# AdafruitDisplay$_{T}extLibrary Documentation$

***Release 1.0***

**Scott Shawcroft**

**Aug 21, 2021**

# Contents

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.

CHAPTER 5

# Table of Contents

## 5.1 Simple test

Ensure your device works with this simple test.

Listing 1: examples/display_text_simpletest.py

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

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
```

## 5.2 Bitmap_label Simple test

Simple test using bitmap_label to display text

Listing 2: examples/display_text_bitmap_label_simpletest.py

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

```

(continues on next page)

(continued from previous page)

```
import board
import terminalio
from adafruit_display_text import bitmap_label


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

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

# Sample for comparing label and bitmap_label positioning with Builtin or loaded BDF␣
↪fonts

# pylint: disable=no-member

import gc
import board
import displayio
import terminalio
from adafruit_bitmap_font import bitmap_font

from adafruit_display_text import bitmap_label
from adafruit_display_text import label

# pylint: disable=no-member


##########
# Use this Boolean variables to select which font style to use
##########
use_builtinfont = False  # Set True to use the terminalio.FONT BuiltinFont,
fontToUse = terminalio.FONT
# Set False to use a BDF loaded font, see "fontFiles" below
##########

if not use_builtinfont:
    # load the fonts
    print("loading font...")

    fontList = []

    # Load some proportional fonts
```

(continues on next page)

(continued from previous page)

```
    fontFile = "fonts/LeagueSpartan-Bold-16.bdf"
    fontToUse = bitmap_font.load_font(fontFile)

# Set scaling factor for display text
my_scale = 1

#  Setup the SPI display
if "DISPLAY" in dir(board):
    # use built in display (PyPortal, PyGamer, PyBadge, CLUE, etc.)
    # see guide for setting up external displays (TFT / OLED breakouts, RGB matrices, etc.)
    # https://learn.adafruit.com/circuitpython-display-support-using-displayio/display-and-display-bus
    display = board.DISPLAY

else:
    # Setup the LCD display with driver
    # You may need to change this to match the display driver for the chipset
    # used on your display
    from adafruit_ili9341 import ILI9341

    displayio.release_displays()

    # setup the SPI bus
    spi = board.SPI()
    tft_cs = board.D9  # arbitrary, pin not used
    tft_dc = board.D10
    tft_backlight = board.D12
    tft_reset = board.D11

    while not spi.try_lock():
        spi.configure(baudrate=32000000)
    spi.unlock()

    display_bus = displayio.FourWire(
        spi,
        command=tft_dc,
        chip_select=tft_cs,
        reset=tft_reset,
        baudrate=32000000,
        polarity=1,
        phase=1,
    )

    # Number of pixels in the display
    DISPLAY_WIDTH = 320
    DISPLAY_HEIGHT = 240

    # create the display
    display = ILI9341(
        display_bus,
        width=DISPLAY_WIDTH,
        height=DISPLAY_HEIGHT,
        rotation=180,  # The rotation can be adjusted to match your configuration.
        auto_refresh=True,
        native_frames_per_second=90,
    )
```

(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\nto 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)

```
    padding_bottom=label2_padding,
    padding_left=0,
    padding_right=0,
    padding_top=label2_padding,
    background_tight=False,
    line_spacing=1.25,
    scale=my_scale,
    anchor_point=(0.0, 0),
    anchored_position=(10, 120),
)

gc.collect()
bitmap_label_end = gc.mem_free()

print("bitmap_label used: {} memory".format(bitmap_label_start - bitmap_label_end))

bmap_group = displayio.Group()  # Create a group for displaying
bmap_group.append(bmap_label1)
bmap_group.append(bmap_label2)


#####
# Create the "label.py" versions of the text labels.

gc.collect()
label_start = gc.mem_free()

label1 = label.Label(
    font=fontToUse,
    text="label",
    color=0xFFFFFF,
    background_color=0xFF0000,
    padding_bottom=0,
    padding_left=0,
    padding_right=0,
    padding_top=0,
    background_tight=True,
    line_spacing=1.25,
    scale=my_scale,
    anchor_point=(1.0, 0),
    anchored_position=(display.width - 10, 60),
)

label2 = label.Label(
    font=fontToUse,
    text=long_string,
    color=0x000000,
    background_color=0xFFFF00,
    padding_bottom=label2_padding,
    padding_left=0,
    padding_right=0,
    padding_top=label2_padding,
    background_tight=False,
    line_spacing=1.25,
    scale=my_scale,
    anchor_point=(1.0, 0),
    anchored_position=(display.width - 10, 120),
```

(continues on next page)

(continued from previous page)

```
)

gc.collect()
label_end = gc.mem_free()

print("label used: {} memory".format(label_start - label_end))
label_group = displayio.Group()  # Create a group for displaying
label_group.append(label1)
label_group.append(label2)


print("***")

main_group = displayio.Group()
main_group.append(label_group)
main_group.append(bmap_group)

display.auto_refresh = True

display.show(main_group)
while True:
    pass
```

## 5.4 Background color example

Show the text backgrounds features

Listing 4: examples/display_text_background_color.py

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

"""
This example shows the use color and background_color
"""
import time
import board
import terminalio
from adafruit_display_text import label

text = " Color Background Hello world"
text_area = label.Label(
    terminalio.FONT, text=text, color=0x0000FF, background_color=0xFFAA00
)
text_area.x = 10
text_area.y = 10

print("background color is {:06x}".format(text_area.background_color))

board.DISPLAY.show(text_area)

time.sleep(2)
text_area.background_color = 0xFF0000
print("background color is {:06x}".format(text_area.background_color))
```

(continues on next page)

(continued from previous page)

