

---

**AdafruitDisplay***TextLibraryDocumentation*  
**Release 1.0**

**Scott Shawcroft**

**Aug 21, 2021**



---

# Contents

---

<b>1</b>	<b>Dependencies</b>	<b>3</b>
<b>2</b>	<b>Usage Example</b>	<b>5</b>
<b>3</b>	<b>Contributing</b>	<b>7</b>
<b>4</b>	<b>Documentation</b>	<b>9</b>
<b>5</b>	<b>Table of Contents</b>	<b>11</b>
5.1	Simple test . . . . .	11
5.2	Bitmap_label Simple test . . . . .	11
5.3	Label vs Bitmap_label Comparison . . . . .	12
5.4	Background color example . . . . .	16
5.5	Text padding example . . . . .	17
5.6	Anchored Position . . . . .	19
5.7	Textarea Boundingbox . . . . .	21
5.8	Align Baseline example . . . . .	22
5.9	Magtag example . . . . .	24
5.10	MatrixPortal example . . . . .	25
5.11	PyPortal example . . . . .	26
5.12	Wraptest example . . . . .	27
5.13	Wrap Pixel Test . . . . .	28
5.14	Library Features Example . . . . .	29
5.15	adafruit_display_text . . . . .	41
5.16	adafruit_display_text.label . . . . .	43
5.16.1	Implementation Notes . . . . .	43
5.17	adafruit_display_text.bitmap_label . . . . .	44
5.17.1	Implementation Notes . . . . .	44
<b>6</b>	<b>Indices and tables</b>	<b>47</b>
	<b>Python Module Index</b>	<b>49</b>
	<b>Index</b>	<b>51</b>



Displays text using CircuitPython's displayio.



# CHAPTER 1

---

## 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](#).





## CHAPTER 2

---

### Usage Example

---

For a board with a built-in display.

```
import board
import terminalio
from adafruit_display_text import label

text = "Hello world"
text_area = label.Label(terminalio.FONT, text=text)
text_area.x = 10
text_area.y = 10
board.DISPLAY.show(text_area)
while True:
    pass
```



## CHAPTER 3

---

### Contributing

---

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



## CHAPTER 4

---

### Documentation

---

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



## 5.1 Simple test

Ensure your device works with this simple test.

Listing 1: examples/display\_text\_simpletest.py

```
1 # SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
2 # SPDX-License-Identifier: MIT
3
4 import board
5 import terminalio
6 from adafruit_display_text import label
7
8
9 text = "Hello world"
10 text_area = label.Label(terminalio.FONT, text=text)
11 text_area.x = 10
12 text_area.y = 10
13 board.DISPLAY.show(text_area)
14 while True:
15     pass
```

## 5.2 Bitmap\_label Simple test

Simple test using bitmap\_label to display text

Listing 2: examples/display\_text\_bitmap\_label\_simpletest.py

```
1 # SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
2 # SPDX-License-Identifier: MIT
3
```

(continues on next page)

(continued from previous page)

```

4 import board
5 import terminalio
6 from adafruit_display_text import bitmap_label
7
8
9 text = "Hello world"
10 text_area = bitmap_label.Label(terminalio.FONT, text=text)
11 text_area.x = 10
12 text_area.y = 10
13 board.DISPLAY.show(text_area)
14 while True:
15     pass

```

## 5.3 Label vs Bitmap\_label Comparison

Example to compare Label and Bitmap\_Label characteristics

Listing 3: examples/display\_text\_label\_vs\_bitmap\_label\_comparison.py

```

1 # SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
2 # SPDX-License-Identifier: MIT
3
4 # Sample for comparing label and bitmap_label positioning with Builtin or loaded BDF_
  ↳ fonts
5
6 # pylint: disable=no-member
7
8 import gc
9 import board
10 import displayio
11 import terminalio
12 from adafruit_bitmap_font import bitmap_font
13
14 from adafruit_display_text import bitmap_label
15 from adafruit_display_text import label
16
17 # pylint: disable=no-member
18
19
20 #####
21 # Use this Boolean variables to select which font style to use
22 #####
23 use_builtinfont = False # Set True to use the terminalio.FONT BuiltinFont,
24 fontToUse = terminalio.FONT
25 # Set False to use a BDF loaded font, see "fontFiles" below
26 #####
27
28 if not use_builtinfont:
29     # load the fonts
30     print("loading font...")
31
32     fontList = []
33
34     # Load some proportional fonts

```

(continues on next page)



(continued from previous page)

```

35     fontFile = "fonts/LeagueSpartan-Bold-16.bdf"
36     fontToUse = bitmap_font.load_font(fontFile)
37
38     # Set scaling factor for display text
39     my_scale = 1
40
41     # Setup the SPI display
42     if "DISPLAY" in dir(board):
43         # use built in display (PyPortal, PyGamer, PyBadge, CLUE, etc.)
44         # see guide for setting up external displays (TFT / OLED breakouts, RGB matrices, ↵
45         ↵etc.)
46         # https://learn.adafruit.com/circuitpython-display-support-using-displayio/
47         ↵display-and-display-bus
48         display = board.DISPLAY
49
50     else:
51         # Setup the LCD display with driver
52         # You may need to change this to match the display driver for the chipset
53         # used on your display
54         from adafruit_ili9341 import ILI9341
55
56         displayio.release_displays()
57
58         # setup the SPI bus
59         spi = board.SPI()
60         tft_cs = board.D9 # arbitrary, pin not used
61         tft_dc = board.D10
62         tft_backlight = board.D12
63         tft_reset = board.D11
64
65         while not spi.try_lock():
66             spi.configure(baudrate=32000000)
67             spi.unlock()
68
69         display_bus = displayio.FourWire(
70             spi,
71             command=tft_dc,
72             chip_select=tft_cs,
73             reset=tft_reset,
74             baudrate=32000000,
75             polarity=1,
76             phase=1,
77         )
78
79         # Number of pixels in the display
80         DISPLAY_WIDTH = 320
81         DISPLAY_HEIGHT = 240
82
83         # create the display
84         display = ILI9341(
85             display_bus,
86             width=DISPLAY_WIDTH,
87             height=DISPLAY_HEIGHT,
88             rotation=180, # The rotation can be adjusted to match your configuration.
89             auto_refresh=True,
90             native_frames_per_second=90,
91         )

```

(continues on next page)

(continued from previous page)

```

90
91     # reset the display to show nothing.
92     display.show(None)
93
94 print("Display is started")
95
96 preload_glyphs = (
97     True # set this to True if you want to preload the font glyphs into memory
98 )
99 # preloading the glyphs will help speed up the rendering of text but will use more RAM
100
101 if preload_glyphs and not use_builtinfont:
102
103     # identify the glyphs to load into memory -> increases rendering speed
104     glyphs = (
105         b"0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ/!@_.,:?!'\n "
106     )
107
108     print("loading glyphs...")
109     fontToUse.load_glyphs(glyphs)
110
111     print("Glyphs are loaded.")
112
113 print("Fonts completed loading.")
114
115 # create group
116
117 long_string = "The purple snake\nbrings python fun\nnto everyone."
118 label2_padding = 10
119
120 #####
121 # Create the "bitmap_label.py" versions of the text labels.
122
123 gc.collect()
124 bitmap_label_start = gc.mem_free()
125
126 bmap_label1 = bitmap_label.Label(
127     font=fontToUse,
128     text="bitmap_label",
129     color=0xFFFFFF,
130     background_color=0xFF0000,
131     padding_bottom=0,
132     padding_left=0,
133     padding_right=0,
134     padding_top=0,
135     background_tight=True,
136     line_spacing=1.25,
137     scale=my_scale,
138     anchor_point=(0.0, 0),
139     anchored_position=(10, 60),
140 )
141
142 bmap_label2 = bitmap_label.Label(
143     font=fontToUse,
144     text=long_string,
145     color=0x000000,
146     background_color=0xFFFF00,

```

(continues on next page)

