Tools for maintaining an open source Python project

Ben Nuttall
@ben_nuttall
Ben Nuttall

- Software engineer at BBC News Labs
- Formerly at Raspberry Pi Foundation
- Creator of gpiozero, piwheels and pyjokes
- Opensource.com correspondent
- twitter.com/ben_nuttall
- github.com/bennuttall
Ben Nuttall

- Software engineer at BBC News Labs
- Formerly at Raspberry Pi Foundation
- Creator of gpiozero, piwheels and pyjokes
- Opensource.com correspondent
- twitter.com/ben_nuttall
- github.com/bennuttall
What this talk covers

- Organising a Python module
- Distributing software
- Using git/GitHub
- Virtual environments
- Testing & automated testing
- Documentation
- Licensing software
What this talk is not

- A thorough follow-along tutorial on how to use each of the ~50 tools mentioned
- Me telling you which tools to use
- Me telling you that you need to know all of these tools inside-out in order to be considered a proper Python programmer
• Python library providing simple API for physical computing with Raspberry Pi
• Eases the learning curve for young people, beginners and educators
• Nice Pythonic API with advanced tooling for experienced programmers
• gpiozero.readthedocs.io
• github.com/gpiozero/gpiozero
piwheels

• Tooling for automating building wheels of everything on PyPI
• piwheels.org – pip-compatible repository hosting Arm wheels
• Natively compiled Arm wheels built on Pi 3 hardware
• Repository hosted on 1 × Pi serves 1 million downloads per month
• piwheels.org
• github.com/piwheels/piwheels
Dave Jones

- Professional programmer, amateur dentist
- Responsible for implementing my crazy ideas
- I write the first 90%, he writes the next 90%
- Co-author of gpiozero and piwheels (also author of picamera, colorzero, picraft, sense-emu, lars, structa, compoundpi, pisense, pibootctl, ...)
- Introduced me to most of the tools in this talk

@ben_nuttall
Writing a Python module

    ├── project.py
GitHub - organisation

@ben_nuttall
GitHub - organisation
GitHub - collaborators

Who has access

PUBLIC REPOSITORY
This repository is public and visible to anyone.
Manage

BASE ROLE
All 3 members can access this repository.
Manage

DIRECT ACCESS
0 teams or members have access to this repository. Only Owners can contribute to this repository.

Manage access

You haven't added any teams or people yet

Organization owners can manage individual and team access to the organization’s repositories. Team maintainers can also manage a team’s repository access. Learn more about organization access

Invite teams or people
## GitHub - branches

### Default branch

<table>
<thead>
<tr>
<th>Branch</th>
<th>Updated</th>
<th>Default</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>master</td>
<td>a month ago by waveform80</td>
<td>✅</td>
<td>![Change default branch]</td>
</tr>
</tbody>
</table>

### Your branches

<table>
<thead>
<tr>
<th>Branch</th>
<th>Updated</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>rgb16board</td>
<td>2 months ago by ben_nuttall</td>
<td>![Open]</td>
</tr>
<tr>
<td>lurch-patch-1</td>
<td>a month ago by lurch</td>
<td>![Open]</td>
</tr>
</tbody>
</table>