```
time.sleep(2)
text_area.background_color = None
print("background color is {}".format(text_area.background_color))
while True:
    pass
```

## 5.5 Text padding example

Show the text padding features in all directions

Listing 5: examples/display_text_background_color.py

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

"""
This example shows the use color and background_color
"""
import time
import board
import displayio

from adafruit_bitmap_font import bitmap_font
from adafruit_display_text import label


#  Setup the SPI display
if "DISPLAY" in dir(board):
    # use built in display (PyPortal, PyGamer, PyBadge, CLUE, etc.)
    # see guide for setting up external displays (TFT / OLED breakouts, RGB matrices, etc.)
    # https://learn.adafruit.com/circuitpython-display-support-using-displayio/display-and-display-bus
    display = board.DISPLAY

else:
    print("Starting external display")  # goes to serial only
    # Setup the LCD display with driver
    # You may need to change this to match the display driver for the chipset
    # used on your display
    from adafruit_ili9341 import ILI9341

    # from adafruit_st7789 import ST7789

    displayio.release_displays()

    # setup the SPI bus
    spi = board.SPI()
    tft_cs = board.D9  # arbitrary, pin not used
    tft_dc = board.D10
    tft_backlight = board.D12
    tft_reset = board.D11

    while not spi.try_lock():
```

(continues on next page)

(continued from previous page)

```
        spi.configure(baudrate=32000000)
    spi.unlock()

    display_bus = displayio.FourWire(
        spi,
        command=tft_dc,
        chip_select=tft_cs,
        reset=tft_reset,
        baudrate=32000000,
        polarity=1,
        phase=1,
    )

    # Number of pixels in the display
    DISPLAY_WIDTH = 320
    DISPLAY_HEIGHT = 240

    # display = ST7789(display_bus, width=240, height=240, rotation=0, rowstart=80,␣
↪colstart=0)

    # create the display
    display = ILI9341(
        display_bus,
        width=DISPLAY_WIDTH,
        height=DISPLAY_HEIGHT,
        rotation=180,  # The rotation can be adjusted to match your configuration.
        auto_refresh=True,
        native_frames_per_second=90,
    )

display.show(None)

# font=terminalio.FONT # this is the Builtin fixed dimension font

font = bitmap_font.load_font("fonts/LeagueSpartan-Bold-16.bdf")

text = []
text.append("none")  # no ascenders or descenders
text.append("pop quops")  # only descenders
text.append("MONSTERs are tall")  # only ascenders
text.append("MONSTERs ate pop quops")  # both ascenders and descenders
text.append("MONSTER quops\nnewline quops")  # with newline

display.auto_refresh = True
myGroup = displayio.Group()
display.show(myGroup)

text_area = []
myPadding = 4

for i, thisText in enumerate(text):
    text_area.append(
        label.Label(
            font,
            text=thisText,
            color=0xFFFFFF,
            background_color=None,
```

(continues on next page)

(continued from previous page)

```
            background_tight=False,
            padding_top=myPadding,
            padding_bottom=myPadding,
            padding_left=myPadding,
            padding_right=myPadding,
        )
    )

    this_x = 10
    this_y = 10 + i * 40
    text_area[i].x = 10
    text_area[i].y = 3 + i * 50
    text_area[i].anchor_point = (0, 0)
    text_area[i].anchored_position = (this_x, this_y)
    myGroup.append(text_area[i])

print("background color is {}".format(text_area[0].background_color))


while True:
    time.sleep(2)
    text_area[0].text = "text"  # change some text in an existing text box
    # Note: changed text must fit within existing number of characters
    # when the Label was created

    for area in text_area:
        area.background_color = 0xFF0000
    print("background color is {:06x}".format(text_area[0].background_color))
    time.sleep(2)
    for area in text_area:
        area.background_color = 0x000088
    print("background color is {:06x}".format(text_area[0].background_color))
    time.sleep(2)
    for area in text_area:
        area.background_color = 0x00FF00
    print("background color is {:06x}".format(text_area[0].background_color))
    time.sleep(2)
    for area in text_area:
        area.background_color = 0xFF0000
    print("background color is {:06x}".format(text_area[0].background_color))
    time.sleep(2)
    for area in text_area:
        area.background_color = None
    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

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

```

(continues on next page)

