Build a cross platform GUI app with BeeWare

Russell Keith-Magee

PyCon US 2025

Acknowledgement of Country

I am from **Whadjuk Noongar Boodja**.

We are on the lands of the **Shawnee** people.

Today's Agenda

- Introduction to BeeWare
- Build a GUI app
- Run the app on desktop
- Package the app for distribution
- Run the app on mobile
- Run the app as a web app
- Run the app as a console app

Today's Agenda

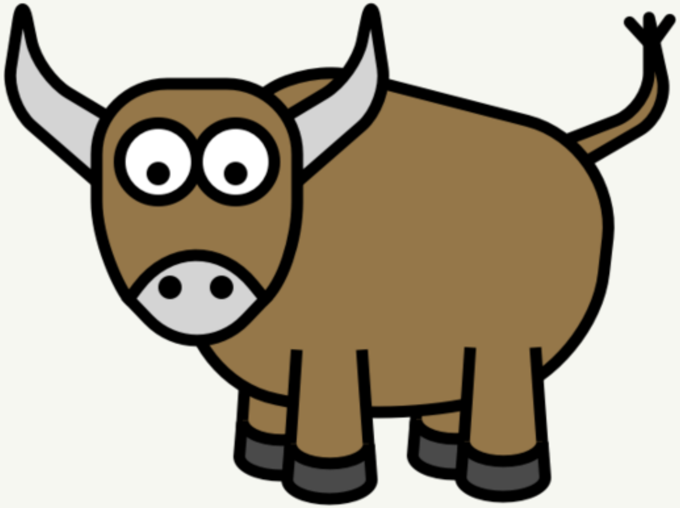
- Using third party packages
- Avoiding beachballs
- Customizing your app
- Permissions and device hardware
- Testing apps
- Slides at https://beeware.org/t/slides

ANACONDA®

BeeWare

Write Python. Deploy everywhere.

Briefcase



Toga

What is Briefcase?

- PEP518-compliant build tool
- Multiple platform support:
  - Windows (MSI and .zip)
  - macOS (DMG, PKG and .app.zip)
  - Linux (.deb, .rpm, .pkg; Flatpak and AppImage)
  - iOS & Android
  - Single Page Web App
- Extensible

What Briefcase isn't

- Dependent on the rest of the BeeWare suite
- A competitor for setuptools/Flit/Hatchling/…

How Briefcase works

- The dumbest approach that could possibly work
- Ships a full, independent Python interpreter*
- Wraps that interpreter in platform-appropriate bundle
- Install dependencies into bundle
- Install app into bundle

What makes Briefcase different

- It doesn't try to be clever
  - PyOxidizer, PyInstaller try to make Python something it isn't
- It's cross platform
  - py2app is macOS only; pynsist is Windows only; ...
  - It supports iOS and Android

What is Toga?

- Python native
- OS native
- Abstracts high level usage
- Not ideal for all use cases
    - … but that's OK.
    - … and you can reproduce Electron if you want

What about you?

Operating Systems?

Mobile platforms?

Python experience level?

Anyone using Conda?

Post-it flags

Starting a new project

Linux: https://beeware.org/t/linux-deps

```
$ mkdir beeware-tutorial
$ cd beeware-tutorial
$ python3 -m venv beeware
$ source beeware/bin/activate
(beeware) $
```

```
C:\...>mkdir beeware-tutorial
C:\...>cd beeware-tutorial
C:\...>py -m venv beeware
C:\...>beeware\Scripts\activate.bat
(beeware)C:\...>
```

Install Briefcase

```
(beeware) $ python -m pip install briefcase
...
Installing collected packages: ... briefcase
Successfully installed ... briefcase-0.3.23

(beeware) $ briefcase --version
0.3.23
```

briefcase new

```
(beeware) $ briefcase new

Let's build a new Briefcase app!

First, we need a formal name for your application. This is the
name that will be displayed to humans whenever the name of the
application is displayed. It can have spaces and punctuation
if you like, and any capitalization will be used as you type it.

Formal Name [Hello World]:
```

`briefcase new`

- Formal Name & App Name (`Hello World` and `helloworld`)
- Bundle (`org.beeware`)
- Project Name
- Description
- Author's Name and email
- URL
- License
- GUI framework

briefcase new

```
(beeware) $ briefcase new

Let's build a new Briefcase app!
...
Application 'Hello World' has been generated. To run your
application, type:

    cd helloworld
    briefcase dev

(beeware) $
```