(continued from previous page)

```

147     padding_bottom=label2_padding,
148     padding_left=0,
149     padding_right=0,
150     padding_top=label2_padding,
151     background_tight=False,
152     line_spacing=1.25,
153     scale=my_scale,
154     anchor_point=(0.0, 0),
155     anchored_position=(10, 120),
156 )
157
158 gc.collect()
159 bitmap_label_end = gc.mem_free()
160
161 print("bitmap_label used: {} memory".format(bitmap_label_start - bitmap_label_end))
162
163 bmap_group = displayio.Group() # Create a group for displaying
164 bmap_group.append(bmap_label1)
165 bmap_group.append(bmap_label2)
166
167
168 #####
169 # Create the "label.py" versions of the text labels.
170
171 gc.collect()
172 label_start = gc.mem_free()
173
174 label1 = label.Label(
175     font=fontToUse,
176     text="label",
177     color=0xFFFFFF,
178     background_color=0xFF0000,
179     padding_bottom=0,
180     padding_left=0,
181     padding_right=0,
182     padding_top=0,
183     background_tight=True,
184     line_spacing=1.25,
185     scale=my_scale,
186     anchor_point=(1.0, 0),
187     anchored_position=(display.width - 10, 60),
188 )
189
190 label2 = label.Label(
191     font=fontToUse,
192     text=long_string,
193     color=0x000000,
194     background_color=0xFFFF00,
195     padding_bottom=label2_padding,
196     padding_left=0,
197     padding_right=0,
198     padding_top=label2_padding,
199     background_tight=False,
200     line_spacing=1.25,
201     scale=my_scale,
202     anchor_point=(1.0, 0),
203     anchored_position=(display.width - 10, 120),

```

(continues on next page)

(continued from previous page)

```

204 )
205
206 gc.collect()
207 label_end = gc.mem_free()
208
209 print("label used: {} memory".format(label_start - label_end))
210 label_group = displayio.Group() # Create a group for displaying
211 label_group.append(label1)
212 label_group.append(label2)
213
214
215 print("***")
216
217 main_group = displayio.Group()
218 main_group.append(label_group)
219 main_group.append(bmap_group)
220
221 display.auto_refresh = True
222
223 display.show(main_group)
224 while True:
225     pass

```

## 5.4 Background color example

Show the text backgrounds features

Listing 4: examples/display\_text\_background\_color.py

```

1 # SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
2 # SPDX-License-Identifier: MIT
3
4 """
5 This example shows the use color and background_color
6 """
7 import time
8 import board
9 import terminalio
10 from adafruit_display_text import label
11
12 text = " Color Background Hello world"
13 text_area = label.Label(
14     terminalio.FONT, text=text, color=0x0000FF, background_color=0xFFAA00
15 )
16 text_area.x = 10
17 text_area.y = 10
18
19 print("background color is {:06x}".format(text_area.background_color))
20
21 board.DISPLAY.show(text_area)
22
23 time.sleep(2)
24 text_area.background_color = 0xFF0000
25 print("background color is {:06x}".format(text_area.background_color))

```

(continues on next page)

(continued from previous page)

```

26 time.sleep(2)
27 text_area.background_color = None
28 print("background color is {}".format(text_area.background_color))
29 while True:
30     pass

```

## 5.5 Text padding example

Show the text padding features in all directions

Listing 5: examples/display\_text\_background\_color.py

```

1  # SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
2  # SPDX-License-Identifier: MIT
3
4  """
5  This example shows the use color and background_color
6  """
7  import time
8  import board
9  import displayio
10
11 from adafruit_bitmap_font import bitmap_font
12 from adafruit_display_text import label
13
14
15 # Setup the SPI display
16 if "DISPLAY" in dir(board):
17     # use built in display (PyPortal, PyGamer, PyBadge, CLUE, etc.)
18     # see guide for setting up external displays (TFT / OLED breakouts, RGB matrices,
19     ↪ etc.)
20     # https://learn.adafruit.com/circuitpython-display-support-using-displayio/
21     ↪ display-and-display-bus
22     display = board.DISPLAY
23
24 else:
25     print("Starting external display") # goes to serial only
26     # Setup the LCD display with driver
27     # You may need to change this to match the display driver for the chipset
28     # used on your display
29     from adafruit_ili9341 import ILI9341
30
31     # from adafruit_st7789 import ST7789
32
33     displayio.release_displays()
34
35     # setup the SPI bus
36     spi = board.SPI()
37     tft_cs = board.D9 # arbitrary, pin not used
38     tft_dc = board.D10
39     tft_backlight = board.D12
40     tft_reset = board.D11
41
42     while not spi.try_lock():

```

(continues on next page)

(continued from previous page)

```

41     spi.configure(baudrate=32000000)
42     spi.unlock()
43
44     display_bus = displayio.FourWire(
45         spi,
46         command=tft_dc,
47         chip_select=tft_cs,
48         reset=tft_reset,
49         baudrate=32000000,
50         polarity=1,
51         phase=1,
52     )
53
54     # Number of pixels in the display
55     DISPLAY_WIDTH = 320
56     DISPLAY_HEIGHT = 240
57
58     # display = ST7789(display_bus, width=240, height=240, rotation=0, rowstart=80,
59     ↪colstart=0)
60
61     # create the display
62     display = ILI9341(
63         display_bus,
64         width=DISPLAY_WIDTH,
65         height=DISPLAY_HEIGHT,
66         rotation=180, # The rotation can be adjusted to match your configuration.
67         auto_refresh=True,
68         native_frames_per_second=90,
69     )
70     display.show(None)
71
72     # font=terminalio.FONT # this is the Builtin fixed dimension font
73
74     font = bitmap_font.load_font("fonts/LeagueSpartan-Bold-16.bdf")
75
76     text = []
77     text.append("none") # no ascenders or descenders
78     text.append("pop quops") # only descenders
79     text.append("MONSTERS are tall") # only ascenders
80     text.append("MONSTERS ate pop quops") # both ascenders and descenders
81     text.append("MONSTER quops\nnewline quops") # with newline
82
83     display.auto_refresh = True
84     myGroup = displayio.Group()
85     display.show(myGroup)
86
87     text_area = []
88     myPadding = 4
89
90     for i, thisText in enumerate(text):
91         text_area.append(
92             label.Label(
93                 font,
94                 text=thisText,
95                 color=0xFFFFFF,
96                 background_color=None,

```

(continues on next page)

(continued from previous page)

```

97         background_tight=False,
98         padding_top=myPadding,
99         padding_bottom=myPadding,
100        padding_left=myPadding,
101        padding_right=myPadding,
102    )
103 )
104
105 this_x = 10
106 this_y = 10 + i * 40
107 text_area[i].x = 10
108 text_area[i].y = 3 + i * 50
109 text_area[i].anchor_point = (0, 0)
110 text_area[i].anchored_position = (this_x, this_y)
111 myGroup.append(text_area[i])
112
113 print("background color is {}".format(text_area[0].background_color))
114
115
116 while True:
117     time.sleep(2)
118     text_area[0].text = "text" # change some text in an existing text box
119     # Note: changed text must fit within existing number of characters
120     # when the Label was created
121
122     for area in text_area:
123         area.background_color = 0xFF0000
124     print("background color is {:06x}".format(text_area[0].background_color))
125     time.sleep(2)
126     for area in text_area:
127         area.background_color = 0x000088
128     print("background color is {:06x}".format(text_area[0].background_color))
129     time.sleep(2)
130     for area in text_area:
131         area.background_color = 0x00FF00
132     print("background color is {:06x}".format(text_area[0].background_color))
133     time.sleep(2)
134     for area in text_area:
135         area.background_color = 0xFF0000
136     print("background color is {:06x}".format(text_area[0].background_color))
137     time.sleep(2)
138     for area in text_area:
139         area.background_color = None
140     print("background color is {}".format(text_area[0].background_color))

```

## 5.6 Anchored Position

Anchored position use illustration

Listing 6: examples/display\_text\_anchored\_position.py

```

1 # SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
2 # SPDX-License-Identifier: MIT
3

```

(continues on next page)

