CircuitPython scales Library Documentation

Release 1.0

Jose David M.

May 21, 2023

CONTENTS

Allows display data in a graduated level

CHAPTER

ONE

DEPENDENCIES

This driver depends on:

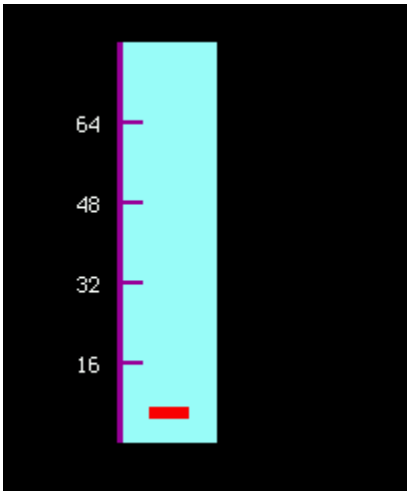
- Adafruit CircuitPython

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



Please se the `scales_simpletest.py` example for initial reference

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/scales_simpletest.py

```
import time
import board
import displayio
from scales import Scale

display = board.DISPLAY
group = displayio.Group()

values = [56, 58, 60, 65, 63, 60, 56, 54, 53, 42, 43, 44, 45, 52, 54]

my_scale = Scale(
    x=50,
    y=220,
    length=200,
    direction="vertical",
    limits=(0, 80),
    ticks=[16, 32, 48, 64, 80],
)

group.append(my_scale)

my_scale2 = Scale(
    x=150,
    y=100,
    length=200,
    direction="horizontal",
    limits=(0, 80),
    ticks=[16, 32, 48, 64, 80],
)
group.append(my_scale2)
display.show(group)


while True:
```

(continues on next page)

(continued from previous page)

```
    for val in values:
        my_scale.animate_pointer(val)
        my_scale2.animate_pointer(val)
        time.sleep(0.1)
```

5.2 Scales Library

5.2.1 scales

Allows display data in a graduated level

- Author(s): Jose David M.

5.2.1.1 Implementation Notes

Scales version in CircuitPython

class scales.**Axes**(**args: Any*, ***kwargs: Any*)

Parameters

- **x** (*int*) – pixel position. Defaults to `0`
- **y** (*int*) – pixel position. Defaults to `0`
- **limits** (*int,int*) – tuple of value range for the scale. Defaults to (0, 100)
- **ticks** (*list*) – list to ticks to display. If this is not enter a equally spaced scale will be created between the given limits.
- **direction** (*str*) – direction of the scale either `horizontal` or `vertical` defaults to `horizontal`
- **stroke** (*int*) – width in pixels of the scale axes. Defaults to `3`
- **length** (*int*) – scale length in pixels. Defaults to `100`
- **color** (*int*) – 24-bit hex value axes line color, Defaults to Purple `0x990099`

class scales.**Scale**(**args: Any*, ***kwargs: Any*)

Parameters

- **x** (*int*) – pixel position. Defaults to `0`
- **y** (*int*) – pixel position. Defaults to `0`
- **direction** (*str*) – direction of the scale either `horizontal` or `vertical` defaults to `horizontal`
- **stroke** (*int*) – width in pixels of the axes line. Defaults to `3` pixels
- **length** (*int*) – scale length in pixels. Defaults to `100` pixels
- **color** (*int*) – 24-bit hex value axes line color, Defaults to Purple `0x990099`
- **width** (*int*) – scale width in pixels. Defaults to `50` pixels
- **limits** – tuple of value range for the scale. Defaults to `(0, 100)`

- **ticks** (*list*) – list to ticks to display. If this is not enter a equally spaced scale will be created between the given limits.
- **back_color** (*int*) – 24-bit hex value axes line color. Defaults to Light Blue `0x9FFFFF`
- **tick_length** (*int*) – Scale tick length in pixels. Defaults to `10`
- **tick_stroke** (*int*) – Scale tick width in pixels. Defaults to `4`
- **pointer_length** (*int*) – length in pixels for the point. Defaults to `20` pixels
- **pointer_stroke** (*int*) – pointer thickness in pixels. Defaults to 6 pixels

Quickstart: Importing and using Scales

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

```
from scale import Scale
```

Now you can create a vertical Scale at pixel position x=50, y=180 and a range of 0 to 80 using:

```
my_scale = Scale(x=50, y=180, direction="vertical", limits=(0, 80))
```

Once you setup your display, you can now add `my_scale` to your display using:

```
display.show(my_scale)
```

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

```
my_scale= Scale(x=20, y=30)
my_group = displayio.Group() # make a group
my_group.append(my_scale) # Add my_slider to the group

#
# Append other display elements to the group
#

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

Summary: Slider Features and input variables

The *Scale* class has some options for controlling its position, visible appearance, and value through a collection of input variables:

- **position**: `x`, `y`
- **size**: `length` and `width`
- **color**: `color`, `back_color`
- **linewidths**: `stroke` and `tick_stroke`
- **range**: `limits`

animate_pointer(*new_value*)

Public function to animate the pointer

Parameters
new_value – value to draw the pointer

Returns
None

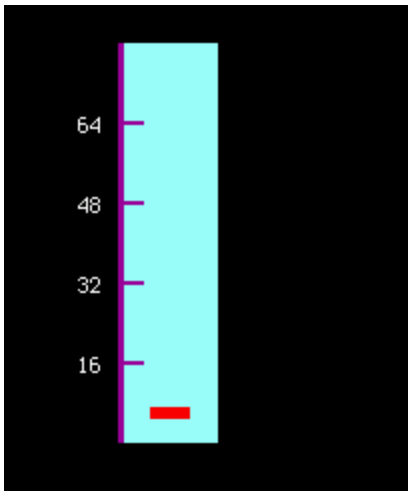


Fig. 1: Diagram showing a simple scale.

scales.**rectangle_draw**(*x0: int*, *y0: int*, *height: int*, *width: int*, *palette*)

rectangle_draw function

Draws a rectangle using or `vectorio.Rectangle`

Parameters

- **x0** (*int*) – rectangle lower corner x position
- **y0** (*int*) – rectangle lower corner y position
- **width** (*int*) – rectangle upper corner x position
- **height** (*int*) – rectangle upper corner y position
- **palette** (*Palette*) – palette object to be used to draw the rectangle

scales.**transform**(*oldrangemin: Union[float, int]*, *oldrangemax: Union[float, int]*, *newrangemin: Union[float, int]*, *newrangemax: Union[float, int]*, *value: Union[float, int]*) → Union[float, int]

This function converts the original value into a new defined value in the new range

Parameters

- **oldrangemin** (*int* / *float*) – minimum of the original range
- **oldrangemax** (*int* / *float*) – maximum of the original range
- **newrangemin** (*int* / *float*) – minimum of the new range
- **newrangemax** (*int* / *float*) – maximum of the new range
- **value** (*int* / *float*) – value to be converted

Return int|float

converted value

PYTHON MODULE INDEX

S

INDEX