(continued from previous page)

```
"""
This examples shows the use of anchor_point and anchored_position.
"""
import board
import terminalio
import displayio
from adafruit_display_text import label

DISPLAY_WIDTH = 320
DISPLAY_HEIGHT = 240
TEXT = "Hello"

text_area_top_left = label.Label(terminalio.FONT, text=TEXT)
text_area_top_left.anchor_point = (0.0, 0.0)
text_area_top_left.anchored_position = (0, 0)

text_area_top_middle = label.Label(terminalio.FONT, text=TEXT)
text_area_top_middle.anchor_point = (0.5, 0.0)
text_area_top_middle.anchored_position = (DISPLAY_WIDTH / 2, 0)

text_area_top_right = label.Label(terminalio.FONT, text=TEXT)
text_area_top_right.anchor_point = (1.0, 0.0)
text_area_top_right.anchored_position = (DISPLAY_WIDTH, 0)

text_area_middle_left = label.Label(terminalio.FONT, text=TEXT)
text_area_middle_left.anchor_point = (0.0, 0.5)
text_area_middle_left.anchored_position = (0, DISPLAY_HEIGHT / 2)

text_area_middle_middle = label.Label(terminalio.FONT, text=TEXT)
text_area_middle_middle.anchor_point = (0.5, 0.5)
text_area_middle_middle.anchored_position = (DISPLAY_WIDTH / 2, DISPLAY_HEIGHT / 2)

text_area_middle_right = label.Label(terminalio.FONT, text=TEXT)
text_area_middle_right.anchor_point = (1.0, 0.5)
text_area_middle_right.anchored_position = (DISPLAY_WIDTH, DISPLAY_HEIGHT / 2)

text_area_bottom_left = label.Label(terminalio.FONT, text=TEXT)
text_area_bottom_left.anchor_point = (0.0, 1.0)
text_area_bottom_left.anchored_position = (0, DISPLAY_HEIGHT)

text_area_bottom_middle = label.Label(terminalio.FONT, text=TEXT)
text_area_bottom_middle.anchor_point = (0.5, 1.0)
text_area_bottom_middle.anchored_position = (DISPLAY_WIDTH / 2, DISPLAY_HEIGHT)

text_area_bottom_right = label.Label(terminalio.FONT, text=TEXT)
text_area_bottom_right.anchor_point = (1.0, 1.0)
text_area_bottom_right.anchored_position = (DISPLAY_WIDTH, DISPLAY_HEIGHT)

text_group = displayio.Group()
text_group.append(text_area_top_middle)
text_group.append(text_area_top_left)
text_group.append(text_area_top_right)
text_group.append(text_area_middle_middle)
text_group.append(text_area_middle_left)
text_group.append(text_area_middle_right)
text_group.append(text_area_bottom_middle)
text_group.append(text_area_bottom_left)
```

(continues on next page)

(continued from previous page)

```
text_group.append(text_area_bottom_right)

board.DISPLAY.show(text_group)

while True:
    pass
```

## 5.7 Textarea Boundingbox

Boundingbox demonstration

Listing 7: examples/display_text_textarea_boundingbox.py

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

import os
import board
import displayio
from adafruit_bitmap_font import bitmap_font
from adafruit_display_text.label import Label


# the current working directory (where this file is)
cwd = ("/" + __file__).rsplit("/", 1)[0]
fonts = [
    file
    for file in os.listdir(cwd + "/fonts/")
    if (file.endswith(".bdf") and not file.startswith("._"))
]
for i, filename in enumerate(fonts):
    fonts[i] = cwd + "/fonts/" + filename
print(fonts)

################################################################################
THE_FONT = fonts[0]
DISPLAY_STRING = "A multi-line-\nexample of\n  font bounding!"
WRAP_CHARS = 40

################################################################################
# Make the display context
splash = displayio.Group()
board.DISPLAY.show(splash)

# Make a background color fill
color_bitmap = displayio.Bitmap(320, 240, 1)
color_palette = displayio.Palette(1)
color_palette[0] = 0xFFFFFF
bg_sprite = displayio.TileGrid(color_bitmap, pixel_shader=color_palette, x=0, y=0)
splash.append(bg_sprite)

# Load the font
font = bitmap_font.load_font(THE_FONT)
font.load_glyphs(DISPLAY_STRING.encode("utf-8"))
```

(continues on next page)

(continued from previous page)

```

print(DISPLAY_STRING)

text = Label(font, text=DISPLAY_STRING)
text.x = 20
text.y = 100
text.color = 0x0

# Make a background color fill
dims = text.bounding_box
print(dims)
textbg_bitmap = displayio.Bitmap(dims[2], dims[3], 1)
textbg_palette = displayio.Palette(1)
textbg_palette[0] = 0xFF0000
textbg_sprite = displayio.TileGrid(
    textbg_bitmap, pixel_shader=textbg_palette, x=text.x + dims[0], y=text.y + dims[1]
)
splash.append(textbg_sprite)
splash.append(text)
try:
    board.DISPLAY.refresh(target_frames_per_second=60)
except AttributeError:
    board.DISPLAY.refresh_soon()
    board.DISPLAY.wait_for_frame()


while True:
    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

```
# SPDX-FileCopyrightText: 2021 Jose David Montoya for Adafruit Industries
# SPDX-License-Identifier: MIT

"""
This example shows the use of base_alignment parameter.
"""

import board
import displayio
from adafruit_bitmap_font import bitmap_font
from adafruit_display_text import label


display = board.DISPLAY

# Font definition. You can choose any two fonts available in your system
MEDIUM_FONT = bitmap_font.load_font("LeagueSpartan-Bold-16.bdf")
BIG_FONT = bitmap_font.load_font("LibreBodoniv2002-Bold-27.bdf")

```

(continues on next page)