(continued from previous page)

```
4 """
5 This examples shows the use of anchor_point and anchored_position.
6 """
7 import board
8 import terminalio
9 import displayio
10 from adafruit_display_text import label
11
12 DISPLAY_WIDTH = 320
13 DISPLAY_HEIGHT = 240
14 TEXT = "Hello"
15
16 text_area_top_left = label.Label(terminalio.FONT, text=TEXT)
17 text_area_top_left.anchor_point = (0.0, 0.0)
18 text_area_top_left.anchored_position = (0, 0)
19
20 text_area_top_middle = label.Label(terminalio.FONT, text=TEXT)
21 text_area_top_middle.anchor_point = (0.5, 0.0)
22 text_area_top_middle.anchored_position = (DISPLAY_WIDTH / 2, 0)
23
24 text_area_top_right = label.Label(terminalio.FONT, text=TEXT)
25 text_area_top_right.anchor_point = (1.0, 0.0)
26 text_area_top_right.anchored_position = (DISPLAY_WIDTH, 0)
27
28 text_area_middle_left = label.Label(terminalio.FONT, text=TEXT)
29 text_area_middle_left.anchor_point = (0.0, 0.5)
30 text_area_middle_left.anchored_position = (0, DISPLAY_HEIGHT / 2)
31
32 text_area_middle_middle = label.Label(terminalio.FONT, text=TEXT)
33 text_area_middle_middle.anchor_point = (0.5, 0.5)
34 text_area_middle_middle.anchored_position = (DISPLAY_WIDTH / 2, DISPLAY_HEIGHT / 2)
35
36 text_area_middle_right = label.Label(terminalio.FONT, text=TEXT)
37 text_area_middle_right.anchor_point = (1.0, 0.5)
38 text_area_middle_right.anchored_position = (DISPLAY_WIDTH, DISPLAY_HEIGHT / 2)
39
40 text_area_bottom_left = label.Label(terminalio.FONT, text=TEXT)
41 text_area_bottom_left.anchor_point = (0.0, 1.0)
42 text_area_bottom_left.anchored_position = (0, DISPLAY_HEIGHT)
43
44 text_area_bottom_middle = label.Label(terminalio.FONT, text=TEXT)
45 text_area_bottom_middle.anchor_point = (0.5, 1.0)
46 text_area_bottom_middle.anchored_position = (DISPLAY_WIDTH / 2, DISPLAY_HEIGHT)
47
48 text_area_bottom_right = label.Label(terminalio.FONT, text=TEXT)
49 text_area_bottom_right.anchor_point = (1.0, 1.0)
50 text_area_bottom_right.anchored_position = (DISPLAY_WIDTH, DISPLAY_HEIGHT)
51
52 text_group = displayio.Group()
53 text_group.append(text_area_top_middle)
54 text_group.append(text_area_top_left)
55 text_group.append(text_area_top_right)
56 text_group.append(text_area_middle_middle)
57 text_group.append(text_area_middle_left)
58 text_group.append(text_area_middle_right)
59 text_group.append(text_area_bottom_middle)
60 text_group.append(text_area_bottom_left)
```

(continues on next page)



(continued from previous page)

```

61 text_group.append(text_area_bottom_right)
62
63 board.DISPLAY.show(text_group)
64
65 while True:
66     pass

```

## 5.7 Textarea Boundingbox

Boundingbox demonstration

Listing 7: examples/display\_text\_textarea\_boundingbox.py

```

1  # SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
2  # SPDX-License-Identifier: MIT
3
4  import os
5  import board
6  import displayio
7  from adafruit_bitmap_font import bitmap_font
8  from adafruit_display_text.label import Label
9
10
11 # the current working directory (where this file is)
12 cwd = ("/" + __file__).rsplit("/", 1)[0]
13 fonts = [
14     file
15     for file in os.listdir(cwd + "/fonts/")
16     if (file.endswith(".bdf") and not file.startswith("_."))
17 ]
18 for i, filename in enumerate(fonts):
19     fonts[i] = cwd + "/fonts/" + filename
20 print(fonts)
21
22 #####
23 THE_FONT = fonts[0]
24 DISPLAY_STRING = "A multi-line-\nexample of\n font bounding!"
25 WRAP_CHARS = 40
26
27 #####
28 # Make the display context
29 splash = displayio.Group()
30 board.DISPLAY.show(splash)
31
32 # Make a background color fill
33 color_bitmap = displayio.Bitmap(320, 240, 1)
34 color_palette = displayio.Palette(1)
35 color_palette[0] = 0xFFFFFF
36 bg_sprite = displayio.TileGrid(color_bitmap, pixel_shader=color_palette, x=0, y=0)
37 splash.append(bg_sprite)
38
39 # Load the font
40 font = bitmap_font.load_font(THE_FONT)
41 font.load_glyphs(DISPLAY_STRING.encode("utf-8"))

```

(continues on next page)

(continued from previous page)

```

42
43 print(DISPLAY_STRING)
44
45 text = Label(font, text=DISPLAY_STRING)
46 text.x = 20
47 text.y = 100
48 text.color = 0x0
49
50 # Make a background color fill
51 dims = text.bounding_box
52 print(dims)
53 textbg_bitmap = displayio.Bitmap(dims[2], dims[3], 1)
54 textbg_palette = displayio.Palette(1)
55 textbg_palette[0] = 0xFF0000
56 textbg_sprite = displayio.TileGrid(
57     textbg_bitmap, pixel_shader=textbg_palette, x=text.x + dims[0], y=text.y + dims[1]
58 )
59 splash.append(textbg_sprite)
60 splash.append(text)
61 try:
62     board.DISPLAY.refresh(target_frames_per_second=60)
63 except AttributeError:
64     board.DISPLAY.refresh_soon()
65     board.DISPLAY.wait_for_frame()
66
67
68 while True:
69     pass

```

## 5.8 Align Baseline example

Demonstrate how to align different labels to a common horizontal line

Listing 8: examples/display\_text\_label\_align\_baseline\_comparison.py

```

1 # SPDX-FileCopyrightText: 2021 Jose David Montoya for Adafruit Industries
2 # SPDX-License-Identifier: MIT
3
4 """
5 This example shows the use of base_alignment parameter.
6 """
7
8 import board
9 import displayio
10 from adafruit_bitmap_font import bitmap_font
11 from adafruit_display_text import label
12
13
14 display = board.DISPLAY
15
16 # Font definition. You can choose any two fonts available in your system
17 MEDIUM_FONT = bitmap_font.load_font("LeagueSpartan-Bold-16.bdf")
18 BIG_FONT = bitmap_font.load_font("LibreBodoni2002-Bold-27.bdf")
19

```

(continues on next page)

(continued from previous page)

```
20 TEXT_RIGHT = "MG"
21 TEXT_LEFT = "32.47"
22
23 main_group = displayio.Group()
24
25 # Create labels
26 # Base Alignment parameter False
27 left_text = label.Label(
28     BIG_FONT,
29     text=TEXT_LEFT,
30     color=0x000000,
31     background_color=0x999999,
32     x=10,
33     y=50,
34     base_alignment=False,
35 )
36 main_group.append(left_text)
37
38 right_text = label.Label(
39     MEDIUM_FONT,
40     text=TEXT_RIGHT,
41     color=0x000000,
42     background_color=0x999999,
43     x=90,
44     y=50,
45     base_alignment=False,
46 )
47 main_group.append(right_text)
48
49 # Base Alignment parameter True
50 left_text_aligned = label.Label(
51     BIG_FONT,
52     text=TEXT_LEFT,
53     color=0x000000,
54     background_color=0x999999,
55     x=10,
56     y=100,
57     base_alignment=True,
58 )
59 main_group.append(left_text_aligned)
60
61 right_text_aligned = label.Label(
62     MEDIUM_FONT,
63     text=TEXT_RIGHT,
64     color=0x000000,
65     background_color=0x999999,
66     x=90,
67     y=100,
68     base_alignment=True,
69 )
70
71 main_group.append(right_text_aligned)
72 display.show(main_group)
73
74 while True:
75     pass
```

## 5.9 Magtag example

Uses the MAGTAG to display some text

Listing 9: examples/display\_text\_magtag.py

```

1  # SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
2  # SPDX-License-Identifier: MIT
3
4  """
5  Basic display_text.label example script
6  adapted for use on MagTag.
7  """
8  import time
9  import board
10 import displayio
11 import terminalio
12 from adafruit_display_text import label
13
14 # use built in display (PyPortal, PyGamer, PyBadge, CLUE, etc.)
15 # see guide for setting up external displays (TFT / OLED breakouts, RGB matrices, etc.
16 # ↪ https://learn.adafruit.com/circuitpython-display-support-using-displayio/display-
17 # ↪ and-display-bus
18 display = board.DISPLAY
19
20 # wait until we can draw
21 time.sleep(display.time_to_refresh)
22
23 # main group to hold everything
24 main_group = displayio.Group()
25
26 # white background. Scaled to save RAM
27 bg_bitmap = displayio.Bitmap(display.width // 8, display.height // 8, 1)
28 bg_palette = displayio.Palette(1)
29 bg_palette[0] = 0xFFFFFF
30 bg_sprite = displayio.TileGrid(bg_bitmap, x=0, y=0, pixel_shader=bg_palette)
31 bg_group = displayio.Group(scale=8)
32 bg_group.append(bg_sprite)
33 main_group.append(bg_group)
34
35 # first example label
36 TEXT = "Hello world"
37 text_area = label.Label(
38     terminalio.FONT,
39     text=TEXT,
40     color=0xFFFFFF,
41     background_color=0x666666,
42     padding_top=1,
43     padding_bottom=3,
44     padding_right=4,
45     padding_left=4,
46 )
47 text_area.x = 10
48 text_area.y = 14
49 main_group.append(text_area)