What you get

```
helloworld/
    .gitignore
    CHANGELOG
    LICENSE
    README.rst
    pyproject.toml
    src/
        helloworld/
            ...
    tests
        ...
```

`pyproject.toml` - The project bits

```
[tool.briefcase]
project_name = "Hello World"
bundle = "com.example"
version = "0.0.1"
url = "https://example.com/hello-world"
license = "BSD license"
author = "Jane Developer"
author_email = "jane@example.com"
...
```

`pyproject.toml` - The app bits

```
[tool.briefcase.app.helloworld]
formal_name = "Hello World"
description = "My first application"
long_description = """More details about the app should go here.
"""
sources = ["src/helloworld"]
test_sources = ["tests"]
requires = []
test_requires = []
```

`pyproject.toml` - The really specific bits

```
[tool.briefcase.app.helloworld.macOS]
requires = [
  "toga-cocoa~=0.5.0",
  "std-nslog~=1.0.3
]
```

`pyproject.toml` - The really specific bits

```
[tool.briefcase.app.helloworld.linux]
requires = [
  "toga-gtk~=0.5.0",
  "pygobject < 3.52.1",
]

[tool.briefcase.app.helloworld.linux.system.debian]
...
[tool.briefcase.app.helloworld.linux.system.rhel]
...
[tool.briefcase.app.helloworld.linux.appimage]
...
[tool.briefcase.app.helloworld.linux.flatpak]
...
```

or

Roll your own pyproject.toml

or

Convert an existing project

PEP 621 metadata

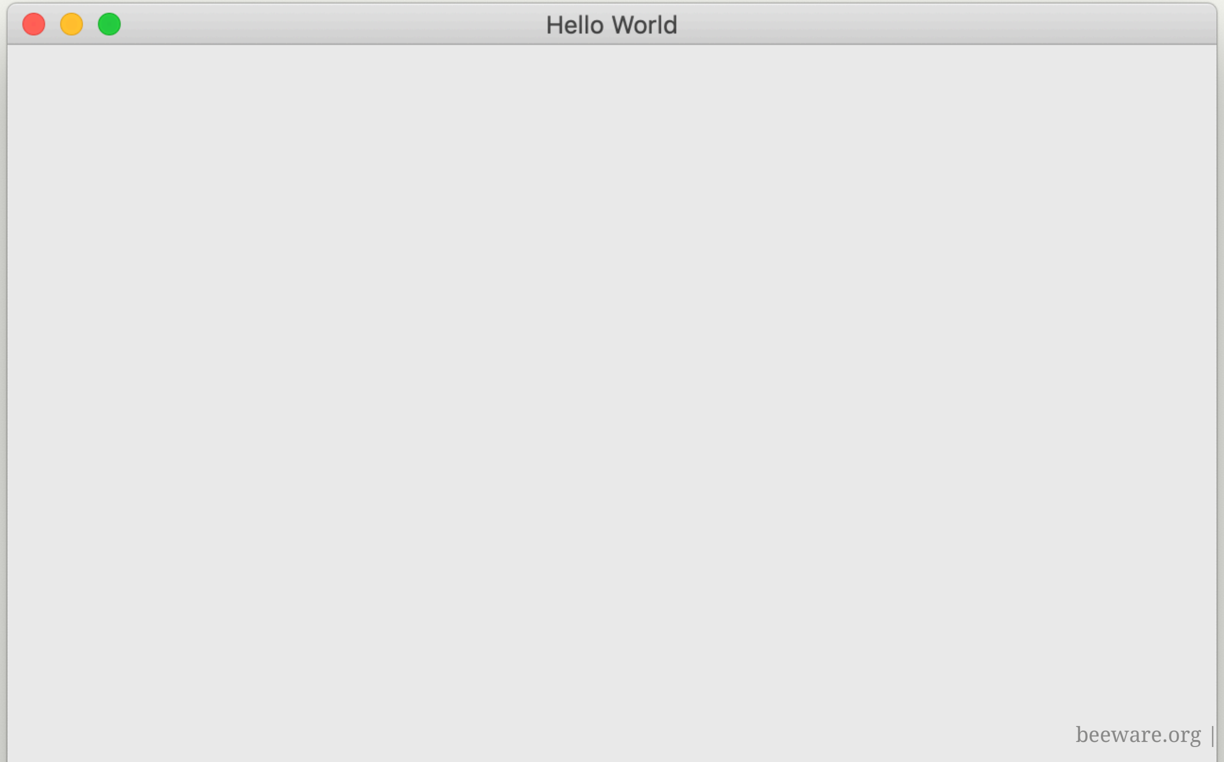
- Briefcase will use *some* PEP 621 definitions.
  - `[project] version` → `version`
  - `name`, `email` of first `[project] author` → `author`, `author_email`
  - `[project] dependencies` → `requires`
  - `[project] description` → `description`
  - `[project.license] text` → `license`
  - `[project.optional-dependencies] test` → `test_requires`
  - `[project.urls] homepage` → `url`

