
Adafruitpyportal Library Documentation

Release 1.0

Melissa LeBlanc-Williams

Jan 30, 2023

CONTENTS

1 Dependencies 3

2 Installing from PyPI 5

3 Contributing 7

4 Documentation 9

5 Table of Contents 11

5.1 Simple test 11

5.2 adafruit_pyportal 12

5.2.1 Implementation Notes 12

6 Indices and tables 15

Python Module Index 17

Index 19

A port of the PyPortal library intended to run on Blinky in CPython. This library will not work with CircuitPython. If you want to run on CircuitPython, you will instead want to use: [Adafruit CircuitPython PyPortal](#)

DEPENDENCIES

This driver depends on:

- [Adafruit Blinka](#)
- [Adafruit Blinka DisplayIO](#)

INSTALLING FROM PYPI

On supported GNU/Linux systems like the Raspberry Pi, you can install the driver locally [from PyPI](#). To install for current user:

```
pip3 install adafruit-blinka-pyportal
```

To install system-wide (this may be required in some cases):

```
sudo pip3 install adafruit-blinka-pyportal
```

To install in a virtual environment in your current project:

```
mkdir project-name && cd project-name  
python3 -m venv .env  
source .env/bin/activate  
pip3 install adafruit-blinka-pyportal
```


CONTRIBUTING

Contributions are welcome! Please read our [Code of Conduct](#) before contributing to help this project stay welcoming.

DOCUMENTATION

For information on building library documentation, please check out [this guide](#).

TABLE OF CONTENTS

5.1 Simple test

Ensure your device works with this simple test.

Listing 1: examples/adafruit_blinka_pyportal_bitcoin.py

```
1  # SPDX-FileCopyrightText: 2017 Scott Shawcroft, written for Adafruit Industries
2  #
3  # SPDX-License-Identifier: Unlicense
4  """
5  This example will access the coindesk API, grab a number like bitcoin value in
6  USD and display it on a screen
7  If you can find something that spits out JSON data, we can display it!
8
9  You can find any resources in the associated Learn Guide at:
10 https://learn.adafruit.com/pyportal-bitcoin-value-display
11
12 Note: This library is designed to run on CPython and not CircuitPython.
13 """
14 import os
15 import time
16 from adafruit_pyportal import PyPortal
17
18 # You can display in 'GBP', 'EUR' or 'USD'
19 CURRENCY = "USD"
20 # Set up where we'll be fetching data from
21 DATA_SOURCE = "https://api.coindesk.com/v1/bpi/currentprice.json"
22 DATA_LOCATION = ["bpi", CURRENCY, "rate_float"]
23
24
25 def text_transform(val):
26     if CURRENCY == "USD":
27         return "$%d" % val
28     if CURRENCY == "EUR":
29         return "€%d" % val
30     if CURRENCY == "GBP":
31         return "£%d" % val
32     return "%d" % val
33
34
```

(continues on next page)

(continued from previous page)

```

35 # the current working directory (where this file is)
36 try:
37     cwd = os.path.dirname(os.path.realpath(__file__))
38 except AttributeError:
39     cwd = ("/" + __file__).rsplit("/", 1)[0]
40
41 pyportal = PyPortal(
42     url=DATA_SOURCE,
43     json_path=DATA_LOCATION,
44     default_bg=cwd + "/bitcoin_background.bmp",
45     text_font=cwd + "/fonts/Arial-Bold-24-Complete.bdf",
46     text_position=(195, 130),
47     text_color=0x0,
48     text_transform=text_transform,
49 )
50 pyportal.preload_font(b"$012345789") # preload numbers
51 pyportal.preload_font((0x00A3, 0x20AC)) # preload gbp/euro symbol
52
53 while True:
54     try:
55         value = pyportal.fetch()
56         print("Response is", value)
57     except (ValueError, RuntimeError) as e:
58         print("Some error occurred, retrying! -", e)
59
60     time.sleep(3 * 60) # wait 3 minutes

```

5.2 adafruit_pyportal

A port of the PyPortal library intended to run on Blinka in CPython.