```

(continues on next page)

(continued from previous page)

```

50 # second example label
51 another_text = label.Label(
52     terminalio.FONT,
53     scale=2,
54     text="MagTag display_text\nexample",
55     color=0x000000,
56     background_color=0x999999,
57     padding_top=1,
58     padding_bottom=3,
59     padding_right=4,
60     padding_left=4,
61 )
62 # centered
63 another_text.anchor_point = (0.5, 0.5)
64 another_text.anchored_position = (display.width // 2, display.height // 2)
65 main_group.append(another_text)
66
67 # show the main group and refresh.
68 display.show(main_group)
69 display.refresh()
70 while True:
71     pass

```

## 5.10 MatrixPortal example

Uses the MatrixPortal to display some text

Listing 10: examples/display\_text\_matrixportal.py

```

1 # SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
2 # SPDX-License-Identifier: MIT
3
4 """
5 This example shows how to create a display_text label and show it
6 with a Matrix Portal
7
8 Requires:
9 adafruit_matrixportal - https://github.com/adafruit/Adafruit\_CircuitPython\_
10 ↪MatrixPortal
11
12 Copy it from the current libraries bundle into the lib folder on your device.
13 """
14 import terminalio
15 from adafruit_matrixportal.matrix import Matrix
16 from adafruit_display_text import label
17
18 matrix = Matrix()
19 display = matrix.display
20
21 text = "Hello\nworld"
22 text_area = label.Label(terminalio.FONT, text=text)
23 text_area.x = 1
24 text_area.y = 4
25 display.show(text_area)

```

(continues on next page)

(continued from previous page)

```

25 while True:
26     pass

```

## 5.11 PyPortal example

Uses the Pyportal to display some text

Listing 11: examples/display\_text\_pyportal.py

```

1  # SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
2  # SPDX-License-Identifier: MIT
3
4  """
5  This example show the use of the backlight as well as using labels to simulate
6  a terminal using a font on the PyPortal
7  """
8
9  import os
10 import time
11 import board
12 import displayio
13
14 from adafruit_bitmap_font import bitmap_font
15 from adafruit_display_text.label import Label
16
17 FONT_DIR = "/fonts/"
18 fonts = list(
19     filter(lambda x: x.endswith("bdf") and not x.startswith("."), os.listdir(FONT_
20     ↪DIR))
21 )
22 fonts = [bitmap_font.load_font(FONT_DIR + x) for x in fonts]
23 if len(fonts) == 0:
24     print("No fonts found in '{}'.format(FONT_DIR)")
25
26 print("fade up")
27 # Fade up the backlight
28 for b in range(100):
29     board.DISPLAY.brightness = b / 100
30     time.sleep(0.01) # default (0.01)
31
32 demos = ["CircuitPython = Code + Community", "accents - ùàèùéáçãíóí", "others - αψ"]
33
34 splash = displayio.Group()
35 board.DISPLAY.show(splash)
36 max_y = 0
37 y = 0
38 for demo_text in demos:
39     for font in fonts:
40         if y >= board.DISPLAY.height:
41             y = 0
42             while len(splash):
43                 splash.pop()
44                 print("Font load {}".format(font.name))
45             area = Label(

```

(continues on next page)

(continued from previous page)

```

45         font, text=demo_text, anchor_point=(0, 0), anchored_position=(0, y)
46     )
47     splash.append(area)
48
49     y += area.height
50
51     # Wait for the image to load.
52     try:
53         board.DISPLAY.refresh(target_frames_per_second=60)
54     except AttributeError:
55         board.DISPLAY.wait_for_frame()
56
57 # Wait for 1 minute (60 seconds)
58 time.sleep(60)
59
60 # Fade down the backlight
61 for b in range(100, -1, -1):
62     board.DISPLAY.brightness = b / 100
63     time.sleep(0.01) # default (0.01)
64
65 print("fade down")
66
67 time.sleep(10)

```

## 5.12 Wraptest example

Illustrates the wraptest feature

Listing 12: examples/display\_text\_wraptest.py

```

1  # SPDX-FileCopyrightText: 2021 ladyada for Adafruit Industries
2  # SPDX-License-Identifier: MIT
3
4  """
5  This example illustrates how to use the wrap_text_to_lines
6  helper function.
7  """
8  import board
9  import terminalio
10 from adafruit_display_text import label, wrap_text_to_lines
11
12 # use built in display (PyPortal, PyGamer, PyBadge, CLUE, etc.)
13 # see guide for setting up external displays (TFT / OLED breakouts, RGB matrices, etc.
14 ↪)
15 # https://learn.adafruit.com/circuitpython-display-support-using-displayio/display-
16 ↪and-display-bus
17 display = board.DISPLAY
18
19 text = (
20     "Lorem ipsum dolor sit amet, consectetur adipiscing elit, "
21     "sed do eiusmod tempor incididunt ut labore et dolore magna "
22     "aliqua. Ut enim ad minim veniam, quis nostrud exercitation "
23     "ullamco laboris nisi ut aliquip ex ea commodo consequat."
24 )

```

(continues on next page)

(continued from previous page)

```

23 text = "\n".join(wrap_text_to_lines(text, 28))
24 text_area = label.Label(terminalio.FONT, text=text)
25 text_area.x = 10
26 text_area.y = 10
27 display.show(text_area)
28 while True:
29     pass

```

## 5.13 Wrap Pixel Test

### Wrap Pixel Test

Listing 13: examples/display\_text\_wrap\_pixels\_test.py

```

1  # SPDX-FileCopyrightText: 2021 Tim C, written for Adafruit Industries
2  #
3  # SPDX-License-Identifier: MIT
4  """
5  Test the wrap_text_to_pixels function. Try changing WRAP_WIDTH or text
6  and observe the results. The red bar represents the full size of
7  WRAP_WIDTH.
8  """
9
10 import board
11 import displayio
12 import terminalio
13 from adafruit_display_text import label, wrap_text_to_pixels
14
15 WRAP_WIDTH = 140
16 text = (
17     "CircuitPython is a programming language designed to simplify experimenting "
18     "and learning to code on low-cost microcontroller boards. "
19 )
20
21 # use built in display (PyPortal, PyGamer, PyBadge, CLUE, etc.)
22 # see guide for setting up external displays (TFT / OLED breakouts, RGB matrices, etc.
23 # ↪ https://learn.adafruit.com/circuitpython-display-support-using-displayio/display-
24 ↪ and-display-bus
25 display = board.DISPLAY
26
27 # Make the display context
28 main_group = displayio.Group()
29 display.show(main_group)
30
31 font = terminalio.FONT
32
33 print(text)
34 print(display.width)
35
36 text_area = label.Label(
37     font,
38     text="\n".join(wrap_text_to_pixels(text, WRAP_WIDTH, font)),
39     background_color=0x0000DD,

```

(continues on next page)



(continued from previous page)

```

39 )
40
41 text_area.anchor_point = (0, 0)
42 text_area.anchored_position = (0, 0)
43
44 main_group.append(text_area)
45
46 # Create a bitmap with two colors
47 size_checker = displayio.Bitmap(WRAP_WIDTH, 10, 2)
48 # Create a two color palette
49 palette = displayio.Palette(2)
50 palette[0] = 0x0000DD
51 palette[1] = 0xDD0000
52
53 # Create a TileGrid using the Bitmap and Palette
54 tile_grid = displayio.TileGrid(size_checker, pixel_shader=palette)
55
56 tile_grid.y = text_area.bounding_box[1] + text_area.bounding_box[3] + 10
57
58 size_checker.fill(1)
59
60 main_group.append(tile_grid)
61
62 while True:
63     pass

```

## 5.14 Library Features Example

This examples shows the label and bitmap\_label capabilities and features

Listing 14: examples/display\_text\_advance\_example.py

```

1 # SPDX-FileCopyrightText: 2021 Jose David M.
2 #
3 # SPDX-License-Identifier: MIT
4 #####
5 """
6 This is an advanced demonstration of the display_text library capabilities
7 """
8
9 import time
10 import board
11 import displayio
12 import terminalio
13 import fontio
14 from adafruit_display_text import label, bitmap_label
15 from adafruit_bitmap_font import bitmap_font
16
17 display = board.DISPLAY
18 main_group = displayio.Group()
19 MEDIUM_FONT = bitmap_font.load_font("fonts/LeagueSpartan-Bold-16.bdf")
20 BIG_FONT = bitmap_font.load_font("fonts/LibreBodoni2002-Bold-27.bdf")
21 TIME_PAUSE = 2
22