### Active branches

<table>
<thead>
<tr>
<th>Branch</th>
<th>Updated</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>coverage_bump</td>
<td>a month ago by lurch</td>
<td>![Open]</td>
</tr>
<tr>
<td>lurch-patch-1</td>
<td>a month ago by lurch</td>
<td>![Open]</td>
</tr>
<tr>
<td>rgb16board</td>
<td>2 months ago by ben_nuttall</td>
<td>![Open]</td>
</tr>
<tr>
<td>Version</td>
<td>Date</td>
<td>Commit Hash</td>
</tr>
<tr>
<td>---------</td>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td>v1.5.0</td>
<td>12 Feb</td>
<td>9cb4a4aa</td>
</tr>
<tr>
<td>v1.4.1</td>
<td>21 Feb 2018</td>
<td>50f5b0c</td>
</tr>
<tr>
<td>v1.4.0</td>
<td>26 Jul 2017</td>
<td>791d22</td>
</tr>
<tr>
<td>v1.3.2</td>
<td>3 Mar 2017</td>
<td>2e7643d</td>
</tr>
<tr>
<td>#</td>
<td>Title</td>
<td>Owner</td>
</tr>
<tr>
<td>---</td>
<td>----------------------------------------------------------------------</td>
<td>-------</td>
</tr>
<tr>
<td>1</td>
<td>ButtonBoard release events?</td>
<td>@ben_nuttall</td>
</tr>
<tr>
<td>2</td>
<td>Add support for 5 channels line follower detector</td>
<td>@Irrumousar</td>
</tr>
<tr>
<td>3</td>
<td>Question: multiprocessing and gpizero</td>
<td>@Irrumousar</td>
</tr>
<tr>
<td>4</td>
<td>Servo docs don't make it clear that it can be driven to any position</td>
<td>@Irrumousar</td>
</tr>
<tr>
<td>5</td>
<td>Internal device values are still boolean</td>
<td>@lurch</td>
</tr>
<tr>
<td>6</td>
<td>Diagram inconsistencies</td>
<td>@lurch</td>
</tr>
<tr>
<td>7</td>
<td>Provide a method of setting per-device init params to composite devices</td>
<td>@ben_nuttall</td>
</tr>
<tr>
<td>8</td>
<td>Why doesn’t pinout -x open in chromium?</td>
<td>@ben_nuttall</td>
</tr>
<tr>
<td>9</td>
<td>New Feature Request - GPIO and Desktop Via Ethernet</td>
<td>@JonnyAlpha</td>
</tr>
<tr>
<td>10</td>
<td>Class init documentation inconsistencies</td>
<td>@lurch</td>
</tr>
<tr>
<td>11</td>
<td>Request for better MockPin docs</td>
<td>@lurch</td>
</tr>
</tbody>
</table>

@ben_nuttall
GitHub - pull requests

Add StepperMotor #757

Commits on Mar 16, 2019
- Add StepperMotor
  - m-alzam committed 10 days ago
- Remove '*.c' files
  - m-alzam committed 10 days ago

Commits on Mar 17, 2019
- Add StepperMotor documentation
  - m-alzam committed 9 days ago

Commits on Mar 20, 2019
- Improve StepperMotor
  - m-alzam committed 6 days ago

@ben_nuttall
# GitHub - project boards

<table>
<thead>
<tr>
<th>Title</th>
<th>Questions</th>
<th>Pending suggestions</th>
<th>In progress</th>
<th>Important / ready</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 Triage</td>
<td>7 Questions</td>
<td>16 Pending suggestions</td>
<td>2 In progress</td>
<td>12 Important / ready</td>
</tr>
<tr>
<td>1 Button bounce time</td>
<td>2 Addon boards compatibility question</td>
<td>3 Asymmetric SmoothedInputDevice?</td>
<td>4 RFC: Polymorphic Parameters for CompositeDevice subclasses</td>
<td>[WIP] Add RGBLEDBoard Class</td>
</tr>
<tr>
<td>2 Addon boards compatibility question</td>
<td>4 Software debounce missing input events</td>
<td>5 Add support for MCP230x8, MCP23017</td>
<td>6 Temperature Sensor</td>
<td>Add Stepper Motor</td>
</tr>
<tr>
<td>3 Software debounce missing input events</td>
<td>5 MCP2201 readings are unstable</td>
<td>6 RGBLED.cyclic()</td>
<td>6 Document Iterable behaviour</td>
<td>Add support for more motors</td>
</tr>
<tr>
<td>4 MCP2201 readings are unstable</td>
<td>6 TimeOfDay questions</td>
<td>7 Add circuit diagrams to docs</td>
<td>7 pigo pin does not support float frequencies</td>
<td>Allow users to differentiate between pressed and held events</td>
</tr>
</tbody>
</table>

@ben_nuttall
Distributing software – how?

Package managers:

- Linux – apt, rpm, yum
- Language package managers – pip, npm, gem
- Linux portable – snap, flatpak, AppImage
- Mac – homebrew

Download from sites:

- GitHub, GitLab, Sourceforge
- Personal websites
- curl
Distributing software – why?

