

---

**KivyMD**

***Release 0.104.2.dev0***

**Andrés Rodríguez, Ivanov Yuri, Artem Bulgakov and KivyMD cont**

**Aug 24, 2020**



# CONTENTS

<b>1</b>	<b>KivyMD</b>	<b>1</b>
<b>2</b>	<b>Contents</b>	<b>3</b>
2.1	Getting Started . . . . .	3
2.2	Themes . . . . .	6
2.3	Components . . . . .	24
2.4	Behaviors . . . . .	227
2.5	Change Log . . . . .	242
2.6	About . . . . .	250
2.7	KivyMD . . . . .	251
<b>3</b>	<b>Indices and tables</b>	<b>277</b>
	<b>Python Module Index</b>	<b>279</b>
	<b>Index</b>	<b>281</b>

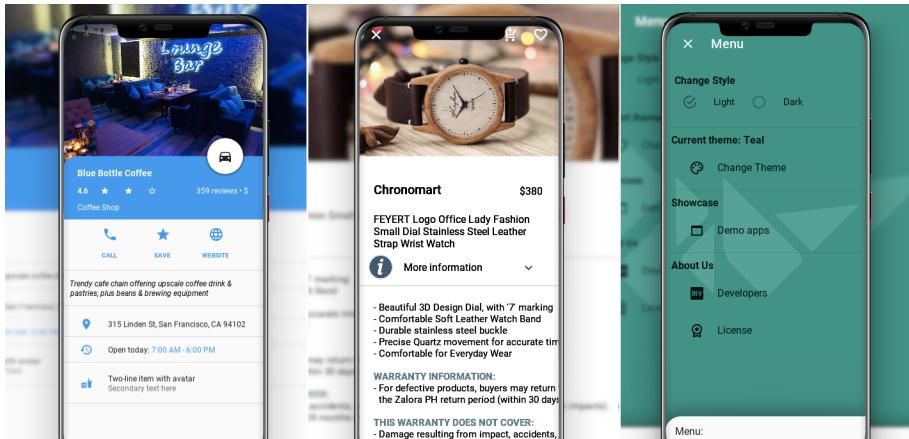


---

# CHAPTER ONE

---

## KIVYMD



Is a collection of Material Design compliant widgets for use with, Kivy cross-platform graphical framework a framework for cross-platform, touch-enabled graphical applications. The project's goal is to approximate Google's [Material Design spec](#) as close as possible without sacrificing ease of use or application performance.

This library is a fork of the [KivyMD project](#) the author of which stopped supporting this project three years ago. We found the strength and brought this project to a new level. Currently we're in **beta** status, so things are changing all the time and we cannot promise any kind of API stability. However it is safe to vendor now and make use of what's currently available.

Join the project! Just fork the project, branch out and submit a pull request when your patch is ready. If any changes are necessary, we'll guide you through the steps that need to be done via PR comments or access to your for may be requested to outright submit them. If you wish to become a project developer (permission to create branches on the project without forking for easier collaboration), have at least one PR approved and ask for it. If you contribute regularly to the project the role may be offered to you without asking too.



---

## CHAPTER TWO

---

## CONTENTS

## 2.1 Getting Started

In order to start using *KivyMD*, you must first install the *Kivy* framework on your computer. Once you have installed *Kivy*, you can install *KivyMD*.

**Warning:** *KivyMD* depends on *Kivy*! Therefore, before using *KivyMD*, first learn how to work with *Kivy*.

### 2.1.1 Installation

```
pip install kivymd
```

Command above will install latest release version of *KivyMD* from PyPI.

If you want to install development version from master branch, you should specify link to zip archive:

```
pip install https://github.com/kivymd/KivyMD/archive/master.zip
```

**\_Tip\_:** Replace *master.zip* with *<commit hash>.zip* (eg *51b8ef0.zip*) to download *KivyMD* from specific commit.

Also you can install manually from sources. Just clone the project and run pip:

```
git clone https://github.com/kivymd/KivyMD.git --depth 1
cd KivyMD
pip install .
```

**\_Speed Tip\_:** If you don't need full commit history (about 320 MiB), you can use a shallow clone (*git clone https://github.com/kivymd/KivyMD.git -depth 1*) to save time. If you need full commit history, then remove *-depth 1*.

### 2.1.2 First KivyMD application

```
from kivymd.app import MDApp
from kivymd.uix.label import MDLabel

class MainApp(MDApp):
    def build(self):
        return MDLabel(text="Hello, World", halign="center")

MainApp().run()
```

And the equivalent with *Kivy*:

```
from kivy.app import App
from kivy.uix.label import Label

class MainApp(App):
    def build(self):
        return Label(text="Hello, World")

MainApp().run()
```

To left - *Kivy*, to right - *KivyMD*:



At first glance, the *KivyMD* example contains more code... However, the following example already demonstrates how difficult it is to create a custom button in *Kivy*:

```
from kivy.app import App
from kivy.metrics import dp
from kivy.uix.behaviors import TouchRippleBehavior
from kivy.uix.button import Button
from kivy.lang import Builder


KV = """
<RectangleFlatButton>:
    ripple_color: 0, 0, 0, .2
    background_color: 0, 0, 0, 0
    color: root.primary_color

    canvas.before:
        Color:
            rgba: root.primary_color
        Line:
            width: 1
            rectangle: (self.x, self.y, self.width, self.height)
```

(continues on next page)

(continued from previous page)

```

Screen:
    canvas:
        Color:
            rgba: 0.9764705882352941, 0.9764705882352941, 0.9764705882352941, 1
    Rectangle:
        pos: self.pos
        size: self.size
"""

class RectangleFlatButton(TouchRippleBehavior, Button):
    primary_color = [
        0.12941176470588237,
        0.5882352941176471,
        0.9529411764705882,
        1
    ]

    def on_touch_down(self, touch):
        collide_point = self.collide_point(touch.x, touch.y)
        if collide_point:
            touch.grab(self)
            self.ripple_show(touch)
            return True
        return False

    def on_touch_up(self, touch):
        if touch.grab_current is self:
            touch.ungrab(self)
            self.ripple_fade()
            return True
        return False

class MainApp(App):
    def build(self):
        screen = Builder.load_string(KV)
        screen.add_widget(
            RectangleFlatButton(
                text="Hello, World",
                pos_hint={"center_x": 0.5, "center_y": 0.5},
                size_hint=(None, None),
                size=(dp(110), dp(35)),
                ripple_color=(0.8, 0.8, 0.8, 0.5),
            )
        )
    return screen

MainApp().run()

```

And the equivalent with *KivyMD*:

```
from kivy.uix.screenmanager import Screen

from kivymd.app import MDApp
from kivymd.uix.button import MDRectangleFlatButton

class MainApp(MDApp):
    def build(self):
        screen = Screen()
        screen.add_widget(
            MDRectangleFlatButton(
                text="Hello, World",
                pos_hint={"center_x": 0.5, "center_y": 0.5},
            )
        )
    return screen

MainApp().run()
```

To left - *Kivy*, to right - *KivyMD*:

## 2.2 Themes

### 2.2.1 Theming

See also:

Material Design spec, Material theming

#### Material App

The main class of your application, which in *Kivy* inherits from the `App` class, in *KivyMD* must inherit from the `MDApp` class. The `MDApp` class has properties that allow you to control application properties such as `color`/`style`/`font` of interface elements and much more.

#### Control material properties

The main application class inherited from the `MDApp` class has the `theme_cls` attribute, with which you control the material properties of your application.

**API - kivymd.theming**

```
class kivymd.theming.ThemeManager(**kwargs)
```

**primary\_palette**

The name of the color scheme that the application will use. All major *material* components will have the color of the specified color theme.

Available options are: ‘Red’, ‘Pink’, ‘Purple’, ‘DeepPurple’, ‘Indigo’, ‘Blue’, ‘LightBlue’, ‘Cyan’, ‘Teal’, ‘Green’, ‘LightGreen’, ‘Lime’, ‘Yellow’, ‘Amber’, ‘Orange’, ‘DeepOrange’, ‘Brown’, ‘Gray’, ‘BlueGray’.

To change the color scheme of an application:

```
from kivy.uix.screenmanager import Screen

from kivymd.app import MDApp
from kivymd.uix.button import MDRectangleFlatButton

class MainApp(MDApp):
    def build(self):
        self.theme_cls.primary_palette = "Green" # "Purple", "Red"

        screen = Screen()
        screen.add_widget(
            MDRectangleFlatButton(
                text="Hello, World",
                pos_hint={"center_x": 0.5, "center_y": 0.5},
            )
        )
    return screen

MainApp().run()
```



`primary_palette` is an `OptionProperty` and defaults to ‘Blue’.

**primary\_hue**

The color hue of the application.

Available options are: ‘50’, ‘100’, ‘200’, ‘300’, ‘400’, ‘500’, ‘600’, ‘700’, ‘800’, ‘900’, ‘A100’, ‘A200’, ‘A400’, ‘A700’.

To change the hue color scheme of an application:

```
from kivy.uix.screenmanager import Screen

from kivymd.app import MDApp
from kivymd.uix.button import MDRectangleFlatButton

class MainApp(MDApp):
    def build(self):
        self.theme_cls.primary_palette = "Green" # "Purple", "Red"
        self.theme_cls.primary_hue = "200" # "500"

        screen = Screen()
        screen.add_widget(
            MDRectangleFlatButton(
                text="Hello, World",
                pos_hint={"center_x": 0.5, "center_y": 0.5},
            )
        )
    return screen

MainApp().run()
```

With a value of `self.theme_cls.primary_hue = "500"`:



With a value of `self.theme_cls.primary_hue = "200"`:



`primary_hue` is an `OptionProperty` and defaults to '500'.

#### `primary_light_hue`

Hue value for `primary_light`.

`primary_light_hue` is an `OptionProperty` and defaults to ‘200’.

### `primary_dark_hue`

Hue value for `primary_dark`.

`primary_light_hue` is an `OptionProperty` and defaults to ‘700’.

### `primary_color`

The color of the current application theme in `rgba` format.

`primary_color` is an `AliasProperty` that returns the value of the current application theme, property is readonly.

### `primary_light`

Colors of the current application color theme in `rgba` format (in lighter color).

```
from kivy.lang import Builder

from kivymd.app import MDApp

KV = '''
Screen:

    MDRaisedButton:
        text: "primary_light"
        pos_hint: {"center_x": 0.5, "center_y": 0.7}
        md_bg_color: app.theme_cls.primary_light

    MDRaisedButton:
        text: "primary_color"
        pos_hint: {"center_x": 0.5, "center_y": 0.5}

    MDRaisedButton:
        text: "primary_dark"
        pos_hint: {"center_x": 0.5, "center_y": 0.3}
        md_bg_color: app.theme_cls.primary_dark
'''


class MainApp(MDApp):
    def build(self):
        self.theme_cls.primary_palette = "Green"
        return Builder.load_string(KV)

MainApp().run()
```



`primary_light` is an `AliasProperty` that returns the value of the current application theme (in lighter color), property is readonly.

#### **primary\_dark**

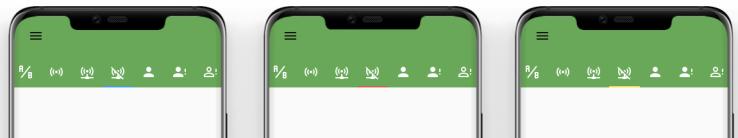
Colors of the current application color theme in `rgba` format (in darker color).

`primary_dark` is an `AliasProperty` that returns the value of the current application theme (in darker color), property is readonly.

#### **accent\_palette**

The application color palette used for items such as the tab indicator in the `MDTabsBar` class and so on...

The image below shows the color schemes with the values `self.theme_cls.accent_palette = 'Blue', Red' and Yellow'`:



`primary_hue` is an `OptionProperty` and defaults to ‘Amber’.

#### **accent\_hue**

Similar to `primary_hue`, but returns a value for `accent_palette`.

`accent_hue` is an `OptionProperty` and defaults to ‘500’.

#### **accent\_light\_hue**

Hue value for `accent_light`.

`accent_light_hue` is an `OptionProperty` and defaults to ‘200’.

#### **accent\_dark\_hue**

Hue value for `accent_dark`.

`accent_dark_hue` is an `OptionProperty` and defaults to ‘700’.

#### **accent\_color**

Similar to `primary_color`, but returns a value for `accent_color`.

`accent_color` is an `AliasProperty` that returns the value in `rgba` format for `accent_color`, property is readonly.

#### **accent\_light**

Similar to `primary_light`, but returns a value for `accent_light`.

`accent_light` is an `AliasProperty` that returns the value in `rgba` format for `accent_light`, property is readonly.

#### **accent\_dark**

Similar to `primary_dark`, but returns a value for `accent_dark`.

`accent_dark` is an `AliasProperty` that returns the value in `rgba` format for `accent_dark`, property is readonly.

#### **theme\_style**

App theme style.

```
from kivy.uix.screenmanager import Screen  
  
from kivymd.app import MDApp  
from kivymd.uix.button import MDRectangleFlatButton
```

(continues on next page)

(continued from previous page)

```

class MainApp(MDApp):
    def build(self):
        self.theme_cls.theme_style = "Dark" # "Light"

        screen = Screen()
        screen.add_widget(
            MDRectangleFlatButton(
                text="Hello, World",
                pos_hint={"center_x": 0.5, "center_y": 0.5},
            )
        )
    return screen

MainApp().run()

```



`theme_style` is an `OptionProperty` and defaults to '`Light`'.

### **bg\_darkest**

Similar to `bg_dark`, but the color values are a tone lower (darker) than `bg_dark`.

```

KV = '''
<Box@BoxLayout>:
    bg: 0, 0, 0, 0

    canvas:
        Color:
            rgba: root.bg
        Rectangle:
            pos: self.pos
            size: self.size

    Box:
        bg: app.theme_cls.bg_light
    Box:

```

(continues on next page)

(continued from previous page)

```

        bg: app.theme_cls.bg_normal
Box:
        bg: app.theme_cls.bg_dark
Box:
        bg: app.theme_cls.bg_darkest
"""

from kivy.lang import Builder

from kivymd.app import MDApp

class MainApp(MDApp):
    def build(self):
        self.theme_cls.theme_style = "Dark" # "Light"
        return Builder.load_string(KV)

MainApp().run()

```



`bg_darkest` is an `AliasProperty` that returns the value in `rgba` format for `bg_darkest`, property is readonly.

#### `opposite_bg_darkest`

The opposite value of color in the `bg_darkest`.

`opposite_bg_darkest` is an `AliasProperty` that returns the value in `rgba` format for `opposite_bg_darkest`, property is readonly.

#### `bg_dark`

Similar to `bg_normal`, but the color values are one tone lower (darker) than `bg_normal`.

`bg_dark` is an `AliasProperty` that returns the value in `rgba` format for `bg_dark`, property is readonly.

#### `opposite_bg_dark`

The opposite value of color in the `bg_dark`.

`opposite_bg_dark` is an `AliasProperty` that returns the value in `rgba` format for `opposite_bg_dark`, property is readonly.

**bg\_normal**

Similar to `bg_light`, but the color values are one tone lower (darker) than `bg_light`.

`bg_normal` is an `AliasProperty` that returns the value in `rgba` format for `bg_normal`, property is readonly.

**opposite\_bg\_normal**

The opposite value of color in the `bg_normal`.

`opposite_bg_normal` is an `AliasProperty` that returns the value in `rgba` format for `opposite_bg_normal`, property is readonly.

**bg\_light**

Depending on the style of the theme ('Dark' or 'Light') that the application uses, `bg_light` contains the color value in `rgba` format for the widgets background.

`bg_light` is an `AliasProperty` that returns the value in `rgba` format for `bg_light`, property is readonly.

**opposite\_bg\_light**

The opposite value of color in the `bg_light`.

`opposite_bg_light` is an `AliasProperty` that returns the value in `rgba` format for `opposite_bg_light`, property is readonly.

**divider\_color**

Color for dividing lines such as `MDSeparator`.

`divider_color` is an `AliasProperty` that returns the value in `rgba` format for `divider_color`, property is readonly.

**opposite\_divider\_color**

The opposite value of color in the `divider_color`.

`opposite_divider_color` is an `AliasProperty` that returns the value in `rgba` format for `opposite_divider_color`, property is readonly.

**text\_color**

Color of the text used in the `MDLabel`.

`text_color` is an `AliasProperty` that returns the value in `rgba` format for `text_color`, property is readonly.

**opposite\_text\_color**

The opposite value of color in the `text_color`.

`opposite_text_color` is an `AliasProperty` that returns the value in `rgba` format for `opposite_text_color`, property is readonly.

**secondary\_text\_color**

The color for the secondary text that is used in classes from the module `TwoLineListItem`.

`secondary_text_color` is an `AliasProperty` that returns the value in `rgba` format for `secondary_text_color`, property is readonly.

**opposite\_secondary\_text\_color**

The opposite value of color in the `secondary_text_color`.

`opposite_secondary_text_color` is an `AliasProperty` that returns the value in `rgba` format for `opposite_secondary_text_color`, property is readonly.

**icon\_color**

Color of the icon used in the `MDIconButton`.

`icon_color` is an `AliasProperty` that returns the value in `rgba` format for `icon_color`, property is readonly.

**opposite\_icon\_color**

The opposite value of color in the `icon_color`.

`opposite_icon_color` is an `AliasProperty` that returns the value in `rgba` format for `opposite_icon_color`, property is readonly.

**disabled\_hint\_text\_color**

Color of the disabled text used in the `MTextField`.

`disabled_hint_text_color` is an `AliasProperty` that returns the value in `rgba` format for `disabled_hint_text_color`, property is readonly.

**opposite\_disabled\_hint\_text\_color**

The opposite value of color in the `disabled_hint_text_color`.

`opposite_disabled_hint_text_color` is an `AliasProperty` that returns the value in `rgba` format for `opposite_disabled_hint_text_color`, property is readonly.

**error\_color**

Color of the error text used in the `MTextField`.

`error_color` is an `AliasProperty` that returns the value in `rgba` format for `error_color`, property is readonly.

**ripple\_color**

Color of ripple effects.

`ripple_color` is an `AliasProperty` that returns the value in `rgba` format for `ripple_color`, property is readonly.

**device\_orientation**

Device orientation.

`device_orientation` is an `StringProperty`.

**standard\_increment**

Value of standard increment.

`standard_increment` is an `AliasProperty` that returns the value in `rgba` format for `standard_increment`, property is readonly.

**horizontal\_margins**

Value of horizontal margins.

`horizontal_margins` is an `AliasProperty` that returns the value in `rgba` format for `horizontal_margins`, property is readonly.

**set\_clearcolor****font\_styles**

Data of default font styles.

Add custom font:

```
KV = '''
Screen:

    MDLabel:
        text: "JetBrainsMono"
        halign: "center"
'''
```

(continues on next page)

(continued from previous page)

```

        font_style: "JetBrainsMono"
    '''

from kivy.core.text import LabelBase

from kivy.lang import Builder

from kivymd.app import MDApp
from kivymd.font_definitions import theme_font_styles


class MainApp(MDApp):
    def build(self):
        LabelBase.register(
            name="JetBrainsMono",
            fn_regular="JetBrainsMono-Regular.ttf")

        theme_font_styles.append('JetBrainsMono')
        self.theme_cls.font_styles["JetBrainsMono"] = [
            "JetBrainsMono",
            16,
            False,
            0.15,
        ]
    return Builder.load_string(KV)

MainApp().run()

```



`font_styles` is an `DictProperty`.

```

on_theme_style(self, instance, value)
set_clearcolor_by_theme_style(self, theme_style)
class kivymd.theming.ThemableBehavior(**kwargs)

theme_cls
Instance of ThemeManager class.

```

*theme\_cls* is an ObjectProperty.

**device\_ios**

True if device is iOS.

*device\_ios* is an BooleanProperty.

**opposite\_colors**

## 2.2.2 Material App

This module contains *MDApp* class that is inherited from *App*. *MDApp* has some properties needed for KivyMD library (like *theme\_cls*).

You can turn on the monitor displaying the current FPS value in your application:

```
KV = '''
Screen:

    MDLabel:
        text: "Hello, World!"
        halign: "center"
    ...

from kivy.lang import Builder

from kivymd.app import MDApp

class MainApp(MDApp):
    def build(self):
        return Builder.load_string(KV)

    def on_start(self):
        self.fps_monitor_start()

MainApp().run()
```



## API - kivymd.app

```
class kivymd.app.MDApp(**kwargs)
```

Application class, see module documentation for more information.

### Events

***on\_start***: Fired when the application is being started (before the `runTouchApp()` call).

***on\_stop***: Fired when the application stops.

***on\_pause***: Fired when the application is paused by the OS.

***on\_resume***: Fired when the application is resumed from pause by the OS. Beware: you have no guarantee that this event will be fired after the `on_pause` event has been called.

Changed in version 1.7.0: Parameter `kv_file` added.

Changed in version 1.8.0: Parameters `kv_file` and `kv_directory` are now properties of App.

**theme\_cls**

Instance of ThemeManager class.

**Warning:** The `theme_cls` attribute is already available in a class that is inherited from the `MDApp` class. The following code will result in an error!

```
class MainApp(MDApp):
    theme_cls = ThemeManager()
    theme_cls.primary_palette = "Teal"
```

**Note:** Correctly do as shown below!

```
class MainApp(MDApp):
    def build(self):
        self.theme_cls.primary_palette = "Teal"
```

`theme_cls` is an `ObjectProperty`.

## 2.2.3 Color Definitions

### See also:

Material Design spec, The color system

Material colors palette to use in `kivymd.theming.ThemeManager.colors` is a dict-in-dict where the first key is a value from `palette` and the second key is a value from `hue`. Color is a hex value, a string of 6 characters (0-9, A-F) written in uppercase.

For example, `colors["Red"]["900"]` is "B71C1C".

### API - kivymd.color\_definitions

#### kivymd.color\_definitions.colors

Color palette. Taken from 2014 Material Design color palettes.

To demonstrate the shades of the palette, you can run the following code:

```
from kivy.lang import Builder
from kivy.uix.boxlayout import BoxLayout
from kivy.utils import get_color_from_hex
from kivy.properties import ListProperty, StringProperty

from kivymd.color_definitions import colors
from kivymd.uix.tab import MDTabsBase

demo = '''
<Root@BoxLayout>
    orientation: 'vertical'

    MDToolbar:
        title: app.title
```

(continues on next page)

(continued from previous page)

```

MDTabs:
    id: android_tabs
    on_tab_switch: app.on_tab_switch(*args)
    size_hint_y: None
    height: "48dp"
    tab_indicator_anim: False

ScrollView:

    MDList:
        id: box


<ItemColor>:
    size_hint_y: None
    height: "42dp"

    canvas:
        Color:
            rgba: root.color
        Rectangle:
            size: self.size
            pos: self.pos

    MDLabel:
        text: root.text
        halign: "center"


<Tab>:
''''

from kivy.factory import Factory
from kivymd.app import MDApp


class Tab(BoxLayout, MDTabsBase):
    pass


class ItemColor(BoxLayout):
    text = StringProperty()
    color = ListProperty()


class Palette(MDApp):
    title = "Colors definitions"

    def build(self):
        Builder.load_string(demo)
        self.screen = Factory.Root()

        for name_tab in colors.keys():
            tab = Tab(text=name_tab)
            self.screen.ids.android_tabs.add_widget(tab)

    return self.screen

```

(continues on next page)

(continued from previous page)

```
def on_tab_switch(self, instance_tabs, instance_tab, instance_tabs_label, tab_text):
    self.screen.ids.box.clear_widgets()
    for value_color in colors[tab_text]:
        self.screen.ids.box.add_widget(
            ItemColor(
                color=get_color_from_hex(colors[tab_text][value_color]),
                text=value_color,
            )
        )

def on_start(self):
    self.on_tab_switch(
        None,
        None,
        None,
        self.screen.ids.android_tabs.ids.layout.children[-1].text,
    )

Palette().run()
```

kivymd.color\_definitions.palette = ['Red', 'Pink', 'Purple', 'DeepPurple', 'Indigo', 'Blue']  
Valid values for color palette selecting.

kivymd.color\_definitions.hue = ['50', '100', '200', '300', '400', '500', '600', '700', '800']  
Valid values for color hue selecting.

kivymd.color\_definitions.light\_colors  
Which colors are light. Other are dark.

kivymd.color\_definitions.text\_colors  
Text colors generated from [light\\_colors](#). “000000” for light and “FFFFFF” for dark.

How to generate text\_colors dict

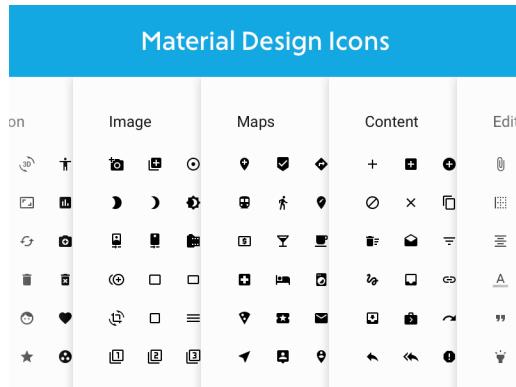
```
text_colors = {}
for p in palette:
    text_colors[p] = {}
    for h in hue:
        if h in light_colors[p]:
            text_colors[p][h] = "000000"
        else:
            text_colors[p][h] = "FFFFFF"
```

kivymd.color\_definitions.theme\_colors = ['Primary', 'Secondary', 'Background', 'Surface', 'Error']  
Valid theme colors.

## 2.2.4 Icon Definitions

See also:

[Material Design Icons](#)



List of icons from [materialdesignicons.com](#). These expanded material design icons are maintained by Austin Andrews (Templarian on Github).

LAST UPDATED: Version 5.5.55

**To preview the icons and their names, you can use the following application:**

```
from kivy.lang import Builder
from kivy.properties import StringProperty
from kivy.uix.screenmanager import Screen

from kivymd.icon_definitions import md_icons
from kivymd.app import MDApp
from kivymd.list import OneLineIconListItem

Builder.load_string(
    '''
#:import images_path kivymd.images_path

<CustomOneLineIconListItem>:

    IconLeftWidget:
        icon: root.icon

<PreviousMDIcons>:

    BoxLayout:
        orientation: 'vertical'
        spacing: dp(10)
        padding: dp(20)

    BoxLayout:
        size_hint_y: None
        height: self.minimum_height
    
```

(continues on next page)

(continued from previous page)

```

MDIconButton:
    icon: 'magnify'

MDTextField:
    id: search_field
    hint_text: 'Search icon'
    on_text: root.set_list_md_icons(self.text, True)

RecycleView:
    id: rv
    key_viewclass: 'viewclass'
    key_size: 'height'

RecycleBoxLayout:
    padding: dp(10)
    default_size: None, dp(48)
    default_size_hint: 1, None
    size_hint_y: None
    height: self.minimum_height
    orientation: 'vertical'
    ...
)

class CustomOneLineIconListItem(OneLineIconListItem):
    icon = StringProperty()

class PreviousMDIcons(Screen):

    def set_list_md_icons(self, text="", search=False):
        '''Builds a list of icons for the screen MDIcons.'''
        def add_icon_item(name_icon):
            self.ids.rv.data.append(
                {
                    "viewclass": "CustomOneLineIconListItem",
                    "icon": name_icon,
                    "text": name_icon,
                    "callback": lambda x: x,
                }
            )
        self.ids.rv.data = []
        for name_icon in md_icons.keys():
            if search:
                if text in name_icon:
                    add_icon_item(name_icon)
            else:
                add_icon_item(name_icon)

class MainApp(MDApp):
    def __init__(self, **kwargs):
        super().__init__(**kwargs)
        self.screen = PreviousMDIcons()

```

(continues on next page)

(continued from previous page)

```
def build(self):
    return self.screen

def on_start(self):
    self.screen.set_list_md_icons()

MainApp().run()
```

## API - kivymd.icon\_definitions

`kivymd.icon_definitions.md_icons`

## 2.2.5 Font Definitions

#### **See also:**

## Material Design spec, The type system

## API - kivymd.font\_definitions

```
kivymd.font_definitions.fonts  
kivymd.font_definitions.theme
```

Scale Category	Typeface	Font	Size	Case	Letter spacing
H1	Roboto	Light	96	Sentence	-1.5
H2	Roboto	Light	60	Sentence	-0.5
H3	Roboto	Regular	48	Sentence	0
H4	Roboto	Regular	34	Sentence	0.25
H5	Roboto	Regular	24	Sentence	0
H6	Roboto	Medium	20	Sentence	0.15
Subtitle 1	Roboto	Regular	16	Sentence	0.15
Subtitle 2	Roboto	Medium	14	Sentence	0.1
Body 1	Roboto	Regular	16	Sentence	0.5
Body 2	Roboto	Regular	14	Sentence	0.25
BUTTON	Roboto	Medium	14	All caps	1.25
Caption	Roboto	Regular	12	Sentence	0.4
OVERLINE	Roboto	Regular	10	All caps	1.5

## 2.3 Components

### 2.3.1 Spinner

Circular progress indicator in Google's Material Design.

#### Usage

```
from kivy.lang import Builder
from kivymd.app import MDApp
KV = '''
Screen:
```

(continues on next page)

(continued from previous page)

```

MDSpinner:
    size_hint: None, None
    size: dp(46), dp(46)
    pos_hint: {'center_x': .5, 'center_y': .5}
    active: True if check.active else False

MDCheckbox:
    id: check
    size_hint: None, None
    size: dp(48), dp(48)
    pos_hint: {'center_x': .5, 'center_y': .4}
    active: True

'''

class Test(MDApp):
    def build(self):
        return Builder.load_string(KV)

Test().run()

```

## Spinner palette

```

MDSpinner:
    # The number of color values can be any.
    palette:
        [0.28627450980392155, 0.8431372549019608, 0.596078431372549, 1],
        [0.3568627450980392, 0.3215686274509804, 0.8666666666666667, 1],
        [0.8862745098039215, 0.36470588235294116, 0.592156862745098, 1],
        [0.8784313725490196, 0.9058823529411765, 0.40784313725490196, 1]

```

```

MDSpinner(
    size_hint=(None, None),
    size=(dp(46), dp(46)),
    pos_hint={'center_x': .5, 'center_y': .5},
    active=True,
    palette=[
        [0.28627450980392155, 0.8431372549019608, 0.596078431372549, 1],
        [0.3568627450980392, 0.3215686274509804, 0.8666666666666667, 1],
        [0.8862745098039215, 0.36470588235294116, 0.592156862745098, 1],
        [0.8784313725490196, 0.9058823529411765, 0.40784313725490196, 1],
    ]
)

```

### API - kivymd.uix.spinner

```
class kivymd.uix.spinner.MDSpinner(**kwargs)
```

*MDSpinner* is an implementation of the circular progress indicator in *Google's Material Design*.

It can be used either as an indeterminate indicator that loops while the user waits for something to happen, or as a determinate indicator.

Set *determinate* to **True** to activate determinate mode, and *determinate\_time* to set the duration of the animation.

#### **determinate**

Determinate value.

*determinate* is a `BooleanProperty` and defaults to *False*.

#### **determinate\_time**

Determinate time value.

*determinate\_time* is a `NumericProperty` and defaults to 2.

#### **active**

Use *active* to start or stop the spinner.

*active* is a `BooleanProperty` and defaults to *True*.

#### **color**

Spinner color.

*color* is a `ListProperty` and defaults to `self.theme_cls.primary_color`.

#### **palette**

A set of colors. Changes with each completed spinner cycle.

*palette* is a `ListProperty` and defaults to `[]`.

`on__rotation_angle(self, *args)`

`on_palette(self, instance, value)`

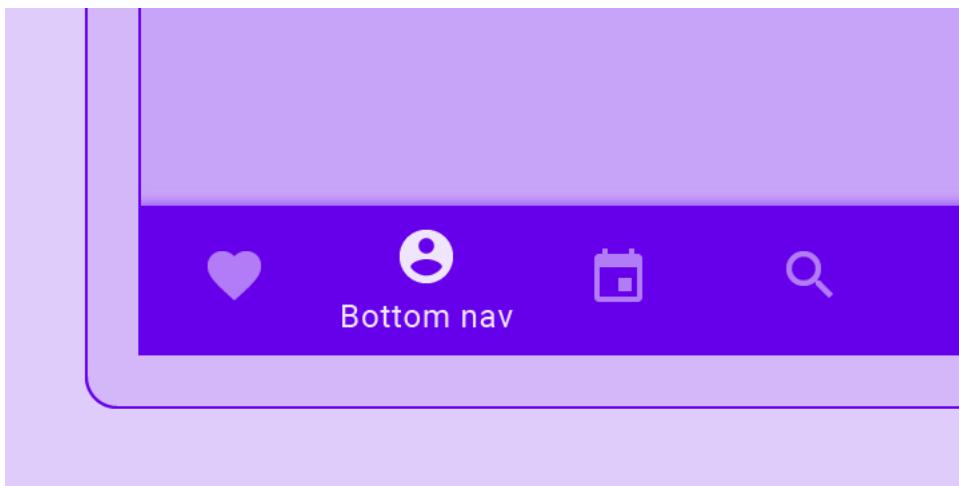
`on_active(self, *args)`

## 2.3.2 Bottom Navigation

### See also:

Material Design spec, Bottom navigation

Bottom navigation bars allow movement between primary destinations in an app:



## Usage

```
<Root>>:
    MDBottomNavigation:
        MDBottomNavigationItem:
            name: "screen 1"
            YourContent:
        MDBottomNavigationItem:
            name: "screen 2"
            YourContent:
        MDBottomNavigationItem:
            name: "screen 3"
            YourContent:
```

For ease of understanding, this code works like this:

```
<Root>>:
    ScreenManager:
        Screen:
            name: "screen 1"
            YourContent:
        Screen:
            name: "screen 2"
            YourContent:
```

(continues on next page)

(continued from previous page)

```
Screen:  
    name: "screen 3"
```

```
YourContent:
```

## Example

```
from kivymd.app import MDApp  
from kivy.lang import Builder  
  
class Test(MDApp):  
  
    def build(self):  
        self.theme_cls.primary_palette = "Gray"  
        return Builder.load_string(  
            '''  
BoxLayout:  
    orientation:'vertical'  
  
MDToolbar:  
    title: 'Bottom navigation'  
    md_bg_color: .2, .2, .2, 1  
    specific_text_color: 1, 1, 1, 1  
  
MDBottomNavigation:  
    panel_color: .2, .2, .2, 1  
  
    MDBottomNavigationItem:  
        name: 'screen 1'  
        text: 'Python'  
        icon: 'language-python'  
  
        MDLabel:  
            text: 'Python'  
            halign: 'center'  
  
    MDBottomNavigationItem:  
        name: 'screen 2'  
        text: 'C++'  
        icon: 'language-cpp'  
  
        MDLabel:  
            text: 'I programming of C++'  
            halign: 'center'  
  
    MDBottomNavigationItem:  
        name: 'screen 3'  
        text: 'JS'  
        icon: 'language-javascript'  
  
        MDLabel:  
            text: 'JS'
```

(continues on next page)

(continued from previous page)

```

        halign: 'center'
    ...
)

Test().run()

```

**MDBottomNavigationItem** provides the following events for use:

```

__events__ = (
    "on_tab_touch_down",
    "on_tab_touch_move",
    "on_tab_touch_up",
    "on_tab_press",
    "on_tab_release",
)

```

See also:

See \_\_events\_\_

**Root:**

**MDBottomNavigation:**

```

MDBottomNavigationItem:
    on_tab_touch_down: print("on_tab_touch_down")
    on_tab_touch_move: print("on_tab_touch_move")
    on_tab_touch_up: print("on_tab_touch_up")
    on_tab_press: print("on_tab_press")
    on_tab_release: print("on_tab_release")

```

**YourContent:**

## How to automatically switch a tab?

Use method `switch_tab` which takes as argument the name of the tab you want to switch to.