```

(continues on next page)

(continued from previous page)

```

23 bitmap = displayio.Bitmap(4, display.width, 2)
24 palette = displayio.Palette(2)
25 palette[0] = 0x004400
26 palette[1] = 0x00FFFF
27 horizontal_line = displayio.TileGrid(bitmap, pixel_shader=palette, x=155, y=0)
28 main_group.append(horizontal_line)
29
30 bitmap = displayio.Bitmap(display.width, 4, 2)
31 vertical_line = displayio.TileGrid(bitmap, pixel_shader=palette, x=0, y=110)
32 main_group.append(vertical_line)
33
34 # Tests
35 text_area = label.Label(terminalio.FONT, text="Circuit Python")
36 main_group.append(text_area)
37 display.show(main_group)
38 time.sleep(TIME_PAUSE)
39
40 # Testing position setter
41 text_area.x = 10
42 text_area.y = 10
43 display.show(main_group)
44 time.sleep(TIME_PAUSE)
45
46 # Testing creating label with initial position
47 text_area.text = "Testing initiating without text"
48 try:
49     text_middle = label.Label(terminalio.FONT)
50 except SyntaxError:
51     print("Fail setting-up label without text")
52     warning_text = label.Label(
53         BIG_FONT,
54         text="Test Fail",
55         x=display.width // 2,
56         y=display.height // 4,
57         background_color=0x004499,
58     )
59     main_group.append(warning_text)
60 display.show(main_group)
61 time.sleep(TIME_PAUSE)
62
63 text_area.text = "Testing Position"
64 text_middle = label.Label(
65     terminalio.FONT, text="Circuit", x=display.width // 2, y=display.height // 2
66 )
67 main_group.append(text_middle)
68 display.show(main_group)
69 time.sleep(TIME_PAUSE)
70
71 # Testing Text Setter
72 text_area.text = "Testing Changing Text"
73 text_middle.text = "Python"
74 display.show(main_group)
75 time.sleep(TIME_PAUSE)
76
77 # Testing a and y getter and setter
78 text_area.text = "Testing Changing Position"
79 text_middle.x = text_middle.x - 50

```

(continues on next page)

(continued from previous page)

```
80 text_middle.y = text_middle.y - 50
81 display.show(main_group)
82 time.sleep(TIME_PAUSE)
83
84 # Testing font Getter and setter
85 text_area.text = "Testing Changing FONT"
86 if isinstance(text_middle.font, fontio.BuiltinFont):
87     text_middle.font = MEDIUM_FONT
88 display.show(main_group)
89 time.sleep(TIME_PAUSE)
90
91 # Once this working we create another label with all the initial specs
92 main_group.pop()
93
94 # Testing Color
95 text_area.text = "Testing Color"
96 text_initial_specs = label.Label(
97     MEDIUM_FONT,
98     text="Circuit Python",
99     x=display.width // 2,
100    y=display.height // 2,
101 )
102 main_group.append(text_initial_specs)
103 display.show(main_group)
104 time.sleep(TIME_PAUSE)
105
106 text_initial_specs.color = 0x004400
107 display.show(main_group)
108 time.sleep(TIME_PAUSE)
109 main_group.pop()
110
111 # Testing Background Color
112 text_area.text = "Testing Background Color"
113 text_initial_specs = label.Label(
114     MEDIUM_FONT,
115     text="CircuitPython",
116     x=display.width // 2,
117     y=display.height // 2,
118     color=0xFFFFFFFF,
119 )
120 main_group.append(text_initial_specs)
121 display.show(main_group)
122 time.sleep(TIME_PAUSE)
123
124 text_initial_specs.background_color = 0x990099
125 display.show(main_group)
126 time.sleep(TIME_PAUSE)
127 main_group.pop()
128
129 # Testing Background Color
130 text_area.text = "Testing Background Tight"
131 text_initial_specs = label.Label(
132     BIG_FONT,
133     text="aaaaq~",
134     x=0,
135     y=display.height // 2,
136     color=0xFFFFFFFF,
```

(continues on next page)

(continued from previous page)

```
137     background_color=0x990099,
138     background_tight=True,
139 )
140 main_group.append(text_initial_specs)
141 text_initial_specs = label.Label(
142     BIG_FONT,
143     text="aaaaq~",
144     x=90,
145     y=display.height // 2,
146     color=0xFFFFFFFF,
147     background_color=0x990099,
148     background_tight=False,
149 )
150 main_group.append(text_initial_specs)
151 display.show(main_group)
152 time.sleep(TIME_PAUSE)
153 main_group.pop()
154 main_group.pop()
155
156 # Testing Padding
157 text_area.text = "Testing Padding"
158 text_initial_specs = label.Label(
159     BIG_FONT,
160     text="CircuitPython",
161     x=display.width // 4,
162     y=display.height // 2,
163     color=0xFFFFFFFF,
164     background_color=0x990099,
165     padding_right=10,
166     padding_top=10,
167     padding_bottom=10,
168     padding_left=10,
169 )
170 main_group.append(text_initial_specs)
171 display.show(main_group)
172 time.sleep(TIME_PAUSE)
173 main_group.pop()
174
175 # Testing Anchor Point/ Anchored Position
176 text_area.text = "Testing Anchor Point/Anchored Position"
177 text_initial_specs = label.Label(
178     MEDIUM_FONT,
179     text="CircuitPython",
180     x=display.width // 2,
181     y=display.height // 2,
182     color=0xFFFFFFFF,
183     background_color=0x990099,
184     padding_right=10,
185     padding_top=10,
186     padding_bottom=10,
187     padding_left=10,
188 )
189 main_group.append(text_initial_specs)
190 display.show(main_group)
191 time.sleep(TIME_PAUSE)
192
193 try:
```

(continues on next page)

(continued from previous page)

```

194     text_initial_specs.anchored_position = (100, 100)
195     text_initial_specs.anchor_point = (0.5, 0.5)
196
197 except TypeError:
198     print("Test is failing here")
199     main_group.pop()
200     warning_text = label.Label(
201         BIG_FONT,
202         text="Test Fail",
203         x=display.width // 2,
204         y=display.height // 4,
205         background_color=0x004499,
206     )
207     main_group.append(warning_text)
208     time.sleep(TIME_PAUSE)
209     display.show(main_group)
210
211 main_group.pop()
212
213 # Testing Scale
214 text_area.text = "Testing Scale"
215 text_initial_specs = label.Label(
216     MEDIUM_FONT,
217     text="CircuitPython",
218     x=display.width // 2,
219     y=display.height // 2,
220     color=0xFFFFFF,
221     background_color=0x990099,
222     padding_right=10,
223     padding_top=10,
224     padding_bottom=10,
225     padding_left=10,
226     anchored_position=(display.width // 2, display.height // 2),
227     anchor_point=(0.5, 0.5),
228 )
229 main_group.append(text_initial_specs)
230 display.show(main_group)
231 time.sleep(TIME_PAUSE)
232
233 text_initial_specs.scale = 2
234 display.show(main_group)
235 time.sleep(TIME_PAUSE)
236 main_group.pop()
237
238 # Testing Base Alignment
239 text_area.text = "Testing Base Alignment"
240 text_initial_specs = label.Label(
241     MEDIUM_FONT,
242     text="python",
243     x=display.width // 2,
244     y=display.height // 2,
245     color=0xFFFFFF,
246     background_color=0x990099,
247     base_alignment=True,
248 )
249 main_group.append(text_initial_specs)
250 text_initial_specs = label.Label(

```

(continues on next page)