- Author(s): Melissa LeBlanc-Williams

5.2.1 Implementation Notes

Software and Dependencies:

- Adafruit Blinka for supported boards: https://github.com/adafruit/Adafruit_Blinka/releases

```

class adafruit_pyportal.PyPortal(*, url=None, headers=None, json_path=None, regexp_path=None,
                                convert_image=True, default_bg=0, status_neopixel=None,
                                text_font=<fontio.BuiltinFont object>, text_position=None,
                                text_color=8421504, text_wrap=False, text_maxlen=0,
                                text_transform=None, text_scale=1, json_transform=None,
                                image_json_path=None, image_resize=None, image_position=None,
                                image_dim_json_path=None, caption_text=None, caption_font=None,
                                caption_position=None, caption_color=8421504, image_url_path=None,
                                success_callback=None, esp=None, external_spi=None, debug=False,
                                display=None, touchscreen=None, secrets=None)

```

Class representing the Adafruit PyPortal.

Parameters

- **url** – The URL of your data source. Defaults to `None`.
- **headers** – The headers for authentication, typically used by Azure API's.
- **json_path** – The list of json traversal to get data out of. Can be list of lists for multiple data points. Defaults to `None` to not use json.
- **regex_path** – The list of regex strings to get data out (use a single regex group). Can be list of regexps for multiple data points. Defaults to `None` to not use regex.
- **convert_image** – Determine whether or not to use the AdafruitIO image converter service. Set as `False` if your image is already resized. Defaults to `True`.
- **default_bg** – The path to your default background image file or a hex color. Defaults to `0x000000`.
- **status_neopixel** – The pin for the status NeoPixel. Use `board.NEOPIXEL` for the on-board NeoPixel. Defaults to `None`, not the status LED
- **text_font** (*str*) – The path to your font file for your data text display.
- **text_position** – The position of your extracted text on the display in an (x, y) tuple. Can be a list of tuples for when there's a list of json_paths, for example
- **text_color** – The color of the text, in 0xRRGGBB format. Can be a list of colors for when there's multiple texts. Defaults to `None`.
- **text_wrap** – Whether or not to wrap text (for long text data chunks). Defaults to `False`, no wrapping.
- **text_maxlen** – The max length of the text for text wrapping. Defaults to 0.
- **text_transform** – A function that will be called on the text before display
- **text_scale** (*int*) – The factor to scale the default size of the text by
- **json_transform** – A function or a list of functions to call with the parsed JSON. Changes and additions are permitted for the dict object.
- **image_json_path** – The JSON traversal path for a background image to display. Defaults to `None`.
- **image_resize** – What size to resize the image we got from the json_path, make this a tuple of the width and height you want. Defaults to `None`.
- **image_position** – The position of the image on the display as an (x, y) tuple. Defaults to `None`.
- **image_dim_json_path** – The JSON traversal path for the original dimensions of image tuple. Used with `fetch()`. Defaults to `None`.
- **success_callback** – A function we'll call if you like, when we fetch data successfully. Defaults to `None`.
- **caption_text** (*str*) – The text of your caption, a fixed text not changed by the data we get. Defaults to `None`.
- **caption_font** (*str*) – The path to the font file for your caption. Defaults to `None`.
- **caption_position** – The position of your caption on the display as an (x, y) tuple. Defaults to `None`.
- **caption_color** – The color of your caption. Must be a hex value, e.g. `0x808000`.

- **image_url_path** – The HTTP traversal path for a background image to display. Defaults to `None`.
- **esp** – A passed ESP32 object, Can be used in cases where the ESP32 chip needs to be used before calling the pyportal class. Defaults to `None`.
- **external_spi** (*busio.SPI*) – A previously declared spi object. Defaults to `None`.
- **debug** – Turn on debug print outs. Defaults to `False`.
- **display** – The displayio display object to use
- **touchscreen** – The touchscreen object to use. Usually STMPE610 or FocalTouch.
- **secrets** – The secrets object to use. If not supplied we will attempt to import.

fetch(*refresh_url=None, timeout=10*)

Fetch data from the url we initialized with, perform any parsing, and display text or graphics. This function does pretty much everything Optionally update the URL

set_caption(*caption_text, caption_position, caption_color*)

A caption. Requires setting `caption_font` in `init`!

Parameters

- **caption_text** – The text of the caption.
- **caption_position** – The position of the caption text.
- **caption_color** – The color of your caption text. Must be a hex value, e.g. `0x808000`.

INDICES AND TABLES

- `genindex`
- `modindex`
- `search`

PYTHON MODULE INDEX

a

adafruit_pyportal, [12](#)

INDEX

A

`adafruit_pyportal`
module, [12](#)

F

`fetch()` (*adafruit_pyportal.PyPortal method*), [14](#)

M

module
adafruit_pyportal, [12](#)

P

`PyPortal` (*class in adafruit_pyportal*), [12](#)

S

`set_caption()` (*adafruit_pyportal.PyPortal method*),
[14](#)