bundle now contains zipfile entries for directories as well. This is needed to work around a bug in the zipimporter for Python 3.3: it won't consider 'pkg/foo.py' to be in namespace package 'pkg' unless there is a zipfile entry for the 'pkg' folder (or there is a 'pkg/__init__.py' entry).
- Issue #97: Fixes a problem with the pyside and sip recipes when the 'qt_plugins' option is used for 'image_plugins'.
- Issue #96: py2app should work with python 2.6 again (previous releases didn't work due to using the sysconfig module introduced in python 2.7)
- Issue #99: appstore requires a number of symlinks in embedded frameworks.

  (Version 0.7 already added a link Python.frameworks/Versions/Current, this versions also adds Python.framework/Python and Python.framework/Resources with the value required by the appstore upload tool).
- Py2app copied stdlib packages into the app bundle for semi-standalone builds when they are mentioned in the '–packages' option (either explicitly or by a recipe). This was unintentional, semi-standlone builds should rely on the external Python framework for the stdlib.

  Note: Because of this bug parts of the stdlib of `/usr/bin/python` could be copied into app bundles created with py2app.

1.13.32 py2app 0.7.3

py2app 0.7.3 is a bugfix release

- Issue #82: Remove debug print statement from py2app.util.LOADER that caused problems with Python 3.
- Issue #81: Py2app now fails with an error when trying to build a bundle for a unix-style shared library build of Python (`--enable-shared`) unless you are using a recent enough patchlevel of python (2.7.4, 3.2.3, 3.3.1, 3.4.0, all of them are not released yet).

  The build failure was added to avoid a very confusing error when trying to start the generated application due to a bug in the way python reads the environment (for shared library builds on Mac OS X).
- Py2app will also give an error message when the python binary does not have a shared library (or framework) at all.

- Issue #87: Ignore '.git' and '.hg' directories while copying package data ('.svn' and 'CVS' were already ignored).
- Issue #65: the fix in 0.7 to avoid copying a symlinked library twice caused problems for some users because only one of the file names ended up in the application bundle. This release ensures that both names exist (one as a symbolic name to the other).
- Issue #88: Ensure that the fix for #65 won't try to create a symlink that points to itself. This could for example occur with homebrew, where the exposed lib directory contains symlinks to a cellar, while tye install_name does mention the "public" lib directory:

```
$ ls -l /opt/homebrew/lib
...
libglib-2.0.0.dylib -> ../Cellar/glib/2.32.4/lib/libglib-2.0.0.dylib
...

$ otool -vL /opt/homebrew/lib/libglib-2.0.0.dylib
/opt/homebrew/lib/libglib-2.0.0.dylib:
   /opt/homebrew/lib/libglib-2.0.0.dylib (compatibility version 3201.0.0, current␣
↪version 3201.4.0)
   time stamp 1 Thu Jan  1 01:00:01 1970
...
```

1.13.33 py2app 0.7.2

py2app 0.7.2 is a bugfix release

- Issue #75: Don't remove `--dist-dir`, but only remove the old version of the objects we're trying to build (if that exists).

  This once again makes it possible to have a number of setup.py files that build plugins into the same target folder (such as the plugins folder of an application)
- Issue #78: Packages added using the `--packages` option didn't end up on `sys.path` for semi-standalone applications.

  Reported by Steve Strassmann
- Issue #76: Semi-standalone packages using extensions modules coudn't use extensions unless they also used the `--site-packages` option (and the extensions are in the site-packages directory).

  Fixes some problems with PyQt and wxWidgets when using the system installation of Python.

  Patch by Dan Horner.
- It is currently not possible to use a subpackage ("foo.bar") in the list of packages for the "packages" option. Py2app now explicitly checks for this and prints an error message instead of building an application that doesn't work.

  Issue: #39

1.13.34 py2app 0.7.1

py2app 0.7.1 is a bugfix release

- Always include 'pkg_resources', this is needed to correctly work with setuptools namespace packages, the __init__.py files of those contain `__import__('pkg_resources')` and that call isn't recognized as an import by the bytecode scanner.

- Issue #67: py2applet didn't work with python 3 due to the use of 'raw_input'

  Reported by Andrew Barnert.

- Issue #68: the "extra-scripts" feature introduced in 0.7 couldn't copy scripts that aren't in the same directory as "setup.py".