briefcase dev

```
(beeware) $ cd helloworld
(beeware) $ briefcase dev

[helloworld] Installing dependencies...
Collecting toga-cocoa~=0.5.0
...
Successfully installed ... std-nslog-1.0.3 toga-cocoa-0.5.1
toga-core-0.5.1 travertino-0.5.1
Installing dev requirements... done

[helloworld] Starting in dev mode...
===========================================================
```


Hello World

`briefcase dev`

```
(beeware) $ cd helloworld
(beeware) $ briefcase dev
```

To see verbose instructions:

```
(beeware) $ briefcase dev -vv
```

src/helloworld/app.py

```python
import toga
from toga.style import Pack
from toga.style.pack import COLUMN, ROW

class HelloWorld(toga.App):
  def startup(self):
    main_box = toga.Box()

    self.main_window = toga.MainWindow(title=self.formal_name)
    self.main_window.content = main_box
    self.main_window.show()

def main():
    return HelloWorld()
```

beeware.org/t/code

src/helloworld/app.py, v2

```
class HelloWorld(toga.App):
    def startup(self):
        main_box = toga.Box(style=Pack(direction=COLUMN))

        name_label = toga.Label(
            "Your name: ",
            style=Pack(margin=(0, 5))
        )
        self.name_input = toga.TextInput(style=Pack(flex=1))

        name_box = toga.Box(style=Pack(direction=ROW, margin=5))
        name_box.add(name_label)
        name_box.add(self.name_input)
...
```

src/helloworld/app.py, v2 (2)

```
...
        button = toga.Button(
            "Say Hello!",
            on_press=self.say_hello,
            style=Pack(margin=5)
        )

        main_box.add(name_box)
        main_box.add(button)
...
```

src/helloworld/app.py, v2 (3)

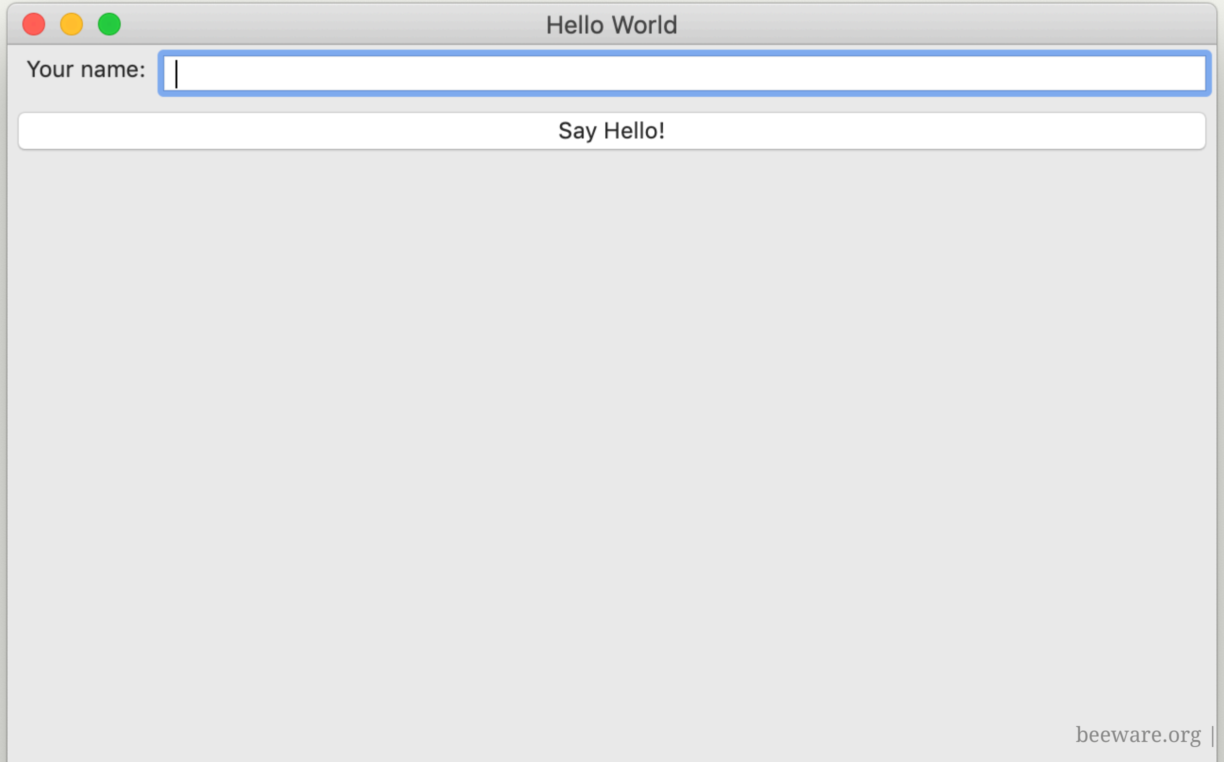
```
...
        self.main_window = toga.MainWindow(
            title=self.formal_name
        )
        self.main_window.content = main_box
        self.main_window.show()

    def say_hello(self, widget, **kwargs):
        print(f"Hello, {self.name_input.value}")
```