(continued from previous page)

```
251     BIG_FONT,
252     text="circuit",
253     x=display.width // 2 - 100,
254     y=display.height // 2,
255     color=0xFFFFFF,
256     background_color=0x990099,
257     base_alignment=True,
258 )
259 main_group.append(text_initial_specs)
260 display.show(main_group)
261 time.sleep(TIME_PAUSE)
262 main_group.pop()
263 main_group.pop()
264
265 # Testing Direction
266 text_area.text = "Testing Direction-UPR"
267 text_initial_specs = label.Label(
268     MEDIUM_FONT,
269     text="CircuitPython",
270     x=display.width // 2,
271     y=display.height // 2,
272     color=0xFFFFFF,
273     background_color=0x990099,
274     padding_right=10,
275     padding_top=10,
276     padding_bottom=10,
277     padding_left=10,
278     anchored_position=(display.width // 2, display.height // 2),
279     anchor_point=(0.5, 0.5),
280     label_direction="UPR",
281 )
282 main_group.append(text_initial_specs)
283 display.show(main_group)
284 time.sleep(TIME_PAUSE)
285 main_group.pop()
286
287 text_area.text = "Testing Direction-DWR"
288 text_initial_specs = label.Label(
289     MEDIUM_FONT,
290     text="CircuitPython",
291     x=display.width // 2,
292     y=display.height // 2,
293     color=0xFFFFFF,
294     background_color=0x990099,
295     padding_right=10,
296     padding_top=10,
297     padding_bottom=10,
298     padding_left=10,
299     anchored_position=(display.width // 2, display.height // 2),
300     anchor_point=(0.5, 0.5),
301     label_direction="DWR",
302 )
303 main_group.append(text_initial_specs)
304 display.show(main_group)
305 time.sleep(TIME_PAUSE)
306 main_group.pop()
307
```

(continues on next page)

(continued from previous page)

```

308 text_area.text = "Testing Direction-TTB"
309 text_initial_specs = label.Label(
310     MEDIUM_FONT,
311     text="CircuitPython",
312     x=display.width // 2,
313     y=display.height // 2,
314     color=0xFFFFFF,
315     background_color=0x990099,
316     padding_right=10,
317     padding_top=10,
318     padding_bottom=10,
319     padding_left=10,
320     anchored_position=(display.width // 2, display.height // 2),
321     anchor_point=(0.5, 0.5),
322     label_direction="TTB",
323 )
324 main_group.append(text_initial_specs)
325 display.show(main_group)
326 time.sleep(TIME_PAUSE)
327 main_group.pop()
328
329 text_area.text = "Testing Direction-RTL"
330 text_initial_specs = label.Label(
331     MEDIUM_FONT,
332     text="CircuitPython",
333     x=display.width // 2,
334     y=display.height // 2,
335     color=0xFFFFFF,
336     background_color=0x990099,
337     padding_right=10,
338     padding_top=10,
339     padding_bottom=10,
340     padding_left=10,
341     anchored_position=(display.width // 2, display.height // 2),
342     anchor_point=(0.5, 0.5),
343     label_direction="RTL",
344 )
345 main_group.append(text_initial_specs)
346 display.show(main_group)
347 time.sleep(TIME_PAUSE)
348 main_group.pop()
349
350 main_group.pop()
351
352 # Testing creating label with initial position
353 display.show(main_group)
354 time.sleep(TIME_PAUSE)
355 text_area = bitmap_label.Label(terminalio.FONT, text="Circuit Python")
356 main_group.append(text_area)
357 display.show(main_group)
358 time.sleep(TIME_PAUSE)
359 # Testing position setter
360 text_area.x = 10
361 text_area.y = 10
362 display.show(main_group)
363 time.sleep(TIME_PAUSE)
364 text_area.text = "Testing initiating without text"

```

(continues on next page)

(continued from previous page)

```
365 try:
366     text_middle = label.Label(terminalio.FONT)
367 except TypeError:
368     print("Fail setting-up label without text")
369     warning_text = label.Label(
370         BIG_FONT,
371         text="Test Fail",
372         x=display.width // 2,
373         y=display.height // 4,
374         background_color=0x004499,
375     )
376     main_group.append(warning_text)
377
378 # Testing creating label with initial position
379 text_area.text = "Testing Position"
380 text_middle = bitmap_label.Label(
381     terminalio.FONT, text="Circuit", x=display.width // 2, y=display.height // 2
382 )
383 main_group.append(text_middle)
384 display.show(main_group)
385 time.sleep(TIME_PAUSE)
386
387 # Testing Text Setter
388 text_area.text = "Testing Changing Text"
389 text_middle.text = "Python"
390 display.show(main_group)
391 time.sleep(TIME_PAUSE)
392
393 # Testing a and y getter and setter
394 text_area.text = "Testing Changing Position"
395 text_middle.x = text_middle.x - 50
396 text_middle.y = text_middle.y - 50
397 display.show(main_group)
398 time.sleep(TIME_PAUSE)
399
400 # Testing font Getter and setter
401 text_area.text = "Testing Changing FONT"
402 if isinstance(text_middle.font, fontio.BuiltinFont):
403     print("Font was BuiltinFont")
404     text_middle.font = MEDIUM_FONT
405 display.show(main_group)
406 time.sleep(TIME_PAUSE)
407
408 # Once this working we create another label with all the initial specs
409 main_group.pop()
410
411 # Testing Color
412 text_area.text = "Testing Color"
413 text_initial_specs = bitmap_label.Label(
414     MEDIUM_FONT,
415     text="Circuit Python",
416     x=display.width // 2,
417     y=display.height // 2,
418 )
419 main_group.append(text_initial_specs)
420 display.show(main_group)
421 time.sleep(TIME_PAUSE)
```

(continues on next page)



(continued from previous page)

```
422
423 text_initial_specs.color = 0x004400
424 display.show(main_group)
425 time.sleep(TIME_PAUSE)
426 main_group.pop()
427
428 # Testing Background Color
429 text_area.text = "Testing Background Color"
430 text_initial_specs = bitmap_label.Label(
431     MEDIUM_FONT,
432     text="CircuitPython",
433     x=display.width // 2,
434     y=display.height // 2,
435     color=0xFFFFFF,
436 )
437 main_group.append(text_initial_specs)
438 display.show(main_group)
439 time.sleep(TIME_PAUSE)
440
441 text_initial_specs.background_color = 0x990099
442 display.show(main_group)
443 time.sleep(TIME_PAUSE)
444 main_group.pop()
445
446 # Testing Background Color
447 text_area.text = "Testing Background Tight"
448 text_initial_specs = bitmap_label.Label(
449     BIG_FONT,
450     text="aaaaq~",
451     x=0,
452     y=display.height // 2,
453     color=0xFFFFFF,
454     background_color=0x990099,
455     background_tight=True,
456 )
457 main_group.append(text_initial_specs)
458 text_initial_specs = bitmap_label.Label(
459     BIG_FONT,
460     text="aaaaq~",
461     x=90,
462     y=display.height // 2,
463     color=0xFFFFFF,
464     background_color=0x990099,
465     background_tight=False,
466 )
467 main_group.append(text_initial_specs)
468 display.show(main_group)
469 time.sleep(TIME_PAUSE)
470 main_group.pop()
471 main_group.pop()
472
473 # Testing Padding
474 text_area.text = "Testing Padding"
475 text_initial_specs = bitmap_label.Label(
476     BIG_FONT,
477     text="CircuitPython",
478     x=display.width // 4,
```

(continues on next page)

(continued from previous page)

```
479     y=display.height // 2,
480     color=0xFFFFFF,
481     background_color=0x990099,
482     padding_right=10,
483     padding_top=10,
484     padding_bottom=10,
485     padding_left=10,
486 )
487 main_group.append(text_initial_specs)
488 display.show(main_group)
489 time.sleep(TIME_PAUSE)
490 main_group.pop()
491
492 # Testing Anchor Point/ Anchored Position
493 text_area.text = "Testing Anchor Point/Anchored Position"
494 text_initial_specs = bitmap_label.Label(
495     MEDIUM_FONT,
496     text="CircuitPython",
497     x=display.width // 2,
498     y=display.height // 2,
499     color=0xFFFFFF,
500     background_color=0x990099,
501     padding_right=10,
502     padding_top=10,
503     padding_bottom=10,
504     padding_left=10,
505 )
506 main_group.append(text_initial_specs)
507 display.show(main_group)
508 time.sleep(TIME_PAUSE)
509
510 try:
511     text_initial_specs.anchored_position = (100, 100)
512     text_initial_specs.anchor_point = (0.5, 0.5)
513
514 except TypeError:
515     print("Test is failing here")
516     main_group.pop()
517     warning_text = bitmap_label.Label(
518         BIG_FONT,
519         text="Test Fail",
520         x=display.width // 2,
521         y=display.height // 4,
522         background_color=0x004499,
523     )
524     main_group.append(warning_text)
525     time.sleep(TIME_PAUSE)
526     display.show(main_group)
527
528 main_group.pop()
529
530 # Testing Scale
531 text_area.text = "Testing Scale"
532 text_initial_specs = bitmap_label.Label(
533     MEDIUM_FONT,
534     text="CircuitPython",
535     x=display.width // 2,
```