  Reported by Andrew Barnert.

- For semi-standalone applications the "lib-dynload" directory inside the application was not on "sys.path", which resulted in launch failures when using an extension that is not in the stdlib.

- Issue #70: application fails to launch when script uses Windows line endings

  Reported by Luc Jean.

1.13.35 py2app 0.7

py2app 0.7 is a bugfix release

- Issue #65: generated bundle would crash when two libraries linked to the same library using different names (one refering to the real name, the other to a symlink).

  An example if this is an application using wxWidgets when wxWidgets is installed using homebrew.

  Reported by "Bouke".

- Issue #13: It is now possible to add helper scripts to a bundle, for example for creating a GUI that starts a helper script in the background.

  This can be done by using the option "–extra-scripts", the value of which is a list of script files (".py" or ".pyw" files).

- Smarter matplotlib recipe, it is now possible to specify which backends should be included. Issue #44, reported by Adam Kovics.

  The argument to `--matplotlib-backends` (or 'matplotlib_backends' in setup.py) is a list of plugins to include. Use '-' to not include backends other than those found by the import statement analysis, and '*' to include all backends (without necessarily including all of matplotlib)

  As an example, use `--matplotlib-backends=wxagg` to include just the wxagg backend.

  Default is to include the entire matplotlib package.

- The packages included by a py2app recipe weren't processed by modulegraph and hence their dependencies were not always included.

- Fix virtualenv support: alias builds in a virtual environment failed to work.

  (There are still issues with semi-standalone and alias plugin bundles in a virtualenv environment).

- issue #18: improved PyQt and PySide support.

  Py2app now has a new option named "–qt-plugins" (or "qt_plugins" in setup.py), this option specify a list of plugins that should be included in the application bundle. The items of the list can have a number of forms:

  - "plugintype/libplugin.dylib"

    Specify one particular plugin

  - "plugintype/*foo*"

    Specify one or more plugins using a glob pattern

  - "plugintype"

    Include all plugins of a type, equivalent to "plugintype/*".

  The plugins are copied into "Resources/qt_plugins" and py2app adds a "qt.conf" file that points to that location for plugins.

- issue #49: package data that is a zipfile is now correctly copied into the bundle instead of extracting the archive.
- issue #59: compile site.py to ensure that the generated bundle doesn't change on first run.

  This is nice to have in general, and essential when using code signing because the signature will break when a new file is added after signing.

  Reported by Michael McCracken.

- issue #60: recipe for "email" package was not loaded

  Reported by Chris Beaumont

- issue #46: py2app no longer warns about the Qt license. We don't warn about other possibly GPL licensed software either and py2app is not a license-enforcement tool.

  Reported by briank_in_la.

- Generated bundles always started with python optimization active (that is, as if running as 'python -O').
- Fix issue #53: py2app would crash if a data file happened to be a zipfile.
- py2app copies data files in the directory for a package into the application bundle. It also did this for directories that represent subpackages, which made it impossible to exclude subpackages.
- added recipe for wxPython because some subpackages of wxPython use `__path__` trickery that confuses modulegraph.
- recipes can now return a list of additional entries for the 'includes' list.
- rewritten the recipe for matplotlib. The recipe no longer includes the entire package, but just the "mpl-data" directory.

  WARNING: This recipe has had limited testing.

- fix mixed indentation (tabs and spaces) in argv_emulation.py, which caused installation failures on python 3.x (issue #40)
- Issue #43: py2app now creates a symlink named "Current" in the 'Versions' directory of the embedded Python framework to comply with a requirement for the Mac App-store.
- on some OSX releases the application receives both the "open application" and "open documents" Apple Events during startup, which broke an assumption in argv_emulation.py.
- py2app is more strict w.r.t. explictly closing files, this avoids ResourceWarnings for unclosed files.
- fix test issue with semi-standalone builds on Python 3.2
- added recipe for pyzmq
- Don't use the version information from Python.framework's Info.plist, but use `sys.version_info`. This fixes a build problem with EPD.
- Ignore some more files when copying package data:
  - VIM swap files (`.foo.py.swp`)
  - Backup files for a number of tools: `foo.orig` and `foo~`

1.13.36 py2app 0.6.4

py2app 0.6.4 is a bugfix and minor feature release

- Issue #28: the argv emulator crashes in 64-bit mode on OSX 10.5

  Fixing this issue required yet another rewrite of the argv_emulator code.