## How to change icon color?

```

MDBottomNavigation:
    text_color_active: 1, 0, 1, 1

```



PYTHON



C++



JS

```

MDBottomNavigation:
    text_color_normal: 1, 0, 1, 1

```



PYTHON



C++



JS

**See also:**[See Tab auto switch example](#)[See full example](#)**API - kivymd.uix.bottomnavigation****class** kivymd.uix.bottomnavigation.**MDTab**(\*\*kwargs)

A tab is simply a screen with meta information that defines the content that goes in the tab header.

**text**

Tab header text.

*text* is an `StringProperty` and defaults to ''.**icon**

Tab header icon.

*icon* is an `StringProperty` and defaults to '*checkbox-blank-circle*'.**on\_tab\_touch\_down**(self, \*args)**on\_tab\_touch\_move**(self, \*args)**on\_tab\_touch\_up**(self, \*args)**on\_tab\_press**(self, \*args)**on\_tab\_release**(self, \*args)**class** kivymd.uix.bottomnavigation.**MDBottomNavigationItem**(\*\*kwargs)

A tab is simply a screen with meta information that defines the content that goes in the tab header.

**header***header* is an `MDBottomNavigationHeader` and defaults to *None*.**on\_tab\_press**(self, \*args)**on\_leave**(self, \*args)**class** kivymd.uix.bottomnavigation.**TabbedPanelBase**(\*\*kwargs)

A class that contains all variables a TabPannel must have. It is here so I (zingballyhoo) don't get mad about the TabbedPanels not being DRY.

**current**

Current tab name.

*current* is an `StringProperty` and defaults to *None*.**previous\_tab***previous\_tab* is an `MDTab` and defaults to *None*.**panel\_color**

Panel color of bottom navigation.

*panel\_color* is an `ListProperty` and defaults to `[]`.**tabs**

```
class kivymd.uix.bottomnavigation.MDBottomNavigation (**kwargs)
A bottom navigation that is implemented by delegating all items to a ScreenManager.

first_widget
first_widget is an MDBottomNavigationItem and defaults to None.

tab_header
tab_header is an MDBottomNavigationHeader and defaults to None.

text_color_normal
Text color of the label when it is not selected.

text_color_normal is an ListProperty and defaults to [1, 1, 1, 1].

text_color_active
Text color of the label when it is selected.

text_color_active is an ListProperty and defaults to [1, 1, 1, 1].

on_panel_color (self, instance, value)
on_text_color_normal (self, instance, value)
on_text_color_active (self, instance, value)
switch_tab (self, name_tab)
Switching the tab by name.

refresh_tabs (self)
Refresh all tabs.

on_resize (self, instance=None, width=None, do_again=True)
Called when the application window is resized.

add_widget (self, widget, **kwargs)
Add a new widget as a child of this widget.
```

### Parameters

**widget: Widget** Widget to add to our list of children.

**index: int, defaults to 0** Index to insert the widget in the list. Notice that the default of 0 means the widget is inserted at the beginning of the list and will thus be drawn on top of other sibling widgets. For a full discussion of the index and widget hierarchy, please see the [Widgets Programming Guide](#).

New in version 1.0.5.

**canvas: str, defaults to None** Canvas to add widget's canvas to. Can be 'before', 'after' or None for the default canvas.

New in version 1.9.0.

```
>>> from kivy.uix.button import Button
>>> from kivy.uix.slider import Slider
>>> root = Widget()
>>> root.add_widget(Button())
>>> slider = Slider()
>>> root.add_widget(slider)
```

**remove\_widget** (self, widget)
Remove a widget from the children of this widget.

### Parameters

**widget:** `Widget` Widget to remove from our children list.

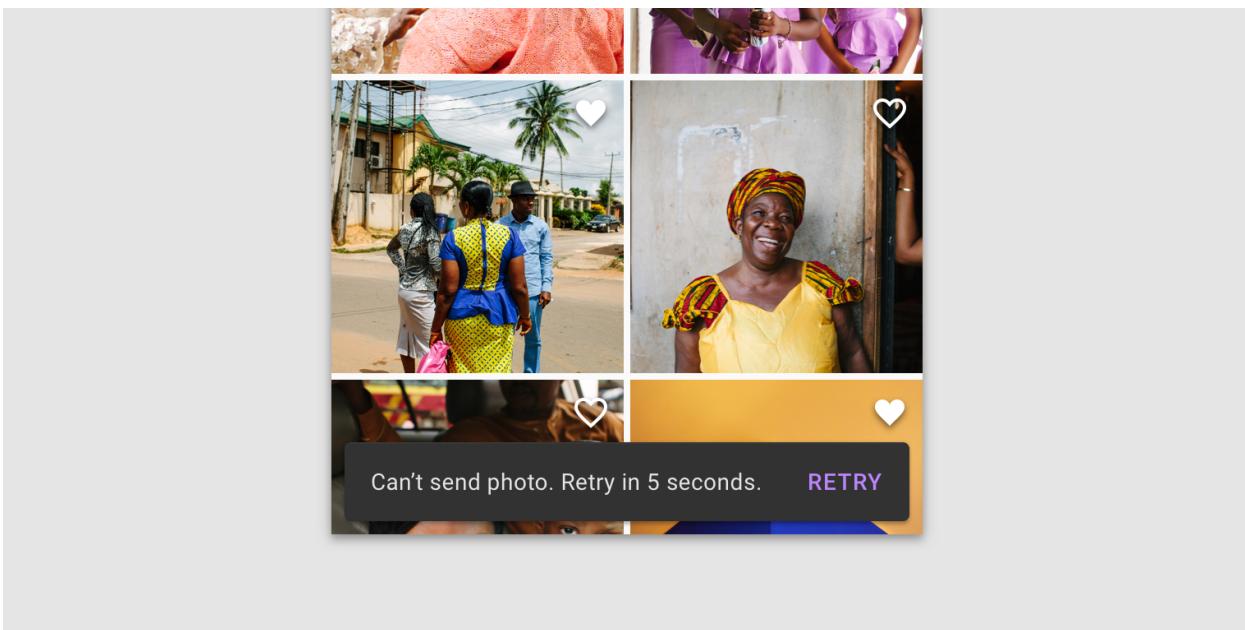
```
>>> from kivy.uix.button import Button
>>> root = Widget()
>>> button = Button()
>>> root.add_widget(button)
>>> root.remove_widget(button)
```

### 2.3.3 Snackbar

See also:

Material Design spec, Snackbars

Snackbars provide brief messages about app processes at the bottom of the screen.



### Usage

```
from kivy.lang import Builder

from kivymd.app import MDApp

KV = '''
#:import Snackbar kivymd.uix.snackbar.Snackbar

Screen:

    MDRaisedButton:
        text: "Create simple snackbar"
        on_release: Snackbar(text="This is a snackbar!").show()
```

(continues on next page)

(continued from previous page)

```

    pos_hint: {"center_x": .5, "center_y": .5}
    ...

class Test(MDApp):
    def build(self):
        return Builder.load_string(KV)

Test().run()

```

## Usage with padding

```
Snackbar(text="This is a snackbar!", padding="20dp").show()
```

## Usage with button

```

Snackbar(
    text="This is a snackbar",
    button_text="BUTTON",
    button_callback=app.callback
).show()

```

## Using a button with custom color

```

Snackbar(
    text="This is a snackbar!",
    padding="20dp",
    button_text="ACTION",
    button_color=(1, 0, 1, 1)
).show()

```

## Custom usage

```

from kivy.lang import Builder
from kivy.animation import Animation
from kivy.clock import Clock
from kivy.metrics import dp

from kivymd.app import MDApp
from kivymd.uix.snackbar import Snackbar

```

(continues on next page)

(continued from previous page)

```
KV = '''
Screen:

    MDFloatingActionButton:
        id: button
        x: root.width - self.width - dp(10)
        y: dp(10)
        on_release: app.snackbar_show()
'''


class Test(MDApp):
    def __init__(self, **kwargs):
        super().__init__(**kwargs)
        self.screen = Builder.load_string(KV)
        self.snackbar = None
        self._interval = 0

    def build(self):
        return self.screen

    def wait_interval(self, interval):
        self._interval += interval
        if self._interval > self.snackbar.duration:
            anim = Animation(y=dp(10), d=.2)
            anim.start(self.screen.ids.button)
            Clock.unschedule(self.wait_interval)
            self._interval = 0
            self.snackbar = None

    def snackbar_show(self):
        if not self.snackbar:
            self.snackbar = Snackbar(text="This is a snackbar!")
            self.snackbar.show()
            anim = Animation(y=dp(72), d=.2)
            anim.bind(on_complete=lambda *args: Clock.schedule_interval(
                self.wait_interval, 0))
            anim.start(self.screen.ids.button)

Test().run()
```

## Custom Snackbar

```

from kivy.lang import Builder
from kivy.properties import StringProperty

from kivymd.app import MDApp
from kivymd.uix.snackbar import Snackbar

KV = '''
<-Snackbar>

    MDCard:
        id: box
        size_hint_y: None
        height: dp(58)
        spacing: dp(5)
        padding: dp(10)
        y: -self.height
        x: root.padding
        md_bg_color: get_color_from_hex('323232')
        radius: (5, 5, 5, 5) if root.padding else (0, 0, 0, 0)
        elevation: 11 if root.padding else 0

        MDIconButton:
            pos_hint: {'center_y': .5}
            icon: root.icon
            opposite_colors: True

        MDLabel:
            id: text_bar
            size_hint_y: None
            height: self.texture_size[1]
            text: root.text
            font_size: root.font_size
            theme_text_color: 'Custom'
            text_color: get_color_from_hex('ffffff')
            shorten: True
            shorten_from: 'right'
            pos_hint: {'center_y': .5}

Screen:

    MDRaisedButton:
        text: "SHOW"
        pos_hint: {"center_x": .5, "center_y": .45}
        on_press: app.show()
    '''

class CustomSnackbar(Snackbar):
    icon = StringProperty()

class Test(MDApp):
    def build(self):
        return Builder.load_string(KV)

```

(continues on next page)

(continued from previous page)

```
def show(self):
    CustomSnackbar(
        text="This is a snackbar!",
        icon="information",
        padding="20dp",
        button_text="ACTION",
        button_color=(1, 0, 1, 1)
    ).show()

Test().run()
```

**API - kivymd.uix.snackbar****class kivymd.uix.snackbar.Snackbar(\*\*kwargs)**

Float layout class. See module documentation for more information.

**text**

The text that will appear in the snackbar.

*text* is a `StringProperty` and defaults to ''.**font\_size**

The font size of the text that will appear in the snackbar.

*font\_size* is a `NumericProperty` and defaults to '15sp'.**button\_text**

The text that will appear in the snackbar's button.

**Note:** If this variable is None, the snackbar will have no button.*button\_text* is a `StringProperty` and defaults to ''.**button\_callback**

The callback that will be triggered when the snackbar's button is pressed.

**Note:** If this variable is None, the snackbar will have no button.*button\_callback* is a `ObjectProperty` and defaults to *None*.**button\_color**

Button color.

*button\_color* is a `ListProperty` and defaults to `[]`.**duration**

The amount of time that the snackbar will stay on screen for.

*duration* is a `NumericProperty` and defaults to 3.

**padding**

Snackbar padding.

*padding* is a `NumericProperty` and defaults to ‘0dp’.

**show(self)**

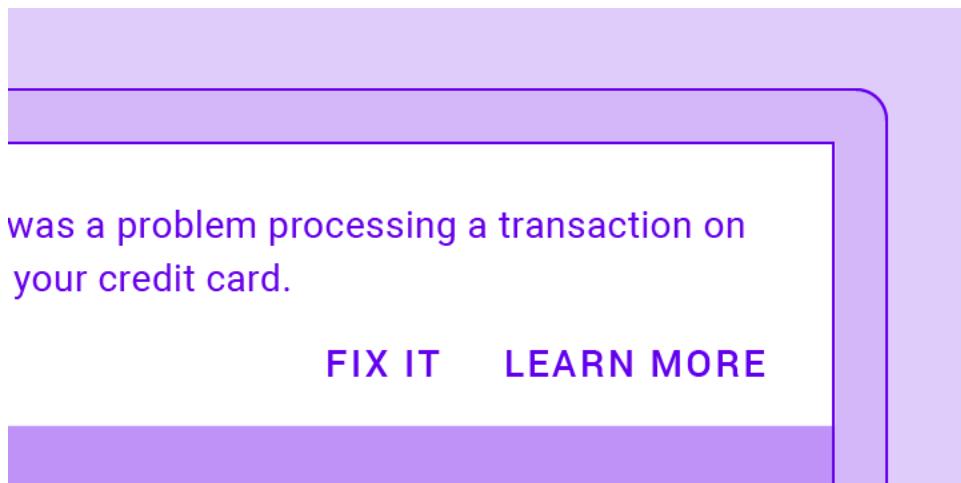
Show the snackbar.

### 2.3.4 Banner

#### See also:

Material Design spec, Banner

A banner displays a prominent message and related optional actions.



#### Usage

```
from kivy.lang import Builder
from kivy.factory import Factory

from kivymd.app import MDApp

Builder.load_string('''
<ExampleBanner@Screen>

    MDBanner:
        id: banner
        text: ["One line string text example without actions."]
        # The widget that is under the banner.
        # It will be shifted down to the height of the banner.
        over_widget: screen
        vertical_pad: toolbar.height

    MDToolbar:
        id: toolbar
        title: "Example Banners"
```

(continues on next page)

(continued from previous page)

```

        elevation: 10
        pos_hint: {'top': 1}

BoxLayout:
    id: screen
    orientation: "vertical"
    size_hint_y: None
    height: Window.height - toolbar.height

    OneLineListItem:
        text: "Banner without actions"
        on_release: banner.show()

    Widget:
''')

class Test(MDApp):
    def build(self):
        return Factory.ExampleBanner()

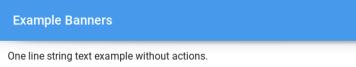
Test().run()

```

## Banner type.

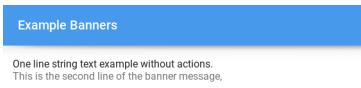
By default, the banner is of the type 'one-line':

```
MDBanner:
    text: ["One line string text example without actions."]
```



To use a two-line banner, specify the 'two-line' `MDBanner.type` for the banner and pass the list of two lines to the `MDBanner.text` parameter:

```
MDBanner:
    type: "two-line"
    text:
        ["One line string text example without actions.", "This is the second line of the banner message."]
        ↪
```



Similarly, create a three-line banner:

```
MDBanner:
    type: "three-line"
    text:
        ["One line string text example without actions.", "This is the second line of the banner message."]
        ↪
        ["and this is the third line of the banner message."]
        ↪
```

(continues on next page)

(continued from previous page)

**Example Banners**

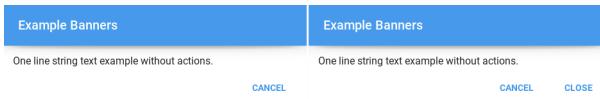
One line string text example without actions.  
 This is the second line of the banner message,  
 and this is the third line of the banner message.

To add buttons to any type of banner, use the `MDBanner.left_action` and `MDBanner.right_action` parameters, which should take a list [‘Button name’, function]:

```
MDBanner:
text: ["One line string text example without actions."]
left_action: ["CANCEL", lambda x: None]
```

Or two buttons:

```
MDBanner:
text: ["One line string text example without actions."]
left_action: ["CANCEL", lambda x: None]
right_action: ["CLOSE", lambda x: None]
```



If you want to use the icon on the left in the banner, add the prefix ‘-icon’ to the banner type:

```
MDBanner:
type: "one-line-icon"
icon: f"{images_path}/kivymd.png"
text: ["One line string text example without actions."]
```



**Note:** See full example

## API - kivymd.uix.banner

```
class kivymd.uix.banner.MDBanner(**kwargs)
Widget class. See module documentation for more information.
```

### Events

**on\_touch\_down: (touch, )** Fired when a new touch event occurs. *touch* is the touch object.

**on\_touch\_move: (touch, )** Fired when an existing touch moves. *touch* is the touch object.

**on\_touch\_up: (touch, )** Fired when an existing touch disappears. *touch* is the touch object.

**on\_kv\_post: (base\_widget, )** Fired after all the kv rules associated with the widget and all other widgets that are in any of those rules have had all their kv rules applied. *base\_widget* is the base-most widget whose instantiation triggered the kv rules (i.e. the widget instantiated from Python, e.g. `MyWidget()`).

Changed in version 1.11.0.

**Warning:** Adding a `__del__` method to a class derived from Widget with Python prior to 3.4 will disable automatic garbage collection for instances of that class. This is because the Widget class creates reference cycles, thereby [preventing garbage collection](#).

Changed in version 1.0.9: Everything related to event properties has been moved to the `EventDispatcher`. Event properties can now be used when contructing a simple class without subclassing `Widget`.

Changed in version 1.5.0: The constructor now accepts `on_*` arguments to automatically bind callbacks to properties or events, as in the Kv language.

#### **vertical\_pad**

Indent the banner at the top of the screen.

`vertical_pad` is an `NumericProperty` and defaults to `dp(68)`.

#### **opening\_transition**

The name of the animation transition.

`opening_transition` is an `StringProperty` and defaults to `'in_quad'`.

#### **icon**

Icon banner.

`icon` is an `StringProperty` and defaults to `'data/logo/kivy-icon-128.png'`.

#### **over\_widget**

The widget that is under the banner. It will be shifted down to the height of the banner.

`over_widget` is an `ObjectProperty` and defaults to `None`.

#### **text**

List of lines for banner text. Must contain no more than three lines for a `'one-line'`, `'two-line'` and `'three-line'` banner, respectively.

`text` is an `ListProperty` and defaults to `[]`.

#### **left\_action**

The action of banner.

To add one action, make a list `[name_action, callback]` where `'name_action'` is a string that corresponds to an action name and `callback` is the function called on a touch release event.

`left_action` is an `ListProperty` and defaults to `[]`.

#### **right\_action**

Works the same way as `left_action`.

`right_action` is an `ListProperty` and defaults to `[]`.

#### **type**

Banner type. . Available options are: (`"one-line"`, `"two-line"`, `"three-line"`, `"one-line-icon"`, `"two-line-icon"`, `"three-line-icon"`).

`type` is an `OptionProperty` and defaults to `'one-line'`.

`add_actions_buttons(self, box, data)`

`set_left_action(self)`

`set_right_action(self)`

`set_type_banner(self)`

`add_banner_to_container(self)`

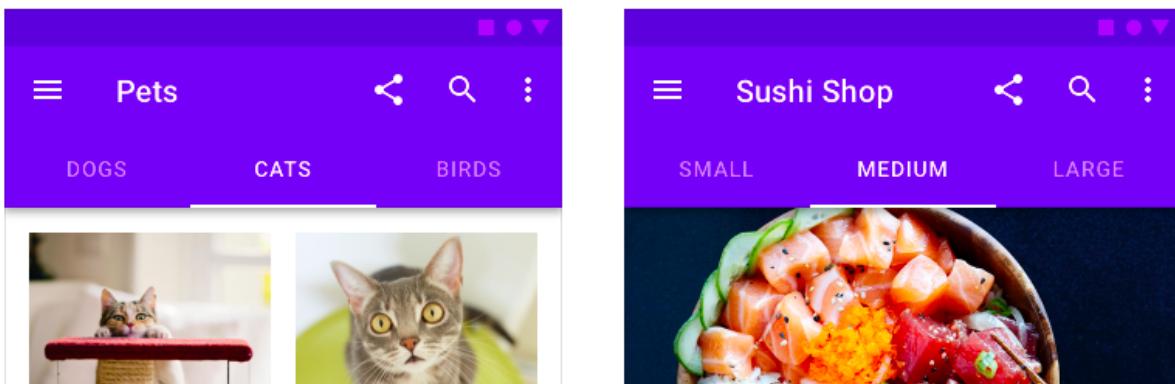
```
show(self)
animation_display_banner(self, i)
hide(self)
```

### 2.3.5 Tabs

See also:

Material Design spec, Tabs

Tabs organize content across different screens, data sets, and other interactions.




---

**Note:** Module provides tabs in the form of icons or text.

---

### Usage

To create a tab, you must create a new class that inherits from the `MDTabsBase` class and the `Kivy` container, in which you will create content for the tab.

```
class Tab(FloatLayout, MDTabsBase):
    '''Class implementing content for a tab.'''
```

```
<Tab>:

    MDLabel:
        text: "Content"
        pos_hint: {"center_x": .5, "center_y": .5}
```

Tabs must be placed in the `MDTabs` container:

```
Root:

    MDTabs:

        Tab:
            text: "Tab 1"
```

(continues on next page)

(continued from previous page)

```
Tab:
    text: "Tab 1"
```

```
...]
```

### Example with tab icon

```
from kivy.lang import Builder
from kivy.uix.floatlayout import FloatLayout

from kivymd.app import MDApp
from kivymd.uix.tab import MDTabsBase
from kivymd.icon_definitions import md_icons

KV = '''
BoxLayout:
    orientation: "vertical"

    MDToolbar:
        title: "Example Tabs"

    MDTabs:
        id: tabs
        on_tab_switch: app.on_tab_switch(*args)

<Tab>:

    MDIconButton:
        id: icon
        icon: app.icons[0]
        user_font_size: "48sp"
        pos_hint: {"center_x": .5, "center_y": .5}
'''


class Tab(FloatLayout, MDTabsBase):
    '''Class implementing content for a tab.'''


class Example(MDApp):
    icons = list(md_icons.keys())[15:30]

    def build(self):
        return Builder.load_string(KV)

    def on_start(self):
        for name_tab in self.icons:
            self.root.ids.tabs.add_widget(Tab(text=name_tab))

    def on_tab_switch(
            self, instance_tabs, instance_tab, instance_tab_label, tab_text
    ):

```

(continues on next page)

(continued from previous page)

```
'''Called when switching tabs.

:type instance_tabs: <kivymd.uix.tab.MDTabs object>;
:param instance_tab: <__main__.Tab object>;
:param instance_tab_label: <kivymd.uix.tab.MDTabsLabel object>;
:param tab_text: text or name icon of tab;
'''

count_icon = [k for k, v in md_icons.items() if v == tab_text]
instance_tab.ids.icon.icon = count_icon[0]
```

Example().run()

## Example with tab text

**Note:** The `MDTabsBase` class has an icon parameter and, by default, tries to find the name of the icon in the file `kivymd/icon_definitions.py`. If the name of the icon is not found, then the name of the tab will be plain text, if found, the tab will look like the corresponding icon.

```
from kivy.lang import Builder
from kivy.uix.floatlayout import FloatLayout

from kivymd.app import MDApp
from kivymd.uix.tab import MDTabsBase

KV = '''
BoxLayout:
    orientation: "vertical"

    MDToolbar:
        title: "Example Tabs"

    MDTabs:
        id: tabs
        on_tab_switch: app.on_tab_switch(*args)

<Tab>:

    MDLabel:
        id: label
        text: "Tab 0"
        halign: "center"
'''


class Tab(FloatLayout, MDTabsBase):
    '''Class implementing content for a tab.'''

```

(continues on next page)

(continued from previous page)

```

class Example(MDApp):
    def build(self):
        return Builder.load_string(KV)

    def on_start(self):
        for i in range(20):
            self.root.ids.tabs.add_widget(Tab(text=f"Tab {i}"))

    def on_tab_switch(
        self, instance_tabs, instance_tab, instance_tab_label, tab_text
    ):
        '''Called when switching tabs.

        :type instance_tabs: <kivymd.uix.tab.MDTabs object>;
        :param instance_tab: <__main__.Tab object>;
        :param instance_tab_label: <kivymd.uix.tab.MDTabsLabel object>;
        :param tab_text: text or name icon of tab;
        '''

        instance_tab.ids.label.text = tab_text

Example().run()

```

## Example with tab icon and text

```

from kivy.lang import Builder
from kivy.uix.floatlayout import FloatLayout

from kivymd.app import MDApp
from kivymd.uix.tab import MDTabsBase
from kivymd.font_definitions import fonts
from kivymd.icon_definitions import md_icons

KV = '''
BoxLayout:
    orientation: "vertical"

    MDToolbar:
        title: "Example Tabs"

    MDTabs:
        id: tabs
'''


class Tab(FloatLayout, MDTabsBase):
    pass


class Example(MDApp):
    def build(self):
        return Builder.load_string(KV)

```

(continues on next page)

(continued from previous page)

```
def on_start(self):
    for name_tab in list(md_icons.keys())[15:30]:
        self.root.ids.tabs.add_widget(
            Tab(
                text=f"[size=20][font={fonts[-1]['fn_regular']}]{md_icons[name_
→tab]}[/size][/font] {name_tab}"
            )
        )

Example().run()
```

## Example Tabs



## Dynamic tab management

```
from kivy.lang import Builder
from kivy.uix.scrollview import ScrollView

from kivymd.app import MDApp
from kivymd.uix.tab import MDTabsBase

KV = '''
BoxLayout:
    orientation: "vertical"

    MDToolbar:
        title: "Example Tabs"

    MDTabs:
        id: tabs

<Tab>:

    MDList:

        MDBBoxLayout:
            adaptive_height: True

            MDFlatButton:
                text: "ADD TAB"
                on_release: app.add_tab()

            MDFlatButton:
                text: "REMOVE LAST TAB"
                on_release: app.remove_tab()

        MDFlatButton:
```

(continues on next page)

(continued from previous page)

```

        text: "GET TAB LIST"
        on_release: app.get_tab_list()
    ...

class Tab(ScrollView, MDTabsBase):
    '''Class implementing content for a tab.'''

class Example(MDApp):
    index = 0

    def build(self):
        return Builder.load_string(KV)

    def on_start(self):
        self.add_tab()

    def get_tab_list(self):
        '''Prints a list of tab objects.'''

        print(self.root.ids.tabs.get_tab_list())

    def add_tab(self):
        self.index += 1
        self.root.ids.tabs.add_widget(Tab(text=f"{self.index} tab"))

    def remove_tab(self):
        if self.index > 1:
            self.index -= 1
            self.root.ids.tabs.remove_widget(
                self.root.ids.tabs.get_tab_list()[0]
            )

Example().run()

```

## Use on\_ref\_press method

You can use markup for the text of the tabs and use the `on_ref_press` method accordingly:

```

from kivy.lang import Builder
from kivy.uix.floatlayout import FloatLayout

from kivymd.app import MDApp
from kivymd.font_definitions import fonts
from kivymd.uix.tab import MDTabsBase
from kivymd.icon_definitions import md_icons

KV = '''
BoxLayout:
    orientation: "vertical"

```

(continues on next page)

(continued from previous page)

```

MDToolbar:
    title: "Example Tabs"

MDTabs:
    id: tabs
    on_ref_press: app.on_ref_press(*args)

<Tab>:

    MDIconButton:
        id: icon
        icon: app.icons[0]
        user_font_size: "48sp"
        pos_hint: {"center_x": .5, "center_y": .5}
    ...


class Tab(FloatLayout, MDTabsBase):
    '''Class implementing content for a tab.'''


class Example(MDApp):
    icons = list(md_icons.keys())[15:30]

    def build(self):
        return Builder.load_string(KV)

    def on_start(self):
        for name_tab in self.icons:
            self.root.ids.tabs.add_widget(
                Tab(
                    text=f"[ref={name_tab}] [font={fonts[-1]['fn_regular']}]{md_icons[
→'close']}[/font][/ref] {name_tab}"
                )
            )

    def on_ref_press(
        self,
        instance_tabs,
        instance_tab_label,
        instance_tab,
        instance_tab_bar,
        instance_carousel,
    ):
        ...
        The method will be called when the ``on_ref_press`` event occurs when you, for example, use markup text for tabs.

        :param instance_tabs: <kivymd.uix.tab.MDTabs object>
        :param instance_tab_label: <kivymd.uix.tab.MDTabsLabel object>
        :param instance_tab: <__main__.Tab object>
        :param instance_tab_bar: <kivymd.uix.tab.MDTabsBar object>
        :param instance_carousel: <kivymd.uix.tab.MDTabsCarousel object>
        ...
    
```

# Removes a tab by clicking on the close icon on the left.

(continues on next page)

(continued from previous page)

```
for instance_tab in instance_carousel.slides:
    if instance_tab.text == instance_tab_label.text:
        instance_tabs.remove_widget(instance_tab_label)
        break
```

```
Example().run()
```

## Switching the tab by name

```
from kivy.lang import Builder
from kivy.uix.floatlayout import FloatLayout

from kivymd.app import MDApp
from kivymd.uix.tab import MDTabsBase
from kivymd.icon_definitions import md_icons

KV = '''
BoxLayout:
    orientation: "vertical"

    MDToolbar:
        title: "Example Tabs"

    MDTabs:
        id: tabs

<Tab>:

    MDIconButton:
        id: icon
        icon: "arrow-right"
        user_font_size: "48sp"
        pos_hint: {"center_x": .5, "center_y": .5}
        on_release: app.switch_tab()
    ...

class Tab(FloatLayout, MDTabsBase):
    '''Class implementing content for a tab.'''
    ...

class Example(MDApp):
    icons = list(md_icons.keys())[15:30]

    def build(self):
        self.iter_list = iter(list(self.icons))
        return Builder.load_string(KV)

    def on_start(self):
        for name_tab in list(self.icons):
            self.root.ids.tabs.add_widget(Tab(text=name_tab))
```

(continues on next page)

(continued from previous page)

```
def switch_tab(self):
    '''Switching the tab by name.'''
    try:
        self.root.ids.tabs.switch_tab(next(self.iter_list))
    except StopIteration:
        pass
```

Example().run()

## API - kivymd.uix.tab

**class kivymd.uix.tab.MDTabsBase (\*\*kwargs)**

This class allow you to create a tab. You must create a new class that inherits from MDTabsBase. In this way you have total control over the views of your tabbed panel.

### **text**

It will be the label text of the tab.

`text` is an `StringProperty` and defaults to ‘’.

### **tab\_label**

It is the label object reference of the tab.

`tab_label` is an `ObjectProperty` and defaults to `None`.

### **on\_text (self, widget, text)**

**class kivymd.uix.tab.MDTabs (\*\*kwargs)**

You can use this class to create your own tabbed panel..

### Events

**on\_tab\_switch** Called when switching tabs.

**on\_slide\_progress** Called while the slide is scrolling.

**on\_ref\_press** The method will be called when the `on_ref_press` event occurs when you, for example, use markup text for tabs.

### **default\_tab**

Index of the default tab.

`default_tab` is an `NumericProperty` and defaults to `0`.

### **tab\_bar\_height**

Height of the tab bar.

`tab_bar_height` is an `NumericProperty` and defaults to ‘`48dp`’.

### **tab\_indicator\_anim**

Tab indicator animation. If you want use animation set it to `True`.

`tab_indicator_anim` is an `BooleanProperty` and defaults to `False`.

### **tab\_indicator\_height**

Height of the tab indicator.

`tab_indicator_height` is an `NumericProperty` and defaults to ‘2dp’.

**anim\_duration**

Duration of the slide animation.

`anim_duration` is an `NumericProperty` and defaults to `0.2`.

**anim\_threshold**

Animation threshold allow you to change the tab indicator animation effect.

`anim_threshold` is an `BoundedNumericProperty` and defaults to `0.8`.

**allow\_stretch**

If `False` - tabs will not stretch to full screen.

`allow_stretch` is an `BooleanProperty` and defaults to `True`.

**background\_color**

Background color of tabs in `rgba` format.

`background_color` is an `ListProperty` and defaults to `[]`.

**text\_color\_normal**

Text color of the label when it is not selected.

`text_color_normal` is an `ListProperty` and defaults to `(1, 1, 1, 1)`.

**text\_color\_active**

Text color of the label when it is selected.

`text_color_active` is an `ListProperty` and defaults to `(1, 1, 1, 1)`.

**elevation**

Tab value elevation.

**See also:**

Behaviors/Elevation

`elevation` is an `NumericProperty` and defaults to `0`.

**color\_indicator**

Color indicator in `rgba` format.

`color_indicator` is an `ListProperty` and defaults to `[]`.

**lock\_swiping**

If `True` - disable switching tabs by swipe.

`lock_swiping` is an `BooleanProperty` and defaults to `False`.

**font\_name**

Font name for tab text.

`font_name` is an `StringProperty` and defaults to ‘`Roboto`’.

**switch\_tab (self, name\_tab)**

Switching the tab by name.

**get\_tab\_list (self)**

Returns a list of tab objects.

**add\_widget (self, widget, index=0, canvas=None)**

Add a new widget as a child of this widget.

**Parameters**

**widget: Widget** Widget to add to our list of children.

**index: int, defaults to 0** Index to insert the widget in the list. Notice that the default of 0 means the widget is inserted at the beginning of the list and will thus be drawn on top of other sibling widgets. For a full discussion of the index and widget hierarchy, please see the [Widgets Programming Guide](#).

New in version 1.0.5.

**canvas: str, defaults to None** Canvas to add widget's canvas to. Can be 'before', 'after' or None for the default canvas.

New in version 1.9.0.

```
>>> from kivy.uix.button import Button
>>> from kivy.uix.slider import Slider
>>> root = Widget()
>>> root.add_widget(Button())
>>> slider = Slider()
>>> root.add_widget(slider)
```

**remove\_widget (self, widget)**

Remove a widget from the children of this widget.

#### Parameters

**widget: Widget** Widget to remove from our children list.

```
>>> from kivy.uix.button import Button
>>> root = Widget()
>>> button = Button()
>>> root.add_widget(button)
>>> root.remove_widget(button)
```

**on\_slide\_progress (self, \*args)**

Called while the slide is scrolling.

**on\_carousel\_index (self, carousel, index)**

Called when the carousel index changes.

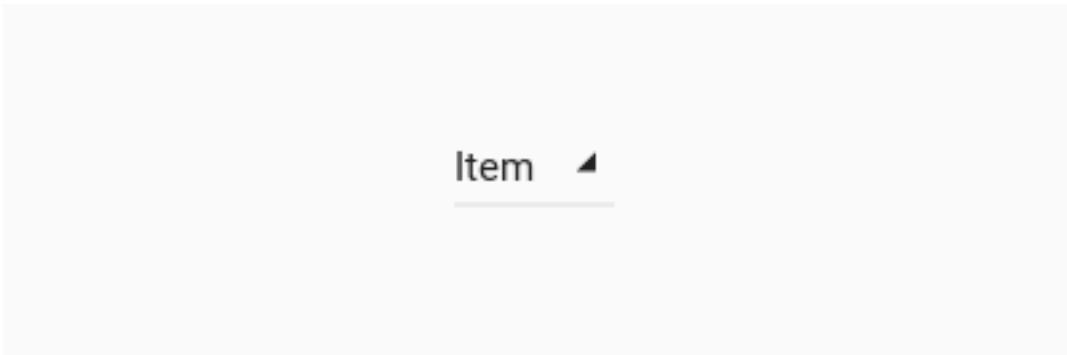
**on\_ref\_press (self, \*args)**

The method will be called when the `on_ref_press` event occurs when you, for example, use markup text for tabs.