briefcase dev

```
(beeware) $ briefcase dev

[helloworld] Starting in dev mode...
===========================================================
```


Hello World
Your name:
Say Hello!

briefcase dev

```
(beeware) $ briefcase dev
```

briefcase create

```
(beeware) $ briefcase create

[helloworld] Generating application template...
Using app template: https://github.com/beeware/briefcase-macOS-
app-template.git
Using existing template ...

[helloworld] Installing support package...
Using support package https://...
Downloading Python-3.13-macOS-support.b3.tar.gz...
################################################## 100%
Unpacking support package...
```

`briefcase create` - The dependencies

```
[helloworld] Installing application code...
Installing src/helloworld... done

[hello-world] Installing dependencies...
Collecting toga-cocoa~=0.5.1
...
Installing collected packages: travertino, std-nslog, rubicon-
objc, toga-core, toga-cocoa
Successfully installed rubicon-objc-0.5.1 std-nslog-1.0.3 toga-
cocoa-0.5.1 toga-core-0.5.1 travertino-0.5.1
```

`briefcase create` - The app bits

```
[helloworld] Installing application resources...

[helloworld] Removing unneeded app content...
Removing unneeded app bundle content... done

[helloworld] Created build/helloworld/macOS/app
```

briefcase create

```
(beeware) $ briefcase create

[helloworld] Generating application template...
...
[helloworld] Created build/helloworld/macOS/app
```

briefcase build

```
(beeware) $ briefcase build
...
[hello-world] Built build/helloworld/macos/app/Hello World.app
```

briefcase run

```
(beeware) $ briefcase run

[hello-world] Starting app...
```

`briefcase package`

```
(beeware) $ briefcase package --adhoc-sign
...
[helloworld] Packaged dist/Hello World-0.0.1.dmg
```

- Debian/Ubuntu:
  - Install: `sudo dpkg -i dist/helloworld_0.0.1-....deb`
  - Uninstall: `sudo apt remove helloworld`
- Fedora:
  - Install: `sudo dnf localinstall dist/helloworld-0.0.1-....rpm`
  - Uninstall: `sudo dnf remove helloworld`

Packaging quirks

- Can't sign on Linux (patches welcome!)
- Package Linux apps for other Linux distros using Docker
  - `briefcase package --target ubuntu:noble`
  - `briefcase run --target ubuntu:noble`
- macOS: Build Linux packages using Docker
  - `briefcase package linux system --target ubuntu:noble`
- Target different package formats
  - `briefcase <command> <platform> <format>`

src/helloworld/app.py, v3

```
    async def say_hello(self, widget, **kwargs):
        await self.dialog(toga.InfoDialog(
            f"Hello, {self.name_input.value}",
            "Hi there!"
        ))
```

then:

```
(beeware) $ briefcase dev
```

```
(beeware) $ briefcase run
```

briefcase update

```
(beeware) $ briefcase update

[helloworld] Updating application code...
Installing src/helloworld... done

[helloworld] Removing unneeded app content...
Removing unneeded app bundle content... done

[helloworld] Application updated.
```

```
(beeware) $ briefcase run
```

src/helloworld/app.py, v4

```
def greeting(name):
    if name:
        return f"Hello, {name}"
    else:
        return "Hello, stranger"
```

```
    async def say_hello(self, widget, **kwargs):
        await self.dialog(toga.InfoDialog(
            greeting(self.name_input.value),
            "Hi there!",
        ))
```

```
(beeware) $ briefcase run -u
```

The story so far...