- Added option '–arch=VALUE' which can be used to select the set of architectures for the main executable. This defaults to the set of architectures supported by the python interpreter and can be used to drop support for some architectures (for example when you're using a python binary that supports both 32-bit and 64-bit code and use a GUI library that does not yet work in 64-bit mode).

  Valid values for the argument are archectures used in the list below and the following groups of architectures:

  - fat: i386, ppc
  - fat3: i386, x86_64, ppc
  - univeral: i386, x86_64, ppc, ppc64
  - intel: i386, x86_64

- Issue #32: fix crash when application uses PySide

  This is partially fixed in macholib (release 1.4.3)

- The '-O' flag of py2app now defaults to the python optimization level when using python 2.6 or later.

- Issue #31: honor optimze flag at runtime.

  Until now an application bundle created by py2app would also run without the "-O" flag, even when the user specified it should. This is now fixed.

- Issue #33: py2app's application bundle launcher now clears the environment variable `PYOBJC_BUNDLE_ADDRESS`, avoids a crash when using PyObjC in an application launched from a py2app based plugin bundle.

- py2app's bundle launcher set the environment variable `PYOBJC_BUNDLE_ADDRESS`, this variable is now deprecated. Use `PYOBJC_BUNDLE_ADDRESS<PID>` instead (replace `<PID>` by the process ID of the current process).

- When using the system python we now explicitly add Apple's additional packages (like PyObjC and Twisted) to `sys.path`.

  This fixes and issue reported by Sean Robinson: py2app used to create a non-working bundle when you used these packages because the packages didn't get included (as intented), but were not available on `sys.path` either.

- Fixed the recipe for sip, which in turn ensures that PyQt4 applications work.

  As before the SIP recipe is rather crude, it will include *all* SIP-based packages into your application bundle when it detects a module that uses SIP.

- The 'Resources' folder is no longer on the python search path, it contains the scripts while Python modules and packages are located in the site-packages directory. This change is related to issue #30.

- The folder 'Resources/Python/site-packages' is no longer on the python search path. This folder is not used by py2app itself, but might by used by custom build scripts that wrap around py2app.

- Issue #30: py2app bundles failed to launch properly when the scriptfile has the same name as a python package used by the application.

- Issue #15: py2app now has an option to emulate the shell environment you get by opening a window in the Terminal.

Usage: `python setup.py py2app --emulate-shell-environment`

This option is experimental, it is far from certain that the implementation works on all systems.

- Issue #16: `--argv-emulation` now works with Python 3.x and in 64-bit executables.
- Issue #17: py2applet script defaults 'argv_emulation' to False when your using a 64-bit build of python, because that option is not supported on such builds.
- py2app now clears the temporary directory in 'build' and the output directory in 'dist' before doing anything. This avoids unwanted interactions between results from a previous builds and the current build.
- Issue #22: py2app will give an error when the specified version is invalid, instead of causing a crash in the generated executable.
- Issue #23: py2app failed to work when an .egg directory was implictly added to `sys.path` by setuptools and the "-O" option was used (for example `python setup.py py2app -O2`)
- Issue #26: py2app copied the wrong executable into the application bundle when using virtualenv with a framework build of Python.

1.13.37 py2app 0.6.3

py2app 0.6.3 is a bugfix release

- py2app failed to compile .xib files (as reported on the pythonmac-sig mail-ing list).

1.13.38 py2app 0.6.2

py2app 0.6.2 is a bugfix release

- py2app failed to copy the iconfile into application bundle (reported by Russel Owen)
- py2app failed to copy resources and data files as well (the `resource` key in the py2ap options dictionary and the `data_files` argument to the setup function).

  Issue #19, reported by bryon(at)spideroak.com.
- py2app failed to build application bundles when using virtualenv due to assumptions about the relation between `sys.prefix` and `sys.executable`.

  Report and fix by Erik van Zijst.
- Ensure that the 'examples' directory is included in the source archive

1.13.39 py2app 0.6.1

py2app 0.6.1 is a bugfix release

Bugfixes:

- py2app failed to build the bundle when python package contained a zipfile with data.