- Ease of access
- Expectations
- Trust and confidence
- Stability
Licensing

- It’s important to choose a licence for a project
- Think about what would annoy you
- It’s important to specify which licence
- It’s important to include the licence with the source code and distributions
- Refer to choosealicense.com
Packaging a Python module

```
.
├── project
│   ├── __init__.py
│   └── project.py
│   └── README.rst
└── setup.py
```
import os
from setuptools import setup, find_packages

def read(fname):
    return open(os.path.join(os.path.dirname(__file__), fname)).read()

setup(
    name="project",
    version="0.1.0",
    author="Ben Nuttall",
    description="Really cool project",
    license="MIT",
    keywords=["sample", "project"],
    url="https://github.com/bennuttall/project",
    packages=find_packages(),
    long_description=read('README.rst'),
)
Publishing a Python module on PyPI

- Register an account on pypi.org
- Put your account details in ~/.pypirc:
  
  ```
  ben@magicman:~ $ cat .pypirc
  [pypi]
  username: bennuttall
  password: correcthorsebatterystaple
  ```
- Install Twine:
  - `pip install twine`
Publishing a Python module on PyPI
User:

```python
from gpiozero import LED
```

__init__.py:

```python
from .input_devices import LED, ...
```

setup.py:

```python
__version__ = '1.5.1'
setup(
    version=__version__
    ...
)
```
__init__.py choices: piwheels

User:

```python
from piwheels.master.pypi import PiWheelsTransport
```

__init__.py:

```python
__version__ = '0.17'
```

setup.py

```python
import piwheels as app

setup(
    version=app.__version__,
    ...)
```
setup.py:

entry_points = {
    'console_scripts': [
        'project = project.cli:main',
    ],
}

setup(
    ...
    entry_points=entry_points,
    ...
  )
Virtual environments

- Virtual environment for a Python project
- You create the environment, pip install into it
- Isolated from your system Python and system packages
- Build your project inside it, with changes "installed" in real time

```
mkvirtualenv -p /usr/bin/python3 project
pip install -e .
pip install ipython
deactivate
workon project
```
all:
  @echo "make install - Install on local system"
  @echo "make develop - Install symlinks for development"

install:
  pip install .

develop:
  pip install -e .
Testing

- Write tests to validate what your code is supposed to do
- Keep your old tests to make sure nothing breaks in future
- For maximum effect, write tests before you write code!
- Testing can be performed quickly locally
- Testing can be automated – e.g. Travis after push
- Be pragmatic! Test edge cases, don’t be exhaustive
from project import add

assert add(2, 2) == 4
Testing - pytest

```python
def test_add():
    assert add(2, 2) == 4
```
Testing - layout

.  ├── Makefile
    │   └── project
    │       ├── __init__.py
    │       └── project.py
    │   └── setup.py
    └── tests
        └── test_add.py
pytest

```
(project) ben@gunter:/Projects/bennuttall/project $ tree

Makefile
project
  __init__.py
  project.py
  setup.py
  tests
    __pycache__
      test_add.cpython-38-pytest-5.4.3.pyc
      test_add.py

3 directories, 6 files
(project) ben@gunter:/Projects/bennuttall/project $ pytest -v

platform linux -- Python 3.8.2, pytest-5.4.3, py-1.9.0, pluggy-0.13.1 -- /home/ben/.environments/project/bin/python
cachedir: .pytest_cache
collectdir: /home/ben/Projects/bennuttall/project
collected 1 item

tests/test_add.py::test_add PASSED

1 passed in 0.01s

(project) ben@gunter:/Projects/bennuttall/project $
```
from project import add
import pytest