**on\_tab\_switch (self, \*args)**

Called when switching tabs.

## 2.3.6 Dropdown Item



### Usage

```
from kivy.lang import Builder

from kivymd.app import MDApp

KV = '''
Screen

    MDDropDownItem:
        id: drop_item
        pos_hint: {'center_x': .5, 'center_y': .5}
        text: 'Item'
        on_release: self.set_item("New Item")
'''


class Test(MDApp):
    def __init__(self, **kwargs):
        super().__init__(**kwargs)
        self.screen = Builder.load_string(KV)

    def build(self):
        return self.screen

Test().run()
```

#### See also:

Work with the class `MDDropdownMenu` see here

**API - kivymd.uix.dropdownitem**

```
class kivymd.uix.dropdownitem.MDDropDownItem(**kwargs)
    Class implements a rectangular ripple effect.

text
    Text item.

    text is a StringProperty and defaults to ''.

current_item
    Current name item.

    current_item is a StringProperty and defaults to ''.

font_size
    Item font size.

    font_size is a NumericProperty and defaults to '16sp'.

on_text(self, instance, value)

set_item(self, name_item)
    Sets new text for an item.
```

### 2.3.7 Pickers

Includes date, time and color picker

KivyMD provides the following classes for use:

- *MDTimePicker*
- *MDDatePicker*
- *MDThemePicker*

#### MDTimePicker

##### Usage

```
from kivy.lang import Builder

from kivymd.app import MDApp
from kivymd.uix.picker import MDTIMEPicker

KV = '''
FloatLayout:

    MDRaisedButton:
        text: "Open time picker"
        pos_hint: {'center_x': .5, 'center_y': .5}
        on_release: app.show_time_picker()
'''


class Test(MDApp):
    def build(self):
```

(continues on next page)

(continued from previous page)

```
return Builder.load_string(KV)

def show_time_picker(self):
    '''Open time picker dialog.'''

    time_dialog = MDTimePicker()
    time_dialog.open()

Test().run()
```

### Binding method returning set time

```
def show_time_picker(self):
    time_dialog = MDTimePicker()
    time_dialog.bind(time=self.get_time)
    time_dialog.open()

def get_time(self, instance, time):
    """
    The method returns the set time.

    :type instance: <kivymd.uix.picker.MDTimePicker object>
    :type time: <class 'datetime.time'>
    """

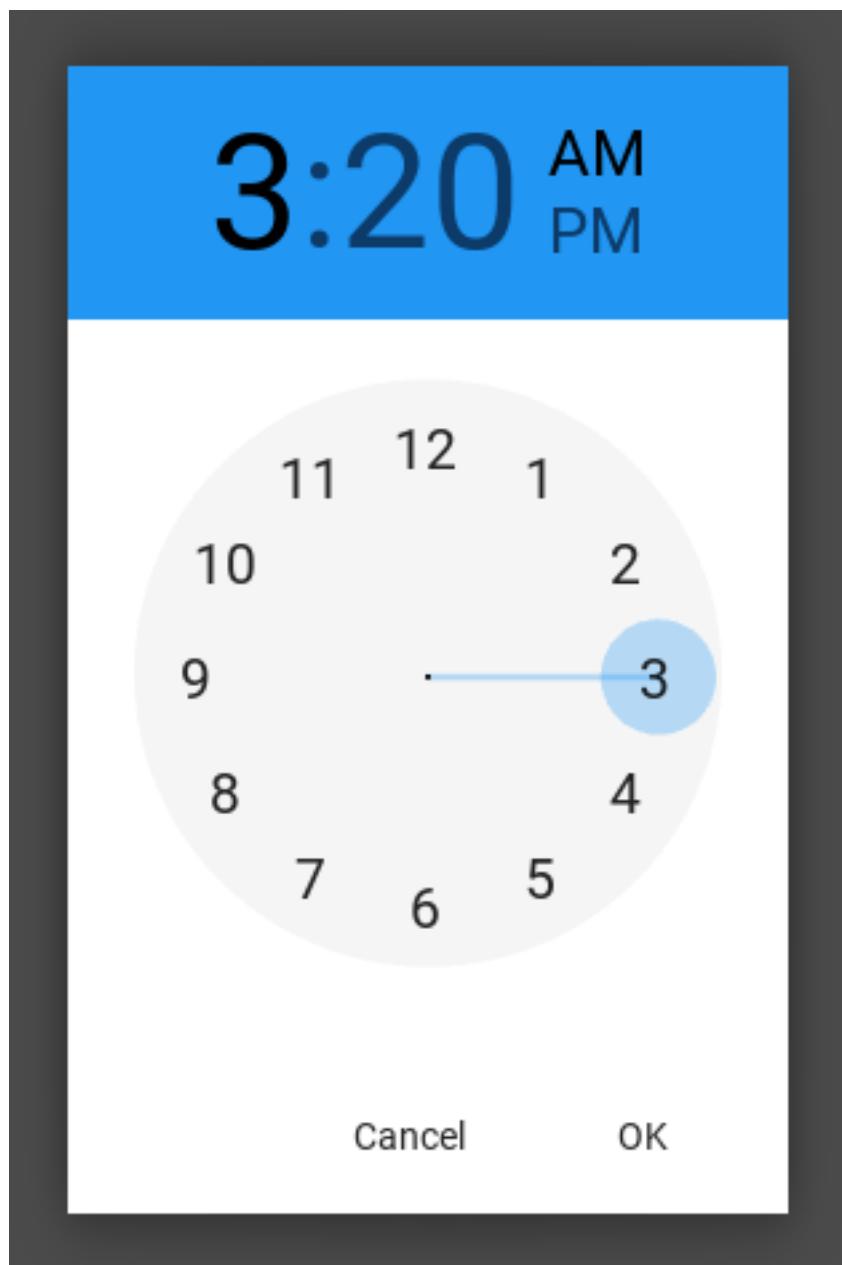
    return time
```

### Open time dialog with the specified time

Use the `set_time` method of the class.

```
def show_time_picker(self):
    from datetime import datetime

    # Must be a datetime object
    previous_time = datetime.strptime("03:20:00", '%H:%M:%S').time()
    time_dialog = MDTimePicker()
    time_dialog.set_time(previous_time)
    time_dialog.open()
```



## MDDatePicker

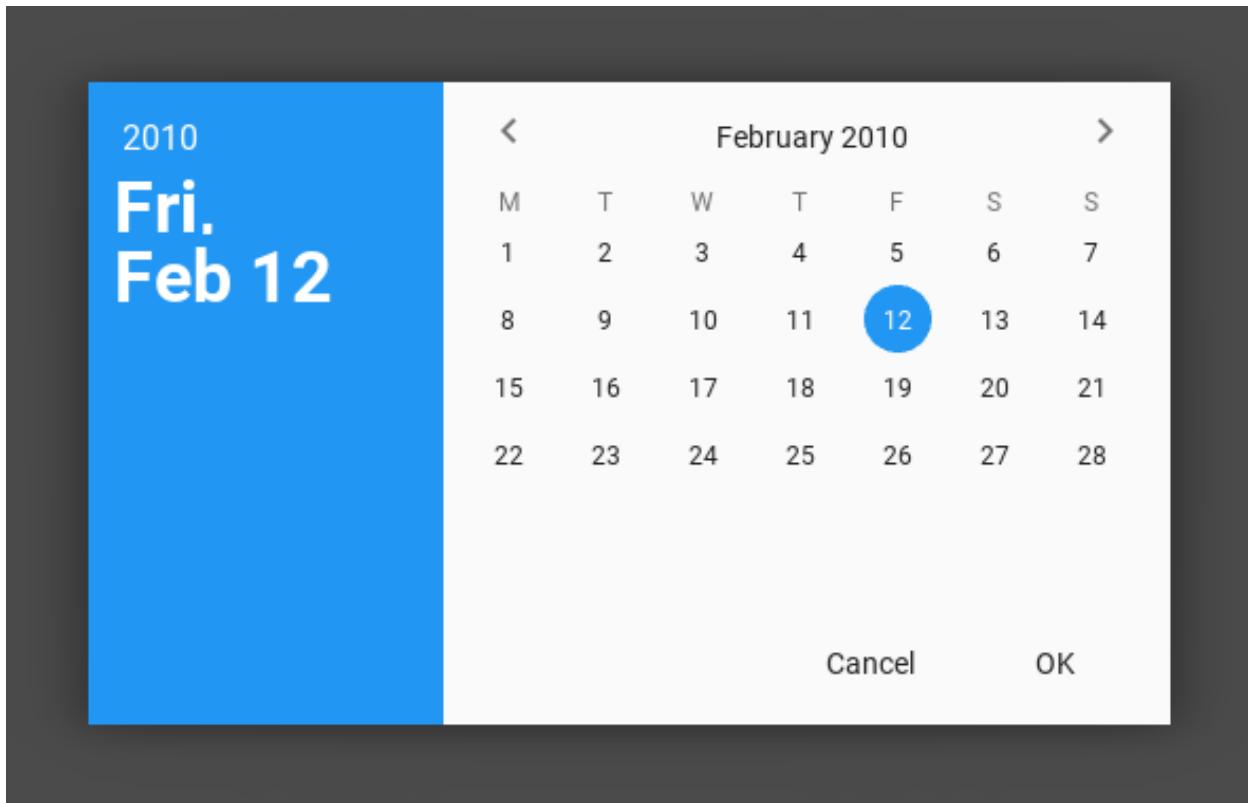
When creating an instance of the `MDDatePicker` class, you must pass as a parameter a method that will take one argument - a `datetime` object.

```
def get_date(self, date):
    """
    :type date: <class 'datetime.date'>
    """

def show_date_picker(self):
    date_dialog = MDDatePicker(callback=self.get_date)
    date_dialog.open()
```

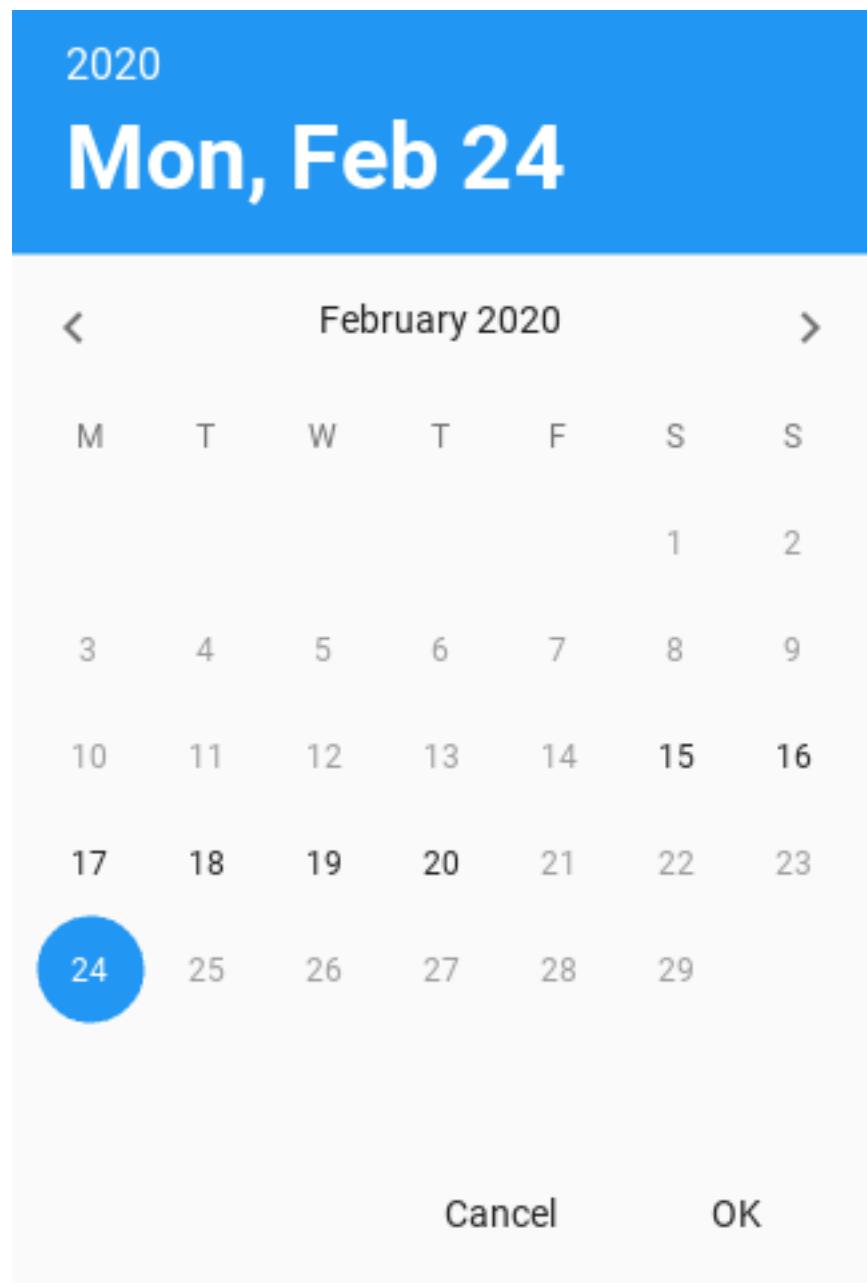
## Open date dialog with the specified date

```
def show_date_picker(self):
    date_dialog = MDDatePicker(
        callback=self.get_date,
        year=2010,
        month=2,
        day=12,
    )
    date_dialog.open()
```



You can set the time interval from and to the set date. All days of the week that are not included in this range will have the status *disabled*.

```
def show_date_picker(self):
    min_date = datetime.strptime("2020:02:15", '%Y:%m:%d').date()
    max_date = datetime.strptime("2020:02:20", '%Y:%m:%d').date()
    date_dialog = MDDatePicker(
        callback=self.get_date,
        min_date=min_date,
        max_date=max_date,
    )
    date_dialog.open()
```



### MDThemePicker

```
def show_theme_picker(self):
    theme_dialog = MDThemePicker()
    theme_dialog.open()
```

**API - kivymd.uix.picker**

```
class kivymd.uix.picker.MDDatePicker(callback, year=None, month=None, day=None,
                                         firstweekday=0, min_date=None, max_date=None,
                                         **kwargs)
```

Float layout class. See module documentation for more information.

```
cal_list
cal_layout
sel_year
sel_month
sel_day
day
month
year
today
callback
background_color
ok_click(self)
fmt_lbl_date(self, year, month, day, orientation)
set_date(self, year, month, day)
set_selected_widget(self, widget)
set_month_day(self, day)
update_cal_matrix(self, year, month)
generate_cal_widgets(self)
change_month(self, operation)
```

```
class kivymd.uix.picker.MDTimePicker(**kwargs)
```

Float layout class. See module documentation for more information.

**time**

Users method. Must take two parameters:

```
def get_time(self, instance, time):
    """
    The method returns the set time.

    :type instance: <kivymd.uix.picker.MDTimePicker object>
    :type time: <class 'datetime.time'>
    """

    return time
```

*time* is an `ObjectProperty` and defaults to `None`.

**set\_time(self, time)**

Sets user time.

```
close_cancel(self)
close_ok(self)

class kivymd.uix.picker.MDThemePicker(**kwargs)
    Float layout class. See module documentation for more information.
```

### 2.3.8 Bottom Sheet

See also:

Material Design spec, Sheets: bottom

**Bottom sheets are surfaces containing supplementary content that are anchored to the bottom of the screen.**

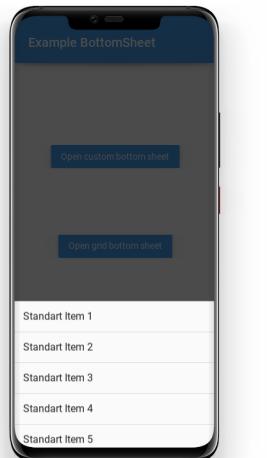


Share

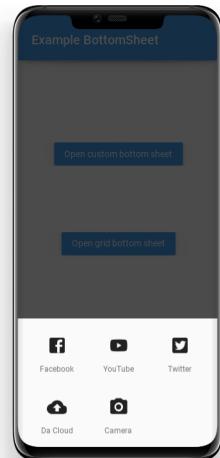


Get link

Two classes are available to you `MDListBottomSheet` and `MDGridBottomSheet` for standard bottom sheets dialogs:



**MDListBottomSheet**



**MDGridBottomSheet**

### Usage MDListBottomSheet

```
from kivy.lang import Builder

from kivymd.toast import toast
from kivymd.uix.bottomsheet import MDListBottomSheet
from kivymd.app import MDApp

KV = '''
Screen:

    MDToolbar:
        title: "Example BottomSheet"
        pos_hint: {"top": 1}
        elevation: 10

    MDRaisedButton:
        text: "Open list bottom sheet"
        on_release: app.show_example_list_bottom_sheet()
        pos_hint: {"center_x": .5, "center_y": .5}
'''


class Example(MDApp):
    def build(self):
        return Builder.load_string(KV)

    def callback_for_menu_items(self, *args):
        toast(args[0])

    def show_example_list_bottom_sheet(self):
        bottom_sheet_menu = MDListBottomSheet()
        for i in range(1, 11):
            bottom_sheet_menu.add_item(
                f"Standart Item {i}",
                lambda x, y=i: self.callback_for_menu_items(
                    f"Standart Item {y}"
                ),
            )
        bottom_sheet_menu.open()

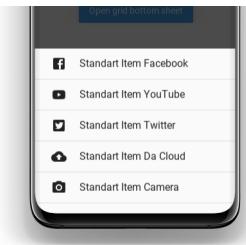
Example().run()
```

The `add_item` method of the `MDListBottomSheet` class takes the following arguments:

`text` - element text;

`callback` - function that will be called when clicking on an item;

There is also an optional argument `icon`, which will be used as an icon to the left of the item:



Using the `MDGridBottomSheet` class is similar to using the `MDListBottomSheet` class:

```
from kivy.lang import Builder

from kivymd.toast import toast
from kivymd.uix.bottomsheet import MDGridBottomSheet
from kivymd.app import MDApp

KV = '''
Screen:

    MDToolbar:
        title: 'Example BottomSheet'
        pos_hint: {"top": 1}
        elevation: 10

    MDRaisedButton:
        text: "Open grid bottom sheet"
        on_release: app.show_example_grid_bottom_sheet()
        pos_hint: {"center_x": .5, "center_y": .5}
'''


class Example(MDApp):
    def build(self):
        return Builder.load_string(KV)

    def callback_for_menu_items(self, *args):
        toast(args[0])

    def show_example_grid_bottom_sheet(self):
        bottom_sheet_menu = MDGridBottomSheet()
        data = {
            "Facebook": "facebook-box",
            "YouTube": "youtube",
            "Twitter": "twitter-box",
            "Da Cloud": "cloud-upload",
            "Camera": "camera",
        }
        for item in data.items():
            bottom_sheet_menu.add_item(
                item[0],
                lambda x, y=item[0]: self.callback_for_menu_items(y),
                icon_src=item[1],
            )
        bottom_sheet_menu.open()
```

(continues on next page)

(continued from previous page)

```
Example().run()
```



### You can use custom content for bottom sheet dialogs:

```
from kivy.lang import Builder
from kivy.factory import Factory

from kivymd.uix.bottomsheet import MDCustomBottomSheet
from kivymd.app import MDApp

KV = """
<ItemForCustomBottomSheet@OneLineIconListItem>
    on_press: app.custom_sheet.dismiss()
    icon: ""

    IconLeftWidget:
        icon: root.icon

<ContentCustomSheet@BoxLayout>:
    orientation: "vertical"
    size_hint_y: None
    height: "400dp"

    MDToolbar:
        title: 'Custom bottom sheet:'

    ScrollView:

        MDGridLayout:
            cols: 1
            adaptive_height: True

            ItemForCustomBottomSheet:
                icon: "page-previous"
                text: "Preview"

            ItemForCustomBottomSheet:
                icon: "exit-to-app"
                text: "Exit"

Screen:

    MDToolbar:
```

(continues on next page)

(continued from previous page)

```

title: 'Example BottomSheet'
pos_hint: {"top": 1}
elevation: 10

MDRaisedButton:
    text: "Open custom bottom sheet"
    on_release: app.show_example_custom_bottom_sheet()
    pos_hint: {"center_x": .5, "center_y": .5}
    ...

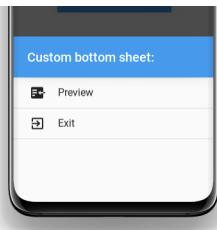
class Example(MDApp):
    custom_sheet = None

    def build(self):
        return Builder.load_string(KV)

    def show_example_custom_bottom_sheet(self):
        self.custom_sheet = MDCustomBottomSheet(screen=Factory.ContentCustomSheet())
        self.custom_sheet.open()

Example().run()

```




---

**Note:** When you use the `MDCustomBottomSheet` class, you must specify the height of the user-defined content exactly, otherwise dp (100) heights will be used for your `ContentCustomSheet` class:

```

<ContentCustomSheet@BoxLayout>:
    orientation: "vertical"
    size_hint_y: None
    height: "400dp"

```

---

**Note:** The height of the bottom sheet dialog will never exceed half the height of the screen!

---

## API - kivymd.uix.bottomsheet

```
class kivymd.uix.bottomsheet.MDBottomSheet(**kwargs)
    ModalView class. See module documentation for more information.
```

### Events

**on\_pre\_open:** Fired before the ModalView is opened. When this event is fired ModalView is not yet added to window.

**on\_open:** Fired when the ModalView is opened.

**on\_pre\_dismiss:** Fired before the ModalView is closed.

**on\_dismiss:** Fired when the ModalView is closed. If the callback returns True, the dismiss will be canceled.

Changed in version 1.11.0: Added events *on\_pre\_open* and *on\_pre\_dismiss*.

### background

Private attribute.

### duration\_opening

The duration of the bottom sheet dialog opening animation.

*duration\_opening* is an [NumericProperty](#) and defaults to *0.15*.

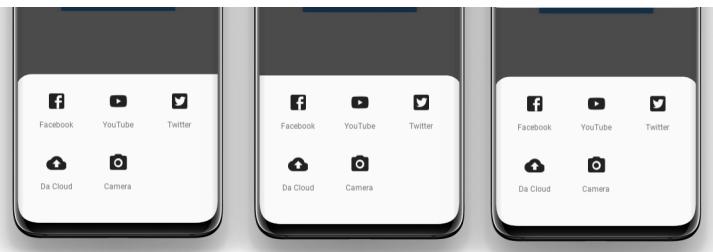
### radius

The value of the rounding of the corners of the dialog.

*radius* is an [NumericProperty](#) and defaults to *25*.

### radius\_from

Sets which corners to cut from the dialog. Available options are: (*"top\_left"*, *"top\_right"*, *"top"*, *"bottom\_right"*, *"bottom\_left"*, *"bottom"*).



*radius\_from* is an [OptionProperty](#) and defaults to *None*.

### animation

To use animation of opening of dialogue of the bottom sheet or not.

*animation* is an [BooleanProperty](#) and defaults to *False*.

### bg\_color

Dialog background color in `rgba` format.

*bg\_color* is an [ListProperty](#) and defaults to *[]*.

### value\_transparent

Background transparency value when opening a dialog.

*value\_transparent* is an [ListProperty](#) and defaults to *[0, 0, 0, 0.8]*.

**open**(*self*, \**largs*)

Show the view window from the `attach_to` widget. If set, it will attach to the nearest window. If the widget is not attached to any window, the view will attach to the global `Window`.

When the view is opened, it will be faded in with an animation. If you don't want the animation, use:

```
view.open(animation=False)
```

**add\_widget**(*self*, *widget*, *index*=0, *canvas*=None)

Add a new widget as a child of this widget.

**Parameters**

**widget: Widget** Widget to add to our list of children.

**index: int, defaults to 0** Index to insert the widget in the list. Notice that the default of 0 means the widget is inserted at the beginning of the list and will thus be drawn on top of other sibling widgets. For a full discussion of the index and widget hierarchy, please see the [Widgets Programming Guide](#).

New in version 1.0.5.

**canvas: str, defaults to None** Canvas to add widget's canvas to. Can be 'before', 'after' or None for the default canvas.

New in version 1.9.0.

```
>>> from kivy.uix.button import Button
>>> from kivy.uix.slider import Slider
>>> root = Widget()
>>> root.add_widget(Button())
>>> slider = Slider()
>>> root.add_widget(slider)
```

**on\_dismiss**(*self*)**resize\_content\_layout**(*self*, *content*, *layout*, *interval*=0)**class** `kivymd.uix.bottomsheet.MDCustomBottomSheet`(\*\**kwargs*)

ModalView class. See module documentation for more information.

**Events**

**on\_pre\_open**: Fired before the ModalView is opened. When this event is fired ModalView is not yet added to window.

**on\_open**: Fired when the ModalView is opened.

**on\_pre\_dismiss**: Fired before the ModalView is closed.

**on\_dismiss**: Fired when the ModalView is closed. If the callback returns True, the dismiss will be canceled.

Changed in version 1.11.0: Added events `on_pre_open` and `on_pre_dismiss`.

**screen**

Custom content.

`screen` is an `ObjectProperty` and defaults to `None`.

**class** `kivymd.uix.bottomsheet.MDListBottomSheet`(\*\**kwargs*)

ModalView class. See module documentation for more information.

**Events**

***on\_pre\_open:*** Fired before the ModalView is opened. When this event is fired ModalView is not yet added to window.

***on\_open:*** Fired when the ModalView is opened.

***on\_pre\_dismiss:*** Fired before the ModalView is closed.

***on\_dismiss:*** Fired when the ModalView is closed. If the callback returns True, the dismiss will be canceled.

Changed in version 1.11.0: Added events *on\_pre\_open* and *on\_pre\_dismiss*.

#### **sheet\_list**

`sheet_list` is an `ObjectProperty` and defaults to `None`.

#### **add\_item(self, text, callback, icon=None)**

##### **Parameters**

- **text** – element text;
- **callback** – function that will be called when clicking on an item;
- **icon\_src** – which will be used as an icon to the left of the item;

**class kivymd.uix.bottomsheet.GridBottomSheetItem(\*\*kwargs)**

This `mixin` class provides `Button` behavior. Please see the `button behaviors module` documentation for more information.

#### **Events**

***on\_press*** Fired when the button is pressed.

***on\_release*** Fired when the button is released (i.e. the touch/click that pressed the button goes away).

#### **source**

Icon path if you use a local image or icon name if you use icon names from a file `kivymd/icon_definitions.py`.

`source` is an `StringProperty` and defaults to ''.

#### **caption**

Item text.

`caption` is an `StringProperty` and defaults to ''.

#### **icon\_size**

Icon size.

`caption` is an `StringProperty` and defaults to '32sp'.

**class kivymd.uix.bottomsheet.MDGridBottomSheet(\*\*kwargs)**

ModalView class. See module documentation for more information.

#### **Events**

***on\_pre\_open:*** Fired before the ModalView is opened. When this event is fired ModalView is not yet added to window.

***on\_open:*** Fired when the ModalView is opened.

***on\_pre\_dismiss:*** Fired before the ModalView is closed.

***on\_dismiss:*** Fired when the ModalView is closed. If the callback returns True, the dismiss will be canceled.

Changed in version 1.11.0: Added events `on_pre_open` and `on_pre_dismiss`.

`add_item(self, text, callback, icon_src)`

#### Parameters

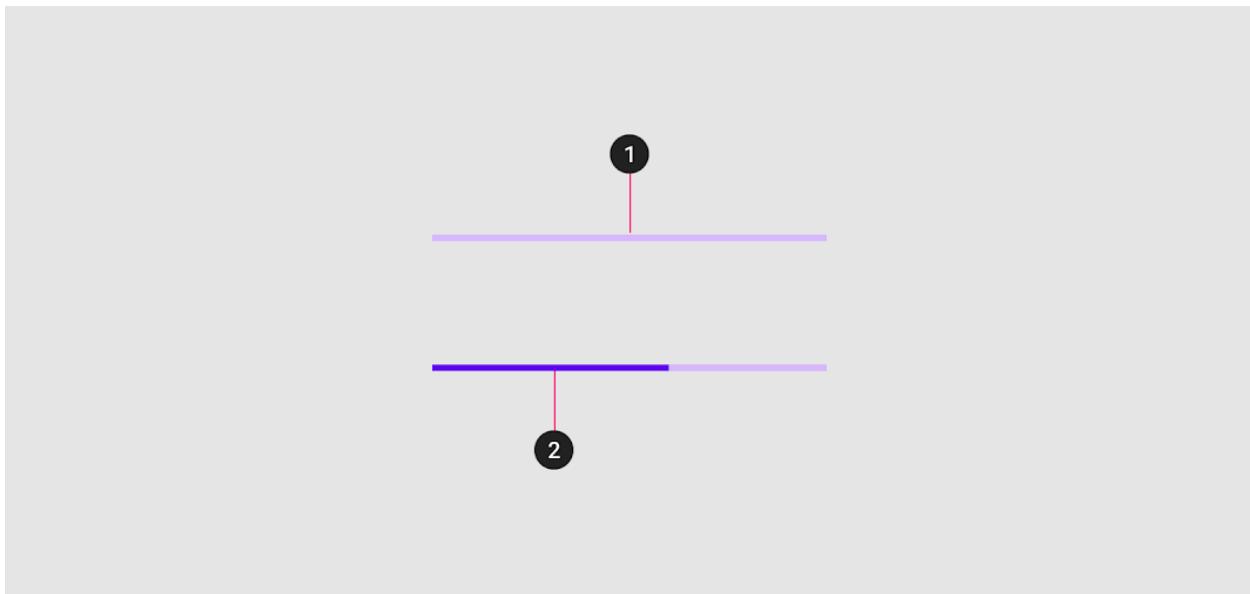
- `text` – element text;
- `callback` – function that will be called when clicking on an item;
- `icon_src` – icon item;

### 2.3.9 Progress Bar

See also:

[`Material Design spec, Progress indicators <https://material.io/components/progress-indicators>`](https://material.io/components/progress-indicators)

Progress indicators express an unspecified wait time or display the length of a process.



KivyMD provides the following bars classes for use:

- `MDProgressBar`
- `Determinate`
- `Indeterminate`

## MDProgressBar

```
from kivy.lang import Builder

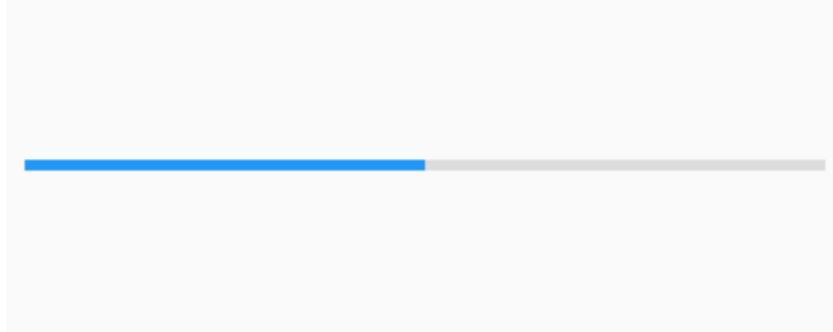
from kivymd.app import MDApp

KV = '''
BoxLayout:
    padding: "10dp"

    MDProgressBar:
        value: 50
'''

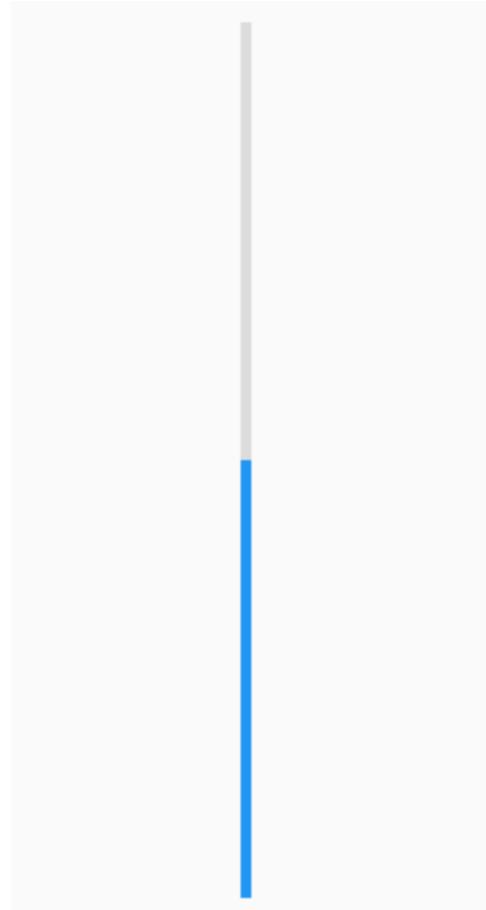

class Test(MDApp):
    def build(self):
        return Builder.load_string(KV)

Test().run()
```



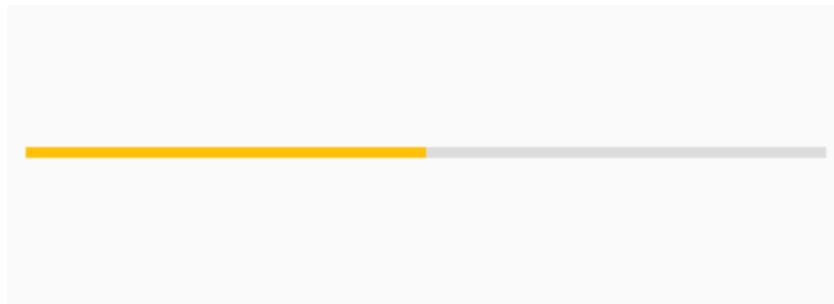
## Vertical orientation

```
MDProgressBar:
    orientation: "vertical"
    value: 50
```



### With custom color

```
MDProgressBar:  
    value: 50  
    color: app.theme_cls.accent_color
```



## Indeterminate

```
from kivy.lang import Builder
from kivy.properties import StringProperty

from kivymd.app import MDApp

KV = '''
Screen:

    MDProgressBar:
        id: progress
        pos_hint: {"center_y": .6}
        type: "indeterminate"

    MDRaisedButton:
        text: "START"
        pos_hint: {"center_x": .5, "center_y": .45}
        on_press: app.state = "stop" if app.state == "start" else "start"
'''

class Test(MDApp):
    state = StringProperty("stop")

    def build(self):
        return Builder.load_string(KV)

    def on_state(self, instance, value):
        {
            "start": self.root.ids.progress.start,
            "stop": self.root.ids.progress.stop,
        }.get(value)()
```

Test().run()

## Determinate

```
MDProgressBar:
    type: "determinate"
    running_duration: 1
    catching_duration: 1.5
```

**API - kivymd.uix.progressbar****class** kivymd.uix.progressbar.MDProgressBar(\*\*kwargs)

Class for creating a progress bar widget.