  This version solves most of that problem using a rough workaround (the issue is fixed when the filename ends with '.zip').
- The code that recreates the stub executables when they are older than the source code now uses `xcode-select` to find the root of SDKs.

  This makes it possible to recreate these executables on machines where both Xcode 3 and Xcode 4 are installed and Xcode 3 is the default Xcode.

- The stub executables were regenerated using Xcode 3

  As a word of warning: Xcode 4 cannot be used to rebuild the stub executables, in particular not those that have support for the PPC architecture.

- Don't rebuild the stub executables automaticly, that's unsafe with Xcode 4 and could trigger accidently when files are installed in a different order than expected.
- Small tweaks to the testsuite to ensure that they work on systems with both Xcode3 and Xcode4 (Xcode3 must be the selected version).
- Better cleanup in the testsuite when `setupClass` fails.

1.13.40 py2app 0.6

py2app 0.6 is a minor feature release

Features:

- it is now possible to specify which python distributions must be availble when building the bundle by using the "install_requires" argument of the `setup()` function:

```
setup(

    ...
    install_requires = [
            "pyobjc == 2.2"
    ],
)
```

- py2app can now package namespace packages that were installed using *pip <http://pypi.python.org/pypi/pip>* or the setuptools install option `--single-version-externally-managed`.
- the bundle template now supports python3, based on a patch by Virgil Dupras.
- alias builds no longer use Carbon Aliases and therefore are supported with python3 as well (patch by Virgil Dupras)
- argv emulation doesn't work in python 3, this release will tell you abou this instead of silently failing to build a working bundle.
- add support for custom URLs to the argv emulation code (patch by Brendan Simon).

  You will have to add a "CFBundleURLTypes" key to your Info.plist to use this, the argv emulation code will ensure that the URL to open will end up in `sys.argv`.

- `py2app.util` contains a number of functions that are now deprecated an will be removed in a future version, specifically: `os_path_islink`, `os_path_isdir`, `path_to_zip`, `get_zip_data`, `get_mtime`, and `os_readlink`.
- The module `py2app.simpleio` no longer exists, and should never have been in the repository (it was part of a failed rewrite of the I/O layer).

Bug fixes:

- fix problem with symlinks in copied framework, as reported by Dan Ross.
- py2applet didn't work in python 3.x.
- The `--alias` option didn't work when building a plugin bundle (issue #10, fix by Virgil Dupras)
- Avoid copying the __pycache__ directory in python versions that implement PEP 3147 (Python 3.2 and later)

- App bundles with Python 3 now work when the application is stored in a directory with non-ASCII characters in the full name.
- Do not compile `.nib` files, it is not strictly needed and breaks PyObjC projects that still use the NibClassBuilder code.
- Better error messsages when trying to include a non-existing file as a resource.
- Don't drop into PDB when an exception occurs.
- Issue #5: Avoid a possible stack overflow in the bundle executable
- Issue #9: Work with python 3.2
- Fix build issues with python 2.5 (due to usage of too modern distutils command subclasses)
- The source distribution didn't include all files that needed to be it ever since switching to mercurial, I've added a MANIFEST.in file rather than relying on setuptool's autoguessing of files to include.
- Bundle template works again with semi-standalone builds (such as when using a system python), this rewrites the fix for issue #10 mentioned earlier.
- Ensure py2app works correctly when the sources are located in a directory with non-ascii characters in its name.

1.13.41 py2app 0.5.2

py2app 0.5.2 is a bugfix release

Bug fixes:

- Ensure that the right stub executable gets found when using the system python 2.5

1.13.42 py2app 0.5.1

py2app 0.5.1 is a bugfix release

Bug fixes:

- Ensure stub executables get included in the egg files
- Fix name of the bundletemplate stub executable for 32-bit builds

1.13.43 py2app 0.5

py2app 0.5 is a minor feature release.

Features:

- Add support for the `--with-framework-name` option of Python's configure script, that is: py2app now also works when the Python framework is not named 'Python.framework'.
- Add support for various build flavours of Python (32bit, 3-way, ...)
- py2app now actually works for me (ronaldoussoren@mac.com) with a python interpreter in a virtualenv environment.
- Experimental support for python 3

Bug fixes:

- Fix recipe for matplotlib: that recipe caused an exception with current versions of matplotlib and pytz.