- `briefcase new`
- `briefcase dev`
- `briefcase create`
- `briefcase build`
- `briefcase run`
- `briefcase package --adhoc-sign`
- `briefcase update` (or `briefcase run -u`)

Briefcase command shorthand

- General form: `briefcase <command> <platform> <format>`
- `briefcase create`
  - shorthand for `briefcase create macOS app`
- `briefcase create macOS Xcode`
- `briefcase create windows VisualStudio`
- `briefcase create linux flatpak`
- `briefcase create linux appimage`

iOS

```
(beeware) $ briefcase create iOS
(beeware) $ briefcase build iOS
(beeware) $ briefcase run iOS
```

Android

```
(beeware) $ briefcase create android
(beeware) $ briefcase build android
(beeware) $ briefcase run android
```

Web

```
(beeware) $ briefcase create web
(beeware) $ briefcase build web
(beeware) $ briefcase run web
```

Console

```
(beeware) $ pip install toga-textual
(beeware) $ TOGA_BACKEND=toga_textual briefcase dev
(beeware) $ pip uninstall toga-textual
```

Third party packages

```python
import httpx
```

```python
async def say_hello(self, widget):
    with httpx.Client() as client:
        response = client.get(
            "https://jsonplaceholder.typicode.com/posts/42"
        )

    payload = response.json()

    await self.dialog(toga.InfoDialog(
        greeting(self.name_input.value),
        payload["body"],
    ))
```

Third party packages

```
(beeware) $ pip install httpx
(beeware) $ briefcase dev
```

Third party packages

```
(beeware) $ briefcase run
```

Third party packages

```
(beeware) $ briefcase run -u
```

Third party packages

In `pyproject.toml`:

```
requires = ["httpx"]
```

Then:

```
(beeware) $ briefcase run -u -r
```

Third party packages

```
import time
```

```
    ...
    payload = response.json()
    time.sleep(5)
    ...
    await self.dialog(toga.InfoDialog(...))
```

Event Loops

- All GUI frameworks are, at some level:

```
while not app.quit_requested():
  app.process_events()
  app.redraw()
```

- Event handlers can't be blocking
- `httpx.get()` is blocking

One approach: Threads

- Spawn a thread
- Do the heavy lifting in the thread
- No special API; use Python's `threading` library
- **WARNING**: Some GUI frameworks don't like this
    - Can't reliably perform GUI actions on the non-GUI thread.
    - Pass results back to GUI thread to be displayed.

Asynchronous event handlers

```
import asyncio
```

```
async def say_hello(self, widget):
    async with httpx.AsyncClient() as client:
        response = await client.get(
            "https://jsonplaceholder.typicode.com/posts/42"
        )
    payload = response.json()
    await asyncio.sleep(5)
    await self.dialog(toga.InfoDialog(
        greeting(self.name_input.value),
        payload["body"],
    ))
```

Native Third party packages

- Anything that is Pure Python should Just Work
  - Look for `-py3-none-any.whl` as a download (and dependencies)
- Anything with binary components:
  - Look for `-cp310-cp310-macosx_11_0_arm64.whl` (or similar)
  - will Just Work on desktop
  - can *probably* be made to work on mobile…
    - cibuildwheel 3.0 supports iOS and Android
    - BeeWare publishes a bunch of common packages

Native Third party packages

- Try adding `numpy` to your app
- Use numpy to generate a random number, add to the message. Hint:

```
import numpy as np
value = np.random.randint(100)
```

- Run in dev mode
- Run as a desktop app
- Run as a mobile app

Custom icons

Custom icons

- Add `icon = "icons/helloworld"` to your app config
- Run `briefcase update --update-resources`

```
[helloworld] Updating application resources...
Unable to find icons/helloworld.icns for application icon;
using default
```

Custom icons

- Download https://beeware.org/t/icons
- Unpack zip file into root of your project
- Run `briefcase run --update-resources`

```
...
[helloworld] Updating application resources...
Installing icons/helloworld.icns as application icon... done
...
```

