CircuitPython Candlesticks Library Documentation

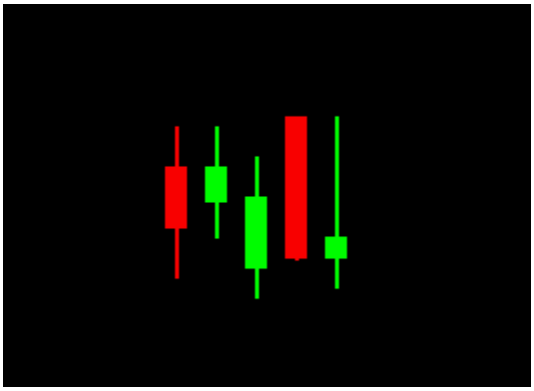
Release 1.0

Jose David

Feb 01, 2023

CONTENTS

Graphical representation of the stock movement in candlestick form



CHAPTER

ONE

DEPENDENCIES

This driver depends on:

- Adafruit CircuitPython

Please ensure all dependencies are available on the CircuitPython filesystem. This is easily achieved by downloading the Adafruit library and driver bundle or individual libraries can be installed using circup.

CHAPTER

TWO

USAGE EXAMPLE

See scripts in the examples directory of this repository.

CHAPTER

THREE

CONTRIBUTING

Contributions are welcome! Please read our Code of Conduct before contributing to help this project stay welcoming.

CHAPTER

FOUR

DOCUMENTATION

For information on building library documentation, please check out this guide.

CHAPTER

FIVE

TABLE OF CONTENTS

5.1 Simple test

Ensure your device works with this simple test.

Listing 1: examples/candlesticks_simpletest.py

```
# SPDX-FileCopyrightText: 2021 Jose David M.
#
# SPDX-License-Identifier: MIT
##############################
"""
This is a simple example of the use of the class candlestick.
"""

# import candlesticks_simpletest

import displayio
import board
from candlesticks import Candlestick

display = board.DISPLAY

my_candle = Candlestick(
    100,
    60,
    30,
    80,
    5,
    color_green=0x00FF00,
    color_red=0xFF0000,
    screen_ref=180,
)

my_candle2 = Candlestick(
    120,
    43,
    60,
    80,
    25,
    color_green=0x00FF00,
```

(continues on next page)

(continued from previous page)

```
    color_red=0xFF0000,
    screen_ref=180,
)

my_candle3 = Candlestick(
    140,
    10,
    45,
    65,
    25,
    color_green=0x00FF00,
    color_red=0xFF0000,
    screen_ref=180,
)

my_candle4 = Candlestick(
    160,
    85,
    15,
    85,
    15,
    color_green=0x00FF00,
    color_red=0xFF0000,
    screen_ref=180,
)
my_candle5 = Candlestick(
    180,
    15,
    25,
    85,
    30,
    color_green=0x00FF00,
    color_red=0xFF0000,
    screen_ref=180,
)


main_group = displayio.Group()
main_group.append(my_candle.my_rep)
main_group.append(my_candle2.my_rep)
main_group.append(my_candle3.my_rep)
main_group.append(my_candle4.my_rep)
main_group.append(my_candle5.my_rep)

display.show(main_group)

while True:
    pass
```

5.2 candlesticks

Graphical representation of the stock movement in candlestick form

- Author(s): Jose David

5.2.1 Implementation Notes

- Adafruit CircuitPython firmware for the supported boards: https://github.com/adafruit/circuitpython/releases

class candlesticks.**Candlestick**(*dist_x: int*, *openp: int*, *close: int*, *high: int*, *low: int*, *color_green: int = 65280*, *color_red: int = 16711680*, *screen_ref: int = 180*)

A graphical candlestick representation

Parameters

- **dist_x** (*int*) – number of segments in each bar
- **openp** (*int*) – Stock open price
- **close** (*int*) – Stock close price
- **high** (*int*) – Stock high price
- **low** (*int*) – Stock low price
- **color_green** (*int*) – When stock close price is higher thant the price opening candlestick are representing by a green color. This allows the selection of the color of your choice
- **color_red** (*int*) – When stock close price is lower thant the price opening candlestick are representing by a red color. This allows the selection of the color of your choice
- **screen_ref** (*int*) – Distance in pixels from the left to the screem to locate the candlestick. This allows to present different candlesticks in the same screen

Quickstart: Importing and using Candlestick

Here is one way of importing the *Candlestick* class, so you can use it as the name `my_candle`:

```
from CircuitPython_Candlesticks.candlesticks import Candlestick as Candlestick
```

Now you can create a plane at pixel position x=100, open price=60 close price=30 high price=80 low price=5 using:

```
my_candle = Candlestick(100, 60, 30, 80, 5)
```

Once you set up your display, you can now add `my_candle` to your display using:

```
display.show(my_plane) # add the group to the display
```

If you want to have multiple display elements, you can create a group and then append the plane and the other elements to the group. Then, you can add the full group to the display as in this example:

```
my_candle = Candlestick(100, 60, 30, 80, 5)
my_group = displayio.Group() # make a group
my_group.append(my_plane) # Add my_plane to the group

#
# Append other display elements to the group
```

(continues on next page)

(continued from previous page)

```
#

display.show(my_group) # add the group to the display
```

Summary: Cartesian Features and input variables

The *Candlestick* class has some options for controlling its position, appearance, through a collection of input variables:

- **position**: `x`
- **color**: `color_green`, `color_red`

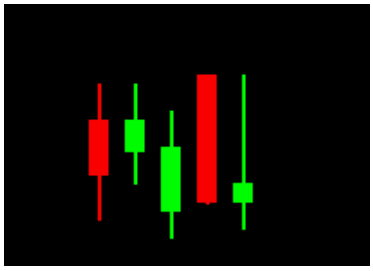


Fig. 1: Diagram showing 5 different candlesticks.

CHAPTER

SIX

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

INDEX

C

M