- Use modern API's in the alias-build bootstrap code, without this 'py2app -A' will result in broken bundles on a 64-bit build of Python. (Patch contributed by James R Eagan)
- Try both 'import Image' and 'from PIL import Image' in the PIL recipe. (Patch contributed by Christopher Barker)
- The stub executable now works for 64-bit application bundles
- (Lowlevel) The application stub was rewritten to use `dlopen` instead of `dyld` APIs. This removes deprecation warnings during compilation.

1.13.44 py2app 0.4.3

py2app 0.4.3 is a bugfix release

Bug fixes:

- A bad format string in build_app.py made it impossible to copy the Python framework into an app bundle.

1.13.45 py2app 0.4.2

py2app 0.4.2 is a minor feature release

Features:

- When the '–strip' option is specified we now also remove '.dSYM' directories from the bundle.
- Remove dependency on a 'version.plist' file in the python framework
- A new recipe for PyQt 4.x. This recipe was donated by Kevin Walzer.
- A new recipe for virtualenv, this allows you to use py2app from a virtual environment.
- Adds support for converting `.xib` files (NIB files for Interface Builder 3)
- Introduces an experimental plugin API for data converters.

  A conversion plugin should be defined as an entry-point in the `py2app.converter` group:

```
 setup(
    ...
    entry_points = {
            'py2app.converter': [
                "label          = some_module:converter_function",
             ]
     },
     ...
)
```

  The conversion function should be defined like this:

```
from py2app.decorators import converts

@converts('.png')
def optimze_png(source, proposed_destionation, dryrun=0):
    # Copy 'source' to 'proposed_destination'
    # The conversion is allowed to change the proposed
    # destination to another name in the same directory.
    pass
```

Buf fixes:

- This fixes an issue with copying a different version of Python over to an app/plugin bundle than the one used to run py2app with.

1.13.46 py2app 0.4.0

py2app 0.4.0 is a minor feature release (and was never formally released).

Features:

- Support for CoreData mapping models (introduced in Mac OS X 10.5)
- Support for python packages that are stored in zipfiles (such as `zip_safe` python eggs).

Bug fixes:

- Fix incorrect symlink target creation with an alias bundle that has included frameworks.
- Stuffit tends to extract archives recursively, which results in unzipped code archives inside py2app-created bundles. This version has a workaround for this "feature" for Stuffit.
- Be more carefull about passing non-constant strings as the template argumenti of string formatting functions (in the app and bundle templates), to avoid crashes under some conditions.

1.13.47 py2app 0.3.6

py2app 0.3.6 is a minor bugfix release.

Bug fixes:

- Ensure that custom icons are copied into the output bundle
- Solve compatibility problem with some haxies and inputmanager plugins

1.13.48 py2app 0.3.5

py2app 0.3.5 is a minor bugfix release.

Bug fixes:

- Resolve disable_linecache issue
- Fix Info.plist and Python path for plugins

1.13.49 py2app 0.3.4

py2app 0.3.4 is a minor bugfix release.

Bug fixes:

- Fixed a typo in the py2applet script
- Removed some, but not all, compiler warnings from the bundle template (which is still probably broken anyway)

1.13.50 py2app 0.3.3

py2app 0.3.3 is a minor bugfix release.

Bug Fixes:

- Fixed a typo in the argv emulation code
- Removed the unnecessary py2app.install hack (setuptools does that already)

1.13.51 py2app 0.3.2

py2app 0.3.2 is a major bugfix release.

Functional changes:

- Massively updated documentation
- New prefer-ppc option
- New recipes: numpy, scipy, matplotlib
- Updated py2applet script to take options, provide –make-setup

Bug Fixes:

- No longer defaults to LSPrefersPPC
- Replaced stdlib usage of argvemulator to inline version for i386 compatibility

1.13.52 py2app 0.3.1

py2app 0.3.1 is a minor bugfix release.

Functional changes:

- New EggInstaller example

Bug Fixes:

- Now ensures that the executable is +x (when installed from egg this may not be the case)

1.13.53 py2app 0.3.0

py2app 0.3.0 is a major feature enhancements release.