assert add(2, 2) == 4
with pytest.raises(TypeError):
    add("foo", "bar")
>>> from unittest.mock import Mock
>>> m = Mock(msg=Mock(return_value="hello"))
>>> m
<Mock id='140656083514272'>
>>> m.msg()
'hello'
def test_timeofday_value(mock_factory):
    with TimeOfDay(time(7), time(8), utc=False) as tod:
        assert repr(tod).startswith('<gpiozero.TimeOfDay object')
        assert tod.start_time == time(7)
        assert tod.end_time == time(8)
        assert not tod.utc
        with patch('gpiozero.internal_devices.datetime') as dt:
            dt.now.return_value = datetime(2018, 1, 1, 6, 59, 0)
            assert not tod.is_active
            dt.now.return_value = datetime(2018, 1, 1, 7, 0, 0)
            assert tod.is_active
            dt.now.return_value = datetime(2018, 1, 2, 8, 0, 0)
            assert tod.is_active
            dt.now.return_value = datetime(2018, 1, 2, 8, 1, 0)
            assert not tod.is_active
Tox

- Run tests in multiple Python versions simultaneously
- Ubuntu users – look for "Deadsnakes PPA"
- tox.ini:
  ```
  [tox]
  envlist = {py27, py35, py36, py37, py38}
  ```
Coverage.py

- Measuring code coverage of Python programs
- Monitors your program, noting which parts of the code have been executed
- Analyses the source to identify code that could have been executed but was not
- Typically used to gauge the effectiveness of tests
- Shows which parts of your code are being touched by your tests, and which are not
Coverage

The coverage report shows the following:

<table>
<thead>
<tr>
<th>Name</th>
<th>Stmts</th>
<th>Miss Branch</th>
<th>Br Part</th>
<th>Cover</th>
<th>Missing</th>
</tr>
</thead>
<tbody>
<tr>
<td>gptzerof/int.py</td>
<td>11</td>
<td>0</td>
<td>0</td>
<td>100%</td>
<td></td>
</tr>
</tbody>
</table>
| gptzerof/boards.py        | 403   | 1           | 818     | 99%   | 0/0/60 
| gptzerof/compat.py        | 96    | 4           | 30      | 97%   | 52/53, 98, 120 |
| gptzerof/devices.py       | 232   | 2           | 78      | 1     | 99%     | 442-443, 462-461 |
| gptzerof/enc.py           | 60    | 0           | 0       | 100%  |         |
| gptzerof/internal_devices.py | 141  | 10         | 20      | 94%   | 180-101, 192-193, 234-235, 380-381, 475-476 |
| gptzerof/mains.py         | 236   | 8           | 74      | 97%   | 48-69, 87, 179, 300-308, 358-368, 113=exit |
| 723-1727, 1779-1786       |       |             |         |       |         |
| gptzerof/plans/date.py    | 305   | 5           | 108     | 97%   | 104-1049, 1188, 1255, 1331, 713=exit, 1187=exit, 1234=1252, 1295=exit, 1330=1331 |
| gptzerof/plans/local.py   | 165   | 8           | 18      | 1%    | 48-49, 98, 179-188, 230, 253-254, 97=98 |
| gptzerof/plans/pl.py      | 121   | 10          | 40      | 93%   | 44-46, 107-113, 291, 290=291 |
| gptzerof/plans/pig.py     | 297   | 293         | 68      | 1%    | 40-580 |
| gptzerof/plans/rpt.py     | 126   | 122         | 39      | 3%    | 43-255 |
| gptzerof/plans/sp.py      | 123   | 121         | 10      | 3%    | 43-267 |
| gptzerof/plans/spl.py     | 67    | 2           | 28      | 94%   | 102-103, 57-59, 59=exit, 101-102, 111=114 |
| gptzerof/spl_devices.py   | 100   | 7           | 34      | 98%   | 45-46, 85-88, 194, 113-114, 105-106, 215-229 |
| gptzerof/thrds.py         | 32    | 2           | 10      | 97%   | 77-80, 59-65, 76-77 |
| gptzerof/tones.py         | 86    | 8           | 28      | 3%    | 16-17, 91, 108, 118-117, 122-123, 90-91, 94-102, 105-100 |
| gptzerof/tools.py         | 231   | 4           | 130     | 98%   | 54-55, 58-59, 406=exit, 497=exit, 648=exit, 675=exit |

TOTAL: 4450 879 1319 56 81%
GitHub – Travis CI & codecov integration

Codecov Report

Merging #757 into master will decrease coverage by 1.19%.
The diff coverage is 19.78%.