See module documentation for more details.

**reversed**

Reverse the direction the progressbar moves.

*reversed* is an `BooleanProperty` and defaults to `False`.**orientation**

Orientation of progressbar. Available options are: ‘horizontal’, ‘vertical’.

*orientation* is an `OptionProperty` and defaults to ‘horizontal’.**color**Progress bar color in `rgba` format.*color* is an `OptionProperty` and defaults to `[]`.**running\_transition**

Running transition.

*running\_transition* is an `StringProperty` and defaults to ‘`in_cubic`’.**catching\_transition**

Catching transition.

*catching\_transition* is an `StringProperty` and defaults to ‘`out_quart`’.**running\_duration**

Running duration.

*running\_duration* is an `NumericProperty` and defaults to `0.5`.**catching\_duration**

Catching duration.

*running\_duration* is an `NumericProperty` and defaults to `0.8`.**type**Type of progressbar. Available options are: ‘`indeterminate`’, ‘`determinate`’.*type* is an `OptionProperty` and defaults to `None`.**start(self)**

Start animation.

**stop(self)**

Stop animation.

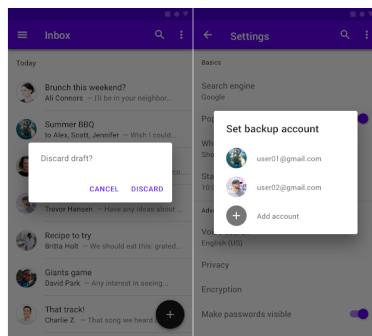
**running\_away(self, \*args)****catching\_up(self, \*args)**

## 2.3.10 Dialog

See also:

Material Design spec, Dialogs

**Dialogs inform users about a task and can contain critical information, require decisions, or involve multiple tasks.**



## Usage

```
from kivy.lang import Builder

from kivymd.app import MDApp
from kivymd.uix.button import MDFlatButton
from kivymd.uix.dialog import MDDialog

KV = '''
FloatLayout:

    MDFlatButton:
        text: "ALERT DIALOG"
        pos_hint: {'center_x': .5, 'center_y': .5}
        on_release: app.show_alert_dialog()
'''


class Example(MDApp):
    dialog = None

    def build(self):
        return Builder.load_string(KV)

    def show_alert_dialog(self):
        if not self.dialog:
            self.dialog = MDDialog(
                text="Discard draft?",
                buttons=[
                    MDFlatButton(
                        text="CANCEL", text_color=self.theme_cls.primary_color
                    ),
                    MDFlatButton(
                        text="DISCARD", text_color=self.theme_cls.primary_color
                    )
                ]
            )
            self.dialog.open()
```

(continues on next page)

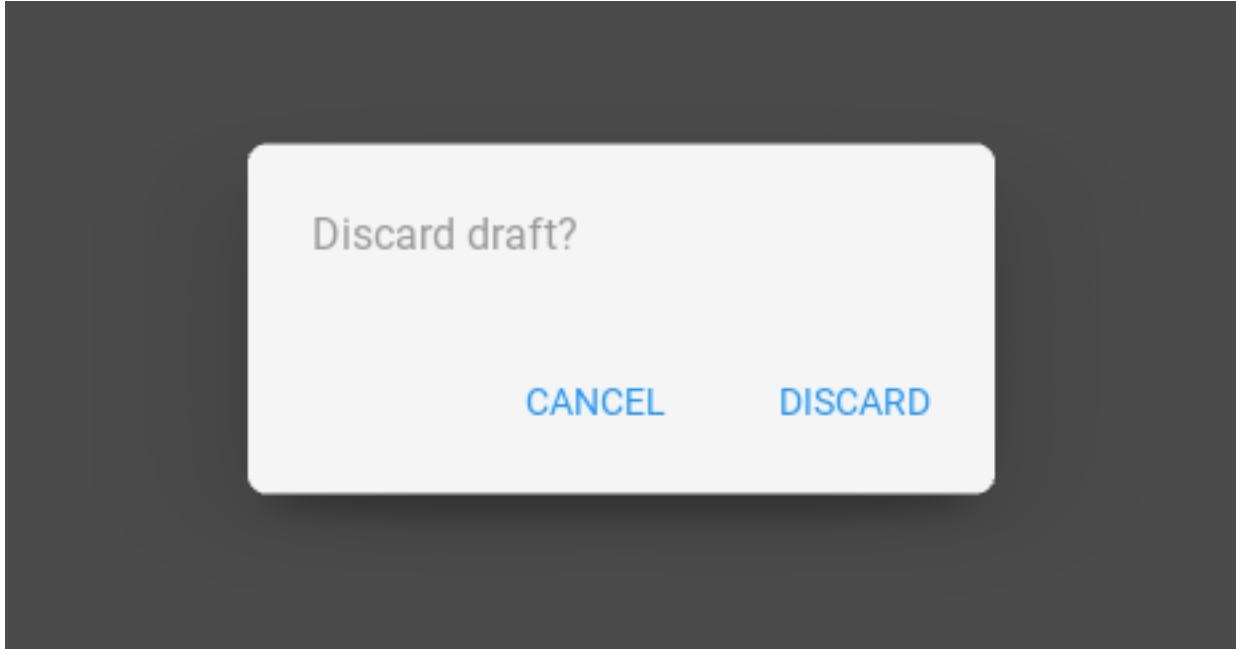
(continued from previous page)

```

        ) ,
    ],
)
self.dialog.open()

```

```
Example().run()
```



## API - kivymd.uix.dialog

**class** kivymd.uix.dialog.**MDDialog**(\*\*kwargs)

ModalView class. See module documentation for more information.

### Events

**on\_pre\_open:** Fired before the ModalView is opened. When this event is fired ModalView is not yet added to window.

**on\_open:** Fired when the ModalView is opened.

**on\_pre\_dismiss:** Fired before the ModalView is closed.

**on\_dismiss:** Fired when the ModalView is closed. If the callback returns True, the dismiss will be canceled.

Changed in version 1.11.0: Added events *on\_pre\_open* and *on\_pre\_dismiss*.

### title

Title dialog.

```

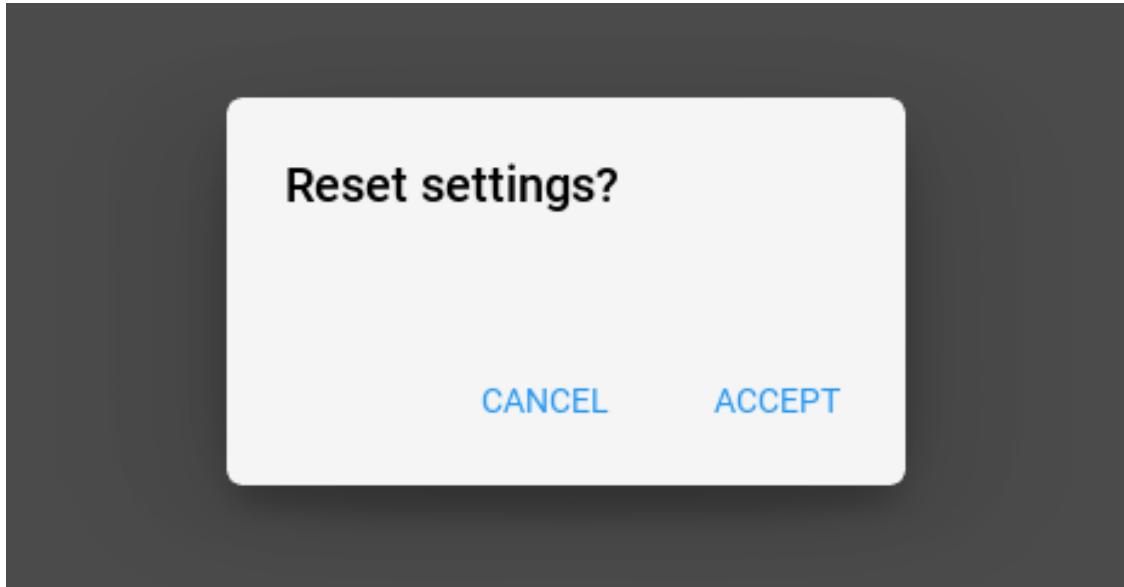
self.dialog = MDDialog(
    title="Reset settings?",
    buttons=[
        MDFlatButton(

```

(continues on next page)

(continued from previous page)

```
        text="CANCEL", text_color=self.theme_cls.primary_color
    ),
    MDFFlatButton(
        text="ACCEPT", text_color=self.theme_cls.primary_color
    ),
],
)
```

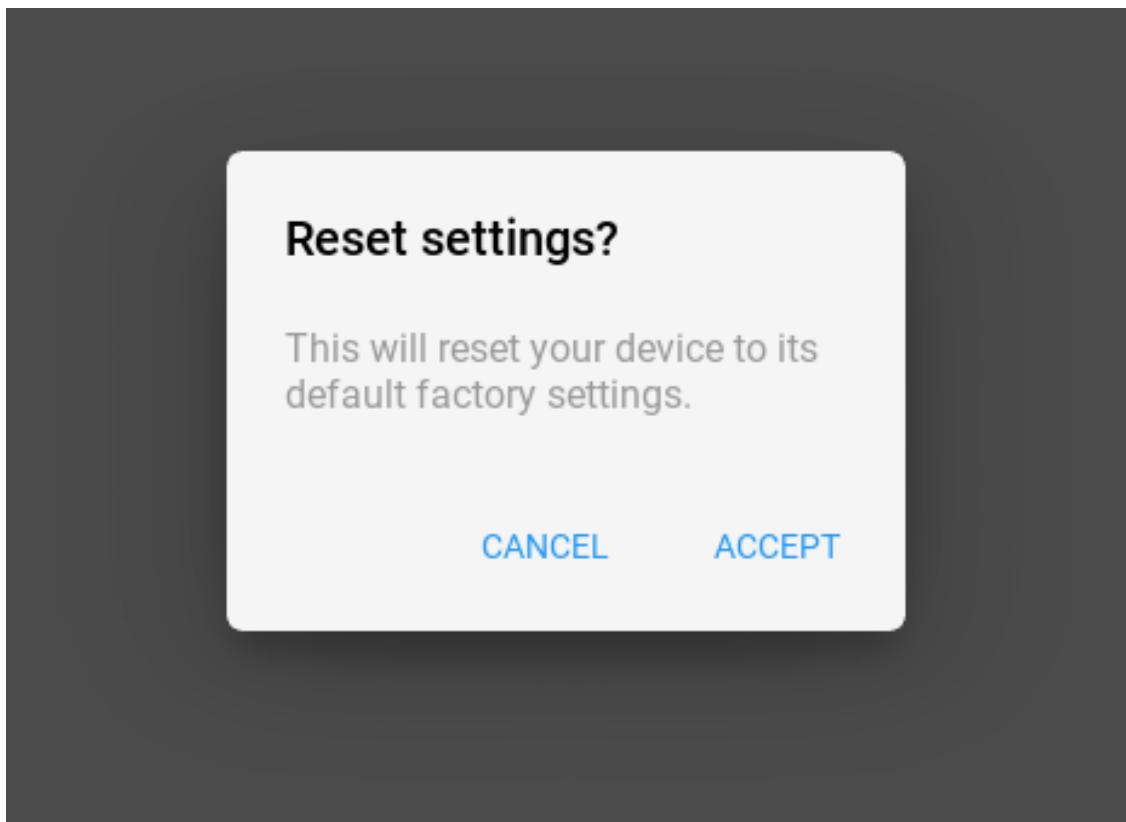


`title` is an `StringProperty` and defaults to ''.

#### `text`

Text dialog.

```
self.dialog = MDDialog(
    title="Reset settings?",
    text="This will reset your device to its default factory settings.",
    buttons=[
        MDFFlatButton(
            text="CANCEL", text_color=self.theme_cls.primary_color
        ),
        MDFFlatButton(
            text="ACCEPT", text_color=self.theme_cls.primary_color
        ),
    ],
)
```

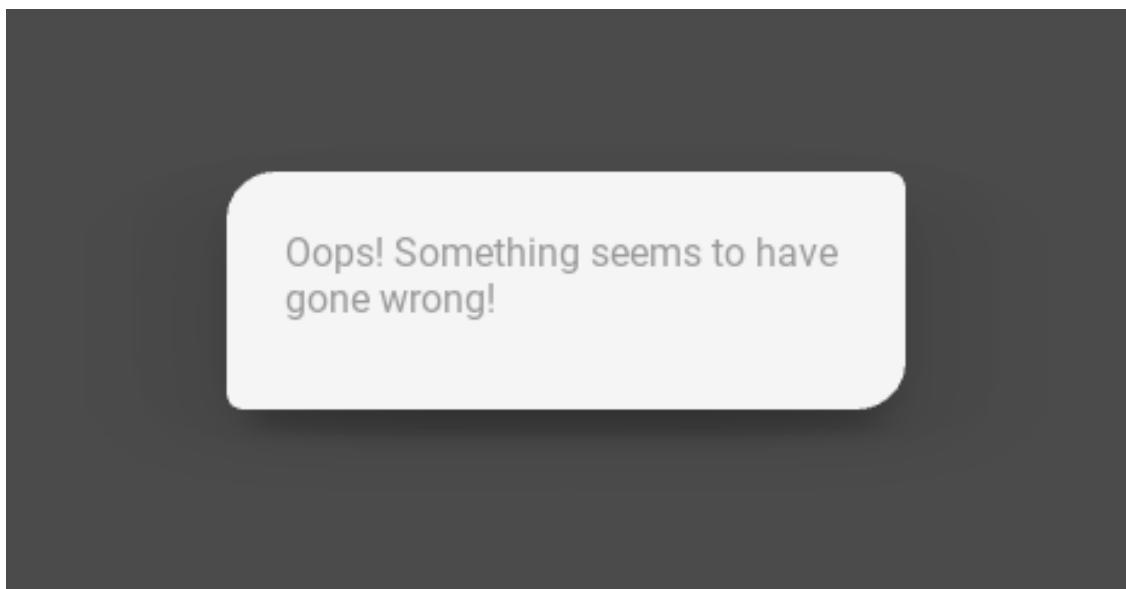


`text` is an `StringProperty` and defaults to ''.

#### **radius**

Dialog corners rounding value.

```
self.dialog = MDDialog(  
    text="Oops! Something seems to have gone wrong!",  
    radius=[20, 7, 20, 7],  
)
```

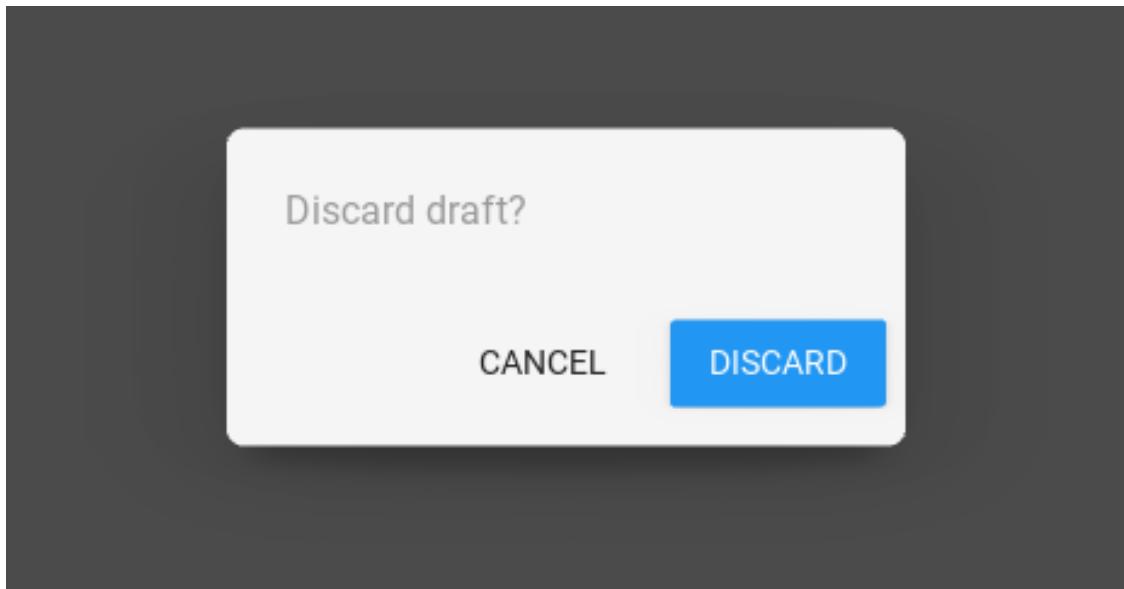


`radius` is an `ListProperty` and defaults to [7, 7, 7, 7].

#### buttons

List of button objects for dialog. Objects must be inherited from `BaseButton` class.

```
self.dialog = MDDialog(
    text="Discard draft?",
    buttons=[
        MDFlatButton(text="CANCEL"),
        MDRaisedButton(text="DISCARD"),
    ],
)
```



`buttons` is an `ListProperty` and defaults to [].

#### items

List of items objects for dialog. Objects must be inherited from `BaseListItem` class.

```
from kivy.lang import Builder
from kivy.properties import StringProperty

from kivymd.app import MDApp
from kivymd.uix.dialog import MDDialog
from kivymd.uix.list import OneLineAvatarListItem

KV = '''
<Item>

    ImageLeftWidget:
        source: root.source

    FloatLayout:

        MDFlatButton:
            text: "ALERT DIALOG"
            pos_hint: {'center_x': .5, 'center_y': .5}
            on_release: app.show_simple_dialog()
'''
```

(continues on next page)

(continued from previous page)

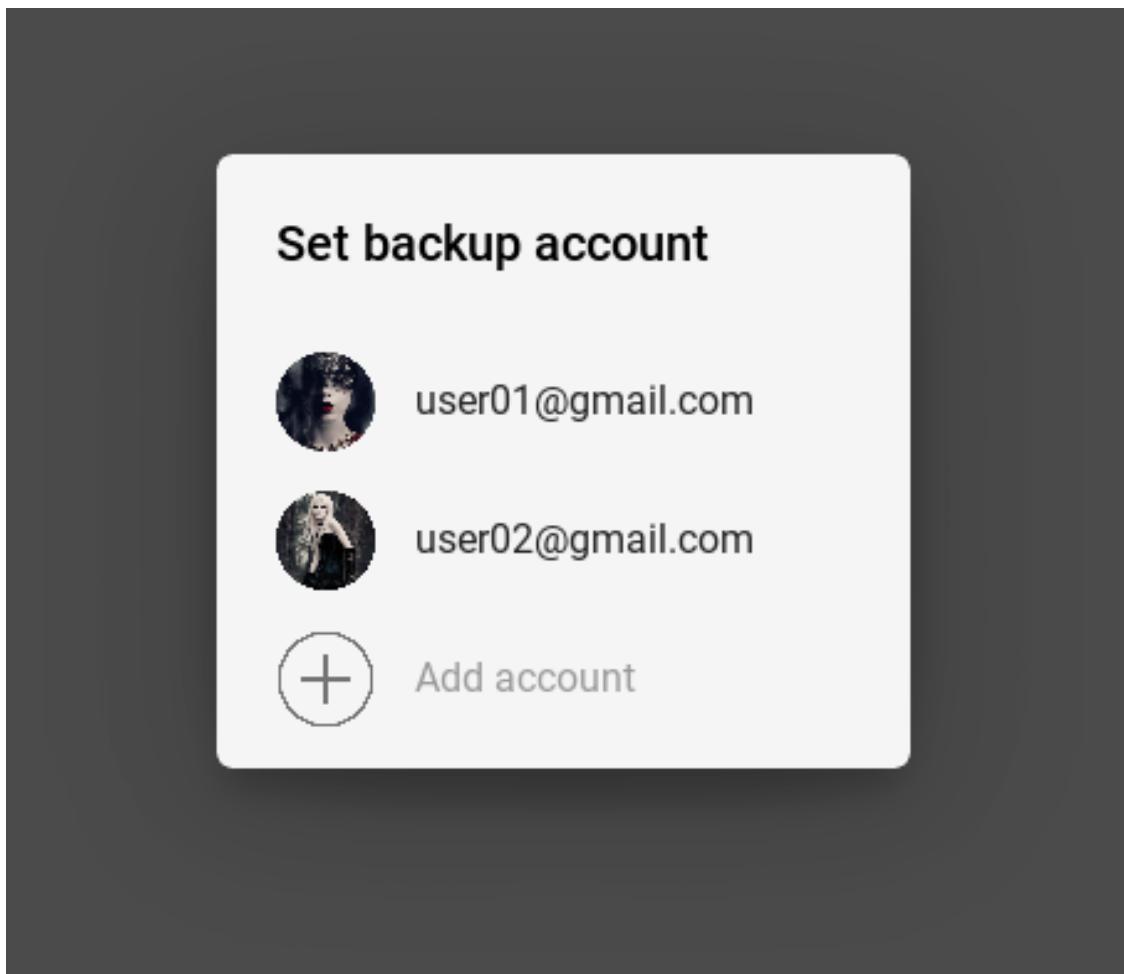
```
class Item(OneLineAvatarListItem):
    divider = None
    source = StringProperty()

class Example(MDApp):
    dialog = None

    def build(self):
        return Builder.load_string(KV)

    def show_simple_dialog(self):
        if not self.dialog:
            self.dialog = MDDialog(
                title="Set backup account",
                type="simple",
                items=[
                    Item(text="user01@gmail.com", source="user-1.png"),
                    Item(text="user02@gmail.com", source="user-2.png"),
                    Item(text="Add account", source="add-icon.png"),
                ],
            )
        self.dialog.open()

Example().run()
```



```
from kivy.lang import Builder

from kivymd.app import MDApp
from kivymd.uix.button import MDFlatButton
from kivymd.uix.dialog import MDDialog
from kivymd.uix.list import OneLineAvatarIconListItem

KV = '''
<ItemConfirm>
    on_release: root.set_icon(check)

    CheckboxLeftWidget:
        id: check
        group: "check"

FloatLayout:

    MDFlatButton:
        text: "ALERT DIALOG"
        pos_hint: {'center_x': .5, 'center_y': .5}
        on_release: app.show_confirmation_dialog()
'''
```

(continues on next page)

(continued from previous page)

```

class ItemConfirm(OneLineAvatarIconListItem):
    divider = None

    def set_icon(self, instance_check):
        instance_check.active = True
        check_list = instance_check.get_widgets(instance_check.group)
        for check in check_list:
            if check != instance_check:
                check.active = False

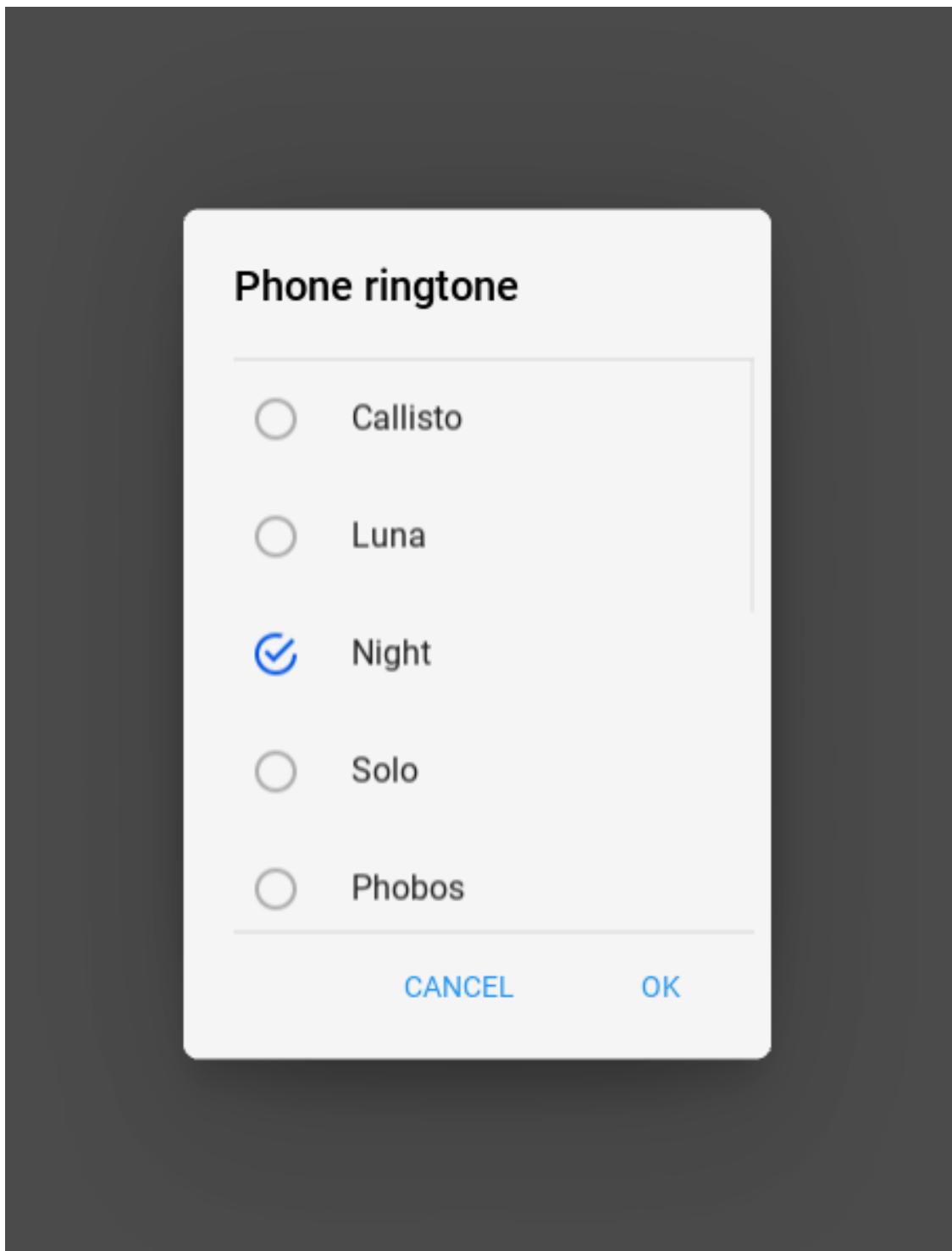

class Example(MDApp):
    dialog = None

    def build(self):
        return Builder.load_string(KV)

    def show_confirmation_dialog(self):
        if not self.dialog:
            self.dialog = MDDialog(
                title="Phone ringtone",
                type="confirmation",
                items=[
                    ItemConfirm(text="Callisto"),
                    ItemConfirm(text="Luna"),
                    ItemConfirm(text="Night"),
                    ItemConfirm(text="Solo"),
                    ItemConfirm(text="Phobos"),
                    ItemConfirm(text="Diamond"),
                    ItemConfirm(text="Sirena"),
                    ItemConfirm(text="Red music"),
                    ItemConfirm(text="Allergio"),
                    ItemConfirm(text="Magic"),
                    ItemConfirm(text="Tic-tac"),
                ],
                buttons=[
                    MDFlatButton(
                        text="CANCEL", text_color=self.theme_cls.primary_color
                    ),
                    MDFlatButton(
                        text="OK", text_color=self.theme_cls.primary_color
                    ),
                ],
            )
            self.dialog.open()

Example().run()

```



`items` is an `ListProperty` and defaults to `[]`.

**type**

Dialog type. Available option are `'alert'`, `'simple'`, `'confirmation'`, `'custom'`.

`type` is an `OptionProperty` and defaults to `'alert'`.

**content\_cls**

Custom content class.

```

from kivy.lang import Builder
from kivy.uix.boxlayout import BoxLayout

from kivymd.app import MDApp
from kivymd.uix.button import MDFlatButton
from kivymd.uix.dialog import MDDialog

KV = '''
<Content>
    orientation: "vertical"
    spacing: "12dp"
    size_hint_y: None
    height: "120dp"

    MDTextField:
        hint_text: "City"

    MDTextField:
        hint_text: "Street"

FloatLayout:

    MDFlatButton:
        text: "ALERT DIALOG"
        pos_hint: {'center_x': .5, 'center_y': .5}
        on_release: app.show_confirmation_dialog()
'''


class Content(BoxLayout):
    pass


class Example(MDApp):
    dialog = None

    def build(self):
        return Builder.load_string(KV)

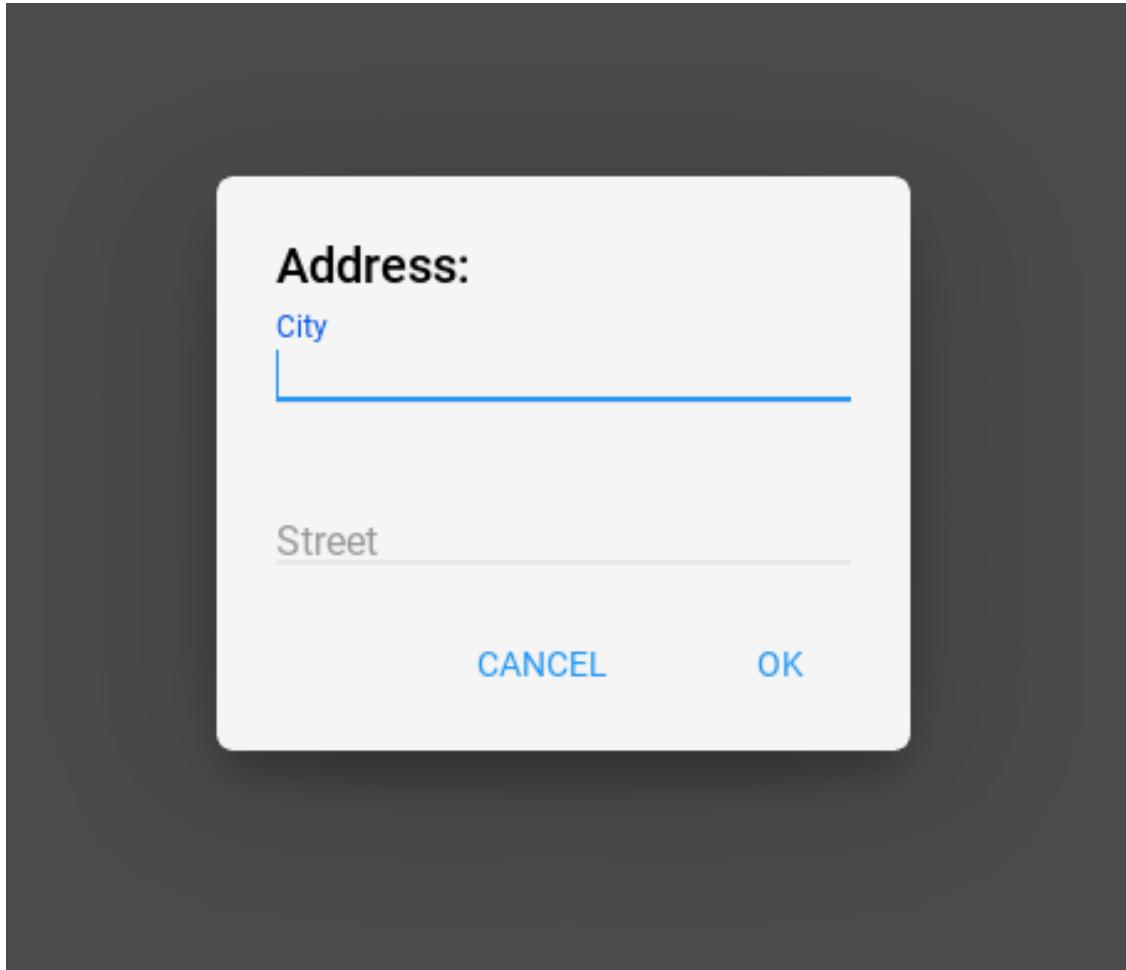
    def show_confirmation_dialog(self):
        if not self.dialog:
            self.dialog = MDDialog(
                title="Address:",
                type="custom",
                content_cls=Content(),
                buttons=[
                    MDFlatButton(
                        text="CANCEL", text_color=self.theme_cls.primary_color
                    ),
                    MDFlatButton(
                        text="OK", text_color=self.theme_cls.primary_color
                    ),
                ],
            )
            self.dialog.open()

```

(continues on next page)

(continued from previous page)

```
Example().run()
```



`content_cls` is an `ObjectProperty` and defaults to '`None`'.

```
update_height(self, *_)
on_open(self)
set_normal_height(self)
get_normal_height(self)
edit_padding_for_item(self, instance_item)
create_items(self)
create_buttons(self)
```

### 2.3.11 User Animation Card

#### Example

```

from kivymd.app import MDApp
from kivy.lang import Builder
from kivy.factory import Factory

from kivymd.toast import toast
from kivymd.theming import ThemeManager
from kivymd.uix.useranimationcard import MDUserAnimationCard
from kivymd.uix.button import MDIconButton
from kivymd.uix.list import ILeftBodyTouch

# Your content for a contact card.
Builder.load_string('''
#:import get_hex_from_color kivy.utils.get_hex_from_color


<TestAnimationCard@MDBoxLayout>
    orientation: 'vertical'
    padding: dp(10)
    spacing: dp(10)
    adaptive_height: True

    MDBBoxLayout:
        adaptive_height: True

        Widget:
        MDRoundFlatButton:
            text: "Free call"
        Widget:
        MDRoundFlatButton:
            text: "Free message"
        Widget:

        OneLineIconListItem:
            text: "Video call"
            IconLeftSampleWidget:
                icon: 'camera-front-variant'

        TwoLineIconListItem:
            text: "Call Viber Out"
            secondary_text: "[color=%s]Advantageous rates for calls[/color]" % get_hex_
            ↪from_color(app.theme_cls.primary_color)
            IconLeftSampleWidget:
                icon: 'phone'

        TwoLineIconListItem:
            text: "Call over mobile network"
            secondary_text: "[color=%s]Operator's tariffs apply[/color]" % get_hex_from_
            ↪color(app.theme_cls.primary_color)
            IconLeftSampleWidget:
                icon: 'remote'
''')

```

(continues on next page)

(continued from previous page)

```

class IconLeftSampleWidget(ILeftBodyTouch, MDIconButton):
    pass

class Example(MDApp):
    title = "Example Animation Card"

    def __init__(self, **kwargs):
        super().__init__(**kwargs)
        self.user_animation_card = None

    def build(self):
        def main_back_callback():
            toast('Close card')

        if not self.user_animation_card:
            self.user_animation_card = MDUserAnimationCard(
                user_name="Lion Lion",
                path_to_avatar="./assets/african-lion-951778_1280.jpg",
                callback=main_back_callback)
            self.user_animation_card.box_content.add_widget(
                Factory.TestAnimationCard())
        self.user_animation_card.open()

Example().run()

```

## API - kivymd.uix.useranimationcard

**class** kivymd.uix.useranimationcard.**MDUserAnimationCard**(*\*\*kwargs*)  
ModalView class. See module documentation for more information.

### Events

**on\_pre\_open:** Fired before the ModalView is opened. When this event is fired ModalView is not yet added to window.

**on\_open:** Fired when the ModalView is opened.

**on\_pre\_dismiss:** Fired before the ModalView is closed.

**on\_dismiss:** Fired when the ModalView is closed. If the callback returns True, the dismiss will be canceled.

Changed in version 1.11.0: Added events *on\_pre\_open* and *on\_pre\_dismiss*.