Functional changes:

- New –xref (-x) option similar to py2exe's that produces a list of modules and their interdependencies as a HTML file
- sys.executable now points to a regular Python interpreter alongside the regular executable, so spawning sub-interpreters should work much more reliably
- Application bootstrap now detects paths containing ":" and will provide a "friendly" error message instead of just crashing <http://python.org/sf/1507224>.
- Application bootstrap now sets PYTHONHOME instead of a large PYTHONPATH
- Application bootstrap rewritten in C that links to CoreFoundation and Cocoa dynamically as needed, so it doesn't imply any particular version of the runtime.

- Documentation and examples changed to use setuptools instead of distutils.core, which removes the need for the py2app import
- Refactored to use setuptools, distributed as an egg.
- macholib, bdist_mpkg, modulegraph, and altgraph are now separately maintained packages available on PyPI as eggs
- macholib now supports little endian architectures, 64-bit Mach-O headers, and reading/writing of multiple headers per file (fat / universal binaries)

1.13.54 py2app 0.2.1

py2app 0.2.1 is a minor bug fix release.

Bug Fixes:

- macholib.util.in_system_path understands SDKs now
- DYLD_LIBRARY_PATH searching is fixed
- Frameworks and excludes options should work again.

1.13.55 py2app 0.2.0

py2app 0.2.0 is a minor bug fix release.

Functional changes:

- New datamodels option to support CoreData. Compiles .xcdatamodel files and places them in the Resources dir (as .mom).
- New use-pythonpath option. The py2app application bootstrap will no longer use entries from PYTHONPATH unless this option is used.
- py2app now persists information about the build environment (python version, executable, build style, etc.) in the Info.plist and will clean the executable before rebuilding if anything at all has changed.
- bdist_mpkg now builds packages with the full platform info, so that installing a package for one platform combination will not look like an upgrade to another platform combination.

Bug Fixes:

- Fixed a bug in standalone building, where a rebuild could cause an unlaunchable executable.
- Plugin bootstrap should compile/link correctly with gcc 4.
- Plugin bootstrap no longer sets PYTHONHOME and will restore PYTHONPATH after initialization.
- Plugin bootstrap swaps out thread state upon plug-in load if it is the first to initialize Python. This fixes threading issues.

1.13.56 py2app 0.1.9

py2app 0.1.9 is a minor bug fix release.

Bugs fixed:

- bdist_mpkg now builds zip files that are correctly unzipped by all known tools.

- The behavior of the bootstrap has changed slightly such that `__file__` should now point to your main script, rather than the bootstrap. The main script has also moved to `Resources`, from `Resources/Python`, so that `__file__` relative resource paths should still work.

1.13.57 py2app 0.1.8

py2app 0.1.8 is a major enhancements release:

Bugs fixed:

- Symlinks in included frameworks should be preserved correctly (fixes Tcl/Tk)
- Fixes some minor issues with alias bundles
- Removed implicit SpiderImagePlugin -> ImageTk reference in PIL recipe
- The `--optimize` option should work now
- `weakref` is now included by default
- `anydbm`'s dynamic dependencies are now in the standard implies list
- Errors on app launch are brought to the front so the user does not miss them
- bdist_mpkg now compatible with pychecker (data_files had issues)

Options changed:

- deprecated `--strip`, it is now on by default
- new `--no-strip` option to turn off stripping of executables

New features:

- Looks for a hacked version of the PyOpenGL __init__.py so that it doesn't have to include the whole package in order to get at the stupid version file.
- New `loader_files` key that a recipe can return in order to ensure that non-code ends up in the .zip (the pygame recipe uses this)
- Now scans all files in the bundle and normalizes Mach-O load commands, not just extensions. This helps out when using the `--package` option, when including frameworks that have plugins, etc.
- An embedded Python interpreter is now included in the executable bundle (`sys.executable` points to it), this currently only works for framework builds of Python
- New `macho_standalone` tool
- New `macho_find` tool
- Major enhancements to the way plugins are built
- bdist_mpkg now has a `--zipdist` option to build zip files from the built package
- The bdist_mpkg "Installed to:" description is now based on the package install root, rather than the build root

1.13.58 py2app 0.1.7

py2app 0.1.7 is a bug fix release:

- The `bdist_mpkg` script will now set up sys.path properly, for setup scripts that require local imports.
- `bdist_mpkg` will now correctly accept `ReadMe`, `License`, `Welcome`, and `background` files by parameter.