(continued from previous page)

```
TEXT_RIGHT = "MG"
TEXT_LEFT = "32.47"

main_group = displayio.Group()

# Create labels
# Base Alignment parameter False
left_text = label.Label(
    BIG_FONT,
    text=TEXT_LEFT,
    color=0x000000,
    background_color=0x999999,
    x=10,
    y=50,
    base_alignment=False,
)
main_group.append(left_text)

right_text = label.Label(
    MEDIUM_FONT,
    text=TEXT_RIGHT,
    color=0x000000,
    background_color=0x999999,
    x=90,
    y=50,
    base_alignment=False,
)
main_group.append(right_text)

# Base Alignment parameter True
left_text_aligned = label.Label(
    BIG_FONT,
    text=TEXT_LEFT,
    color=0x000000,
    background_color=0x999999,
    x=10,
    y=100,
    base_alignment=True,
)
main_group.append(left_text_aligned)

right_text_aligned = label.Label(
    MEDIUM_FONT,
    text=TEXT_RIGHT,
    color=0x000000,
    background_color=0x999999,
    x=90,
    y=100,
    base_alignment=True,
)

main_group.append(right_text_aligned)
display.show(main_group)

while True:
    pass
```

## 5.9 Magtag example

Uses the MAGTAG to display some text

Listing 9: examples/display_text_magtag.py

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

"""
Basic display_text.label example script
adapted for use on MagTag.
"""
import time
import board
import displayio
import terminalio
from adafruit_display_text import label

# use built in display (PyPortal, PyGamer, PyBadge, CLUE, etc.)
# see guide for setting up external displays (TFT / OLED breakouts, RGB matrices, etc.)
# https://learn.adafruit.com/circuitpython-display-support-using-displayio/display-and-display-bus
display = board.DISPLAY

# wait until we can draw
time.sleep(display.time_to_refresh)

# main group to hold everything
main_group = displayio.Group()

# white background. Scaled to save RAM
bg_bitmap = displayio.Bitmap(display.width // 8, display.height // 8, 1)
bg_palette = displayio.Palette(1)
bg_palette[0] = 0xFFFFFF
bg_sprite = displayio.TileGrid(bg_bitmap, x=0, y=0, pixel_shader=bg_palette)
bg_group = displayio.Group(scale=8)
bg_group.append(bg_sprite)
main_group.append(bg_group)

# first example label
TEXT = "Hello world"
text_area = label.Label(
    terminalio.FONT,
    text=TEXT,
    color=0xFFFFFF,
    background_color=0x666666,
    padding_top=1,
    padding_bottom=3,
    padding_right=4,
    padding_left=4,
)
text_area.x = 10
text_area.y = 14
main_group.append(text_area)

```

(continues on next page)

(continued from previous page)

```
# second example label
another_text = label.Label(
    terminalio.FONT,
    scale=2,
    text="MagTag display_text\nexample",
    color=0x000000,
    background_color=0x999999,
    padding_top=1,
    padding_bottom=3,
    padding_right=4,
    padding_left=4,
)
# centered
another_text.anchor_point = (0.5, 0.5)
another_text.anchored_position = (display.width // 2, display.height // 2)
main_group.append(another_text)

# show the main group and refresh.
display.show(main_group)
display.refresh()
while True:
    pass
```

## 5.10 MatrixPortal example

Uses the MatrixPortal to display some text

Listing 10: examples/display_text_matrixportal.py

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

"""
This example shows how to create a display_text label and show it
with a Matrix Portal

Requires:
adafruit_matrixportal - https://github.com/adafruit/Adafruit_CircuitPython_
↪MatrixPortal

Copy it from the current libraries bundle into the lib folder on your device.
"""
import terminalio
from adafruit_matrixportal.matrix import Matrix
from adafruit_display_text import label

matrix = Matrix()
display = matrix.display

text = "Hello\nworld"
text_area = label.Label(terminalio.FONT, text=text)
text_area.x = 1
text_area.y = 4
display.show(text_area)
```

(continues on next page)

(continued from previous page)

```
while True:
    pass
```

## 5.11 PyPortal example

Uses the Pyportal to display some text

Listing 11: examples/display_text_pyportal.py

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

"""
This example show the use of the backlight as well as using labels to simulate
a terminal using a font on the PyPortal
"""

import os
import time
import board
import displayio

from adafruit_bitmap_font import bitmap_font
from adafruit_display_text.label import Label

FONT_DIR = "/fonts/"
fonts = list(
    filter(lambda x: x.endswith("bdf") and not x.startswith("."), os.listdir(FONT_
↪DIR))
)
fonts = [bitmap_font.load_font(FONT_DIR + x) for x in fonts]
if len(fonts) == 0:
    print("No fonts found in '{}'".format(FONT_DIR))

print("fade up")
# Fade up the backlight
for b in range(100):
    board.DISPLAY.brightness = b / 100
    time.sleep(0.01)  # default (0.01)

demos = ["CircuitPython = Code + Community", "accents - üàêùéáçãÎói", "others - αψ"]

splash = displayio.Group()
board.DISPLAY.show(splash)
max_y = 0
y = 0
for demo_text in demos:
    for font in fonts:
        if y >= board.DISPLAY.height:
            y = 0
            while len(splash):
                splash.pop()
        print("Font load {}".format(font.name))
        area = Label(
```

(continues on next page)

(continued from previous page)

```
            font, text=demo_text, anchor_point=(0, 0), anchored_position=(0, y)
        )
        splash.append(area)

        y += area.height

        # Wait for the image to load.
        try:
            board.DISPLAY.refresh(target_frames_per_second=60)
        except AttributeError:
            board.DISPLAY.wait_for_frame()