(continues on next page)

(continued from previous page)

```

536     y=display.height // 2,
537     color=0xFFFFFFFF,
538     background_color=0x990099,
539     padding_right=10,
540     padding_top=10,
541     padding_bottom=10,
542     padding_left=10,
543     anchored_position=(display.width // 2, display.height // 2),
544     anchor_point=(0.5, 0.5),
545 )
546 main_group.append(text_initial_specs)
547 display.show(main_group)
548 time.sleep(TIME_PAUSE)
549
550 text_initial_specs.scale = 2
551 display.show(main_group)
552 time.sleep(TIME_PAUSE)
553 main_group.pop()
554
555 # Testing Base Alignment
556 text_area.text = "Testing Base Alignment"
557 text_initial_specs = bitmap_label.Label(
558     MEDIUM_FONT,
559     text="python",
560     x=display.width // 2,
561     y=display.height // 2,
562     color=0xFFFFFFFF,
563     background_color=0x990099,
564     base_alignment=True,
565 )
566 main_group.append(text_initial_specs)
567 text_initial_specs = bitmap_label.Label(
568     BIG_FONT,
569     text="circuit",
570     x=display.width // 2 - 100,
571     y=display.height // 2,
572     color=0xFFFFFFFF,
573     background_color=0x990099,
574     base_alignment=True,
575 )
576 main_group.append(text_initial_specs)
577 display.show(main_group)
578 time.sleep(TIME_PAUSE)
579 main_group.pop()
580 main_group.pop()
581
582 # Testing Direction
583 text_area.text = "Testing Direction-UPR"
584 text_initial_specs = bitmap_label.Label(
585     MEDIUM_FONT,
586     text="CircuitPython",
587     x=display.width // 2,
588     y=display.height // 2,
589     color=0xFFFFFFFF,
590     background_color=0x990099,
591     padding_right=10,
592     padding_top=10,

```

(continues on next page)

(continued from previous page)

```
593     padding_bottom=10,
594     padding_left=10,
595     anchored_position=(display.width // 2, display.height // 2),
596     anchor_point=(0.5, 0.5),
597     label_direction="UPR",
598 )
599 main_group.append(text_initial_specs)
600 display.show(main_group)
601 time.sleep(TIME_PAUSE)
602 main_group.pop()
603
604 text_area.text = "Testing Direction-DWR"
605 text_initial_specs = bitmap_label.Label(
606     MEDIUM_FONT,
607     text="CircuitPython",
608     x=display.width // 2,
609     y=display.height // 2,
610     color=0xFFFFFF,
611     background_color=0x990099,
612     padding_right=10,
613     padding_top=10,
614     padding_bottom=10,
615     padding_left=10,
616     anchored_position=(display.width // 2, display.height // 2),
617     anchor_point=(0.5, 0.5),
618     label_direction="DWR",
619 )
620 main_group.append(text_initial_specs)
621 display.show(main_group)
622 time.sleep(TIME_PAUSE)
623 main_group.pop()
624
625 text_area.text = "Testing Direction-UPD"
626 text_initial_specs = bitmap_label.Label(
627     MEDIUM_FONT,
628     text="CircuitPython",
629     x=display.width // 2,
630     y=display.height // 2,
631     color=0xFFFFFF,
632     background_color=0x990099,
633     padding_right=10,
634     padding_top=10,
635     padding_bottom=10,
636     padding_left=10,
637     anchored_position=(display.width // 2, display.height // 2),
638     anchor_point=(0.5, 0.5),
639     label_direction="UPD",
640 )
641 main_group.append(text_initial_specs)
642 display.show(main_group)
643 time.sleep(TIME_PAUSE)
644 main_group.pop()
645
646 text_area.text = "Testing Direction-RTL"
647 text_initial_specs = bitmap_label.Label(
648     MEDIUM_FONT,
649     text="CircuitPython",
```

(continues on next page)

(continued from previous page)

```

650     x=display.width // 2,
651     y=display.height // 2,
652     color=0xFFFFFF,
653     background_color=0x990099,
654     padding_right=10,
655     padding_top=10,
656     padding_bottom=10,
657     padding_left=10,
658     anchored_position=(display.width // 2, display.height // 2),
659     anchor_point=(0.5, 0.5),
660     label_direction="RTL",
661 )
662 main_group.append(text_initial_specs)
663 display.show(main_group)
664 time.sleep(TIME_PAUSE)
665 main_group.pop()
666
667 text_area.text = "Finished"
668 print("Tests finished")

```

## 5.15 adafruit\_display\_text

**class** `adafruit_display_text.LabelBase` (*font*, *x*: *int* = 0, *y*: *int* = 0, *text*: *str* = "", *color*: *int* = 16777215, *background\_color*: *int* = None, *line\_spacing*: *float* = 1.25, *background\_tight*: *bool* = False, *padding\_top*: *int* = 0, *padding\_bottom*: *int* = 0, *padding\_left*: *int* = 0, *padding\_right*: *int* = 0, *anchor\_point*: *Tuple*[*float*, *float*] = None, *anchored\_position*: *Tuple*[*int*, *int*] = None, *scale*: *int* = 1, *base\_alignment*: *bool* = False, *tab\_replacement*: *Tuple*[*int*, *str*] = (4, ' '), *label\_direction*: *str* = 'LTR', *\*\*kwargs*)

Superclass that all other types of labels will extend. This contains all of the properties and functions that work the same way in all labels.

**Note:** This should be treated as an abstract base class.

Subclasses should implement `_set_text`, `_set_font`, and `_set_line_spacing` to have the correct behavior for that type of label.

### Parameters

- **font** (*Font*) – A font class that has `get_bounding_box` and `get_glyph`. Must include a capital M for measuring character size.
- **text** (*str*) – Text to display
- **color** (*int*) – Color of all text in RGB hex
- **background\_color** (*int*) – Color of the background, use `None` for transparent
- **line\_spacing** (*float*) – Line spacing of text to display
- **background\_tight** (*bool*) – Set `True` only if you want background box to tightly surround text. When set to `True` Padding parameters will be ignored.
- **padding\_top** (*int*) – Additional pixels added to background bounding box at top

- **padding\_bottom** (*int*) – Additional pixels added to background bounding box at bottom
- **padding\_left** (*int*) – Additional pixels added to background bounding box at left
- **padding\_right** (*int*) – Additional pixels added to background bounding box at right
- **anchor\_point** (*(float, float)*) – Point that anchored\_position moves relative to. Tuple with decimal percentage of width and height. (E.g. (0,0) is top left, (1.0, 0.5): is middle right.)
- **anchored\_position** (*(int, int)*) – Position relative to the anchor\_point. Tuple containing x,y pixel coordinates.
- **scale** (*int*) – Integer value of the pixel scaling
- **base\_alignment** (*bool*) – when True allows to align text label to the baseline. This is helpful when two or more labels need to be aligned to the same baseline
- **tab\_replacement** (*(int, str)*) – tuple with tab character replace information. When (4, " ") will indicate a tab replacement of 4 spaces, defaults to 4 spaces by tab character
- **label\_direction** (*str*) – string defining the label text orientation. See the subclass documentation for the possible values.

**anchor\_point**

Point that anchored\_position moves relative to. Tuple with decimal percentage of width and height. (E.g. (0,0) is top left, (1.0, 0.5): is middle right.)

**anchored\_position**

Position relative to the anchor\_point. Tuple containing x,y pixel coordinates.

**background\_color**

Color of the background as an RGB hex number.

**bounding\_box**

An (x, y, w, h) tuple that completely covers all glyphs. The first two numbers are offset from the x, y origin of this group

**color**

Color of the text as an RGB hex number.

**font**

Font to use for text display.

**height**

The height of the label determined from the bounding box.

**label\_direction**

Set the text direction of the label

**line\_spacing**

The amount of space between lines of text, in multiples of the font's bounding-box height. (E.g. 1.0 is the bounding-box height)

**scale**

Set the scaling of the label, in integer values

**text**

Text to be displayed.

**width**

The width of the label determined from the bounding box.