- bdist_mpkg can now display a custom background again (0.1.6 broke this).
- bdist_mpkg now accepts a build-base= argument, to put build files in an alternate location.
- py2app will now accept main scripts with a .pyw extension.
- py2app's not_stdlib_filter will now ignore a site-python directory as well as site-packages.
- py2app's plugin bundle template no longer displays GUI dialogs by default, but still links to AppKit.
- py2app now ensures that the directory of the main script is now added to sys.path when scanning modules.
- The py2app build command has been refactored such that it would be easier to change its behavior by subclassing.
- py2app alias bundles can now cope with editors that do atomic saves (write new file, swap names with existing file).
- macholib now has minimal support for fat binaries. It still assumes big endian and will not make any changes to a little endian header.
- Add a warning message when using the install command rather than installing from a package.
- New simple/structured example that shows how you could package an application that is organized into several folders.
- New PyObjC/pbplugin Xcode Plug-In example.

1.13.59 py2app 0.1.6

Since I have been slacking and the last announcement was for 0.1.4, here are the changes for the soft-launched releases 0.1.5 and 0.1.6:

py2app 0.1.6 was a major feature enhancements release:

- py2applet and bdist_mpkg scripts have been moved to Python modules so that the functionality can be shared with the tools.
- Generic graph-related functionality from py2app was moved to altgraph.ObjectGraph and altgraph.GraphUtil.
- bdist_mpkg now outputs more specific plist requirements (for future compatibility).
- py2app can now create plugin bundles (MH_BUNDLE) as well as executables. New recipe for supporting extensions built with sip, such as PyQt. Note that due to the way that sip works, when one sip-based extension is used, *all* sip-based extensions are included in your application. In practice, this means anything provided by Riverbank, I don't think anyone else uses sip (publicly).
- New recipe for PyOpenGL. This is very naive and simply includes the whole thing, rather than trying to monkeypatch their brain-dead version acquisition routine in __init__.
- Bootstrap now sets ARGVZERO and EXECUTABLEPATH environment variables, corresponding to the argv[0] and the _NSGetExecutablePath(...) that the bundle saw. This is only really useful if you need to relaunch your own application.
- More correct dyld search behavior.
- Refactored macholib to use altgraph, can now generate GraphViz graphs and more complex analysis of dependencies can be done.
- macholib was refactored to be easier to maintain, and the structure handling has been optimized a bit.
- The few tests that there are were refactored in py.test style.
- New PyQt example.

- New PyOpenGL example.

See also:

- http://mail.python.org/pipermail/pythonmac-sig/2004-December/012272.html

1.13.60 py2app 0.1.5

py2app 0.1.5 is a major feature enhancements release:

- Added a `bdist_mpkg` distutils extension, for creating Installer an metapackage from any distutils script.
  - Includes PackageInstaller tool
  - bdist_mpkg script
  - setup.py enhancements to support bdist_mpkg functionality
- **Added a `PackageInstaller` tool, a droplet that performs the same function** as the `bdist_mpkg` script.
- Create a custom `bdist_mpkg` subclass for py2app's setup script.
- Source package now includes PJE's setuptools extension to distutils.
- Added lots of metadata to the setup script.
- `py2app.modulegraph` is now a top-level package, `modulegraph`.
- `py2app.find_modules` is now `modulegraph.find_modules`.
- Should now correctly handle paths (and application names) with unicode characters in them.
- New `--strip` option for `py2app` build command, strips all Mach-O files in output application bundle.
- New `--bdist-base=` option for `py2app` build command, allows an alternate build directory to be specified.
- New docutils recipe. Support for non-framework Python, such as the one provided by DarwinPorts.

See also:

- http://mail.python.org/pipermail/pythonmac-sig/2004-October/011933.html

1.13.61 py2app 0.1.4

py2app 0.1.4 is a minor bugfix release:

- The `altgraph` from 0.1.3 had a pretty nasty bug in it that prevented filtering from working properly, so I fixed it and bumped to 0.1.4.