# Wait for 1 minute (60 seconds)
time.sleep(60)

# Fade down the backlight
for b in range(100, -1, -1):
    board.DISPLAY.brightness = b / 100
    time.sleep(0.01)  # default (0.01)

print("fade down")

time.sleep(10)
```

## 5.12 Wraptest example

Illustrates the wraptest feature

Listing 12: examples/display_text_wraptest.py

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

"""
This example illustrates how to use the wrap_text_to_lines
helper function.
"""
import board
import terminalio
from adafruit_display_text import label, wrap_text_to_lines

# use built in display (PyPortal, PyGamer, PyBadge, CLUE, etc.)
# see guide for setting up external displays (TFT / OLED breakouts, RGB matrices, etc.)
# https://learn.adafruit.com/circuitpython-display-support-using-displayio/display-and-display-bus
display = board.DISPLAY

text = (
    "Lorem ipsum dolor sit amet, consectetur adipiscing elit, "
    "sed do eiusmod tempor incididunt ut labore et dolore magna "
    "aliqua. Ut enim ad minim veniam, quis nostrud exercitation "
    "ullamco laboris nisi ut aliquip ex ea commodo consequat."
)
```

(continues on next page)

(continued from previous page)

```
text = "\n".join(wrap_text_to_lines(text, 28))
text_area = label.Label(terminalio.FONT, text=text)
text_area.x = 10
text_area.y = 10
display.show(text_area)
while True:
    pass
```

## 5.13 Wrap Pixel Test

Wrap Pixel Test

Listing 13: examples/display_text_wrap_pixels_test.py

```
# SPDX-FileCopyrightText: 2021 Tim C, written for Adafruit Industries
#
# SPDX-License-Identifier: MIT
"""
Test the wrap_text_to_pixels function. Try changing WRAP_WIDTH or text
and observe the results. The red bar represents the full size of
WRAP_WIDTH.
"""

import board
import displayio
import terminalio
from adafruit_display_text import label, wrap_text_to_pixels

WRAP_WIDTH = 140
text = (
    "CircuitPython is a programming language designed to simplify experimenting "
    "and learning to code on low-cost microcontroller boards. "
)

# use built in display (PyPortal, PyGamer, PyBadge, CLUE, etc.)
# see guide for setting up external displays (TFT / OLED breakouts, RGB matrices, etc.)
# https://learn.adafruit.com/circuitpython-display-support-using-displayio/display-and-display-bus
display = board.DISPLAY

# Make the display context
main_group = displayio.Group()
display.show(main_group)

font = terminalio.FONT

print(text)
print(display.width)

text_area = label.Label(
    font,
    text="\n".join(wrap_text_to_pixels(text, WRAP_WIDTH, font)),
    background_color=0x0000DD,
```

(continues on next page)

(continued from previous page)

```
)

text_area.anchor_point = (0, 0)
text_area.anchored_position = (0, 0)

main_group.append(text_area)

# Create a bitmap with two colors
size_checker = displayio.Bitmap(WRAP_WIDTH, 10, 2)
# Create a two color palette
palette = displayio.Palette(2)
palette[0] = 0x0000DD
palette[1] = 0xDD0000

# Create a TileGrid using the Bitmap and Palette
tile_grid = displayio.TileGrid(size_checker, pixel_shader=palette)

tile_grid.y = text_area.bounding_box[1] + text_area.bounding_box[3] + 10

size_checker.fill(1)

main_group.append(tile_grid)

while True:
    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

```
# SPDX-FileCopyrightText: 2021 Jose David M.
#
# SPDX-License-Identifier: MIT
##############################
"""
This is an advanced demonstration of the display_text library capabilities
"""

import time
import board
import displayio
import terminalio
import fontio
from adafruit_display_text import label, bitmap_label
from adafruit_bitmap_font import bitmap_font

display = board.DISPLAY
main_group = displayio.Group()
MEDIUM_FONT = bitmap_font.load_font("fonts/LeagueSpartan-Bold-16.bdf")
BIG_FONT = bitmap_font.load_font("fonts/LibreBodoniv2002-Bold-27.bdf")
TIME_PAUSE = 2

```

(continues on next page)

(continued from previous page)

```
bitmap = displayio.Bitmap(4, display.width, 2)
palette = displayio.Palette(2)
palette[0] = 0x004400
palette[1] = 0x00FFFF
horizontal_line = displayio.TileGrid(bitmap, pixel_shader=palette, x=155, y=0)
main_group.append(horizontal_line)

bitmap = displayio.Bitmap(display.width, 4, 2)
vertical_line = displayio.TileGrid(bitmap, pixel_shader=palette, x=0, y=110)
main_group.append(vertical_line)

# Tests
text_area = label.Label(terminalio.FONT, text="Circuit Python")
main_group.append(text_area)
display.show(main_group)
time.sleep(TIME_PAUSE)

# Testing position setter
text_area.x = 10
text_area.y = 10
display.show(main_group)
time.sleep(TIME_PAUSE)

# Testing creating label with initial position
text_area.text = "Testing initiating without text"
try:
    text_middle = label.Label(terminalio.FONT)
except SyntaxError:
    print("Fail setting-up label without text")
    warning_text = label.Label(
        BIG_FONT,
        text="Test Fail",
        x=display.width // 2,
        y=display.height // 4,
        background_color=0x004499,
    )
    main_group.append(warning_text)
display.show(main_group)
time.sleep(TIME_PAUSE)

text_area.text = "Testing Position"
text_middle = label.Label(
    terminalio.FONT, text="Circuit", x=display.width // 2, y=display.height // 2
)
main_group.append(text_middle)
display.show(main_group)
time.sleep(TIME_PAUSE)