**user\_name**

**path\_to\_avatar**

**box\_content**

**callback**

**on\_open** (*self*)

**on\_touch\_move** (*self, touch*)

Receive a touch move event. The touch is in parent coordinates.

See [on\\_touch\\_down\(\)](#) for more information.

**on\_touch\_down** (*self, touch*)  
 Receive a touch down event.

#### Parameters

**touch: MotionEvent class** Touch received. The touch is in parent coordinates. See [relativelayout](#) for a discussion on coordinate systems.

**Returns** bool If True, the dispatching of the touch event will stop. If False, the event will continue to be dispatched to the rest of the widget tree.

**on\_touch\_up** (*self, touch*)  
 Receive a touch up event. The touch is in parent coordinates.

See [on\\_touch\\_down \(\)](#) for more information.

**animation\_to\_bottom** (*self*)

**animation\_to\_top** (*self*)

**class** kivymd.uix.useranimationcard.**UserAnimationCard** (\*\*kwargs)  
 Float layout class. See module documentation for more information.

**user\_name**

**path\_to\_avatar**

**class** kivymd.uix.useranimationcard.**ModifiedToolbar** (\*\*kwargs)  
 Widget class. See module documentation for more information.

#### Events

**on\_touch\_down: (touch, )** Fired when a new touch event occurs. *touch* is the touch object.

**on\_touch\_move: (touch, )** Fired when an existing touch moves. *touch* is the touch object.

**on\_touch\_up: (touch, )** Fired when an existing touch disappears. *touch* is the touch object.

**on\_kv\_post: (base\_widget, )** Fired after all the kv rules associated with the widget and all other widgets that are in any of those rules have had all their kv rules applied. *base\_widget* is the base-most widget whose instantiation triggered the kv rules (i.e. the widget instantiated from Python, e.g. `MyWidget ()`).

Changed in version 1.11.0.

**Warning:** Adding a `__del__` method to a class derived from Widget with Python prior to 3.4 will disable automatic garbage collection for instances of that class. This is because the Widget class creates reference cycles, thereby [preventing garbage collection](#).

Changed in version 1.0.9: Everything related to event properties has been moved to the [EventDispatcher](#). Event properties can now be used when contructing a simple class without subclassing Widget.

Changed in version 1.5.0: The constructor now accepts `on_*` arguments to automatically bind callbacks to properties or events, as in the Kv language.

**left\_action\_items**

**title**

**on\_left\_action\_items** (*self, instance, value*)

**update\_action\_bar** (*self, action\_bar, action\_bar\_items*)

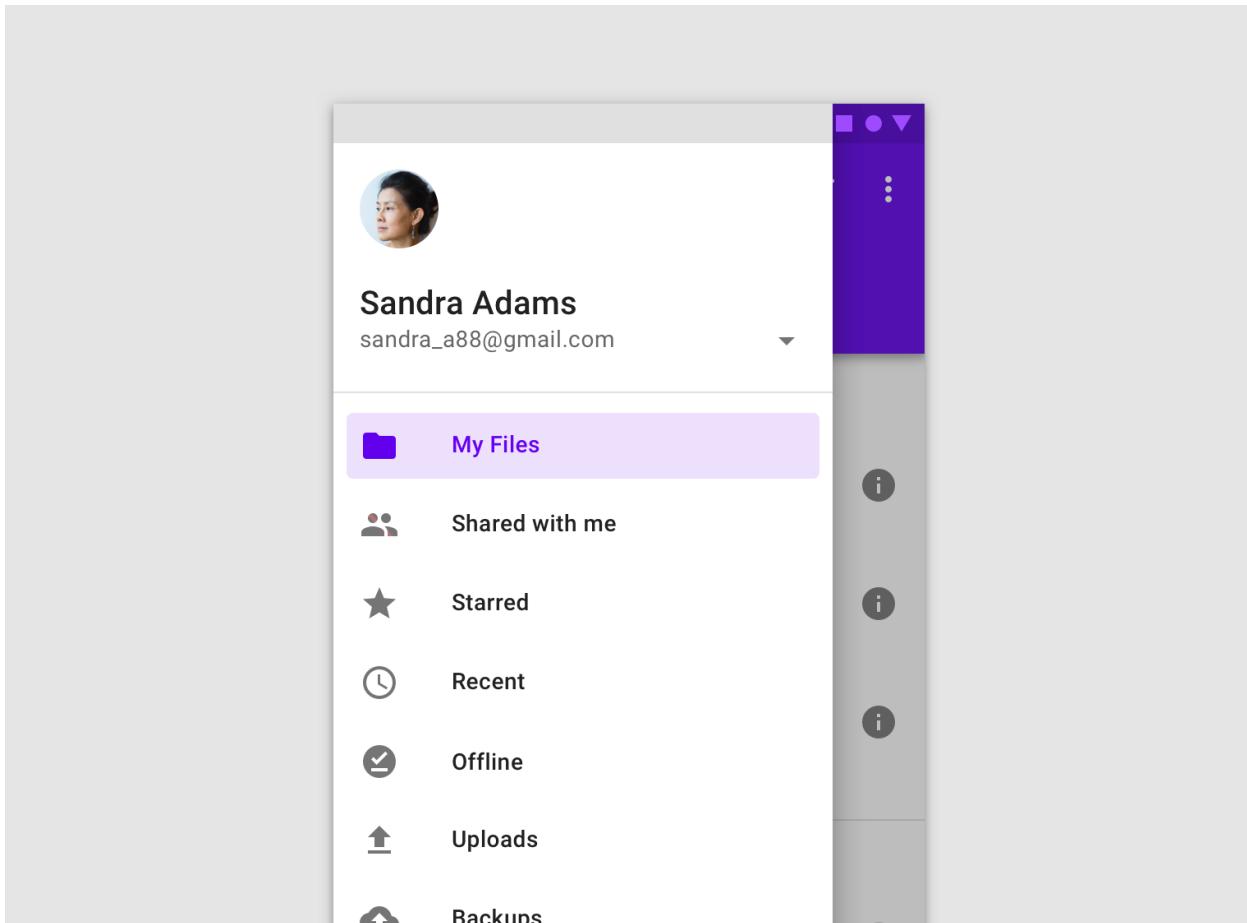
**update\_action\_bar\_text\_colors** (*self, instance, value*)

### 2.3.12 Navigation Drawer

See also:

Material Design spec, Navigation drawer

**Navigation drawers provide access to destinations in your app.**



When using the class `MDNavigationDrawer` skeleton of your KV markup should look like this:

```

Root :

    NavigationLayout :

        ScreenManager :

            Screen_1 :

            Screen_2 :

        MDNavigationDrawer :
            # This custom rule should implement what will be appear in your ↵
            ↵MDNavigationDrawer
            ContentNavigationDrawer

```

A simple example:

```
from kivy.lang import Builder
from kivy.uix.boxlayout import BoxLayout

from kivymd.app import MDApp

KV = '''
Screen:

    NavigationLayout:

        ScreenManager:

            Screen:

                BoxLayout:
                    orientation: 'vertical'

                    MDToolbar:
                        title: "Navigation Drawer"
                        elevation: 10
                        left_action_items: [ ['menu', lambda x: nav_drawer.toggle_nav_
→drawer() ] ]

                Widget:

            MDNavigationDrawer:
                id: nav_drawer

            ContentNavigationDrawer:
        '''

class ContentNavigationDrawer(BoxLayout):
    pass


class TestNavigationDrawer(MDApp):
    def build(self):
        return Builder.load_string(KV)

TestNavigationDrawer().run()
```

---

**Note:** `MDNavigationDrawer` is an empty `MDCard` panel.

---

Let's extend the `ContentNavigationDrawer` class from the above example and create content for our `MDNavigationDrawer` panel:

```
# Menu item in the DrawerList list.
<ItemDrawer>:
    theme_text_color: "Custom"
```

(continues on next page)

(continued from previous page)

```
on_release: self.parent.set_color_item(self)
```

```
IconLeftWidget:
    id: icon
    icon: root.icon
    theme_text_color: "Custom"
    text_color: root.text_color
```

```
class ItemDrawer(OneLineIconListItem):
    icon = StringProperty()
```



My files

Top of ContentNavigationDrawer and DrawerList for menu items:

```
<ContentNavigationDrawer>:
    orientation: "vertical"
    padding: "8dp"
    spacing: "8dp"

    AnchorLayout:
        anchor_x: "left"
        size_hint_y: None
        height: avatar.height

        Image:
            id: avatar
            size_hint: None, None
            size: "56dp", "56dp"
            source: "kivymd.png"

        MDLabel:
            text: "KivyMD library"
            font_style: "Button"
            size_hint_y: None
            height: self.texture_size[1]

        MDLabel:
            text: "kivydevelopment@gmail.com"
            font_style: "Caption"
            size_hint_y: None
            height: self.texture_size[1]

    ScrollView:

        DrawerList:
            id: md_list
```

```
class ContentNavigationDrawer(BoxLayout):
    pass
```

(continues on next page)

(continued from previous page)

```
class DrawerList(ThemableBehavior, MDList):
    def set_color_item(self, instance_item):
        '''Called when tap on a menu item.'''

        # Set the color of the icon and text for the menu item.
        for item in self.children:
            if item.text_color == self.theme_cls.primary_color:
                item.text_color = self.theme_cls.text_color
                break
        instance_item.text_color = self.theme_cls.primary_color
```

**KIVYMD LIBRARY**[kivydevelopment@gmail.com](mailto:kivydevelopment@gmail.com)

Create a menu list for ContentNavigationDrawer:

```
def on_start(self):
    icons_item = {
        "folder": "My files",
        "account-multiple": "Shared with me",
        "star": "Starred",
        "history": "Recent",
        "checkbox-marked": "Shared with me",
        "upload": "Upload",
    }
    for icon_name in icons_item.keys():
        self.root.ids.content_drawer.ids.md_list.add_widget(
            ItemDrawer(icon=icon_name, text=icons_item[icon_name]))
    )
```

**Switching screens in the ScreenManager and using the common MDToolbar**

```
from kivy.lang import Builder
from kivy.uix.boxlayout import BoxLayout
from kivy.properties import ObjectProperty

from kivymd.app import MDApp

KV = '''
<ContentNavigationDrawer>:

    ScrollView:
```

(continues on next page)

(continued from previous page)

```

MDList:

    OneLineListItem:
        text: "Screen 1"
        on_press:
            root.nav_drawer.set_state("close")
            root.screen_manager.current = "scr 1"

    OneLineListItem:
        text: "Screen 2"
        on_press:
            root.nav_drawer.set_state("close")
            root.screen_manager.current = "scr 2"

Screen:

MDToolbar:
    id: toolbar
    pos_hint: {"top": 1}
    elevation: 10
    title: "MDNavigationDrawer"
    left_action_items: [{"menu", lambda x: nav_drawer.set_state("open")}]]

NavigationLayout:
    x: toolbar.height

ScreenManager:
    id: screen_manager

    Screen:
        name: "scr 1"

        MDLabel:
            text: "Screen 1"
            halign: "center"

    Screen:
        name: "scr 2"

        MDLabel:
            text: "Screen 2"
            halign: "center"

MDNavigationDrawer:
    id: nav_drawer

ContentNavigationDrawer:
    screen_manager: screen_manager
    nav_drawer: nav_drawer
'''


class ContentNavigationDrawer(BoxLayout):
    screen_manager = ObjectProperty()
    nav_drawer = ObjectProperty()

```

(continues on next page)

(continued from previous page)

```
class TestNavigationDrawer(MDApp):
    def build(self):
        return Builder.load_string(KV)

TestNavigationDrawer().run()
```

## NavigationDrawer with type standard

You can use the standard behavior type for the NavigationDrawer:

```
MDNavigationDrawer:
    type: "standard"
```

### See also:

Full example of Components-Navigation-Drawer

### API - kivymd.uix.navigationdrawer

```
class kivymd.uix.navigationdrawer.NavigationLayout(**kwargs)
    Float layout class. See module documentation for more information.

    update_pos(self, *args)
    add_scrim(self, widget)
    update_scrim_rectangle(self, *args)
    add_widget(self, widget, index=0, canvas=None)
        Only two layouts are allowed: ScreenManager and MDNavigationDrawer.

class kivymd.uix.navigationdrawer.MDNavigationDrawer(**kwargs)
    Widget class. See module documentation for more information.
```

#### Events

- on\_touch\_down: (touch, )** Fired when a new touch event occurs. *touch* is the touch object.
- on\_touch\_move: (touch, )** Fired when an existing touch moves. *touch* is the touch object.
- on\_touch\_up: (touch, )** Fired when an existing touch disappears. *touch* is the touch object.
- on\_kv\_post: (base\_widget, )** Fired after all the kv rules associated with the widget and all other widgets that are in any of those rules have had all their kv rules applied. *base\_widget* is the base-most widget whose instantiation triggered the kv rules (i.e. the widget instantiated from Python, e.g. MyWidget ()).

Changed in version 1.11.0.

**Warning:** Adding a `__del__` method to a class derived from Widget with Python prior to 3.4 will disable automatic garbage collection for instances of that class. This is because the Widget class creates reference cycles, thereby preventing garbage collection.

Changed in version 1.0.9: Everything related to event properties has been moved to the `EventDispatcher`. Event properties can now be used when contructing a simple class without subclassing `Widget`.

Changed in version 1.5.0: The constructor now accepts `on_*` arguments to automatically bind callbacks to properties or events, as in the Kv language.

**type**

Type of drawer. Modal type will be on top of screen. Standard type will be at left or right of screen. Also it automatically disables `close_on_click` and `enable_swiping` to prevent closing drawer for standard type.

`type` is a `OptionProperty` and defaults to `modal`.

**anchor**

Anchoring screen edge for drawer. Set it to ‘right’ for right-to-left languages. Available options are: ‘left’, ‘right’.

`anchor` is a `OptionProperty` and defaults to `left`.

**close\_on\_click**

Close when click on scrim or keyboard escape. It automatically sets to False for “standard” type.

`close_on_click` is a `BooleanProperty` and defaults to `True`.

**state**

Indicates if panel closed or opened. Sets after `status` change. Available options are: ‘close’, ‘open’.

`state` is a `OptionProperty` and defaults to ‘close’.

**status**

Detailed state. Sets before `state`. Bind to `state` instead of `status`. Available options are: ‘closed’, ‘opening\_withSwipe’, ‘opening\_withAnimation’, ‘opened’, ‘closing\_withSwipe’, ‘closing\_withAnimation’.

`status` is a `OptionProperty` and defaults to ‘closed’.

**open\_progress**

Percent of visible part of side panel. The percent is specified as a floating point number in the range 0-1. 0.0 if panel is closed and 1.0 if panel is opened.

`open_progress` is a `NumericProperty` and defaults to `0.0`.

**enable\_swiping**

Allow to open or close navigation drawer with swipe. It automatically sets to False for “standard” type.

`enable_swiping` is a `BooleanProperty` and defaults to `True`.

**swipe\_distance**

The distance of the swipe with which the movement of navigation drawer begins.

`swipe_distance` is a `NumericProperty` and defaults to `10`.

**swipe\_edge\_width**

The size of the area in px inside which should start swipe to drag navigation drawer.

`swipe_edge_width` is a `NumericProperty` and defaults to `20`.

**scrim\_color**

Color for scrim. Alpha channel will be multiplied with `_scrim_alpha`. Set fourth channel to 0 if you want to disable scrim.

`scrim_color` is a `ListProperty` and defaults to `[0, 0, 0, 0.5]`.

**scrim\_alpha\_transition**

The name of the animation transition type to use for changing `scrim_alpha`.

`scrim_alpha_transition` is a `StringProperty` and defaults to ‘`linear`’.

**opening\_transition**

The name of the animation transition type to use when animating to the `state` ‘`open`’.

`opening_transition` is a `StringProperty` and defaults to ‘`out_cubic`’.

**opening\_time**

The time taken for the panel to slide to the `state` ‘`open`’.

`opening_time` is a `NumericProperty` and defaults to `0.2`.

**closing\_transition**

The name of the animation transition type to use when animating to the `state` ‘`close`’.

`closing_transition` is a `StringProperty` and defaults to ‘`out_sine`’.

**closing\_time**

The time taken for the panel to slide to the `state` ‘`close`’.

`closing_time` is a `NumericProperty` and defaults to `0.2`.

**on\_type(self, \*args)****set\_state(self, new\_state='toggle', animation=True)**

Change state of the side panel. New\_state can be one of “`toggle`”, “`open`” or “`close`”.

**toggle\_nav\_drawer(self)****update\_status(self, \*\_)****get\_dist\_from\_side(self, x)****on\_touch\_down(self, touch)**

Receive a touch down event.

**Parameters**

**touch: MotionEvent class** Touch received. The touch is in parent coordinates. See `relativelayout` for a discussion on coordinate systems.

**Returns** bool If True, the dispatching of the touch event will stop. If False, the event will continue to be dispatched to the rest of the widget tree.

**on\_touch\_move(self, touch)**

Receive a touch move event. The touch is in parent coordinates.

See `on_touch_down()` for more information.

**on\_touch\_up(self, touch)**

Receive a touch up event. The touch is in parent coordinates.

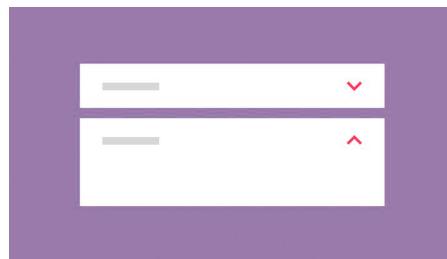
See `on_touch_down()` for more information.

### 2.3.13 Expansion Panel

See also:

Material Design spec, Expansion panel

**Expansion panels contain creation flows and allow lightweight editing of an element.**



#### Usage

```
self.add_widget(  
    MDExpansionPanel(  
        icon="logo.png", # panel icon  
        content=Content(), # panel content  
        panel_cls=MDExpansionPanelOneLine(text="Secondary text"), # panel class  
    )  
)
```

To use `MDExpansionPanel` you must pass one of the following classes to the `panel_cls` parameter:

- `MDExpansionPanelOneLine`
- `MDExpansionPanelTwoLine`
- `MDExpansionPanelThreeLine`

These classes are inherited from the following classes:

- `OneLineAvatarIconListItem`
- `TwoLineAvatarIconListItem`
- `ThreeLineAvatarIconListItem`

```
self.root.ids.box.add_widget(  
    MDExpansionPanel(  
        icon="logo.png",  
        content=Content(),  
        panel_cls=MDExpansionPanelThreeLine(  
            text="Text",  
            secondary_text="Secondary text",  
            tertiary_text="Tertiary text",  
        )  
    )  
)
```

## Example

```

from kivy.lang import Builder

from kivymd.app import MDApp
from kivymd.uix.boxlayout import MDBBoxLayout
from kivymd.uix.expansionpanel import MDExpansionPanel, MDExpansionPanelThreeLine
from kivymd import images_path

KV = """
<Content>
    adaptive_height: True

    TwoLineIconListItem:
        text: "(050)-123-45-67"
        secondary_text: "Mobile"

        IconLeftWidget:
            icon: 'phone'

ScrollView:

    MDGridLayout:
        id: box
        cols: 1
        adaptive_height: True
"""

class Content(MDBBoxLayout):
    '''Custom content.'''


class Test(MDApp):
    def build(self):
        return Builder.load_string(KV)

    def on_start(self):
        for i in range(10):
            self.root.ids.box.add_widget(
                MDExpansionPanel(
                    icon=f'{images_path}/kivymd.png',
                    content=Content(),
                    panel_cls=MDExpansionPanelThreeLine(
                        text="Text",
                        secondary_text="Secondary text",
                        tertiary_text="Tertiary text",
                    )
                )
            )

Test().run()

```

## Two events are available for MDExpansionPanel

- `on_open`
- `on_close`

### `MDExpansionPanel`:

```
    on_open: app.on_panel_open(args)
    on_close: app.on_panel_close(args)
```

The user function takes one argument - the object of the panel:

```
def on_panel_open(self, instance_panel):
    print(instance_panel)
```

### See also:

See Expansion panel example

Expansion panel and MDCard

## API - kivymd.uix.expansionpanel

```
class kivymd.uix.expansionpanel.MDExpansionPanelOneLine(**kwargs)
    Single line panel.

class kivymd.uix.expansionpanel.MDExpansionPanelTwoLine(**kwargs)
    Two-line panel.

class kivymd.uix.expansionpanel.MDExpansionPanelThreeLine(**kwargs)
    Three-line panel.

class kivymd.uix.expansionpanel.MDExpansionPanel(**kwargs)
```

### Events

`on_open` Called when a panel is opened.

`on_close` Called when a panel is closed.

### content

Content of panel. Must be *Kivy* widget.

`content` is an `ObjectProperty` and defaults to `None`.

### icon

Icon of panel.

Icon Should be either be a path to an image or a logo name in `md_icons`

`icon` is an `StringProperty` and defaults to ‘’.

### opening\_transition

The name of the animation transition type to use when animating to the state ‘open’.

`opening_transition` is a `StringProperty` and defaults to ‘out\_cubic’.

### opening\_time

The time taken for the panel to slide to the state ‘open’.

`opening_time` is a `NumericProperty` and defaults to `0.2`.

**closing\_transition**

The name of the animation transition type to use when animating to the state ‘close’.

*closing\_transition* is a `StringProperty` and defaults to ‘*out\_sine*’.

**closing\_time**

The time taken for the panel to slide to the state ‘close’.

*closing\_time* is a `NumericProperty` and defaults to *0.2*.

**panel\_cls**

Panel object. The object must be one of the classes `MDExpansionPanelOneLine`, `MDExpansionPanelTwoLine` or `MDExpansionPanelThreeLine`.

*panel\_cls* is a `ObjectProperty` and defaults to *None*.

**on\_open(self, \*args)**

Called when a panel is opened.

**on\_close(self, \*args)**

Called when a panel is closed.

**check\_open\_panel(self, instance)**

Called when you click on the panel. Called methods to open or close a panel.

**set\_chevron\_down(self)**

Sets the chevron down.

**set\_chevron\_up(self, instance\_chevron)**

Sets the chevron up.

**close\_panel(self, instance\_panel)**

Method closes the panel.

**open\_panel(self, \*args)**

Method opens a panel.

**add\_widget(self, widget, index=0, canvas=None)**

Add a new widget as a child of this widget.

**Parameters**

**widget: Widget** Widget to add to our list of children.

**index: int, defaults to 0** Index to insert the widget in the list. Notice that the default of 0 means the widget is inserted at the beginning of the list and will thus be drawn on top of other sibling widgets. For a full discussion of the index and widget hierarchy, please see the [Widgets Programming Guide](#).

New in version 1.0.5.

**canvas: str, defaults to None** Canvas to add widget’s canvas to. Can be ‘before’, ‘after’ or None for the default canvas.

New in version 1.9.0.

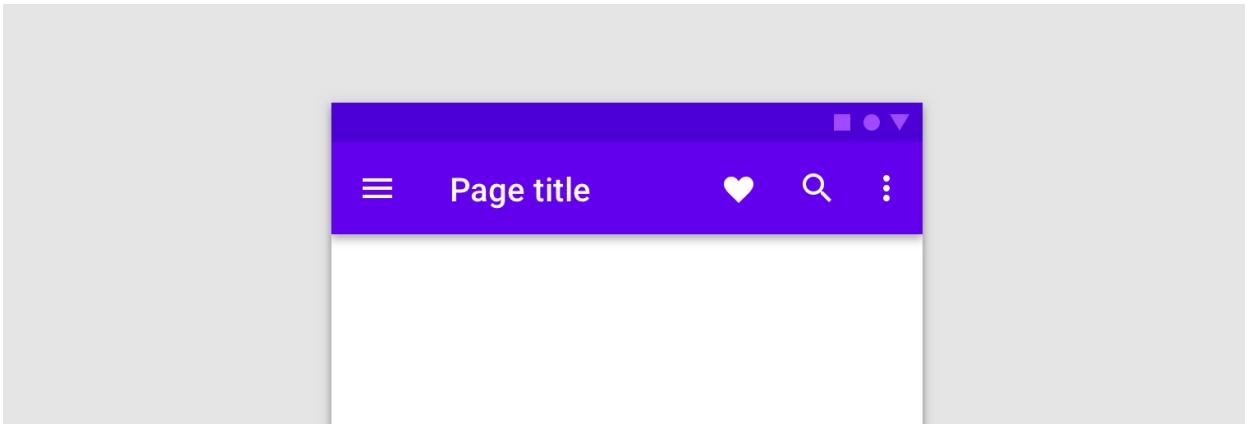
```
>>> from kivy.uix.button import Button
>>> from kivy.uix.slider import Slider
>>> root = Widget()
>>> root.add_widget(Button())
>>> slider = Slider()
>>> root.add_widget(slider)
```

## 2.3.14 Toolbar

See also:

[Material Design spec, App bars: top](#)

[Material Design spec, App bars: bottom](#)



KivyMD provides the following toolbar positions for use:

- *Top*
- *Bottom*

### Top

```
from kivy.lang import Builder

from kivymd.app import MDApp

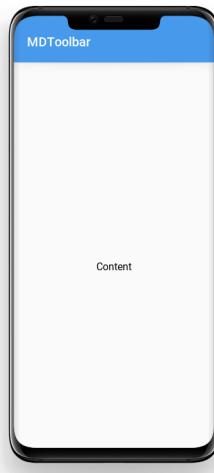
KV = '''
BoxLayout:
    orientation: "vertical"

    MDToolbar:
        title: "MDToolbar"

    MDLabel:
        text: "Content"
        halign: "center"
    '''

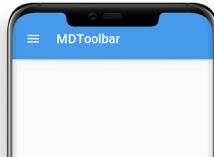
class Test(MDApp):
    def build(self):
        return Builder.load_string(KV)

Test().run()
```



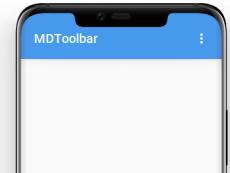
### Add left menu

```
MDToolbar:  
    title: "MDToolbar"  
    left_action_items: [ ["menu", lambda x: app.callback() ] ]
```



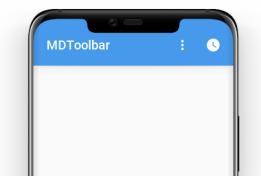
### Add right menu

```
MDToolbar:  
    title: "MDToolbar"  
    right_action_items: [ ["dots-vertical", lambda x: app.callback() ] ]
```



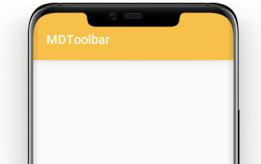
### Add two item to the right menu

```
MDToolbar:  
    title: "MDToolbar"  
    right_action_items: [ ["dots-vertical", lambda x: app.callback_1()], ["clock",  
    ↪lambda x: app.callback_2()]]
```



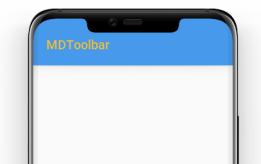
### Change toolbar color

```
MDToolbar:  
    title: "MDToolbar"  
    md_bg_color: app.theme_cls.accent_color
```



### Change toolbar text color

```
MDToolbar:  
    title: "MDToolbar"  
    specific_text_color: app.theme_cls.accent_color
```

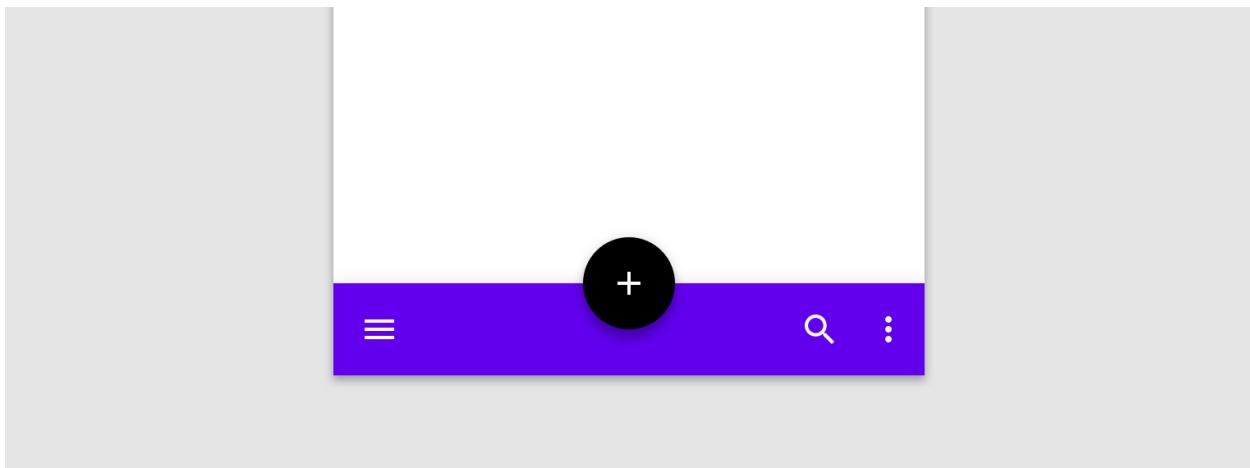


### Shadow elevation control

```
MDToolbar:  
    title: "Elevation 10"  
    elevation: 10
```



## Bottom



## Usage

```
from kivy.lang import Builder

from kivymd.app import MDApp

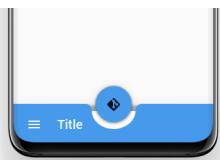
KV = '''
BoxLayout:

    # Will always be at the bottom of the screen.
    MDBottomAppBar:

        MDToolbar:
            title: "Title"
            icon: "git"
            type: "bottom"
            left_action_items: [["menu", lambda x: x]]
'''

class Test(MDApp):
    def build(self):
        return Builder.load_string(KV)

Test().run()
```



## Event on floating button

Event on\_action\_button:

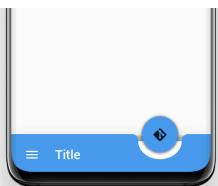
```
MDBottomAppBar:  
    MDToolbar:  
        title: "Title"  
        icon: "git"  
        type: "bottom"  
        left_action_items: [ ["menu", lambda x: x]]  
        on_action_button: app.callback(self.icon)
```

## Floating button position

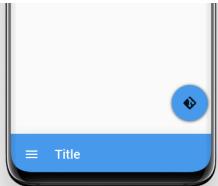
Mode:

- '*free-end*'
- '*free-center*'
- '*end*'
- '*center*'

```
MDBottomAppBar:  
    MDToolbar:  
        title: "Title"  
        icon: "git"  
        type: "bottom"  
        left_action_items: [ ["menu", lambda x: x]]  
        mode: "end"
```



```
MDBottomAppBar:  
    MDToolbar:  
        title: "Title"  
        icon: "git"  
        type: "bottom"  
        left_action_items: [ ["menu", lambda x: x]]  
        mode: "free-end"
```



**See also:**

[Components-Bottom-AppBar](#)

**API - kivymd.uix.toolbar**

```
class kivymd.uix.toolbar.MDActionBottomAppBarButton(**kwargs)
    Abstract base class for all round buttons, bringing in the appropriate on-touch behavior

class kivymd.uix.toolbar.MToolbar(**kwargs)
```

**Events**

**on\_action\_button** Method for the button used for the [MDBottomAppBar](#) class.

**elevation**

Elevation value.

`elevation` is an [NumericProperty](#) and defaults to `6`.

**left\_action\_items**

The icons on the left of the toolbar. To add one, append a list like the following:

```
left_action_items: [icon_name, callback]
```

where `'icon_name'` is a string that corresponds to an icon definition and `callback` is the function called on a touch release event.

`left_action_items` is an [ListProperty](#) and defaults to `[]`.

**right\_action\_items**

The icons on the right of the toolbar. Works the same way as `left_action_items`.

`right_action_items` is an [ListProperty](#) and defaults to `[]`.

**title**

Text toolbar.

`title` is an [StringProperty](#) and defaults to `''`.

**md\_bg\_color**

Color toolbar.

`md_bg_color` is an [ListProperty](#) and defaults to `[0, 0, 0, 0]`.

**anchor\_title****mode**

Floating button position. Only for [MDBottomAppBar](#) class. Available options are: `'free-end'`, `'free-center'`, `'end'`, `'center'`.

`mode` is an [OptionProperty](#) and defaults to `'center'`.

**round**

Rounding the corners at the notch for a button. Only for [MDBottomAppBar](#) class.

`round` is an [NumericProperty](#) and defaults to `'10dp'`.

**icon**

Floating button. Only for [MDBottomAppBar](#) class.

`icon` is an [StringProperty](#) and defaults to `'android'`.

**icon\_color**

Color action button. Only for `MDBottomAppBar` class.

`icon_color` is an `ListProperty` and defaults to `[]`.

**type**

When using the `MDBottomAppBar` class, the parameter `type` must be set to ‘bottom’:

```
MDBottomAppBar:
```

```
MDToolbar:  
    type: "bottom"
```

Available options are: ‘top’, ‘bottom’.

`type` is an `OptionProperty` and defaults to ‘top’.

`on_action_button(self, *args)`

`on_md_bg_color(self, instance, value)`

`on_left_action_items(self, instance, value)`

`on_right_action_items(self, instance, value)`

`update_action_bar(self, action_bar, action_bar_items)`

`update_action_bar_text_colors(self, instance, value)`

`on_icon(self, instance, value)`

`on_icon_color(self, instance, value)`

`on_mode(self, instance, value)`

`remove_notch(self)`

`set_notch(self)`

`remove_shadow(self)`

`set_shadow(self, *args)`

**class** `kivymd.uix.toolbar.MDBottomAppBar(**kwargs)`

Float layout class. See module documentation for more information.

`add_widget(self, widget, index=0, canvas=None)`

Add a new widget as a child of this widget.

### Parameters

`widget: Widget` Widget to add to our list of children.

`index: int, defaults to 0` Index to insert the widget in the list. Notice that the default of 0 means the widget is inserted at the beginning of the list and will thus be drawn on top of other sibling widgets. For a full discussion of the index and widget hierarchy, please see the [Widgets Programming Guide](#).

New in version 1.0.5.

`canvas: str, defaults to None` Canvas to add widget’s canvas to. Can be ‘before’, ‘after’ or None for the default canvas.

New in version 1.9.0.

```
>>> from kivy.uix.button import Button
>>> from kivy.uix.slider import Slider
>>> root = Widget()
>>> root.add_widget(Button())
>>> slider = Slider()
>>> root.add_widget(slider)
```

### 2.3.15 Menu

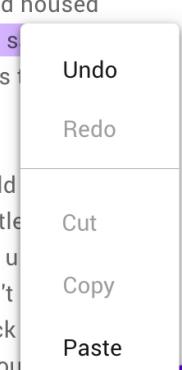
See also:

[Material Design spec, Menus](#)

**Menus display a list of choices on temporary surfaces.**

es lay spread out on the table - Samsa was a travelling salesman - and above a picture that he had recently cut out of an illustrated magazine and housed in a gold frame. It showed a lady fitted out with a fur hat and fur boa who was sitting a heavy fur muff that covered the whole of her lower arm towards the right.

urned to look out the window at the dull weather. Drops of rain could be seen falling on the pane, which made him feel quite sad. "How about if I sleep a little longer", he thought, but that was something he was used to doing. He was used to sleeping on his right, and in his present state couldn't do otherwise. However hard he threw himself onto his right, he always rolled back onto his left. He must have tried it a hundred times, shut his eyes so that he would not notice the floundering legs, and only stopped when he began to feel a mild, dull pain like he had never felt before.