1.13.62 py2app 0.1.3

py2app 0.1.3 is a refactoring and new features release:

- `altgraph`, my fork of Istvan Albert's graphlib, is now part of the distribution
- `py2app.modulegraph` has been refactored to use `altgraph`
- py2app can now create GraphViz DOT graphs with the `-g` option (TinyTinyEdit example)
- Moved the filter stack into `py2app.modulegraph`
- Fixed a bug that may have been in 0.1.2 where explicitly included packages would not be scanned by `macholib`

- `py2app.apptemplate` now contains a stripped down `site` module as opposed to a `sitecustomize`
- Alias builds are now the only ones that contain the system and user `site-packages` directory in `sys.path`
- The `pydoc` recipe has been beefed up to also exclude `BaseHTTPServer`, etc.

Known issues:

- Commands marked with XXX in the help are not implemented
- Includes *all* files from packages, it should be smart enough to strip unused .py/.pyc/.pyo files (to save space, depending on which optimization flag is used)
- `macholib` should be refactored to use `altgraph`
- `py2app.build_app` and `py2app.modulegraph` should be refactored to search for dependencies on a per-application basis

1.13.63 py2app 0.1.2

py2app 0.2 is primarily a bugfix release:

- The encodings package now gets included in the zip file (saves space)
- A copy of the Python interpreter is not included anymore in standalone builds (saves space)
- The executable bootstrap is now stripped by default (saves a little space)
- `sys.argv` is set correctly now, it used to point to the executable, now it points to the boot script. This should enhance compatibility with some applications.
- Adds an "Alias" feature to modulegraph, so that `sys.modules` craziness such as `wxPython.wx -> wx` can be accomodated (this particular craziness is also now handled by default)
- A `sys.path` alternative may be passed to `find_modules` now, though this is not used yet
- The `Command` instance is now passed to recipes instead of the `Distribution` instance (though no recipes currently use either)
- The post-filtering of modules and extensions is now generalized into a stack and can be modified by recipes
- A wxPython example demonstrating how to package wxGlade has been added (this is a good example of how to write your own recipe, and how to deal with complex applications that mix code and data files)
- `PyRuntimeLocations` is now set to (only) the location of the current interpreter's `Python.framework` for alias and semi-standalone build modes (enhances compatibility with extensions built with an unpatched Makefile with Mac OS X 10.3's Python 2.3.0)

Known issues:

- Includes *all* files from packages, it should be smart enough to strip unused .py/.pyc/.pyo files (to save space, depending on which optimization flag is used).

1.13.64 py2app 0.1.1

py2app 0.1.1 is primarily a bugfix release:

- **Several problems related to Mac OS X 10.2 compatibility and standalone** building have been resolved
- Scripts that are not in the same directory as setup.py now work
- A new recipe has been added that removes the pydoc -> Tkinter dependency
- A recipe has been added for py2app itself

- a wxPython example (superdoodle) has been added. Demonstrates not only how easy it is (finally!) to bundle wxPython applications, but also how one setup.py can deal with both py2exe and py2app.
- A new experimental tool, py2applet, has been added. Once you've built it (`python setup.py py2app`, of course), you should be able to build simple applications simply by dragging your main script and optionally any packages, data files, Info.plist and icon it needs.

Known issues:

- Includes *all* files from packages, it should be smart enough to strip unused .py/.pyc/.pyo files (to save space, depending on which optimization flag is used).
- The default `PyRuntimeLocations` can cause problems on machines that have a /Library/Frameworks/Python.framework installed. Workaround is to set a plist that has the following key: `PyRuntimeLocations=['/System/Library/Frameworks/Python.framework/Versions/2.3/Python']` (this will be resolved soon)

1.13.65 py2app 0.1

(first public release) py2app is the bundlebuilder replacement we've all been waiting for. It is implemented as a distutils command, similar to py2exe.

CHAPTER 2

Online Resources

There are several online resources to help you get along with py2app.

Mailing list: http://www.python.org/community/sigs/current/pythonmac-sig/

Issue tracker: https://github.com/ronaldoussoren/py2app/issues

Source code repository: https://github.com/ronaldoussoren/py2app

PyPI Entry: https://pypi.org/project/py2app/

If you're looking for help, pay special attention to the `examples` folder in the source, which demonstrates many common use cases.

CHAPTER 3

License

py2app may be distributed under the MIT or PSF open source licenses.