# Testing Text Setter
text_area.text = "Testing Changing Text"
text_middle.text = "Python"
display.show(main_group)
time.sleep(TIME_PAUSE)

# Testing a and y getter and setter
text_area.text = "Testing Changing Position"
text_middle.x = text_middle.x - 50
```

(continues on next page)

(continued from previous page)

```
text_middle.y = text_middle.y - 50
display.show(main_group)
time.sleep(TIME_PAUSE)

# Testing font Getter and setter
text_area.text = "Testing Changing FONT"
if isinstance(text_middle.font, fontio.BuiltinFont):
    text_middle.font = MEDIUM_FONT
display.show(main_group)
time.sleep(TIME_PAUSE)

# Once this working we create another label with all the initial specs
main_group.pop()

# Testing Color
text_area.text = "Testing Color"
text_initial_specs = label.Label(
    MEDIUM_FONT,
    text="Circuit Python",
    x=display.width // 2,
    y=display.height // 2,
)
main_group.append(text_initial_specs)
display.show(main_group)
time.sleep(TIME_PAUSE)

text_initial_specs.color = 0x004400
display.show(main_group)
time.sleep(TIME_PAUSE)
main_group.pop()

# Testing Background Color
text_area.text = "Testing Background Color"
text_initial_specs = label.Label(
    MEDIUM_FONT,
    text="CircuitPython",
    x=display.width // 2,
    y=display.height // 2,
    color=0xFFFFFF,
)
main_group.append(text_initial_specs)
display.show(main_group)
time.sleep(TIME_PAUSE)

text_initial_specs.background_color = 0x990099
display.show(main_group)
time.sleep(TIME_PAUSE)
main_group.pop()

# Testing Background Color
text_area.text = "Testing Background Tight"
text_initial_specs = label.Label(
    BIG_FONT,
    text="aaaaq~",
    x=0,
    y=display.height // 2,
    color=0xFFFFFF,
```

(continues on next page)

(continued from previous page)

```
    background_color=0x990099,
    background_tight=True,
)
main_group.append(text_initial_specs)
text_initial_specs = label.Label(
    BIG_FONT,
    text="aaaaq~",
    x=90,
    y=display.height // 2,
    color=0xFFFFFF,
    background_color=0x990099,
    background_tight=False,
)
main_group.append(text_initial_specs)
display.show(main_group)
time.sleep(TIME_PAUSE)
main_group.pop()
main_group.pop()

# Testing Padding
text_area.text = "Testing Padding"
text_initial_specs = label.Label(
    BIG_FONT,
    text="CircuitPython",
    x=display.width // 4,
    y=display.height // 2,
    color=0xFFFFFF,
    background_color=0x990099,
    padding_right=10,
    padding_top=10,
    padding_bottom=10,
    padding_left=10,
)
main_group.append(text_initial_specs)
display.show(main_group)
time.sleep(TIME_PAUSE)
main_group.pop()

# Testing Anchor Point/ Anchored Position
text_area.text = "Testing Anchor Point/Anchored Position"
text_initial_specs = label.Label(
    MEDIUM_FONT,
    text="CircuitPython",
    x=display.width // 2,
    y=display.height // 2,
    color=0xFFFFFF,
    background_color=0x990099,
    padding_right=10,
    padding_top=10,
    padding_bottom=10,
    padding_left=10,
)
main_group.append(text_initial_specs)
display.show(main_group)
time.sleep(TIME_PAUSE)

try:
```

(continues on next page)

(continued from previous page)

```
    text_initial_specs.anchored_position = (100, 100)
    text_initial_specs.anchor_point = (0.5, 0.5)

except TypeError:
    print("Test is failing here")
    main_group.pop()
    warning_text = label.Label(
        BIG_FONT,
        text="Test Fail",
        x=display.width // 2,
        y=display.height // 4,
        background_color=0x004499,
    )
    main_group.append(warning_text)
    time.sleep(TIME_PAUSE)
    display.show(main_group)

main_group.pop()

# Testing Scale
text_area.text = "Testing Scale"
text_initial_specs = label.Label(
    MEDIUM_FONT,
    text="CircuitPython",
    x=display.width // 2,
    y=display.height // 2,
    color=0xFFFFFF,
    background_color=0x990099,
    padding_right=10,
    padding_top=10,
    padding_bottom=10,
    padding_left=10,
    anchored_position=(display.width // 2, display.height // 2),
    anchor_point=(0.5, 0.5),
)
main_group.append(text_initial_specs)
display.show(main_group)
time.sleep(TIME_PAUSE)

text_initial_specs.scale = 2
display.show(main_group)
time.sleep(TIME_PAUSE)
main_group.pop()

# Testing Base Alignment
text_area.text = "Testing Base Alignment"
text_initial_specs = label.Label(
    MEDIUM_FONT,
    text="python",
    x=display.width // 2,
    y=display.height // 2,
    color=0xFFFFFF,
    background_color=0x990099,
    base_alignment=True,
)
main_group.append(text_initial_specs)
text_initial_specs = label.Label(
```

(continues on next page)

(continued from previous page)