### Usage

```
from kivy.lang import Builder

from kivymd.app import MDApp
from kivymd.uix.menu import MDDropdownMenu

KV = '''
Screen:

    MDRaisedButton:
        id: button
        text: "PRESS ME"
        pos_hint: {"center_x": .5, "center_y": .5}
        on_release: app.menu.open()
'''
```

(continues on next page)

(continued from previous page)

```
'''
```

```

class Test(MDApp):
    def __init__(self, **kwargs):
        super().__init__(**kwargs)
        self.screen = Builder.load_string(KV)
        menu_items = [{"text": f"Item {i}"} for i in range(5)]
        self.menu = MDDropdownMenu(
            caller=self.screen.ids.button,
            items=menu_items,
            width_mult=4,
        )
        self.menu.bind(on_release=self.menu_callback)

    def menu_callback(self, instance_menu, instance_menu_item):
        print(instance_menu, instance_menu_item)

    def build(self):
        return self.screen

Test().run()

```

**Warning:** Do not create the `MDDropdownMenu` object when you open the menu window. Because on a mobile device this one will be very slow!

## Wrong

```
menu = MDDropdownMenu(caller=self.screen.ids.button, items=menu_items)
menu.open()
```

## Customization of menu item

You must create a new class that inherits from the `RightContent` class:

```
class RightContentCls(RightContent):
    pass
```

Now in the KV rule you can create your own elements that will be displayed in the menu item on the right:

```
<RightContentCls>
    disabled: True

    MDIconButton:
        icon: root.icon
```

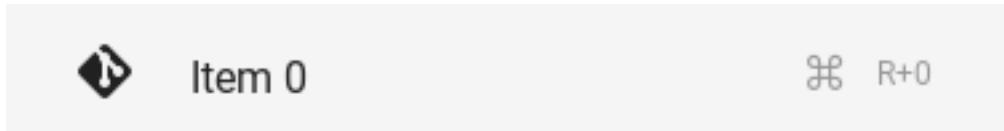
(continues on next page)

(continued from previous page)

```

        user_font_size: "16sp"
        pos_hint: {"center_y": .5}

MDLabel:
    text: root.text
    font_style: "Caption"
    size_hint_x: None
    width: self.texture_size[0]
    text_size: None, None
```



Now create menu items as usual, but add the key `right_content_cls` whose value is the class `RightContentCls` that you created:

```

menu_items = [
    {
        "right_content_cls": RightContentCls(
            text=f"R+{i}", icon="apple-keyboard-command",
        ),
        "icon": "git",
        "text": f"Item {i}",
    }
    for i in range(5)
]
self.menu = MDDropdownMenu(
    caller=self.screen.ids.button, items=menu_items, width_mult=4
)
```

## Full example

```

from kivy.lang import Builder

from kivymd.app import MDApp
from kivymd.uix.menu import MDDropdownMenu, RightContent

KV = '''
<RightContentCls>
    disabled: True

    MDIconButton:
        icon: root.icon
        user_font_size: "16sp"
        pos_hint: {"center_y": .5}

    MDLabel:
        text: root.text
        font_style: "Caption"
        size_hint_x: None
        width: self.texture_size[0]
        text_size: None, None
```

(continues on next page)

(continued from previous page)

Screen:

```

MDRaisedButton:
    id: button
    text: "PRESS ME"
    pos_hint: {"center_x": .5, "center_y": .5}
    on_release: app.menu.open()
    ...

class RightContentCls(RightContent):
    pass

class Test(MDApp):
    def __init__(self, **kwargs):
        super().__init__(**kwargs)
        self.screen = Builder.load_string(KV)
        menu_items = [
            {
                "right_content_cls": RightContentCls(
                    text=f"R+{i}", icon="apple-keyboard-command",
                ),
                "icon": "git",
                "text": f"Item {i}",
            }
            for i in range(5)
        ]
        self.menu = MDDropdownMenu(
            caller=self.screen.ids.button, items=menu_items, width_mult=4
        )
        self.menu.bind(on_release=self.menu_callback)

    def menu_callback(self, instance_menu, instance_menu_item):
        instance_menu.dismiss()

    def build(self):
        return self.screen

Test().run()

```

## Menu without icons on the left

If you do not want to use the icons in the menu items on the left, then do not use the “icon” key when creating menu items:

```

menu_items = [
    {
        "right_content_cls": RightContentCls(
            text=f"R+{i}", icon="apple-keyboard-command",
        ),
        "text": f"Item {i}",
    }
]

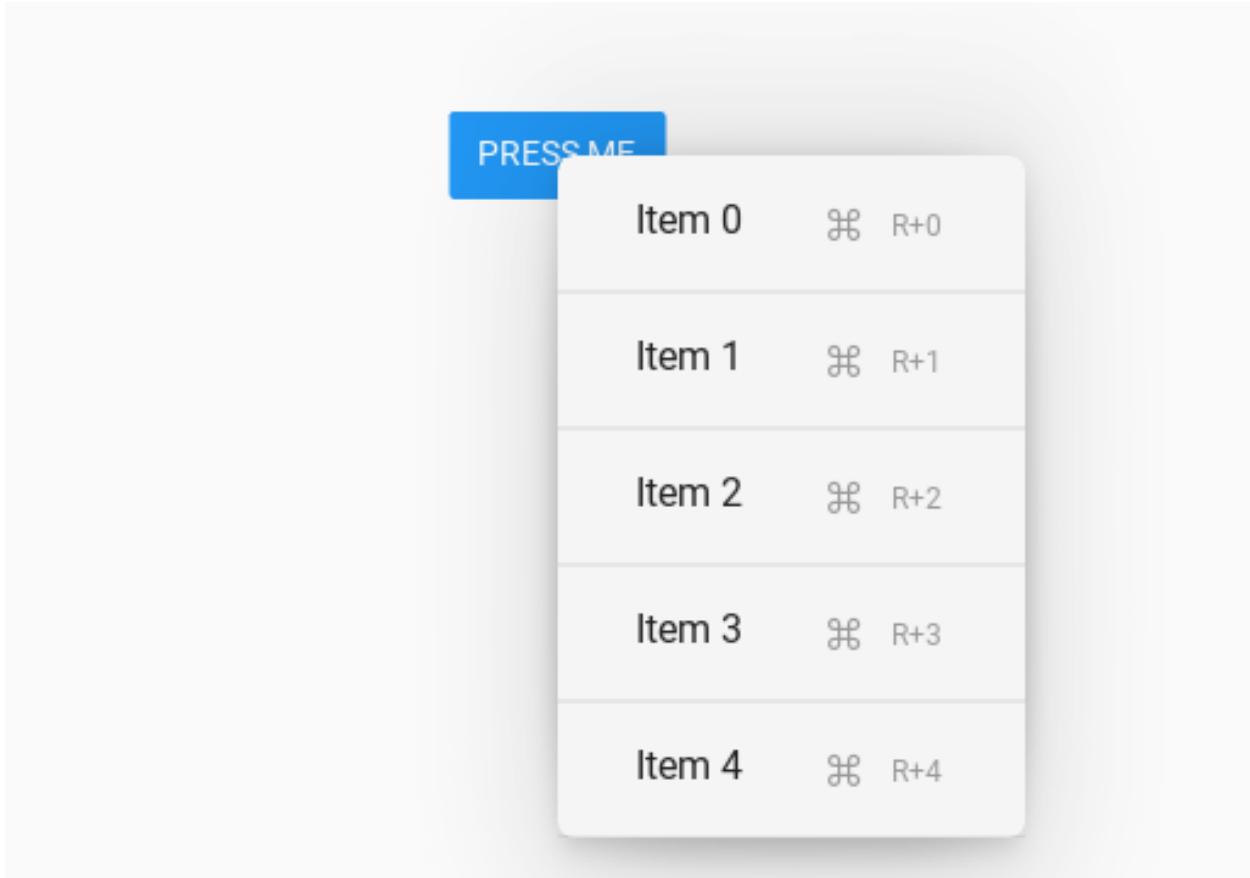
```

(continues on next page)

(continued from previous page)

```

    }
    for i in range(5)
]
```



### Item height adjustment

```

menu_items = [
    {
        "right_content_cls": RightContentCls(
            text=f"R+{i}",
            icon="apple-keyboard-command",
        ),
        "text": f"Item {i}",
        "height": "36dp",
        "top_pad": "10dp",
        "bot_pad": "10dp",
    }
    for i in range(5)
]
```



## Mixin items

```

from kivy.lang import Builder

from kivymd.app import MDApp
from kivymd.uix.menu import MDDropdownMenu, RightContent

KV = '''
<RightContentCls>
    disabled: True

    MDIconButton:
        icon: root.icon
        user_font_size: "16sp"
        pos_hint: {"center_y": .5}

    MDLabel:
        text: root.text
        font_style: "Caption"
        size_hint_x: None
        width: self.texture_size[0]
        text_size: None, None

Screen:

    MDRaisedButton:
        id: button
        text: "PRESS ME"
        pos_hint: {"center_x": .5, "center_y": .5}
        on_release: app.menu.open()
    ...

class RightContentCls(RightContent):
    pass

class Test(MDApp):
    def __init__(self, **kwargs):
        super().__init__(**kwargs)
        self.screen = Builder.load_string(KV)

        menu_items = []
        data = [
            {"": "Open"},
            {},
            {"open-in-app": "Open in app >"},
            {"trash-can-outline": "Move to Trash"},

    ...

```

(continues on next page)

(continued from previous page)

```

        {"rename_box": "Rename"},  

        {"zip_box_outline": "Create zip"},  

        {},  

        {"": "Properties"},  

    ]  
  

    for data_item in data:  

        if data_item:  

            if list(data_item.items())[0][1].endswith(">"):  

                menu_items.append(  

                    {  

                        "right_content_cls": RightContentCls(  

                            icon="menu-right-outline",  

                        ),  

                        "icon": list(data_item.items())[0][0],  

                        "text": list(data_item.items())[0][1][:-2],  

                        "height": "36dp",  

                        "top_pad": "10dp",  

                        "bot_pad": "10dp",  

                        "divider": None,  

                    }  

                )  

            )  

        else:  

            menu_items.append(  

                {  

                    "text": list(data_item.items())[0][1],  

                    "icon": list(data_item.items())[0][0],  

                    "font_style": "Caption",  

                    "height": "36dp",  

                    "top_pad": "10dp",  

                    "bot_pad": "10dp",  

                    "divider": None,  

                }  

            )  

        )  

    else:  

        menu_items.append(  

            {"viewclass": "MDSeparator", "height": 1}  

        )  

self.menu = MDDropdownMenu(  

    caller=self.screen.ids.button,  

    items=menu_items,  

    width_mult=4,  

)
self.menu.bind(on_release=self.menu_callback)  
  

def menu_callback(self, instance_menu, instance_menu_item):  

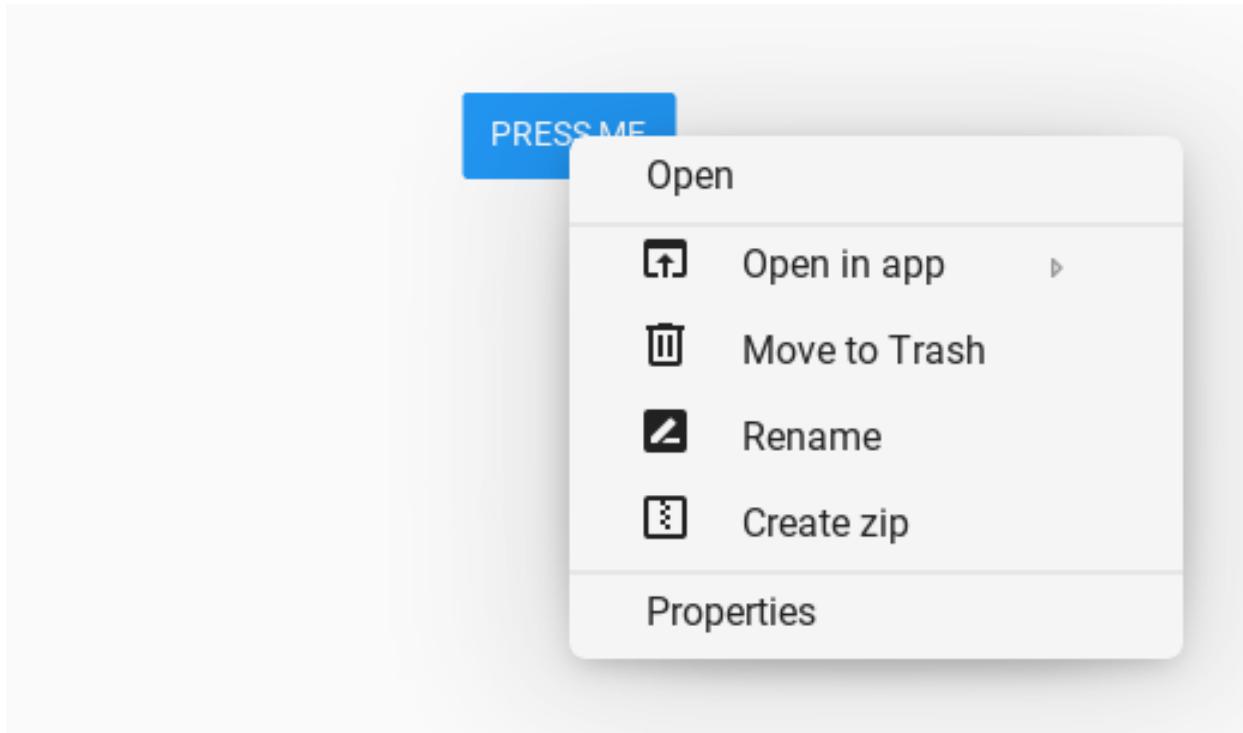
    print(instance_menu, instance_menu_item)  
  

def build(self):  

    return self.screen

```

Test().run()



### Hover Behavior

```
self.menu = MDDropdownMenu(  
    ...,  
    ...,  
    selected_color=self.theme_cls.primary_dark_hue,  
)
```

### Create submenu

```
from kivy.lang import Builder  
  
from kivymd.app import MDApp  
from kivymd.uix.menu import MDDropdownMenu  
  
KV = '''  
Screen:  
  
    MDRaisedButton:  
        id: button  
        text: "PRESS ME"  
        pos_hint: {"center_x": .5, "center_y": .5}  
        on_release: app.menu.open()  
'''
```

(continues on next page)

(continued from previous page)

```

class CustomDrop(MDDropdownMenu):
    def set_bg_color_items(self, instance_selected_item):
        if self.selected_color and not MDApp.get_running_app().submenu:
            for item in self.menu.ids.box.children:
                if item is not instance_selected_item:
                    item.bg_color = (0, 0, 0, 0)
        else:
            instance_selected_item.bg_color = self.selected_color

class Test(MDApp):
    submenu = None

    def __init__(self, **kwargs):
        super().__init__(**kwargs)
        self.screen = Builder.load_string(KV)
        menu_items = [
            {
                "icon": "git",
                "text": f"Item {i}" if i != 3 else "Open submenu",
            }
            for i in range(5)
        ]
        self.menu = CustomDrop(
            caller=self.screen.ids.button,
            items=menu_items,
            width_mult=4,
            selected_color=self.theme_cls.bg_darkest
        )
        self.menu.bind(on_enter=self.check_item)

    def check_item(self, menu, item):
        if item.text == "Open submenu" and not self.submenu:
            menu_items = [{"text": f"Item {i}"} for i in range(5)]
            self.submenu = MDDropdownMenu(
                caller=item,
                items=menu_items,
                width_mult=4,
                selected_color=self.theme_cls.bg_darkest,
            )
            self.submenu.bind(on_dismiss=self.set_state_submenu)
            self.submenu.open()

    def set_state_submenu(self, *args):
        self.submenu = None

    def build(self):
        return self.screen

Test().run()

```

## Menu with MDToolbar

**Warning:** The `MDDropdownMenu` does not work with the standard `MDToolbar`. You can use your own `CustomToolbar` and bind the menu window output to its elements.

```
from kivy.lang import Builder

from kivymd.app import MDApp
from kivymd.uix.menu import MDDropdownMenu
from kivymd.theming import ThemableBehavior
from kivymd.uix.behaviors import RectangularElevationBehavior
from kivymd.uix.boxlayout import MDBBoxLayout

KV = '''
<CustomToolbar>:
    size_hint_y: None
    height: self.theme_cls.standard_increment
    padding: "5dp"
    spacing: "12dp"

    MDIconButton:
        id: button_1
        icon: "menu"
        pos_hint: {"center_y": .5}
        on_release: app.menu_1.open()

    MDLabel:
        text: "MDDropdownMenu"
        pos_hint: {"center_y": .5}
        size_hint_x: None
        width: self.texture_size[0]
        text_size: None, None
        font_style: 'H6'

    Widget:

    MDIconButton:
        id: button_2
        icon: "dots-vertical"
        pos_hint: {"center_y": .5}
        on_release: app.menu_2.open()

Screen:
    CustomToolbar:
        id: toolbar
        elevation: 10
        pos_hint: {"top": 1}
    ...

class CustomToolbar(
    ThemableBehavior, RectangularElevationBehavior, MDBBoxLayout,
):
```

(continues on next page)

(continued from previous page)

```

def __init__(self, **kwargs):
    super().__init__(**kwargs)
    self.md_bg_color = self.theme_cls.primary_color


class Test(MDApp):
    def __init__(self, **kwargs):
        super().__init__(**kwargs)
        self.screen = Builder.load_string(KV)
        self.menu_1 = self.create_menu(
            "Button menu", self.screen.ids.toolbar.ids.button_1,
        )
        self.menu_2 = self.create_menu(
            "Button dots", self.screen.ids.toolbar.ids.button_2,
        )

    def create_menu(self, text, instance):
        menu_items = [{"icon": "git", "text": text} for i in range(5)]
        menu = MDDropdownMenu(caller=instance, items=menu_items, width_mult=5)
        menu.bind(on_release=self.menu_callback)
        return menu

    def menu_callback(self, instance_menu, instance_menu_item):
        instance_menu.dismiss()

    def build(self):
        return self.screen


Test().run()

```

## Position menu

### Bottom position

See also:

*position*

```

from kivy.clock import Clock
from kivy.lang import Builder

from kivymd.app import MDApp
from kivymd.uix.menu import MDDropdownMenu

KV = '''
Screen

    MDTextField:
        id: field
        pos_hint: {'center_x': .5, 'center_y': .5}
        size_hint_x: None
        width: "200dp"

```

(continues on next page)

(continued from previous page)

```

        hint_text: "Password"
        on_focus: if self.focus: app.menu.open()
    ...

class Test(MDApp):
    def __init__(self, **kwargs):
        super().__init__(**kwargs)
        self.screen = Builder.load_string(KV)
        menu_items = [{"icon": "git", "text": f"Item {i}"} for i in range(5)]
        self.menu = MDDropdownMenu(
            caller=self.screen.ids.field,
            items=menu_items,
            position="bottom",
            width_mult=4,
        )
        self.menu.bind(on_release=self.set_item)

    def set_item(self, instance_menu, instance_menu_item):
        def set_item(interval):
            self.screen.ids.field.text = instance_menu_item.text
            instance_menu.dismiss()
        Clock.schedule_once(set_item, 0.5)

    def build(self):
        return self.screen

Test().run()

```

## Center position

```

from kivy.lang import Builder

from kivymd.app import MDApp
from kivymd.uix.menu import MDDropdownMenu

KV = """
Screen

    MDDropDownItem:
        id: drop_item
        pos_hint: {'center_x': .5, 'center_y': .5}
        text: 'Item 0'
        on_release: app.menu.open()
    ...

class Test(MDApp):
    def __init__(self, **kwargs):
        super().__init__(**kwargs)
        self.screen = Builder.load_string(KV)
        menu_items = [{"icon": "git", "text": f"Item {i}"} for i in range(5)]

```

(continues on next page)

(continued from previous page)

```

self.menu = MDDropdownMenu(
    caller=self.screen.ids.drop_item,
    items=menu_items,
    position="center",
    width_mult=4,
)
self.menu.bind(on_release=self.set_item)

def set_item(self, instance_menu, instance_menu_item):
    self.screen.ids.drop_item.set_item(instance_menu_item.text)
    self.menu.dismiss()

def build(self):
    return self.screen

Test().run()

```

**API - kivymd.uix.menu****class kivymd.uix.menu.RightContent(\*\*kwargs)**

Same as IRigidBody, but allows the widget to receive touch events instead of triggering the ListItem's ripple effect

**text**

Text item.

*text* is a StringProperty and defaults to ''.

**icon**

Icon item.

*icon* is a StringProperty and defaults to ''.

**class kivymd.uix.menu.MDDropdownMenu(\*\*kwargs)****Events**

**on\_enter** Call when mouse enter the bbox of item menu.

**on\_leave** Call when the mouse exit the item menu.

**on\_dismiss** Call when closes menu.

**on\_release** The method that will be called when you click menu items.

**selected\_color**

Custom color (rgba format) for list item when hover behavior occurs.

*selected\_color* is a ListProperty and defaults to [].

**items**

See `data`.

*items* is a ListProperty and defaults to [].

**width\_mult**

This number multiplied by the standard increment (56dp on mobile, 64dp on desktop, determines the width of the menu items.

If the resulting number were to be too big for the application Window, the multiplier will be adjusted for the biggest possible one.

*width\_mult* is a `NumericProperty` and defaults to *1*.

**max\_height**

The menu will grow no bigger than this number. Set to 0 for no limit.

*max\_height* is a `NumericProperty` and defaults to *0*.

**border\_margin**

Margin between Window border and menu.

*border\_margin* is a `NumericProperty` and defaults to *4dp*.

**ver\_growth**

Where the menu will grow vertically to when opening. Set to None to let the widget pick for you. Available options are: ‘*up*’, ‘*down*’.

*ver\_growth* is a `OptionProperty` and defaults to *None*.

**hor\_growth**

Where the menu will grow horizontally to when opening. Set to None to let the widget pick for you. Available options are: ‘*left*’, ‘*right*’.

*hor\_growth* is a `OptionProperty` and defaults to *None*.

**background\_color**

Color of the background of the menu.

*background\_color* is a `ListProperty` and defaults to *[]*.

**opening\_transition**

Type of animation for opening a menu window.

*opening\_transition* is a `StringProperty` and defaults to ‘*out\_cubic*’.

**opening\_time**

Menu window opening animation time.

*opening\_time* is a `NumericProperty` and defaults to *0.2*.

**caller**

The widget object that caller the menu window.

*caller* is a `ObjectProperty` and defaults to *None*.

**position**

Menu window position relative to parent element. Available options are: ‘*auto*’, ‘*center*’, ‘*bottom*’.

*position* is a `OptionProperty` and defaults to ‘*auto*’.

**check\_position\_caller** (*self, instance, width, height*)**set\_bg\_color\_items** (*self, instance\_selected\_item*)

Called when a Hover Behavior event occurs for a list item.

**create\_menu\_items** (*self*)

Creates menu items.

**set\_menu\_properties (self, interval=0)**  
Sets the size and position for the menu window.

**open (self)**  
Animate the opening of a menu window.

**on\_touch\_down (self, touch)**  
Receive a touch down event.

#### Parameters

**touch: MotionEvent class** Touch received. The touch is in parent coordinates. See `relativelayout` for a discussion on coordinate systems.

**Returns** bool If True, the dispatching of the touch event will stop. If False, the event will continue to be dispatched to the rest of the widget tree.

**on\_touch\_move (self, touch)**  
Receive a touch move event. The touch is in parent coordinates.

See `on_touch_down ()` for more information.

**on\_touch\_up (self, touch)**  
Receive a touch up event. The touch is in parent coordinates.

See `on_touch_down ()` for more information.

**on\_enter (self, instance)**  
Call when mouse enter the bbox of the item of menu.

**on\_leave (self, instance)**  
Call when the mouse exit the item of menu.

**on\_release (self, \*args)**  
The method that will be called when you click menu items.

**on\_dismiss (self)**  
Called when the menu is closed.

**dismiss (self)**  
Closes the menu.

### 2.3.16 FloatLayout

`FloatLayout` class equivalent. Simplifies working with some widget properties. For example:

#### FloatLayout

```
FloatLayout:
    canvas:
        Color:
            rgba: app.theme_cls.primary_color
        RoundedRectangle:
            pos: self.pos
            size: self.size
            radius: [25, 0, 0, 0]
```

## MDFloatLayout

```
MDFloatLayout:  
    radius: [25, 0, 0, 0]  
    md_bg_color: app.theme_cls.primary_color
```

**Warning:** For a `FloatLayout`, the `minimum_size` attributes are always 0, so you cannot use `adaptive_size` and related options.

### API - `kivymd.uix.floatlayout`

```
class kivymd.uix.floatlayout.MDFloatLayout(**kwargs)  
    Float layout class. See module documentation for more information.
```

## 2.3.17 GridLayout

`GridLayout` class equivalent. Simplifies working with some widget properties. For example:

### GridLayout

```
GridLayout:  
    size_hint_y: None  
    height: self.minimum_height  
  
    canvas:  
        Color:  
            rgba: app.theme_cls.primary_color  
        Rectangle:  
            pos: self.pos  
            size: self.size
```

### MDGridLayout

```
MDGridLayout:  
    adaptive_height: True  
    md_bg_color: app.theme_cls.primary_color
```

**Available options are:**

- `adaptive_height`
- `adaptive_width`
- `adaptive_size`

### adaptive\_height

```
adaptive_height: True
```

Equivalent

```
size_hint_y: None  
height: self.minimum_height
```

### adaptive\_width

```
adaptive_width: True
```

Equivalent

```
size_hint_x: None  
height: self.minimum_width
```

### adaptive\_size

```
adaptive_size: True
```

Equivalent

```
size_hint: None, None  
size: self.minimum_size
```

## API - kivymd.uix.gridlayout

```
class kivymd.uix.gridlayout.MDGridLayout(**kwargs)  
    Grid layout class. See module documentation for more information.
```

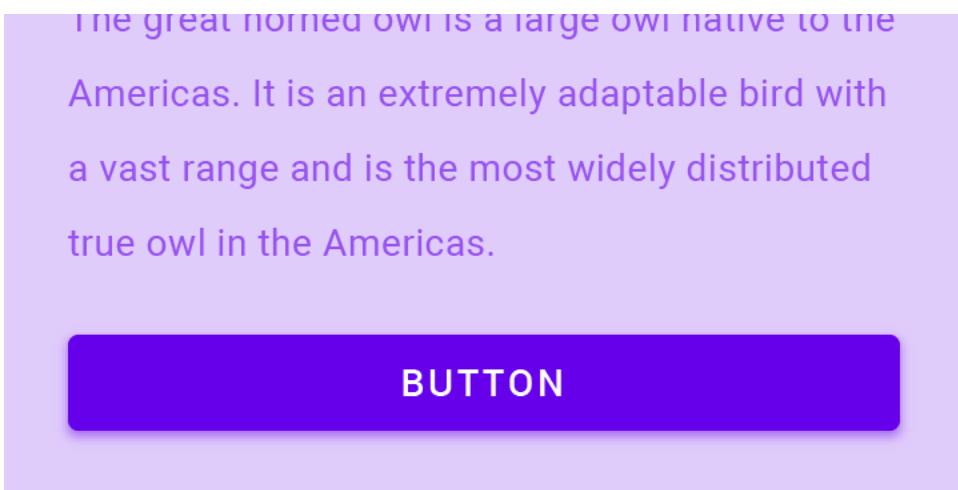
### 2.3.18 Button

See also:

Material Design spec, Buttons

Material Design spec, Buttons: floating action button

Buttons allow users to take actions, and make choices, with a single tap.



KivyMD provides the following button classes for use:

- [MDIconButton](#)
- [MDFloatingActionButton](#)
- [MDFlatButton](#)
- [MDRaisedButton](#)
- [MDRectangleFlatButton](#)
- [MDRectangleFlatIconButton](#)
- [MDRoundFlatButton](#)
- [MDRoundFlatIconButton](#)
- [MDFillRoundFlatButton](#)
- [MDFillRoundFlatIconButton](#)
- [MDTextButton](#)
- [MDFloatingActionButtonSpeedDial](#)

## MDIconButton

## MDIconButton

```
from kivy.lang import Builder

from kivymd.app import MDApp

KV = '''
Screen:

    MDIconButton:
        icon: "language-python"
        pos_hint: {"center_x": .5, "center_y": .5}
'''


class Example(MDApp):
    def build(self):
        return Builder.load_string(KV)

Example().run()
```

The `icon` parameter must have the name of the icon from `kivymd/icon_definitions.py` file.

You can also use custom icons:

```
MDIconButton:
    icon: "data/logo/kivy-icon-256.png"
```

By default, `MDIconButton` button has a size (`dp(48)`, `dp(48)`). Use `user_font_size` attribute to resize the button:

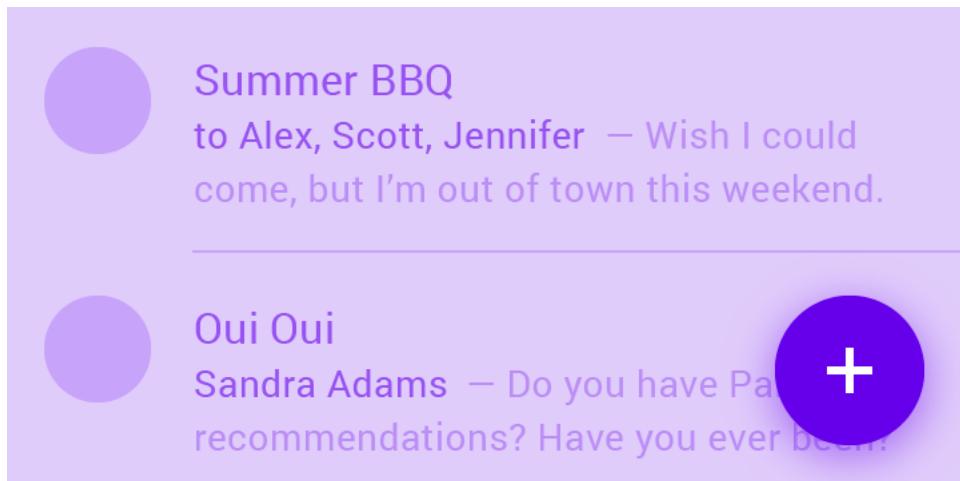
```
MDIconButton:
    icon: "android"
    user_font_size: "64sp"
```

By default, the color of `MDIconButton` (depending on the style of the application) is black or white. You can change the color of `MDIconButton` as the text color of `MDLabel`:

```
MDIconButton:
    icon: "android"
    theme_text_color: "Custom"
    text_color: app.theme_cls.primary_color
```



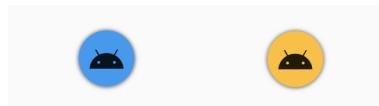
## MDFloatingActionButton



The above parameters for `MDIconButton` apply to `MDFloatingActionButton`.

To change `MDFloatingActionButton` background, use the `md_bg_color` parameter:

```
MDFloatingActionButton:
    icon: "android"
    md_bg_color: app.theme_cls.primary_color
```



The length of the shadow is controlled by the `elevation_normal` parameter:

```
MDFloatingActionButton:
    icon: "android"
    elevation_normal: 12
```



## MDFlatButton

To change the text color of: class:`~MDFlatButton` use the `text_color` parameter:

```
MDFlatButton:
    text: "MDFLATBUTTON"
    text_color: 0, 0, 1, 1
```

MDFLATBUTTON MDFLATBUTTON

Or use markup:

```
MDFlatButton:
    text: "[color=#00ffcc]MDFLATBUTTON[/color]"
    markup: True
```

To specify the font size and font name, use the parameters as in the usual *Kivy* buttons:

```
MDFlatButton:
    text: "MDFLATBUTTON"
    font_size: "18sp"
    font_name: "path/to/font"
```

**Warning:** You cannot use the `size_hint_x` parameter for *KivyMD* buttons (the width of the buttons is set automatically)!