`adafruit_display_text.wrap_text_to_lines` (*string: str, max\_chars: int*) → List[str]  
`wrap_text_to_lines` function A helper that will return a list of lines with word-break wrapping

#### Parameters

- **string** (*str*) – The text to be wrapped
- **max\_chars** (*int*) – The maximum number of characters on a line before wrapping

**Returns** A list of lines where each line is separated based on the amount of `max_chars` provided

**Return type** List[str]

`adafruit_display_text.wrap_text_to_pixels` (*string: str, max\_width: int, font=None, indent0: str = "", indent1: str = ""*) → List[str]

`wrap_text_to_pixels` function A helper that will return a list of lines with word-break wrapping. Leading and trailing whitespace in your string will be removed. If you wish to use leading whitespace see `indent0` and `indent1` parameters.

#### Parameters

- **string** (*str*) – The text to be wrapped.
- **max\_width** (*int*) – The maximum number of pixels on a line before wrapping.
- **font** (*Font*) – The font to use for measuring the text.
- **indent0** (*str*) – Additional character(s) to add to the first line.
- **indent1** (*str*) – Additional character(s) to add to all other lines.

**Returns** A list of the lines resulting from wrapping the input text at `max_width` pixels size

**Return type** List[str]

## 5.16 adafruit\_display\_text.label

Displays text labels using CircuitPython's `displayio`.

- Author(s): Scott Shawcroft

### 5.16.1 Implementation Notes

#### Hardware:

#### Software and Dependencies:

- Adafruit CircuitPython firmware for the supported boards: <https://circuitpython.org/downloads>

**class** `adafruit_display_text.label.Label` (*font, \*\*kwargs*)

A label displaying a string of text. The origin point set by `x` and `y` properties will be the left edge of the bounding box, and in the center of a M glyph (if its one line), or the  $(\text{number of lines} * \text{linespacing} + M)/2$ . That is, it will try to have it be center-left as close as possible.

#### Parameters

- **font** (*Font*) – A font class that has `get_bounding_box` and `get_glyph`. Must include a capital M for measuring character size.
- **text** (*str*) – Text to display
- **color** (*int*) – Color of all text in RGB hex

- **background\_color** (*int*) – Color of the background, use `None` for transparent
- **line\_spacing** (*float*) – Line spacing of text to display
- **background\_tight** (*bool*) – Set `True` only if you want background box to tightly surround text. When set to ‘True’ Padding parameters will be ignored.
- **padding\_top** (*int*) – Additional pixels added to background bounding box at top. This parameter could be negative indicating additional pixels subtracted from the background bounding box.
- **padding\_bottom** (*int*) – Additional pixels added to background bounding box at bottom. This parameter could be negative indicating additional pixels subtracted from the background bounding box.
- **padding\_left** (*int*) – Additional pixels added to background bounding box at left. This parameter could be negative indicating additional pixels subtracted from the background bounding box.
- **padding\_right** (*int*) – Additional pixels added to background bounding box at right. This parameter could be negative indicating additional pixels subtracted from the background bounding box.
- **anchor\_point** (*(float, float)*) – Point that `anchored_position` moves relative to. Tuple with decimal percentage of width and height. (E.g. (0,0) is top left, (1.0, 0.5): is middle right.)
- **anchored\_position** (*(int, int)*) – Position relative to the `anchor_point`. Tuple containing x,y pixel coordinates.
- **scale** (*int*) – Integer value of the pixel scaling
- **base\_alignment** (*bool*) – when `True` allows to align text label to the baseline. This is helpful when two or more labels need to be aligned to the same baseline
- **tab\_replacement** (*(int, str)*) – tuple with tab character replace information. When (4, “”) will indicate a tab replacement of 4 spaces, defaults to 4 spaces by tab character
- **label\_direction** (*str*) – string defining the label text orientation. There are 5 configurations possible LTR-Left-To-Right RTL-Right-To-Left TTB-Top-To-Bottom UPR-Upwards DWR-Downwards. It defaults to LTR

## 5.17 adafruit\_display\_text.bitmap\_label

Text graphics handling for CircuitPython, including text boxes

- Author(s): Kevin Matocha

### 5.17.1 Implementation Notes

**Hardware:**

**Software and Dependencies:**

- Adafruit CircuitPython firmware for the supported boards: <https://circuitpython.org/downloads>

**class** `adafruit_display_text.bitmap_label.Label` (*font, save\_text=True, \*\*kwargs*)

A label displaying a string of text that is stored in a bitmap. Note: This `bitmap_label.py` library utilizes a `Bitmap` to display the text. This method is memory-conserving relative to `label.py`.



For further reduction in memory usage, set `save_text=False` (text string will not be stored and `line_spacing` and `font` are immutable with `save_text` set to `False`).

The origin point set by `x` and `y` properties will be the left edge of the bounding box, and in the center of a M glyph (if its one line), or the  $(\text{number of lines} * \text{linespacing} + M)/2$ . That is, it will try to have it be center-left as close as possible.

### Parameters

- **font** (*Font*) – A font class that has `get_bounding_box` and `get_glyph`. Must include a capital M for measuring character size.
- **text** (*str*) – Text to display
- **color** (*int*) – Color of all text in RGB hex
- **background\_color** (*int*) – Color of the background, use `None` for transparent
- **line\_spacing** (*float*) – Line spacing of text to display
- **background\_tight** (*bool*) – Set `True` only if you want background box to tightly surround text. When set to ‘True’ Padding parameters will be ignored.
- **padding\_top** (*int*) – Additional pixels added to background bounding box at top
- **padding\_bottom** (*int*) – Additional pixels added to background bounding box at bottom
- **padding\_left** (*int*) – Additional pixels added to background bounding box at left
- **padding\_right** (*int*) – Additional pixels added to background bounding box at right
- **anchor\_point** (*(float, float)*) – Point that `anchored_position` moves relative to. Tuple with decimal percentage of width and height. (E.g. (0,0) is top left, (1.0, 0.5): is middle right.)
- **anchored\_position** (*(int, int)*) – Position relative to the `anchor_point`. Tuple containing x,y pixel coordinates.
- **scale** (*int*) – Integer value of the pixel scaling
- **save\_text** (*bool*) – Set `True` to save the text string as a constant in the label structure. Set `False` to reduce memory use.
- **base\_alignment** (*bool*) – when `True` allows to align text label to the baseline. This is helpful when two or more labels need to be aligned to the same baseline
- **tab\_replacement** (*(int, str)*) – tuple with tab character replace information. When (4, " ") will indicate a tab replacement of 4 spaces, defaults to 4 spaces by tab character
- **label\_direction** (*str*) – string defining the label text orientation. There are 5 configurations possible LTR-Left-To-Right RTL-Right-To-Left UPD-Upside Down UPR-Upwards DWR-Downwards. It defaults to LTR

### bitmap

The Bitmap object that the text and background are drawn into.

**Return type** `displayio.Bitmap`



## CHAPTER 6

---

### Indices and tables

---

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



**a**

adafruit\_display\_text, 41  
adafruit\_display\_text.bitmap\_label, 44  
adafruit\_display\_text.label, 43



## A

adafruit\_display\_text (*module*), 41  
adafruit\_display\_text.bitmap\_label (*module*), 44  
adafruit\_display\_text.label (*module*), 43  
anchor\_point (*adafruit\_display\_text.LabelBase attribute*), 42  
anchored\_position (*adafruit\_display\_text.LabelBase attribute*), 42

## B

background\_color (*adafruit\_display\_text.LabelBase attribute*), 42  
bitmap (*adafruit\_display\_text.bitmap\_label.Label attribute*), 45  
bounding\_box (*adafruit\_display\_text.LabelBase attribute*), 42

## C

color (*adafruit\_display\_text.LabelBase attribute*), 42

## F

font (*adafruit\_display\_text.LabelBase attribute*), 42

## H

height (*adafruit\_display\_text.LabelBase attribute*), 42

## L

Label (*class in adafruit\_display\_text.bitmap\_label*), 44  
Label (*class in adafruit\_display\_text.label*), 43  
label\_direction (*adafruit\_display\_text.LabelBase attribute*), 42  
LabelBase (*class in adafruit\_display\_text*), 41  
line\_spacing (*adafruit\_display\_text.LabelBase attribute*), 42

## S

scale (*adafruit\_display\_text.LabelBase attribute*), 42

## T

text (*adafruit\_display\_text.LabelBase attribute*), 42

## W

width (*adafruit\_display\_text.LabelBase attribute*), 42  
wrap\_text\_to\_lines() (*in module adafruit\_display\_text*), 42  
wrap\_text\_to\_pixels() (*in module adafruit\_display\_text*), 43