Complete Guide: Converting Python Scripts to macOS Apps with py2app

This guide will help you convert any Python script (especially tkinter GUI applications) into a clickable macOS application with a dock icon.

Prerequisites

1. **Python 3 installed** on your Mac
2. **Your Python script** (e.g., `MyApp.py`)
3. **Terminal access**

Step 1: Install py2app

Open Terminal and run:

```bash
pip3 install py2app
```

Or if you use a virtual environment:

```bash
python3 -m pip install py2app
```

Step 2: Create setup.py

In the **same directory** as your Python script, create a file called `setup.py`:

```python
"""
Setup script for creating a macOS application bundle using py2app.

Usage:
    python3 setup.py py2app -A       # Alias mode (fast, for testing)
    python3 setup.py py2app          # Standalone mode (distributable)
"""

from setuptools import setup

# CHANGE THIS: Put your main script filename here
APP = ['MyApp.py']

DATA_FILES = []

OPTIONS = {
    'argv_emulation': False,
    'packages': ['tkinter'],  # Add any packages your app uses
    # 'iconfile': 'app_icon.icns',  # Uncomment if you have a custom icon
    'plist': {
        'CFBundleName': 'MyApp',  # CHANGE THIS: Your app's name
        'CFBundleDisplayName': 'MyApp',  # CHANGE THIS: Display name
        'CFBundleGetInfoString': 'My Python Application',  # CHANGE THIS
        'CFBundleIdentifier': 'com.mycompany.myapp',  # CHANGE THIS
        'CFBundleVersion': '1.0.0',
        'CFBundleShortVersionString': '1.0.0',
        'NSHumanReadableCopyright': 'Copyright © 2025',  # CHANGE THIS
    }
}

setup(
    app=APP,
    data_files=DATA_FILES,
    options={'py2app': OPTIONS},
    setup_requires=['py2app'],
)
```

What to Customize:

1. **`APP = ['MyApp.py']`** - Change to your script's filename
2. **`'packages': ['tkinter']`** - Add any packages your script imports (e.g., `'numpy'`, `'requests'`, `'PIL'`)
3. **`CFBundleName`** - Your app's internal name (no spaces recommended)
4. **`CFBundleDisplayName`** - How the app appears in Finder and the Dock
5. **`CFBundleGetInfoString`** - Short description of your app
6. **`CFBundleIdentifier`** - Unique identifier in reverse domain format (e.g., `com.yourname.appname`)

Step 3: Build Your App

Option A: Alias Mode (Recommended for Testing)

This creates a lightweight app that's perfect for personal use:

```bash
python3 setup.py py2app -A
```

Advantages:

- Fast build time
- Easy to update (just edit your original Python script)
- No code signing issues
- Works great for personal use

Disadvantages:

- Not portable (won't work on other Macs)
- Requires Python to be installed

Option B: Standalone Mode (For Distribution)

This creates a fully self-contained app:

```bash
python3 setup.py py2app
```

If you get a code signing error:

```bash
# After the build fails, manually sign the app:
codesign --force --deep --sign - dist/MyApp.app
```

Advantages:

- Portable to other Macs
- Includes all dependencies
- Professional distribution

Disadvantages:

- Slower build time
- Larger file size
- May need code signing

Step 4: Test Your App

The built app will be in the `dist/` folder:

```bash
open dist/MyApp.app
```

If it works correctly, you'll see:

- Your app opens like any other Mac application
- It appears in the Dock
- You can double-click it from Finder

Step 5: Install to Applications (Optional)

Move your app to the Applications folder:

```bash
mv dist/MyApp.app /Applications/
```

Now you can:

- Launch it from Spotlight (Cmd + Space, type your app name)
- Pin it to your Dock
- Open it from Finder's Applications folder

Troubleshooting

Problem: "command 'py2app' has no such option"

Solution: Remove any unsupported options from `setup.py` (like `codesign_identity`)

Problem: "Cannot sign bundle"

Solution:

1. Use alias mode: `python3 setup.py py2app -A`
2. OR manually sign after building: `codesign --force --deep --sign - dist/MyApp.app`

Problem: App won't open / crashes immediately

Solution:

- Check for import errors in your script
- Make sure all required packages are listed in the `'packages'` list
- Test your script directly first: `python3 MyApp.py`

Problem: Deprecation warnings during build

Solution: These are harmless warnings. To suppress them:

```bash
python3 -W ignore setup.py py2app -A
```

Problem: Need to rebuild after changes

Solution:

```bash
rm -rf build dist  # Clean old builds
python3 setup.py py2app -A  # Rebuild
```

Adding a Custom Icon

1. **Create or find an icon file** in `.icns` format
   - You can convert PNG to ICNS using online tools
   - Or use this command:

```bash
sips -s format icns YourIcon.png --out app_icon.icns
```

2. **Save the `.icns` file** in the same directory as `setup.py`
3. **Uncomment the icon line** in `setup.py`:

```python
'iconfile': 'app_icon.icns',
```

4. **Rebuild:**

```bash
rm -rf build dist
python3 setup.py py2app -A
```

Common Python Packages and Dependencies

If your script uses these packages, add them to the `'packages'` list:

```python
OPTIONS = {
    'packages': [
        'tkinter',     # GUI (tkinter apps)
        'numpy',       # Numerical computing
        'pandas',      # Data analysis
        'requests',    # HTTP requests
        'PIL',         # Image processing
        'matplotlib',  # Plotting
        'sqlite3',     # Database
    ],
    # ... rest of options
}
```

Quick Reference Card

Task	Command
Install py2app	`pip3 install py2app`
Build (alias mode)	`python3 setup.py py2app -A`
Build (standalone)	`python3 setup.py py2app`
Clean builds	`rm -rf build dist`
Test app	`open dist/MyApp.app`
Install app	`mv dist/MyApp.app /Applications/`
Sign manually	`codesign --force --deep --sign - dist/MyApp.app`

Example: Complete Workflow

bash

```
# 1. Navigate to your script's directory
cd ~/Documents/MyPythonApp

# 2. Install py2app
pip3 install py2app

# 3. Create setup.py (use the template above)
nano setup.py

# 4. Build the app
python3 setup.py py2app -A

# 5. Test it
open dist/MyApp.app

# 6. If it works, install it
mv dist/MyApp.app /Applications/

# Done! Launch from Spotlight or Dock
```

Tips for Success

1. **Always test your Python script first** before building the app
2. **Use alias mode (`-A`)** for personal use - it's faster and easier
3. **Keep your setup.py** - you'll need it to rebuild after changes
4. **Start simple** - get a basic app working before adding custom icons
5. **Check your imports** - make sure all required packages are listed
6. **Clean between builds** - use `rm -rf build dist` if something goes wrong

---

For More Information

- py2app documentation: https://py2app.readthedocs.io/
- Python on macOS: https://docs.python.org/3/using/mac.html
- macOS .icns format: https://en.wikipedia.org/wiki/Apple_Icon_Image_format

---

Last updated: November 2025