Splash screen customization

- Run:
  - `briefcase run iOS --update-resources`
  - `briefcase run android --update-resources`
- Add `splash_background_color = "#D3E6F5"` to pyproject.
- Re-create then re-run project:
  - `briefcase create iOS`
  - `briefcase run iOS`

`build` should be ephemeral

- `build` folder contains templated output
    - Templated content can't be updated
- Re-create whenever you make a change that isn't code/requires
- Release strategy should assume `build` doesn't exist
- Don't commit `build` to version control
- Modify the *template* if you need customization

Using device hardware

Accessing the camera

- Add code to access the camera
    - https://beeware.org/t/code-camera
- Declare that we need permission to use the camera:
    - `permission.camera = "App will take mugshots."`
- Re-create and re-run project:
    - `briefcase create android`
    - `briefcase run android`

App Testing

- "Real" apps need to be tested
- Testing needs to happen in "real" environments
- Briefcase has a test mode
- The stub app contains a test suite

tests/test_app.py

```python
def test_first():
    "An initial test for the app"
    assert 1 + 1 == 2
```

Run in test mode

```
(beeware) $ briefcase dev --test -r
...
[helloworld] Running test suite in dev environment...
============================================================
=================== test session starts ====================
...
tests/test_app.py::test_first PASSED                  [100%]

==================== 1 passed in 0.00s =====================

[helloworld] Test suite passed!
```

Test driven development

- Write tests to verify existing behavior of `greeting()`
  - Hint: `from helloworld.app import greeting`
- Run those tests (`briefcase dev --test -r`)
- Write a test for new behavior of `greeting()`
- Run the tests, see the test fail (`briefcase dev --test`)
- Update the implementation of `greeting()`; re-run

Test code

```
from helloworld.app import greeting

def test_name():
    assert greeting("Alice") == "Hello, Alice"

def test_empty():
    assert greeting("") == "Hello, stranger"

def test_brutus():
    assert greeting("Brutus") == "BeeWare the IDEs of Python!"
```

```
(beeware) $ briefcase dev --test
```

Run in test mode on device

As an app:

```
(beeware) $ briefcase run --test -r
```

On iOS:

```
(beeware) $ briefcase run iOS --test -r
```

On Android:

```
(beeware) $ briefcase run Android --test -r
```

Wrapping a web site as an app

- Positron: Electron… but in Python
- `pip install toga-positron`
- `briefcase new`; then select
  - Static: Static web pages as an app
  - Django: Django web site as an app
  - Single-page web app: Wrap "github.com" as an app
- Generates a Toga app
  - … which means it can be a hybrid
  - … and it works on mobile

Wrapping a web site as an app

- Positron: Electron… but in Python
- `pip install toga-positron`
- `briefcase new`; then select
  - Static: Static web pages as an app
  - Django: Django web site as an app
  - Single-page web app: Wrap "github.com" as an app
- Generates a Toga app
  - … which means it works on mobile
  - … and it can be a hybrid app

Extension activities

- Customize the app! Some suggestions:
  - Put the 2 pieces of functionality behind tabs
    - Add a toga.OptionContainer
    - Docs at https://toga.readthedocs.io
  - Add a third function: showing a map
    - toga.MapView
- Look at the Toga example projects:
  - https://github.com/beeware/toga

Wrapping up...

```
briefcase open
briefcase upgrade
```

It's Just Python

- If it's legal Python, it's legal BeeWare.
- The only exceptions:
  - `fork/spawn`
  - `tk`
- Supports Python 3.8/PEP566 `importlib.metadata`

```
from importlib import metadata as importlib_metadata

meta = importlib_metadata.metadata('hello_world')
print(meta['Version'])
print(meta['Formal-Name'])
```

Getting down to the metal

- Any native platform API is accessible
- In toga, `widget._impl.native` is the underlying native implementation
  - macOS, iOS: Rubicon-ObjC
  - GTK: gobject Python API
  - Windows: Python.net
  - Android: Chaquopy
  - Web: Pyscript/Pyodide DOM APIs.
- Switch on `toga.current_platform`/`sys.platform`

Going Further

- Briefcase's documentation:
  - briefcase.readthedocs.io
- Toga's documentation:
  - toga.readthedocs.io
- Toga's examples repository
  - github.com/beeware/toga
- BeeWare Tutorial
  - docs.beeware.org

BeeWare

Write Python. Deploy everywhere.

beeware.org/bee/join/

Thank you!

beeware.org

russell@keith-magee.com

@freakboy3742