```
    BIG_FONT,
    text="circuit",
    x=display.width // 2 - 100,
    y=display.height // 2,
    color=0xFFFFFF,
    background_color=0x990099,
    base_alignment=True,
)
main_group.append(text_initial_specs)
display.show(main_group)
time.sleep(TIME_PAUSE)
main_group.pop()
main_group.pop()

# Testing Direction
text_area.text = "Testing Direction-UPR"
text_initial_specs = label.Label(
    MEDIUM_FONT,
    text="CircuitPython",
    x=display.width // 2,
    y=display.height // 2,
    color=0xFFFFFF,
    background_color=0x990099,
    padding_right=10,
    padding_top=10,
    padding_bottom=10,
    padding_left=10,
    anchored_position=(display.width // 2, display.height // 2),
    anchor_point=(0.5, 0.5),
    label_direction="UPR",
)
main_group.append(text_initial_specs)
display.show(main_group)
time.sleep(TIME_PAUSE)
main_group.pop()

text_area.text = "Testing Direction-DWR"
text_initial_specs = label.Label(
    MEDIUM_FONT,
    text="CircuitPython",
    x=display.width // 2,
    y=display.height // 2,
    color=0xFFFFFF,
    background_color=0x990099,
    padding_right=10,
    padding_top=10,
    padding_bottom=10,
    padding_left=10,
    anchored_position=(display.width // 2, display.height // 2),
    anchor_point=(0.5, 0.5),
    label_direction="DWR",
)
main_group.append(text_initial_specs)
display.show(main_group)
time.sleep(TIME_PAUSE)
main_group.pop()

```

(continues on next page)

(continued from previous page)

```
text_area.text = "Testing Direction-TTB"
text_initial_specs = label.Label(
    MEDIUM_FONT,
    text="CircuitPython",
    x=display.width // 2,
    y=display.height // 2,
    color=0xFFFFFF,
    background_color=0x990099,
    padding_right=10,
    padding_top=10,
    padding_bottom=10,
    padding_left=10,
    anchored_position=(display.width // 2, display.height // 2),
    anchor_point=(0.5, 0.5),
    label_direction="TTB",
)
main_group.append(text_initial_specs)
display.show(main_group)
time.sleep(TIME_PAUSE)
main_group.pop()

text_area.text = "Testing Direction-RTL"
text_initial_specs = label.Label(
    MEDIUM_FONT,
    text="CircuitPython",
    x=display.width // 2,
    y=display.height // 2,
    color=0xFFFFFF,
    background_color=0x990099,
    padding_right=10,
    padding_top=10,
    padding_bottom=10,
    padding_left=10,
    anchored_position=(display.width // 2, display.height // 2),
    anchor_point=(0.5, 0.5),
    label_direction="RTL",
)
main_group.append(text_initial_specs)
display.show(main_group)
time.sleep(TIME_PAUSE)
main_group.pop()

main_group.pop()

# Testing creating label with initial position
display.show(main_group)
time.sleep(TIME_PAUSE)
text_area = bitmap_label.Label(terminalio.FONT, text="Circuit Python")
main_group.append(text_area)
display.show(main_group)
time.sleep(TIME_PAUSE)
# Testing position setter
text_area.x = 10
text_area.y = 10
display.show(main_group)
time.sleep(TIME_PAUSE)
text_area.text = "Testing initiating without text"
```

(continues on next page)

(continued from previous page)

```
try:
    text_middle = label.Label(terminalio.FONT)
except TypeError:
    print("Fail setting-up label without text")
    warning_text = label.Label(
        BIG_FONT,
        text="Test Fail",
        x=display.width // 2,
        y=display.height // 4,
        background_color=0x004499,
    )
    main_group.append(warning_text)

# Testing creating label with initial position
text_area.text = "Testing Position"
text_middle = bitmap_label.Label(
    terminalio.FONT, text="Circuit", x=display.width // 2, y=display.height // 2
)
main_group.append(text_middle)
display.show(main_group)
time.sleep(TIME_PAUSE)

# Testing Text Setter
text_area.text = "Testing Changing Text"
text_middle.text = "Python"
display.show(main_group)
time.sleep(TIME_PAUSE)

# Testing a and y getter and setter
text_area.text = "Testing Changing Position"
text_middle.x = text_middle.x - 50
text_middle.y = text_middle.y - 50
display.show(main_group)
time.sleep(TIME_PAUSE)

# Testing font Getter and setter
text_area.text = "Testing Changing FONT"
if isinstance(text_middle.font, fontio.BuiltinFont):
    print("Font was BuiltinFont")
    text_middle.font = MEDIUM_FONT
display.show(main_group)
time.sleep(TIME_PAUSE)