<table>
<thead>
<tr>
<th>Coverage Diff</th>
<th>_coverage</th>
<th>master</th>
<th>#757</th>
<th>+/-</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coverage</td>
<td>79.46%</td>
<td>78.20%</td>
<td>-1.26%</td>
<td></td>
</tr>
<tr>
<td>Files</td>
<td>23</td>
<td>23</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lines</td>
<td>4478</td>
<td>4568</td>
<td>+90</td>
<td></td>
</tr>
<tr>
<td>Branches</td>
<td>650</td>
<td>697</td>
<td>+47</td>
<td></td>
</tr>
<tr>
<td>Hits</td>
<td>3552</td>
<td>3569</td>
<td>+17</td>
<td></td>
</tr>
<tr>
<td>Misses</td>
<td>603</td>
<td>106</td>
<td>+73</td>
<td></td>
</tr>
<tr>
<td>Partials</td>
<td>56</td>
<td>56</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Impacted Files
- gpiozero__init__.py
- gpiozero/output_devices.py

Default branch
- master Updated a month ago by waveform80

Your branches
- rgbledboard Updated 2 months ago by ben_nuttall
- lurch-patch-1 Updated a month ago by lurch

Active branches
- coverage_bump Updated a month ago by lurch
- lurch-patch-1 Updated a month ago by lurch
- rgbledboard Updated 2 months ago by ben_nuttall
Makefiles

all:
@echo "make install - Install on local system"
@echo "make develop - Install symlinks for development"
@echo "make test - Run tests"

install:
    pip install .

develop:
    pip install -e .

test:
    coverage run --rcfile coverage.cfg -m pytest -v tests
    coverage report --rcfile coverage.cfg
• Tutorials
• How-to guides
• Explanation
• Reference
• https://documentation.divio.com/
Run your own instance

1. Sign up for a Twitter account, create an app and get your four API keys.
2. Sign up for an npmjs account and note your username and password.
3. Register for a Stack Exchange App Key
4. Install this project
   ```bash
   sudo pip3 install memoverflow
   ```
5. Copy the example script `example.py` (e.g. `rasperrypi.py`) and edit your copy to specify:
   - the Stack Exchange site you wish to follow (the exact string here) and your Stack Exchange API key
   - your npmjs's username and password
6. Run it:
   ```bash
   python3 rasperrypi.py
   ```

Raspberry Pi revision codes

Each distinct Raspberry Pi model revision has a unique revision code. You can look up a Raspberry Pi's revision code by running:

```
cat /proc/cpuinfo
```

The last three lines show the hardware type, the revision code, and the Pi's unique serial number. For example:

```
Hardware: BCM2835
Revision: a02682
Serial: 0000000000000000
```

Note: As of the 4.9 kernel, all Pis report `rev 0000`, even those with BCM2835 and BCM2837 processors. You should not use this string to detect the processor.

Old-style revision codes

The first set of Raspberry Pi revisions were given sequential hex revision codes from 0002 to 0015:
# Title

Some text

## Header 2

- List item
- [link](http://foo.com/)
Documentation - mkdocs

- Markdown-based documentation builder
- Exports to static HTML
- Easy to write, easy to deploy
- Can host anywhere – e.g. GitHub pages or self-hosted
Some text

Header 2

* List item
* :doc:`api_input`
• ReST
• Extracts docs from docstrings
• Can embed additional bespoke docs
• Multiple outputs:
  • HTML
  • PDF
  • Epub
• Language docs linking (e.g. gpiozero can link to Python docs using ReST)
• Cross-project linking (e.g. gpiozero can link to picamera using ReST)
Regular Classes
===============

The following classes are intended for general use with the devices they represent. All classes in this section are concrete (not abstract).

LED
---

.. autoclass:: LED(pin, *, active_high=True, initial_value=False, pin_factory=None)
   :members: on, off, toggle, blink, pin, is_lit, value

PWMLED
-----

.. autoclass:: PWMLED(pin, *, active_high=True, initial_value=0, frequency=100, pin_factory=None)
   :members: on, off, toggle, blink, pulse, pin, is_lit, value
datetime — Basic date and time types

The `datetime` module supplies classes for manipulating dates and times.