## MDRaisedButton

This button is similar to the `MDFlatButton` button except that you can set the background color for `MDRaisedButton`:

```
MDRaisedButton:
    text: "MDRAISEDBUTTON"
    md_bg_color: 1, 0, 1, 1
```

## MDRectangleFlatButton

Button parameters `MDRectangleFlatButton` are the same as button `MDRaisedButton`:

```
MDRectangleFlatButton:
    text: "MDRECTANGLEFLATBUTTON"
    text_color: 0, 0, 1, 1
    md_bg_color: 1, 1, 0, 1
```

---

**Note:** Note that the frame color will be the same as the text color.

---



## MDRectangleFlatButton



Button parameters `MDRectangleFlatButton` are the same as button `MDRectangleFlatButton`:

```
MDRectangleFlatButton:
    icon: "android"
    text: "MDRECTANGLEFLATICONBUTTON"
```

### Without border

```
from kivy.uix.screenmanager import Screen

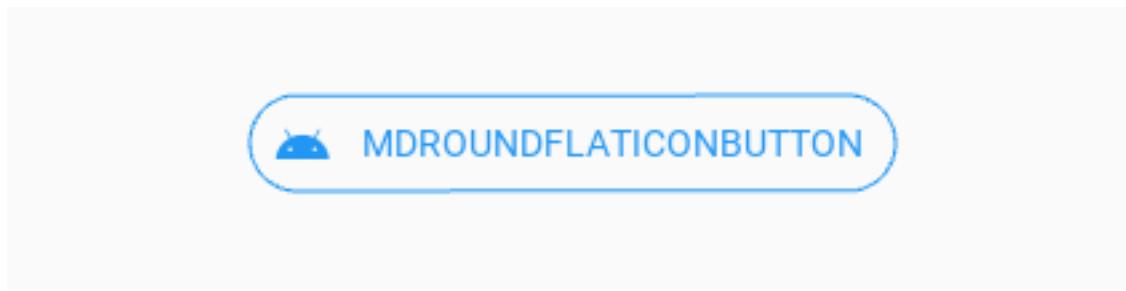
from kivymd.app import MDApp
from kivymd.uix.button import MDRectangleFlatButton

class Example(MDApp):
    def build(self):
        screen = Screen()
        screen.add_widget(
            MDRectangleFlatButton(
                text="MDRectangleFlatButton",
                icon="language-python",
                line_color=(0, 0, 0),
                pos_hint={"center_x": .5, "center_y": .5},
            )
        )
    return screen

Example().run()
```

```
MDRectangleFlatButton:
    text: "MDRectangleFlatButton"
    icon: "language-python"
    line_color: 0, 0, 0
    pos_hint: {"center_x": .5, "center_y": .5}
```

## MDRoundFlatButton

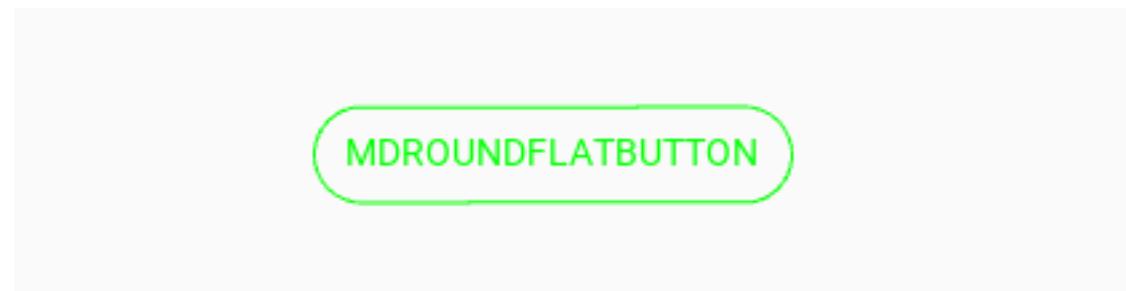


Button parameters `MDRoundFlatButton` are the same as button `MDRectangleFlatButton`:

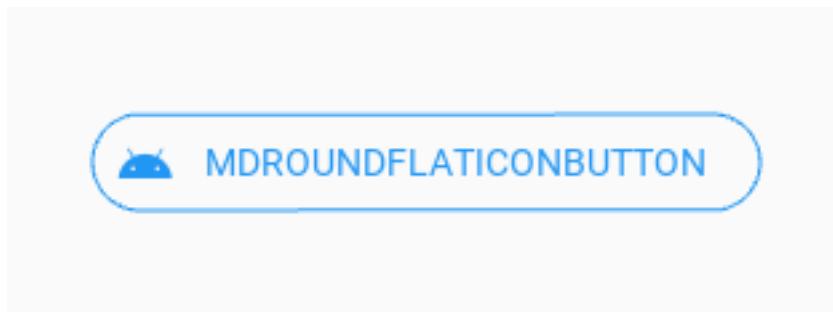
```
MDRoundFlatButton:  
    text: "MDROUNDFLATBUTTON"
```

**Warning:** The border color does change when using `text_color` parameter.

```
MDRoundFlatButton:  
    text: "MDROUNDFLATBUTTON"  
    text_color: 0, 1, 0, 1
```



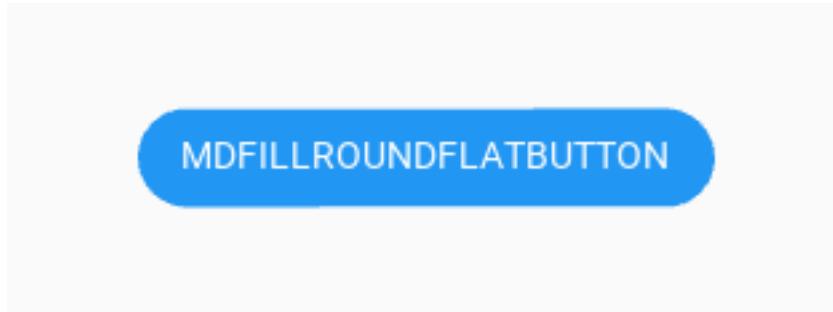
## MDRoundFlatIconButton



Button parameters `MDRoundFlatIconButton` are the same as button `MDRoundFlatButton`:

```
MDRoundFlatIconButton:  
    icon: "android"  
    text: "MDROUNDFLATICONBUTTON"
```

### MDFillRoundFlatButton



Button parameters `MDFillRoundFlatButton` are the same as button `MDRaisedButton`.

### MDFillRoundFlatIconButton



Button parameters `MDFillRoundFlatIconButton` are the same as button `MDRaisedButton`.

---

**Note:** Notice that the width of the `MDFillRoundFlatIconButton` button matches the size of the button text.

---

### MDTextButton



```
MDTextButton:  
    text: "MDTEXTBUTTON"  
    custom_color: 0, 1, 0, 1
```

## MDFloatingActionButtonSpeedDial

**Note:** See the full list of arguments in the class `MDFloatingActionButtonSpeedDial`.

```
from kivy.lang import Builder

from kivymd.app import MDApp

KV = '''
Screen:

    MDFloatingActionButtonSpeedDial:
        data: app.data
        rotation_root_button: True
'''


class Example(MDApp):
    data = {
        'language-python': 'Python',
        'language-php': 'PHP',
        'language-cpp': 'C++',
    }

    def build(self):
        return Builder.load_string(KV)

Example().run()
```

Or without KV Language:

```
from kivy.uix.screenmanager import Screen

from kivymd.app import MDApp
from kivymd.uix.button import MDFloatingActionButtonSpeedDial


class Example(MDApp):
    data = {
        'language-python': 'Python',
        'language-php': 'PHP',
        'language-cpp': 'C++',
    }

    def build(self):
        screen = Screen()
        speed_dial = MDFloatingActionButtonSpeedDial()
        speed_dial.data = self.data
        speed_dial.rotation_root_button = True
        screen.add_widget(speed_dial)
        return screen
```

(continues on next page)

(continued from previous page)

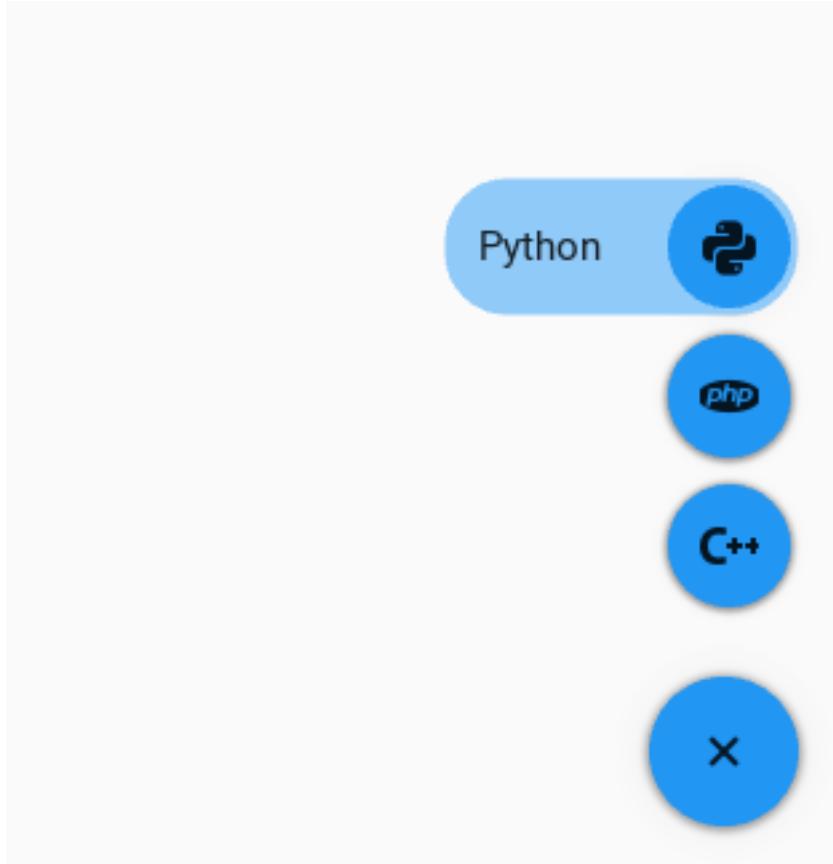
```
Example().run()
```

You can use various types of animation of labels for buttons on the stack:

```
MDFloatingActionButtonSpeedDial:  
    hint_animation: True
```

You can set your color values for background, text of buttons etc:

```
MDFloatingActionButtonSpeedDial:  
    bg_hint_color: app.theme_cls.primary_light
```



**See also:**

[See full example](#)

**API - kivymd.uix.button**

**class** kivymd.uix.button.**MDIconButton**(\*\*kwargs)

Abstract base class for all round buttons, bringing in the appropriate on-touch behavior

**icon**

Button icon.

*icon* is an `StringProperty` and defaults to ‘checkbox-blank-circle’.

**class** kivymd.uix.button.**MDFlatButton**(\*\*kwargs)

Abstract base class for all rectangular buttons, bringing in the appropriate on-touch behavior. Also maintains the correct minimum width as stated in guidelines.

**class** kivymd.uix.button.**MDRaisedButton**(\*\*kwargs)

Abstract base class for all rectangular buttons, bringing in the appropriate on-touch behavior. Also maintains the correct minimum width as stated in guidelines.

**class** kivymd.uix.button.**MDFloatingActionButton**(\*\*kwargs)

Abstract base class for all round buttons, bringing in the appropriate on-touch behavior

**icon**

Button icon.

*icon* is an `StringProperty` and defaults to ‘android’.

**background\_palette**

The name of the palette used for the background color of the button.

*background\_palette* is an `StringProperty` and defaults to ‘Accent’.

**on\_md\_bg\_color(self, instance, value)**

**class** kivymd.uix.button.**MDRectangleFlatButton**(\*\*kwargs)

Abstract base class for all rectangular buttons, bringing in the appropriate on-touch behavior. Also maintains the correct minimum width as stated in guidelines.

**update\_md\_bg\_color(self, instance, value)**

Called when the application color palette changes.

**on\_disabled(self, instance, value)**

**class** kivymd.uix.button.**MDRoundFlatButton**(\*\*kwargs)

Abstract base class for all rectangular buttons, bringing in the appropriate on-touch behavior. Also maintains the correct minimum width as stated in guidelines.

**update\_md\_bg\_color(self, instance, value)**

Called when the application color palette changes.

**lay\_canvas\_instructions(self)**

**class** kivymd.uix.button.**MDTextButton**(\*\*kwargs)

Button class, see module documentation for more information.

Changed in version 1.8.0: The behavior / logic of the button has been moved to `ButtonBehaviors`.

**custom\_color**

Custom user button color in `rgba` format.

*custom\_color* is an `ListProperty` and defaults to `[]`.

**animation\_label(self)****on\_press(self, \*args)**

```
on_disabled(self, instance, value)

class kivymd.uix.button.MDFillRoundFlatButton(**kwargs)
    Base class for circular elevation behavior. Controls the size and position of the shadow.

    update_md_bg_color(self, instance, value)
        Called when the application color palette changes.

    on_md_bg_color(self, instance, value)

    on_elevation(self, instance, value)

    on_disabled(self, instance, value)
        from kivy.lang import Builder

        from kivymd.app import MDApp

        root_kv = """
            Screen:

                MDFillRoundFlatButton: id: btn text: "Click me!" pos_hint: {"center_x": .5, "center_y": .6}
                    elevation: 8 on_press: self.disabled = True

                MDFillRoundFlatButton: text: "UNDISABLED" pos_hint: {"center_x": .5, "center_y": .4}
                    on_press: btn.disabled = False

            """

    class MainApp(MDApp):

        def build(self):
            self.root = Builder.load_string(root_kv)

        MainApp().run()

class kivymd.uix.button.MDRectangleFlatButton(**kwargs)
    Abstract base class for all rectangular buttons, bringing in the appropriate on-touch behavior. Also maintains the correct minimum width as stated in guidelines.

    line_color
        Button border color in rgba format.

        line_color is an ListProperty and defaults to [].

    class kivymd.uix.button.MDRoundFlatButton(**kwargs)
        Abstract base class for all rectangular buttons, bringing in the appropriate on-touch behavior. Also maintains the correct minimum width as stated in guidelines.

    class kivymd.uix.button.MDFillRoundFlatButton(**kwargs)
        Abstract base class for all rectangular buttons, bringing in the appropriate on-touch behavior. Also maintains the correct minimum width as stated in guidelines.

    text_color

    on_md_bg_color(self, instance, value)

    update_md_bg_color(self, instance, value)
        Called when the application color palette changes.

class kivymd.uix.button.MDFloatingActionButtonSpeedDial(**kwargs)
    Events

    on_open Called when a stack is opened.

    on_close Called when a stack is closed.
```

**icon**

Root button icon name.

*icon* is a `StringProperty` and defaults to ‘plus’.

**anchor**

Stack anchor. Available options are: ‘right’.

*anchor* is a `OptionProperty` and defaults to ‘right’.

**callback**

Custom callback.

```
MDFloatingActionButtonSpeedDial:
    callback: app.callback
```

```
def callback(self, instance):
    print(instance.icon)
```

*callback* is a `ObjectProperty` and defaults to *None*.

**label\_text\_color**

Floating text color in `rgba` format.

*label\_text\_color* is a `ListProperty` and defaults to `[0, 0, 0, 1]`.

**data**

Must be a dictionary

```
{
    'name-icon': 'Text label',
    ...,
    ...
}
```

**right\_pad**

If *True*, the button will increase on the right side by 2.5 piesels if the `hint_animation` parameter equal to *True*.

**False**

**True**

*right\_pad* is a `BooleanProperty` and defaults to *False*.

**rotation\_root\_button**

If *True* then the root button will rotate 45 degrees when the stack is opened.

*rotation\_root\_button* is a `BooleanProperty` and defaults to *False*.

**opening\_transition**

The name of the stack opening animation type.

*opening\_transition* is a `StringProperty` and defaults to ‘out\_cubic’.

**closing\_transition**

The name of the stack closing animation type.

*closing\_transition* is a `StringProperty` and defaults to ‘*out\_cubic*’.

**opening\_transition\_button\_rotation**

The name of the animation type to rotate the root button when opening the stack.

*opening\_transition\_button\_rotation* is a `StringProperty` and defaults to ‘*out\_cubic*’.

**closing\_transition\_button\_rotation**

The name of the animation type to rotate the root button when closing the stack.

*closing\_transition\_button\_rotation* is a `StringProperty` and defaults to ‘*out\_cubic*’.

**opening\_time**

Time required for the stack to go to: attr:`state` ‘*open*’.

*opening\_time* is a `NumericProperty` and defaults to 0.2.

**closing\_time**

Time required for the stack to go to: attr:`state` ‘*close*’.

*closing\_time* is a `NumericProperty` and defaults to 0.2.

**opening\_time\_button\_rotation**

Time required to rotate the root button 45 degrees during the stack opening animation.

*opening\_time\_button\_rotation* is a `NumericProperty` and defaults to 0.2.

**closing\_time\_button\_rotation**

Time required to rotate the root button 0 degrees during the stack closing animation.

*closing\_time\_button\_rotation* is a `NumericProperty` and defaults to 0.2.

**state**

Indicates whether the stack is closed or open. Available options are: ‘*close*’, ‘*open*’.

*state* is a `OptionProperty` and defaults to ‘*close*’.

**bg\_color\_root\_button**

Root button color in `rgba` format.

*bg\_color\_root\_button* is a `ListProperty` and defaults to `[]`.

**bg\_color\_stack\_button**

The color of the buttons in the stack `rgba` format.

*bg\_color\_stack\_button* is a `ListProperty` and defaults to `[]`.

**color\_icon\_stack\_button**

The color icon of the buttons in the stack `rgba` format.

*color\_icon\_stack\_button* is a `ListProperty` and defaults to `[]`.

**color\_icon\_root\_button**

The color icon of the root button `rgba` format.

*color\_icon\_root\_button* is a `ListProperty` and defaults to `[]`.

**bg\_hint\_color**

Background color for the text of the buttons in the stack `rgba` format.

*bg\_hint\_color* is a `ListProperty` and defaults to `[]`.

---

**hint\_animation**  
Whether to use button extension animation to display text labels.  
*hint\_animation* is a `BooleanProperty` and defaults to *False*.

**on\_open (self, \*args)**  
Called when a stack is opened.

**on\_close (self, \*args)**  
Called when a stack is closed.

**on\_leave (self, instance)**  
Called when the mouse cursor goes outside the button of stack.

**on\_enter (self, instance)**  
Called when the mouse cursor is over a button from the stack.

**on\_data (self, instance, value)**  
Creates a stack of buttons.

**on\_icon (self, instance, value)**

**on\_label\_text\_color (self, instance, value)**

**on\_color\_icon\_stack\_button (self, instance, value)**

**on\_hint\_animation (self, instance, value)**

**on\_bg\_hint\_color (self, instance, value)**

**on\_color\_icon\_root\_button (self, instance, value)**

**on\_bg\_color\_stack\_button (self, instance, value)**

**on\_bg\_color\_root\_button (self, instance, value)**

**set\_pos\_labels (self, widget)**  
Sets the position of the floating labels.

**set\_pos\_root\_button (self, instance)**  
Sets the position of the root button.

**set\_pos\_bottom\_buttons (self, instance)**  
Sets the position of the bottom buttons in a stack.

**open\_stack (self, instance)**  
Opens a button stack.

**do\_animation\_open\_stack (self, anim\_data)**

**close\_stack (self)**  
Closes the button stack.

### 2.3.19 BoxLayout

BoxLayout class equivalent. Simplifies working with some widget properties. For example:

## BoxLayout

```
BoxLayout:  
    size_hint_y: None  
    height: self.minimum_height  
  
    canvas:  
        Color:  
            rgba: app.theme_cls.primary_color  
        Rectangle:  
            pos: self.pos  
            size: self.size
```

## MDBoxLayout

```
MDBoxLayout:  
    adaptive_height: True  
    md_bg_color: app.theme_cls.primary_color
```

Available options are:

- *adaptive\_height*
- *adaptive\_width*
- *adaptive\_size*

### adaptive\_height

```
adaptive_height: True
```

Equivalent

```
size_hint_y: None  
height: self.minimum_height
```

### adaptive\_width

```
adaptive_width: True
```

Equivalent

```
size_hint_x: None  
height: self.minimum_width
```

### adaptive\_size

```
adaptive_size: True
```

Equivalent

```
size_hint: None, None
size: self.minimum_size
```

### API - kivymd.uix.boxlayout

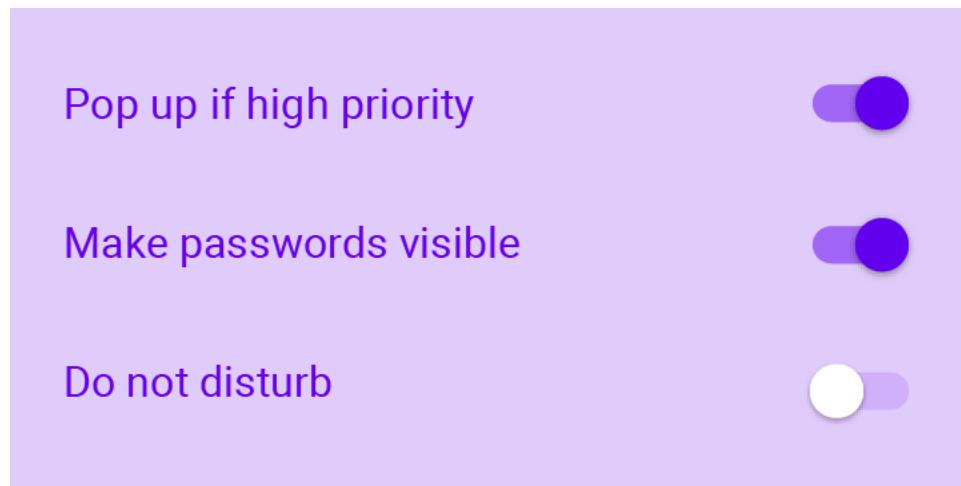
```
class kivymd.uix.boxlayout.MDBoxLayout(**kwargs)
    Box layout class. See module documentation for more information.
```

## 2.3.20 Selection Controls

See also:

Material Design spec, Selection controls

Selection controls allow the user to select options.



KivyMD provides the following selection controls classes for use:

- *MDCheckbox*
- *MDSwitch*

## MDCheckbox

```
from kivy.lang import Builder

from kivymd.app import MDApp

KV = '''
FloatLayout:

    MDCheckbox:
        size_hint: None, None
        size: "48dp", "48dp"
        pos_hint: {'center_x': .5, 'center_y': .5}
'''


class Test(MDApp):
    def build(self):
        return Builder.load_string(KV)

Test().run()
```

---

**Note:** Be sure to specify the size of the checkbox. By default, it is (dp(48), dp(48)), but the ripple effect takes up all the available space.

---

## Control state

```
MDCheckbox:
    on_active: app.on_checkbox_active(*args)

def on_checkbox_active(self, checkbox, value):
    if value:
        print('The checkbox', checkbox, 'is active', 'and', checkbox.state, 'state')
    else:
        print('The checkbox', checkbox, 'is inactive', 'and', checkbox.state, 'state')
```

## MDCheckbox with group

```
from kivy.lang import Builder

from kivymd.app import MDApp

KV = '''
<Check@MDCheckbox>:
    group: 'group'
```

(continues on next page)

(continued from previous page)

```

size_hint: None, None
size: dp(48), dp(48)

FloatLayout:

    Check:
        active: True
        pos_hint: {'center_x': .4, 'center_y': .5}

    Check:
        pos_hint: {'center_x': .6, 'center_y': .5}
    ...

class Test(MDApp):
    def build(self):
        return Builder.load_string(KV)

Test().run()

```

## MDSwitch

```

from kivy.lang import Builder

from kivymd.app import MDApp

KV = '''
FloatLayout:

    MDSwitch:
        pos_hint: {'center_x': .5, 'center_y': .5}
    ...

class Test(MDApp):
    def build(self):
        return Builder.load_string(KV)

Test().run()

```

---

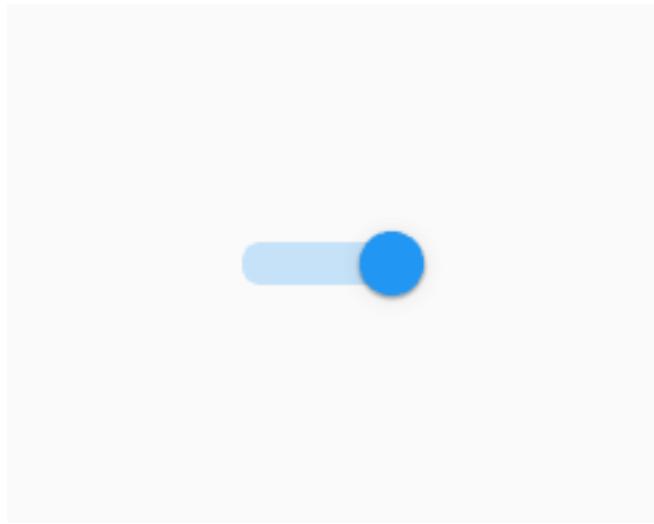
**Note:** For `MDCheckbox` size is not required. By default it is `(dp(36), dp(48))`, but you can increase the width if you want.

---

```

MDSwitch:
    width: dp(64)

```



---

**Note:** Control state of `MDSwitch` same way as in `MDCheckbox`.

---

#### API - kivymd.uix.selectioncontrol

**class** kivymd.uix.selectioncontrol.**MDCheckbox**(\*\*kwargs)

Class implements a circular ripple effect.

##### **active**

Indicates if the checkbox is active or inactive.

`active` is a `BooleanProperty` and defaults to `False`.

##### **checkbox\_icon\_normal**

Background icon of the checkbox used for the default graphical representation when the checkbox is not pressed.

`checkbox_icon_normal` is a `StringProperty` and defaults to ‘checkbox-blank-outline’.

##### **checkbox\_icon\_down**

Background icon of the checkbox used for the default graphical representation when the checkbox is pressed.

`checkbox_icon_down` is a `StringProperty` and defaults to ‘checkbox-marked-outline’.

##### **radio\_icon\_normal**

Background icon (when using the group option) of the checkbox used for the default graphical representation when the checkbox is not pressed.

`radio_icon_normal` is a `StringProperty` and defaults to ‘checkbox-blank-circle-outline’.

##### **radio\_icon\_down**

Background icon (when using the group option) of the checkbox used for the default graphical representation when the checkbox is pressed.

`radio_icon_down` is a `StringProperty` and defaults to ‘checkbox-marked-circle-outline’.

##### **selected\_color**

Selected color in `rgba` format.

`selected_color` is a `ListProperty` and defaults to `[]`.

---

```
unselected_color
Unelected color in rgba format.

unselected_color is a ListProperty and defaults to [].
```

```
disabled_color
Disabled color in rgba format.

disabled_color is a ListProperty and defaults to [].
```

```
update_primary_color(self, instance, value)
update_icon(self, *args)
update_color(self, *args)
on_state(self, *args)
on_active(self, *args)
```

```
class kivymd.uix.selectioncontrol.MDSwitch(**kwargs)
This mixin class provides Button behavior. Please see the button behaviors module documentation for more information.
```

#### Events

**on\_press** Fired when the button is pressed.

**on\_release** Fired when the button is released (i.e. the touch/click that pressed the button goes away).

#### active

Indicates if the switch is active or inactive.

**active** is a `BooleanProperty` and defaults to `False`.

#### thumb\_color

Get thumb color rgba format.

**thumb\_color** is an `AliasProperty` and property is readonly.

#### thumb\_color\_disabled

Get thumb color disabled rgba format.

**thumb\_color\_disabled** is an `AliasProperty` and property is readonly.

#### thumb\_color\_down

Get thumb color down rgba format.

**thumb\_color\_down** is an `AliasProperty` and property is readonly.

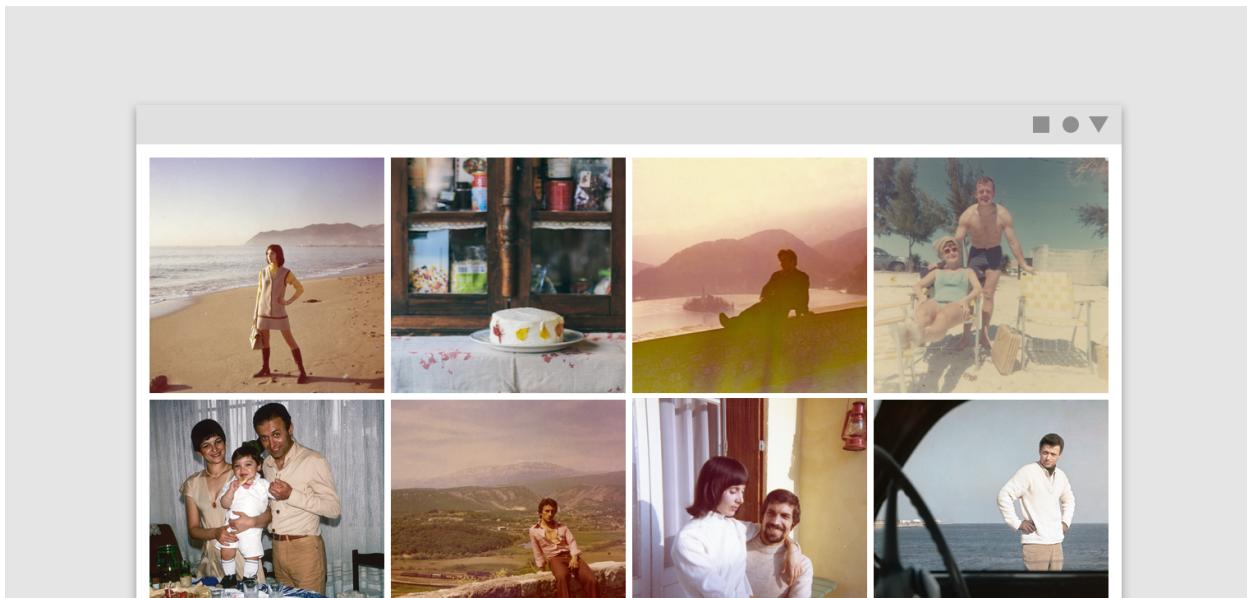
#### on\_size(*self, \*args*)

### 2.3.21 Image List

See also:

Material Design spec, Image lists

Image lists display a collection of images in an organized grid.



KivyMD provides the following tile classes for use:

- *SmartTileWithStar*
- *SmartTileWithLabel*

### SmartTileWithStar

```
from kivymd.app import MDApp
from kivy.lang import Builder

KV = '''
<MyTile@SmartTileWithStar>
    size_hint_y: None
    height: "240dp"

ScrollView:

    MDGridLayout:
        cols: 3
        adaptive_height: True
        padding: dp(4), dp(4)
        spacing: dp(4)

        MyTile:
            stars: 5
            source: "cat-1.jpg"

        MyTile:
            stars: 5
            source: "cat-2.jpg"

        MyTile:
            stars: 5
            source: "cat-3.jpg"
'''
```

(continues on next page)

(continued from previous page)

```

        stars: 5
        source: "cat-3.jpg"
    ...

class MyApp(MDApp):
    def build(self):
        return Builder.load_string(KV)

MyApp().run()

```

### SmartTileWithLabel

```

from kivymd.app import MDApp
from kivy.lang import Builder

KV = '''
<MyTile@SmartTileWithStar>
    size_hint_y: None
    height: "240dp"

ScrollView:

    MDGridLayout:
        cols: 3
        adaptive_height: True
        padding: dp(4), dp(4)
        spacing: dp(4)

    MyTile:
        source: "cat-1.jpg"
        text: "[size=26]Cat 1[/size]\n[size=14]cat-1.jpg[/size]"

    MyTile:
        source: "cat-2.jpg"
        text: "[size=26]Cat 2[/size]\n[size=14]cat-2.jpg[/size]"
        tile_text_color: app.theme_cls.accent_color

    MyTile:
        source: "cat-3.jpg"
        text: "[size=26][color=#ffffff]Cat 3[/color][/size]\n[size=14]cat-3.jpg[/size]"
        tile_text_color: app.theme_cls.accent_color
    ...
    ...

class MyApp(MDApp):
    def build(self):
        return Builder.load_string(KV)

```

(continues on next page)

(continued from previous page)

MyApp().run()

**API - kivymd.uix.imagelist****class** kivymd.uix.imagelist.**SmartTile**(\*\*kwargs)

A tile for more complex needs.

Includes an image, a container to place overlays and a box that can act as a header or a footer, as described in the Material Design specs.

**box\_color**

Sets the color and opacity for the information box.

*box\_color* is a `ListProperty` and defaults to `(0, 0, 0, 0.5)`.**box\_position**

Determines whether the information box acts as a header or footer to the image. Available are options: 'footer', 'header'.

*box\_position* is a `OptionProperty` and defaults to 'footer'.**lines**Number of lines in the *header/footer*. As per *Material Design specs*, only 1 and 2 are valid values. Available are options: 1, 2.*lines* is a `OptionProperty` and defaults to 1.**overlap**Determines if the *header/footer* overlaps on top of the image or not.*overlap* is a `BooleanProperty` and defaults to *True*.**source**Path to tile image. See `source`.*source* is a `StringProperty` and defaults to ''.**reload(self)****class** kivymd.uix.imagelist.**SmartTileWithLabel**(\*\*kwargs)

A tile for more complex needs.

Includes an image, a container to place overlays and a box that can act as a header or a footer, as described in the Material Design specs.

**font\_style**  
 Tile font style.  
`font_style` is a `StringProperty` and defaults to ‘Caption’.

**tile\_text\_color**  
 Tile text color in `rgba` format.  
`tile_text_color` is a `StringProperty` and defaults to `(1, 1, 1, 1)`.

**text**  
 Determines the text for the box *footer/header*.  
`text` is a `StringProperty` and defaults to ‘’.

**class** `kivymd.uix.imagelist.SmartTileWithStar(**kwargs)`  
 A tile for more complex needs.  
 Includes an image, a container to place overlays and a box that can act as a header or a footer, as described in the Material Design specs.

**stars**  
 Tile stars.  
`stars` is a `NumericProperty` and defaults to `1`.

**on\_stars** (`self, *args`)

## 2.3.22 Refresh Layout

### Example

```
from kivymd.app import MDApp
from kivy.clock import Clock
from kivy.lang import Builder
from kivy.factory import Factory
from kivy.properties import StringProperty

from kivymd.uix.button import MDIconButton
from kivymd.icon_definitions import md_icons
from kivymd.uix.list import ILeftBodyTouch, OneLineIconListItem
from kivymd.theming import ThemeManager
from kivymd.utils import asynckivy

Builder.load_string('''
<ItemForList>
    text: root.text

    IconLeftSampleWidget:
        icon: root.icon

<Example@FloatLayout>

    BoxLayout:
        orientation: 'vertical'

        MToolbar:
            title: app.title
```

(continues on next page)

(continued from previous page)

```

        md_bg_color: app.theme_cls.primary_color
        background_palette: 'Primary'
        elevation: 10
        left_action_items: [['menu', lambda x: x]]

    MDScrollViewRefreshLayout:
        id: refresh_layout
        refresh_callback: app.refresh_callback
        root_layout: root

    MDGridLayout:
        id: box
        adaptive_height: True
        cols: 1
    ''')

class IconLeftSampleWidget(ILeftBodyTouch, MDIconButton):
    pass

class ItemForList(OneLineIconListItem):
    icon = StringProperty()

class Example(MDApp):
    title = 'Example Refresh Layout'
    screen = None
    x = 0
    y = 15

    def build(self):
        self.screen = Factory.Example()
        self.set_list()

        return self.screen

    def set_list(self):
        async def set_list():
            names_icons_list = list(md_icons.keys())[self.x:self.y]
            for name_icon in names_icons_list:
                await asynckivy.sleep(0)
                self.screen.ids.box.add_widget(
                    ItemForList(icon=name_icon, text=name_icon))
        asynckivy.start(set_list())

    def refresh_callback(self, *args):
        '''A method that updates the state of your application
        while the spinner remains on the screen.'''
        self.refresh_callback(interval)

    def refresh_callback(interval):
        self.screen.ids.box.clear_widgets()
        if self.x == 0:
            self.x, self.y = 15, 30
        else:
            self.x, self.y = 0, 15
        self.set_list()

```

(continues on next page)

(continued from previous page)

```

    self.screen.ids.refresh_layout.refresh_done()
    self.tick = 0

    Clock.schedule_once(refresh_callback, 1)

Example().run()

```

**API - kivymd.uix.refreshlayout****class** kivymd.uix.refreshlayout.**MDS ScrollView Refresh Layout** (\*\*kargs)

ScrollView class. See module documentation for more information.

**Events****on\_scroll\_start** Generic event fired when scrolling starts from touch.**on\_scroll\_move** Generic event fired when scrolling move from touch.**on\_scroll\_stop** Generic event fired when scrolling stops from touch.Changed in version 1.9.0: *on\_scroll\_start*, *on\_scroll\_move* and *on\_scroll\_stop* events are now dispatched when scrolling to handle nested ScrollViews.Changed in version 1.7.0: *auto\_scroll*, *scroll\_friction*, *scroll\_moves*, *scroll\_stoptime*' has been deprecated, use *:attr:`effect\_cls`* instead.**root\_layout**

The spinner will be attached to this layout.

**on\_touch\_up** (self, \*args)

Receive a touch up event. The touch is in parent coordinates.

See `on_touch_down()` for more information.**refresh\_done** (self)**class** kivymd.uix.refreshlayout.**Refresh Spinner** (\*\*kwargs)

Float layout class. See module documentation for more information.

**spinner\_color****start\_anim\_spinner** (self)**hide\_anim\_spinner** (self)**set\_spinner** (self, \*args)**2.3.23 Text Field****See also:**

Material Design spec, Text fields

Text fields let users enter and edit text.



KivyMD provides the following field classes for use:

- [MDTextField](#)
- [MDTextFieldRound](#)
- [MDTextFieldRect](#)

---

**Note:** [MDTextField](#) inherited from [TextInput](#). Therefore, most parameters and all events of the [TextInput](#) class are also available in the [MDTextField](#) class.