# Once this working we create another label with all the initial specs
main_group.pop()

# Testing Color
text_area.text = "Testing Color"
text_initial_specs = bitmap_label.Label(
    MEDIUM_FONT,
    text="Circuit Python",
    x=display.width // 2,
    y=display.height // 2,
)
main_group.append(text_initial_specs)
display.show(main_group)
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)

```
    y=display.height // 2,
    color=0xFFFFFF,
    background_color=0x990099,
    padding_right=10,
    padding_top=10,
    padding_bottom=10,
    padding_left=10,
)
main_group.append(text_initial_specs)
display.show(main_group)
time.sleep(TIME_PAUSE)
main_group.pop()

# Testing Anchor Point/ Anchored Position
text_area.text = "Testing Anchor Point/Anchored Position"
text_initial_specs = bitmap_label.Label(
    MEDIUM_FONT,
    text="CircuitPython",
    x=display.width // 2,
    y=display.height // 2,
    color=0xFFFFFF,
    background_color=0x990099,
    padding_right=10,
    padding_top=10,
    padding_bottom=10,
    padding_left=10,
)
main_group.append(text_initial_specs)
display.show(main_group)
time.sleep(TIME_PAUSE)

try:
    text_initial_specs.anchored_position = (100, 100)
    text_initial_specs.anchor_point = (0.5, 0.5)

except TypeError:
    print("Test is failing here")
    main_group.pop()
    warning_text = bitmap_label.Label(
        BIG_FONT,
        text="Test Fail",
        x=display.width // 2,
        y=display.height // 4,
        background_color=0x004499,
    )
    main_group.append(warning_text)
    time.sleep(TIME_PAUSE)
    display.show(main_group)

main_group.pop()

# Testing Scale
text_area.text = "Testing Scale"
text_initial_specs = bitmap_label.Label(
    MEDIUM_FONT,
    text="CircuitPython",
    x=display.width // 2,
```

(continues on next page)

(continued from previous page)

```
    y=display.height // 2,
    color=0xFFFFFF,
    background_color=0x990099,
    padding_right=10,
    padding_top=10,
    padding_bottom=10,
    padding_left=10,
    anchored_position=(display.width // 2, display.height // 2),
    anchor_point=(0.5, 0.5),
)
main_group.append(text_initial_specs)
display.show(main_group)
time.sleep(TIME_PAUSE)

text_initial_specs.scale = 2
display.show(main_group)
time.sleep(TIME_PAUSE)
main_group.pop()

# Testing Base Alignment
text_area.text = "Testing Base Alignment"
text_initial_specs = bitmap_label.Label(
    MEDIUM_FONT,
    text="python",
    x=display.width // 2,
    y=display.height // 2,
    color=0xFFFFFF,
    background_color=0x990099,
    base_alignment=True,
)
main_group.append(text_initial_specs)
text_initial_specs = bitmap_label.Label(
    BIG_FONT,
    text="circuit",
    x=display.width // 2 - 100,
    y=display.height // 2,
    color=0xFFFFFF,
    background_color=0x990099,
    base_alignment=True,
)
main_group.append(text_initial_specs)
display.show(main_group)
time.sleep(TIME_PAUSE)
main_group.pop()
main_group.pop()

# Testing Direction
text_area.text = "Testing Direction-UPR"
text_initial_specs = bitmap_label.Label(
    MEDIUM_FONT,
    text="CircuitPython",
    x=display.width // 2,
    y=display.height // 2,
    color=0xFFFFFF,
    background_color=0x990099,
    padding_right=10,
    padding_top=10,
```

(continues on next page)

(continued from previous page)

```
    padding_bottom=10,
    padding_left=10,
    anchored_position=(display.width // 2, display.height // 2),
    anchor_point=(0.5, 0.5),
    label_direction="UPR",
)
main_group.append(text_initial_specs)
display.show(main_group)
time.sleep(TIME_PAUSE)
main_group.pop()

text_area.text = "Testing Direction-DWR"
text_initial_specs = bitmap_label.Label(
    MEDIUM_FONT,
    text="CircuitPython",
    x=display.width // 2,
    y=display.height // 2,
    color=0xFFFFFF,
    background_color=0x990099,
    padding_right=10,
    padding_top=10,
    padding_bottom=10,
    padding_left=10,
    anchored_position=(display.width // 2, display.height // 2),
    anchor_point=(0.5, 0.5),
    label_direction="DWR",
)
main_group.append(text_initial_specs)
display.show(main_group)
time.sleep(TIME_PAUSE)
main_group.pop()

text_area.text = "Testing Direction-UPD"
text_initial_specs = bitmap_label.Label(
    MEDIUM_FONT,
    text="CircuitPython",
    x=display.width // 2,
    y=display.height // 2,
    color=0xFFFFFF,
    background_color=0x990099,
    padding_right=10,
    padding_top=10,
    padding_bottom=10,
    padding_left=10,
    anchored_position=(display.width // 2, display.height // 2),
    anchor_point=(0.5, 0.5),
    label_direction="UPD",
)
main_group.append(text_initial_specs)
display.show(main_group)
time.sleep(TIME_PAUSE)
main_group.pop()

text_area.text = "Testing Direction-RTL"
text_initial_specs = bitmap_label.Label(
    MEDIUM_FONT,
    text="CircuitPython",
```

(continues on next page)

(continued from previous page)

```
    x=display.width // 2,
    y=display.height // 2,
    color=0xFFFFFF,
    background_color=0x990099,
    padding_right=10,
    padding_top=10,
    padding_bottom=10,
    padding_left=10,
    anchored_position=(display.width // 2, display.height // 2),
    anchor_point=(0.5, 0.5),
    label_direction="RTL",
)
main_group.append(text_initial_specs)
display.show(main_group)
time.sleep(TIME_PAUSE)
main_group.pop()

text_area.text = "Finished"
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 (number of lines * 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 possibles `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 (number of lines * 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 possibles `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

# Python Module Index

## a

# Index