While date and time arithmetic is supported, the focus of the implementation is on efficient attribute extraction for output formatting and manipulation.

See also:
- **Module calendar**
  General calendar related functions.
- **Module time**
  Time access and conversions.
- **Package dateutil**
  Third-party library with expanded time zone and parsing support.

Aware and Naive Objects

Date and time objects may be categorized as "aware" or "naive" depending on whether or not they include timezone information.

With sufficient knowledge of applicable algorithms and political time adjustments, such as time zone and daylight saving time information, an aware object can locate itself relative to other aware objects. An aware object represents a specific moment in time that is not open to interpretation.  

A naive object does not contain enough information to unambiguously locate itself relative to other datetime objects. Whether a naive object represents Coordinated Universal Time (UTC), local time, or time in some other timezone is purely up to the program, just like it is up to the program whether a particular number represents metres, miles, or mass. Naive objects are easy to understand and to work with, at the cost of ignoring some aspects of reality.

For applications requiring aware objects, `datetime` and `time` objects have an optional time zone information attribute, `tzinfo`, that can be set to an instance of a subclass of the abstract `tzinfo` class. These `tzinfo` objects capture information about the offset from UTC time, the time zone name, and whether daylight saving
Documentation - ReadTheDocs

- Multiple versions (v1.0, v1.1, v1.2...)
- Branches
- Multi-user management
- Easy to integrate with GitHub to automate building
7. Source/Values

GPIO Zero provides a method of using the declarative programming paradigm to connect devices together: feeding the values of one device into another, for example the values of a button into an LED:

```python
from gpiozero import LED, Button
from signal import pause

led = LED(17)
button = Button(2)

led.source = button
pause()
```

```python
from gpiozero import LED, MotionSensor, LightSensor
from gpiozero.tools import booleanized, all_values
from signal import pause

garden = LED(2)
motion = MotionSensor(4)
light = LightSensor(5)

garden.source = all_values(booleanized(light, 0, 0.1), motion)
pause()
```
digraph {
  graph [rankdir=RL];
  node [shape=rect, style=filled, color="#2980b9", fontname=Sans, fontcolor="#ffffff", fontsize=10];
  edge [arrowhead=normal, style=solid];

  Button -> LED;
}
```plaintext
digraph {
  graph [rankdir=RL];
  edge [arrowhead=normal, style=solid];

  /* Devices */
  node [shape=rect, style=filled, color="#2980b9", fontname=Sans, fontcolor="#ffffff", fontsize=10];

  led [label="Garden light"]
  light [label="Light sensor"]
  motion [label="Motion sensor"]

  /* functions */
  node [shape=oval, style=filled, color="#9ec6e0", fontcolor="#ffffff"];;

  booleanized
  all_values

  all_values -> led;
  booleanized -> all_values;
  motion -> all_values;
  light -> booleanized;
}
```
Documentation - Graphviz
Project structure

```
.
├── coverage.cfg
├── docs/
│   ├── .github/
│   │   └── .gitignore
├── LICENSE
├── Makefile
├── project/
├── setup.py
├── tests/
├── tox.ini
└── .travis.yml
```
What this talk covered

- Organising a Python module – module structure, setup.py, Makefiles
- Distributing software – PyPI, pip
- Using git/GitHub – repositories, users & orgs, collaborators, issues, PRs, project boards, integrations
- Virtual environments – virtualenvwrapper
- Testing & automated testing – assert, pytest, mock, coverage, tox, Travis CI
- Documentation – markdown, ReST, mkdocs, sphinx, graphviz
- Licensing software – choosealicense.org
Tooling Tuesday

- My tooling blog: https://tooling.bennuttall.com/
- Inspired by Les Pounder: https://bigl.es/
- New posts every Tuesday
- New posts every other Tuesday
- New posts every now and then, sometimes on a Tuesday
Tools for maintaining an open source Python project

Ben Nuttall
@ben_nuttall