---

## MDTextField

[MDTextField](#) can be with helper text and without.

### Without helper text mode

```
MDTextField:  
    hint_text: "No helper text"
```

### Helper text mode on on\_focus event

```
MDTextField:
    hint_text: "Helper text on focus"
    helper_text: "This will disappear when you click off"
    helper_text_mode: "on_focus"
```

### Persistent helper text mode

```
MDTextField:
    hint_text: "Persistent helper text"
    helper_text: "Text is always here"
    helper_text_mode: "persistent"
```

### Helper text mode ‘on\_error’

To display an error in a text field when using the `helper_text_mode: "on_error"` parameter, set the “*error*” text field parameter to *True*:

```
from kivy.lang import Builder

from kivymd.app import MDApp

KV = '''
BoxLayout:
    padding: "10dp"

    MDTextField:
        id: text_field_error
        hint_text: "Helper text on error (press 'Enter')"
        helper_text: "There will always be a mistake"
        helper_text_mode: "on_error"
        pos_hint: {"center_y": .5}
'''

class Test(MDApp):
    def __init__(self, **kwargs):
        super().__init__(**kwargs)
        self.screen = Builder.load_string(KV)

    def build(self):
        self.screen.ids.text_field_error.bind(
            on_text_validate=self.set_error_message,
            on_focus=self.set_error_message,
        )
        return self.screen

    def set_error_message(self, instance_textfield):
        self.screen.ids.text_field_error.error = True
```

(continues on next page)

(continued from previous page)

```
Test().run()
```

### Helper text mode ‘on\_error’ (with required)

```
MDTextField:  
    hint_text: "required = True"  
    required: True  
    helper_text_mode: "on_error"  
    helper_text: "Enter text"
```

### Text length control

```
MDTextField:  
    hint_text: "Max text length = 5"  
    max_text_length: 5
```

### Multi line text

```
MDTextField:  
    multiline: True  
    hint_text: "Multi-line text"
```

### Color mode

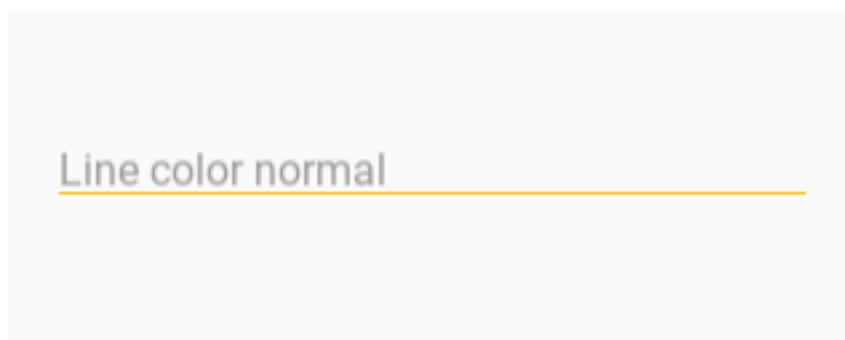
```
MDTextField:  
    hint_text: "color_mode = 'accent'"  
    color_mode: 'accent'
```

Available options are ‘primary’, ‘accent’ or ‘custom’.

```
MDTextField:  
    hint_text: "color_mode = 'custom'"  
    color_mode: 'custom'  
    helper_text_mode: "on_focus"  
    helper_text: "Color is defined by 'line_color_focus' property"  
    line_color_focus: 1, 0, 1, 1
```

**MDTextField:**

```
hint_text: "Line color normal"
line_color_normal: app.theme_cls.accent_color
```

**Rectangle mode****MDTextField:**

```
hint_text: "Rectangle mode"
mode: "rectangle"
```

**Fill mode****MDTextField:**

```
hint_text: "Fill mode"
mode: "fill"
fill_color: 0, 0, 0, .4
```

**MDTextFieldRect**

**Note:** `MDTextFieldRect` inherited from `TextInput`. You can use all parameters and attributes of the `TextInput` class in the `MDTextFieldRect` class.

**MDTextFieldRect:**

```
size_hint: 1, None
height: "30dp"
```

**Warning:** While there is no way to change the color of the border.

## MDTextFieldRound

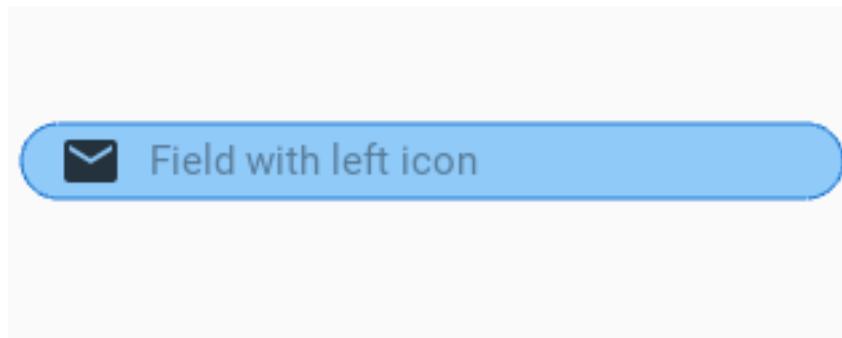
### Without icon

```
MDTextFieldRound:  
    hint_text: 'Empty field'
```

### With left icon

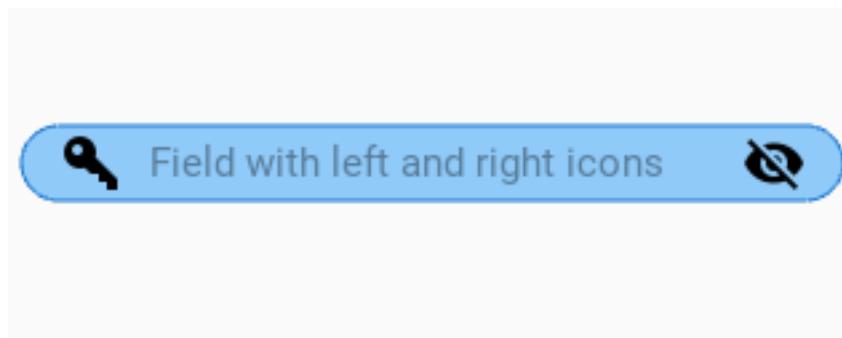
**Warning:** The icons in the `MDTextFieldRound` are static. You cannot bind events to them.

```
MDTextFieldRound:  
    icon_left: "email"  
    hint_text: "Field with left icon"
```



### With left and right icons

```
MDTextFieldRound:  
    icon_left: 'key-variant'  
    icon_right: 'eye-off'  
    hint_text: 'Field with left and right icons'
```



## Control background color

```
MDTextFieldRound:
    icon_left: 'key-variant'
    normal_color: app.theme_cls.accent_color
```

```
MDTextFieldRound:
    icon_left: 'key-variant'
    normal_color: app.theme_cls.accent_color
    color_active: 1, 0, 0, 1
```

## Clickable icon for MDTextFieldRound

```
from kivy.lang import Builder
from kivy.properties import StringProperty

from kivymd.app import MDApp
from kivymd.uix.relativelayout import MDRelativeLayout

KV = '''
<ClickableTextFieldRound>:
    size_hint_y: None
    height: text_field.height

    MDTextFieldRound:
        id: text_field
        hint_text: root.hint_text
        text: root.text
        password: True
        color_active: app.theme_cls.primary_light
        icon_left: "key-variant"
        padding:
            self._lbl_icon_left.texture_size[1] + dp(10) if self.icon_left else_
            ↪dp(15), (self.height / 2) - (self.line_height / 2),
            ↪self._lbl_icon_right.texture_size[1] + dp(20), 0
            ↪
        MDIconButton:
            icon: "eye-off"
            ripple_scale: .5
            pos_hint: {"center_y": .5}
            pos: text_field.width - self.width + dp(8), 0
            on_release:
                self.icon = "eye" if self.icon == "eye-off" else "eye-off"
                text_field.password = False if text_field.password is True else True

    MDScreen:

        ClickableTextFieldRound:
            size_hint_x: None
            width: "300dp"
```

(continues on next page)

(continued from previous page)

```

        hint_text: "Password"
        pos_hint: {"center_x": .5, "center_y": .5}
    ...

class ClickableTextFieldRound(MDRelativeLayout):
    text = StringProperty()
    hint_text = StringProperty()
    # Here specify the required parameters for MDTextFieldRound:
    # [...]

class Test(MDApp):
    def build(self):
        return Builder.load_string(KV)

Test().run()

```

## With right icon

**Note:** The icon on the right is available for use in all text fields.

```

MDTextField:
    hint_text: "Name"
    mode: "fill"
    fill_color: 0, 0, 0, .4
    icon_right: "arrow-down-drop-circle-outline"
    icon_right_color: app.theme_cls.primary_color

```



```

MDTextField:
    hint_text: "Name"
    icon_right: "arrow-down-drop-circle-outline"
    icon_right_color: app.theme_cls.primary_color

```



```

MDTextField:
    hint_text: "Name"
    mode: "rectangle"
    icon_right: "arrow-down-drop-circle-outline"
    icon_right_color: app.theme_cls.primary_color

```

**See also:**

See more information in the [MDTextFieldRect](#) class.

**API - kivymd.uix.textfield**

```
class kivymd.uix.textfield.MDTextFieldRect (**kwargs)
```

TextInput class. See module documentation for more information.

**Events**

**on\_text\_validate** Fired only in multiline=False mode when the user hits ‘enter’. This will also unfocus the textinput.

**on\_double\_tap** Fired when a double tap happens in the text input. The default behavior selects the text around the cursor position. More info at [on\\_double\\_tap\(\)](#).

**on\_triple\_tap** Fired when a triple tap happens in the text input. The default behavior selects the line around the cursor position. More info at [on\\_triple\\_tap\(\)](#).

**on\_quad\_touch** Fired when four fingers are touching the text input. The default behavior selects the whole text. More info at [on\\_quad\\_touch\(\)](#).

**Warning:** When changing a TextInput property that requires re-drawing, e.g. modifying the `text`, the updates occur on the next clock cycle and not instantly. This might cause any changes to the TextInput that occur between the modification and the next cycle to be ignored, or to use previous values. For example, after a update to the `text`, changing the cursor in the same clock frame will move it using the previous text and will likely end up in an incorrect position. The solution is to schedule any updates to occur on the next clock cycle using `schedule_once()`.

---

**Note:** Selection is cancelled when TextInput is focused. If you need to show selection when TextInput is focused, you should delay (use Clock.schedule) the call to the functions for selecting text (`select_all`, `select_text`).

---

Changed in version 1.10.0: `background_disabled_active` has been removed.

Changed in version 1.9.0: TextInput now inherits from `FocusBehavior`. `keyboard_mode`, `show_keyboard()`, `hide_keyboard()`, `focus()`, and `input_type` have been removed since they are now inherited from `FocusBehavior`.

Changed in version 1.7.0: `on_double_tap`, `on_triple_tap` and `on_quad_touch` events added.

**anim\_rect** (self, points, alpha)

```
class kivymd.uix.textfield.MDTextField (**kwargs)
```

TextInput class. See module documentation for more information.

**Events**

**on\_text\_validate** Fired only in multiline=False mode when the user hits ‘enter’. This will also unfocus the textinput.

**`on_double_tap`** Fired when a double tap happens in the text input. The default behavior selects the text around the cursor position. More info at `on_double_tap()`.

**`on_triple_tap`** Fired when a triple tap happens in the text input. The default behavior selects the line around the cursor position. More info at `on_triple_tap()`.

**`on_quad_touch`** Fired when four fingers are touching the text input. The default behavior selects the whole text. More info at `on_quad_touch()`.

**Warning:** When changing a `TextInput` property that requires re-drawing, e.g. modifying the `text`, the updates occur on the next clock cycle and not instantly. This might cause any changes to the `TextInput` that occur between the modification and the next cycle to be ignored, or to use previous values. For example, after a update to the `text`, changing the cursor in the same clock frame will move it using the previous text and will likely end up in an incorrect position. The solution is to schedule any updates to occur on the next clock cycle using `schedule_once()`.

---

**Note:** Selection is cancelled when `TextInput` is focused. If you need to show selection when `TextInput` is focused, you should delay (use `Clock.schedule`) the call to the functions for selecting text (`select_all`, `select_text`).

---

Changed in version 1.10.0: `background_disabled_active` has been removed.

Changed in version 1.9.0: `TextInput` now inherits from `FocusBehavior`. `keyboard_mode`, `show_keyboard()`, `hide_keyboard()`, `focus()`, and `input_type` have been removed since they are now inherited from `FocusBehavior`.

Changed in version 1.7.0: `on_double_tap`, `on_triple_tap` and `on_quad_touch` events added.

#### **`helper_text`**

Text for `helper_text` mode.

`helper_text` is an `StringProperty` and defaults to '*This field is required*'.

#### **`helper_text_mode`**

Helper text mode. Available options are: '`on_error`', '`persistent`', '`on_focus`'.

`helper_text_mode` is an `OptionProperty` and defaults to '`none`'.

#### **`max_text_length`**

Maximum allowed value of characters in a text field.

`max_text_length` is an `NumericProperty` and defaults to `None`.

#### **`required`**

Required text. If True then the text field requires text.

`required` is an `BooleanProperty` and defaults to `False`.

#### **`color_mode`**

Color text mode. Available options are: '`primary`', '`accent`', '`custom`'.

`color_mode` is an `OptionProperty` and defaults to '`primary`'.

#### **`mode`**

Text field mode. Available options are: '`line`', '`rectangle`', '`fill`'.

`mode` is an `OptionProperty` and defaults to '`line`'.

#### **`line_color_normal`**

Line color normal in `rgba` format.

`line_color_normal` is an `ListProperty` and defaults to `[]`.

**line\_color\_focus**  
Line color focus in `rgba` format.

`line_color_focus` is an `ListProperty` and defaults to `[]`.

**error\_color**  
Error color in `rgba` format for `required = True`.

`error_color` is an `ListProperty` and defaults to `[]`.

**fill\_color**  
The background color of the fill in `rgba` format when the `mode` parameter is “fill”.

`fill_color` is an `ListProperty` and defaults to `(0, 0, 0, 0)`.

**active\_line**  
Show active line or not.

`active_line` is an `BooleanProperty` and defaults to `True`.

**error**  
If `True`, then the text field goes into `error` mode.

`error` is an `BooleanProperty` and defaults to `False`.

**current\_hint\_text\_color**  
`hint_text` text color.

`current_hint_text_color` is an `ListProperty` and defaults to `[]`.

**icon\_right**  
Right icon.

`icon_right` is an `StringProperty` and defaults to `''`.

**icon\_right\_color**  
Color of right icon in `rgba` format.

`icon_right_color` is an `ListProperty` and defaults to `(0, 0, 0, 1)`.

**set\_objects\_labels (self)**  
Creates labels objects for the parameters `helper_text`, `hint_text`, etc.

**on\_icon\_right (self, instance, value)**

**on\_icon\_right\_color (self, instance, value)**

**on\_width (self, instance, width)**  
Called when the application window is resized.

**on\_focus (self, \*args)**

**on\_text (self, instance, text)**

**on\_text\_validate (self)**

**on\_color\_mode (self, instance, mode)**

**on\_line\_color\_focus (self, \*args)**

**on\_hint\_text (self, instance, value)**

**class kivymd.uix.textfield.MDTextFieldRound (\*\*kwargs)**  
`TextInput` class. See module documentation for more information.

## Events

**`on_text_validate`** Fired only in multiline=False mode when the user hits ‘enter’. This will also unfocus the TextInput.

**`on_double_tap`** Fired when a double tap happens in the text input. The default behavior selects the text around the cursor position. More info at `on_double_tap()`.

**`on_triple_tap`** Fired when a triple tap happens in the text input. The default behavior selects the line around the cursor position. More info at `on_triple_tap()`.

**`on_quad_touch`** Fired when four fingers are touching the text input. The default behavior selects the whole text. More info at `on_quad_touch()`.

**Warning:** When changing a TextInput property that requires re-drawing, e.g. modifying the `text`, the updates occur on the next clock cycle and not instantly. This might cause any changes to the TextInput that occur between the modification and the next cycle to be ignored, or to use previous values. For example, after a update to the `text`, changing the cursor in the same clock frame will move it using the previous text and will likely end up in an incorrect position. The solution is to schedule any updates to occur on the next clock cycle using `schedule_once()`.

---

**Note:** Selection is cancelled when TextInput is focused. If you need to show selection when TextInput is focused, you should delay (use Clock.schedule) the call to the functions for selecting text (`select_all`, `select_text`).

---

Changed in version 1.10.0: `background_disabled_active` has been removed.

Changed in version 1.9.0: TextInput now inherits from `FocusBehavior`. `keyboard_mode`, `show_keyboard()`, `hide_keyboard()`, `focus()`, and `input_type` have been removed since they are now inherited from `FocusBehavior`.

Changed in version 1.7.0: `on_double_tap`, `on_triple_tap` and `on_quad_touch` events added.

**`icon_left`**

Left icon.

`icon_left` is an `StringProperty` and defaults to “”.

**`icon_left_color`**

Color of left icon in `rgba` format.

`icon_left_color` is an `ListProperty` and defaults to `(0, 0, 0, 1)`.

**`icon_right`**

Right icon.

`icon_right` is an `StringProperty` and defaults to “”.

**`icon_right_color`**

Color of right icon.

`icon_right_color` is an `ListProperty` and defaults to `(0, 0, 0, 1)`.

**`line_color`**

Field line color.

`line_color` is an `ListProperty` and defaults to `[]`.

**`normal_color`**

Field color if `focus` is `False`.

`normal_color` is an `ListProperty` and defaults to `[]`.

**color\_active**

Field color if *focus* is *True*.

*color\_active* is an `ListProperty` and defaults to `[]`.

**on\_focus** (*self, instance, value*)

**on\_icon\_left** (*self, instance, value*)

**on\_icon\_left\_color** (*self, instance, value*)

**on\_icon\_right** (*self, instance, value*)

**on\_icon\_right\_color** (*self, instance, value*)

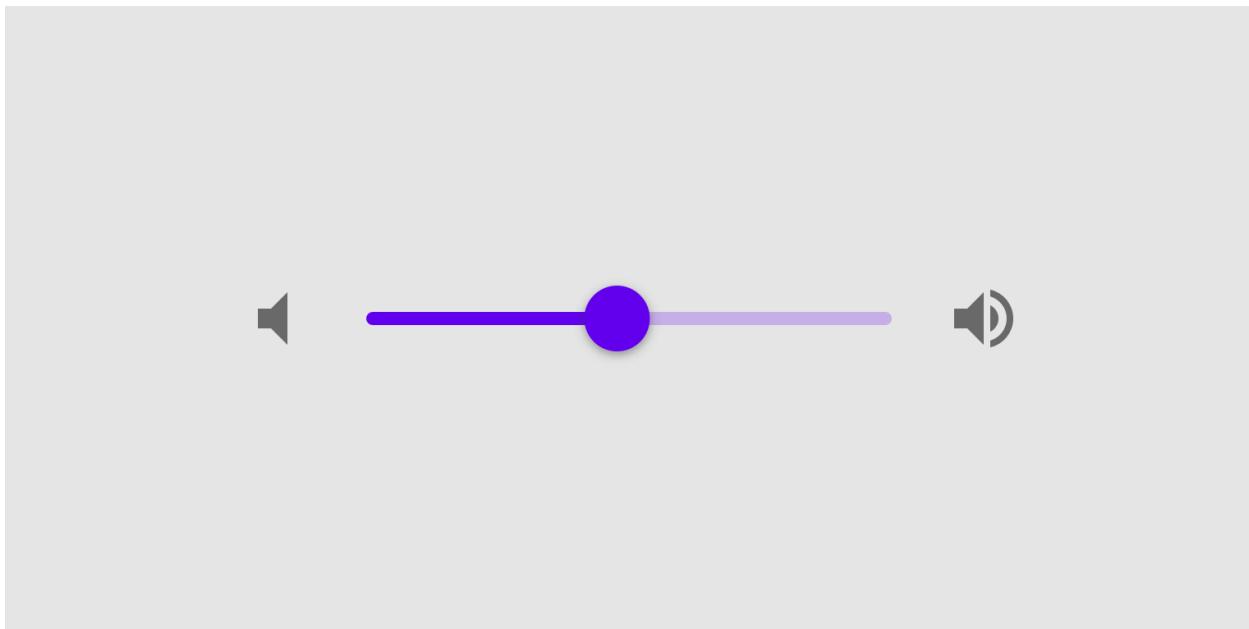
**on\_color\_active** (*self, instance, value*)

### 2.3.24 Slider

See also:

Material Design spec, Sliders

**Sliders allow users to make selections from a range of values.**



**With value hint**

```
from kivy.lang import Builder
from kivymd.app import MDApp
KV = '''
Screen
```

(continues on next page)

(continued from previous page)

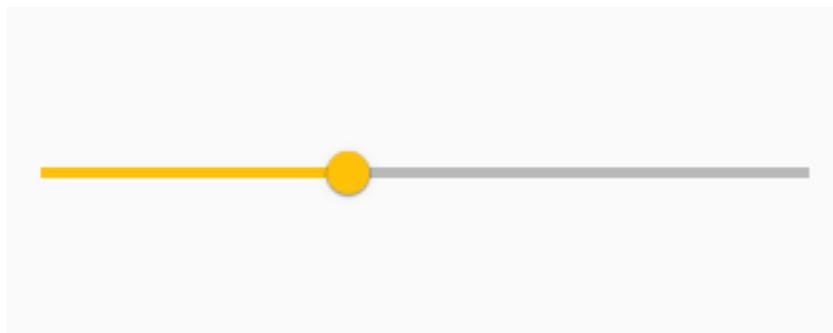
```
MDSlider:  
    min: 0  
    max: 100  
    value: 40  
    ...  
  
class Test(MDApp):  
    def build(self):  
        return Builder.load_string(KV)  
  
Test().run()
```

### Without value hint

```
MDSlider:  
    min: 0  
    max: 100  
    value: 40  
    hint: False
```

### Without custom color

```
MDSlider:  
    min: 0  
    max: 100  
    value: 40  
    hint: False  
    thumb_color_down: app.theme_cls.accent_color
```



**API - kivymd.uix.slider**

```
class kivymd.uix.slider.MDSlider(**kwargs)
```

Class for creating a Slider widget.

Check module documentation for more details.

**active**

If the slider is clicked.

`active` is an `BooleanProperty` and defaults to `False`.

**hint**

If True, then the current value is displayed above the slider.

`hint` is an `BooleanProperty` and defaults to `True`.

**hint\_bg\_color**

Hint rectangle color in `rgba` format.

`hint_bg_color` is an `ListProperty` and defaults to `[]`.

**hint\_text\_color**

Hint text color in `rgba` format.

`hint_text_color` is an `ListProperty` and defaults to `[]`.

**hint\_radius**

Hint radius.

`hint_radius` is an `NumericProperty` and defaults to `4`.

**show\_off**

Show the ‘off’ ring when set to minimum value.

`show_off` is an `BooleanProperty` and defaults to `True`.

**thumb\_color**

Current color slider in `rgba` format.

`thumb_color` is an `AliasProperty` that returns the value of the current color slider, property is readonly.

**thumb\_color\_down**

Color slider in `rgba` format.

`thumb_color_down` is an `AliasProperty` that returns and set the value of color slider.

**on\_hint(self, instance, value)****on\_value\_normalized(self, \*args)**

When the `value == min` set it to ‘off’ state and make slider a ring.

**on\_show\_off(self, \*args)****on\_is\_off(self, \*args)****on\_active(self, \*args)****on\_touch\_down(self, touch)**

Receive a touch down event.

**Parameters**

**touch: MotionEvent class** Touch received. The touch is in parent coordinates. See `relativelayout` for a discussion on coordinate systems.

**Returns** bool If True, the dispatching of the touch event will stop. If False, the event will continue to be dispatched to the rest of the widget tree.

**on\_touch\_up (self, touch)**

Receive a touch up event. The touch is in parent coordinates.

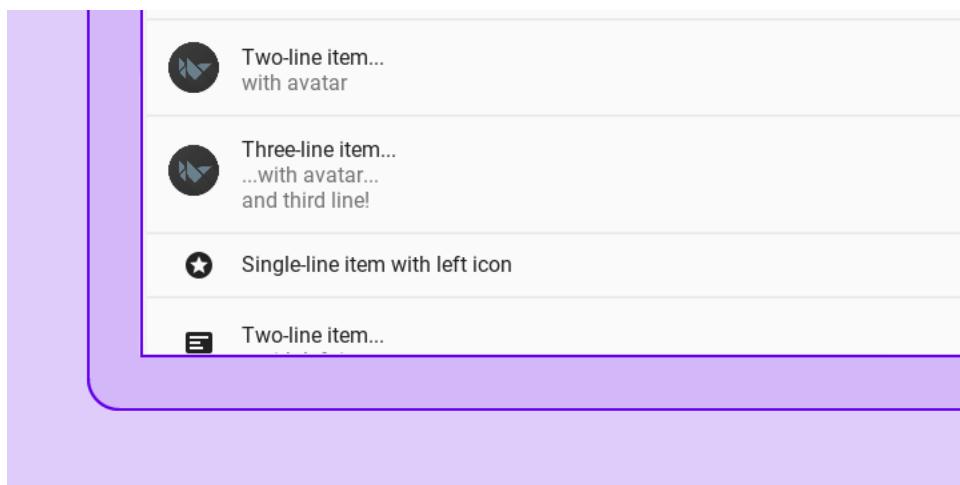
See [on\\_touch\\_down \(\)](#) for more information.

## 2.3.25 List

### See also:

Material Design spec, Lists

**Lists are continuous, vertical indexes of text or images.**



The class [MDList](#) in combination with a [BaseListItem](#) like [OneLineListItem](#) will create a list that expands as items are added to it, working nicely with Kivy's [ScrollView](#).

Due to the variety in sizes and controls in the *Material Design spec*, this module suffers from a certain level of complexity to keep the widgets compliant, flexible and performant.

For this KivyMD provides list items that try to cover the most common usecases, when those are insufficient, there's a base class called [BaseListItem](#) which you can use to create your own list items. This documentation will only cover the provided ones, for custom implementations please refer to this module's source code.

KivyMD provides the following list items classes for use:

### Text only ListItems

- [OneLineListItem](#)
- [TwoLineListItem](#)
- [ThreeLineListItem](#)

## ListItems with widget containers

These widgets will take other widgets that inherit from `ILeftBody`, `ILeftBodyTouch`, `IRightBody` or `IRightBodyTouch` and put them in their corresponding container.

As the name implies, `ILeftBody` and `IRightBody` will signal that the widget goes into the left or right container, respectively.

`ILeftBodyTouch` and `IRightBodyTouch` do the same thing, except these widgets will also receive touch events that occur within their surfaces.

KivyMD provides base classes such as `ImageLeftWidget`, `ImageRightWidget`, `IconRightWidget`, `IconLeftWidget`, based on the above classes.

### Allows the use of items with custom widgets on the left.

- `OneLineAvatarListItem`
- `TwoLineAvatarListItem`
- `ThreeLineAvatarListItem`
- `OneLineIconListItem`
- `TwoLineIconListItem`
- `ThreeLineIconListItem`

### It allows the use of elements with custom widgets on the left and the right.

- `OneLineAvatarIconListItem`
- `TwoLineAvatarIconListItem`
- `ThreeLineAvatarIconListItem`

## Usage

```
from kivy.lang import Builder

from kivymd.app import MDApp
from kivymd.uix.list import OneLineListItem

KV = '''
ScrollView:

    MDList:
        id: container
'''


class Test(MDApp):
    def build(self):
        return Builder.load_string(KV)

    def on_start(self):
        for i in range(20):
```

(continues on next page)

(continued from previous page)

```
        self.root.ids.container.add_widget(
            OneLineListItem(text=f"Single-line item {i}")
        )
Test().run()
```

## Events of List

```
from kivy.lang import Builder

from kivymd.app import MDApp

KV = '''
ScrollView:

    MDList:

        OneLineAvatarIconListItem:
            on_release: print("Click!")

            IconLeftWidget:
                icon: "github"

        OneLineAvatarIconListItem:
            on_release: print("Click 2!")

            IconLeftWidget:
                icon: "gitlab"
        ...

class MainApp(MDApp):
    def build(self):
        return Builder.load_string(KV)

MainApp().run()
```

## OneLineListItem

```
OneLineListItem:
    text: "Single-line item"
```

Single-line item

## TwoLineListItem

```
TwoLineListItem:  
    text: "Two-line item"  
    secondary_text: "Secondary text here"
```

Two-line item  
Secondary text here

## ThreeLineListItem

```
ThreeLineListItem:  
    text: "Three-line item"  
    secondary_text: "This is a multi-line label where you can"  
    tertiary_text: "fit more text than usual"
```

Three-line item  
This is a multi-line label where you...  
fit more text than usual

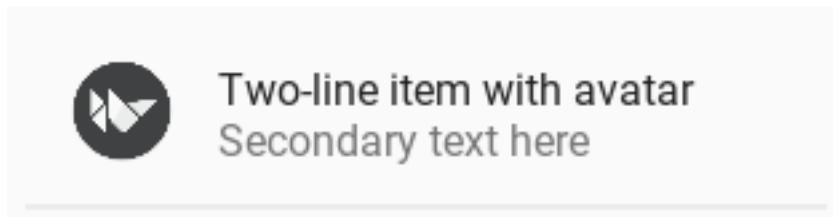
## OneLineAvatarListItem

```
OneLineAvatarListItem:  
    text: "Single-line item with avatar"  
  
    ImageLeftWidget:  
        source: "data/logo/kivy-icon-256.png"
```



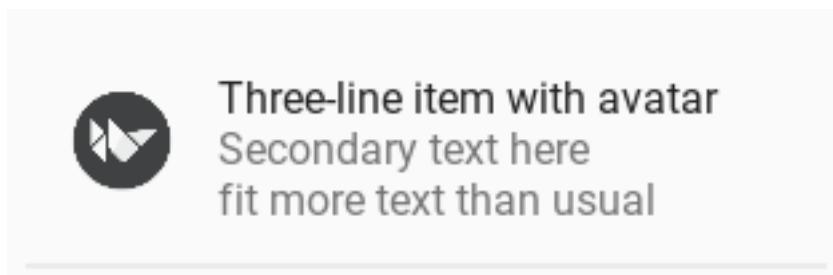
### TwoLineAvatarListItem

```
TwoLineAvatarListItem:  
    text: "Two-line item with avatar"  
    secondary_text: "Secondary text here"  
  
ImageLeftWidget:  
    source: "data/logo/kivy-icon-256.png"
```



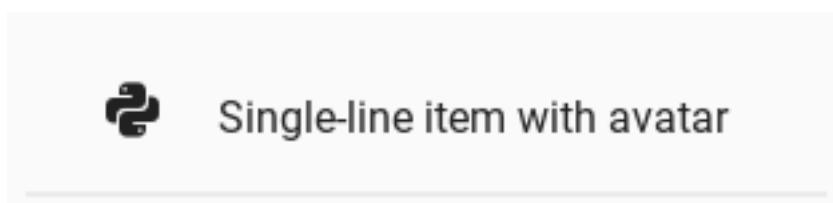
### ThreeLineAvatarListItem

```
ThreeLineAvatarListItem:  
    text: "Three-line item with avatar"  
    secondary_text: "Secondary text here"  
    tertiary_text: "fit more text than usual"  
  
ImageLeftWidget:  
    source: "data/logo/kivy-icon-256.png"
```



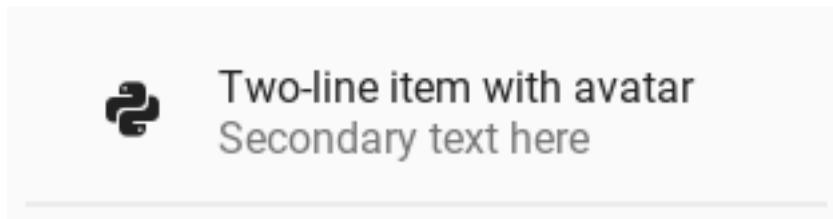
### OneLineIconListItem

```
OneLineAvatarListItem:  
    text: "Single-line item with avatar"  
  
IconLeftWidget:  
    icon: "language-python"
```



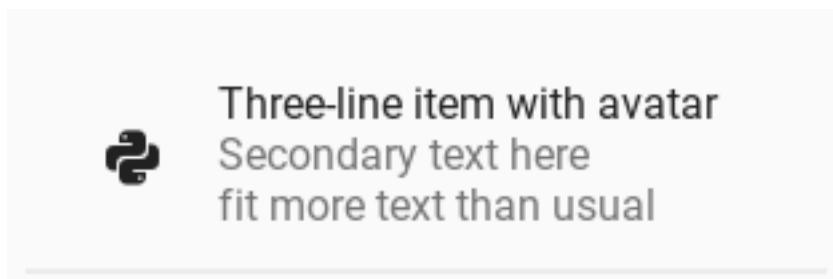
### TwoLineIconListItem

```
TwoLineIconListItem:  
    text: "Two-line item with avatar"  
    secondary_text: "Secondary text here"  
  
    IconLeftWidget:  
        icon: "language-python"
```



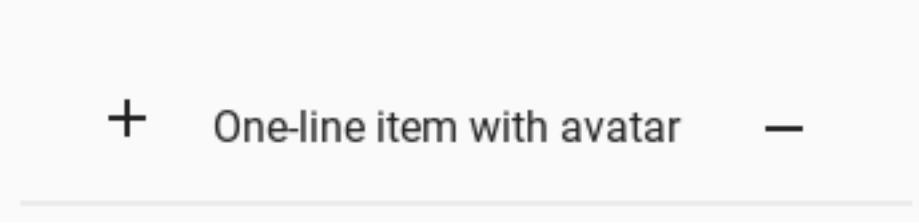
### ThreeLineIconListItem

```
ThreeLineIconListItem:  
    text: "Three-line item with avatar"  
    secondary_text: "Secondary text here"  
    tertiary_text: "fit more text than usual"  
  
    IconLeftWidget:  
        icon: "language-python"
```



### OneLineAvatarIconListItem

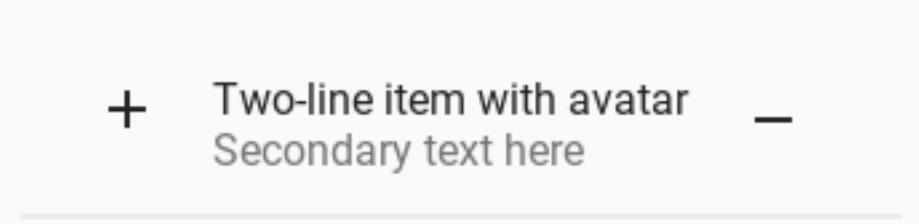
```
OneLineAvatarIconListItem:  
    text: "One-line item with avatar"  
  
    IconLeftWidget:  
        icon: "plus"  
  
    IconRightWidget:  
        icon: "minus"
```



+ One-line item with avatar -

### TwoLineAvatarIconListItem

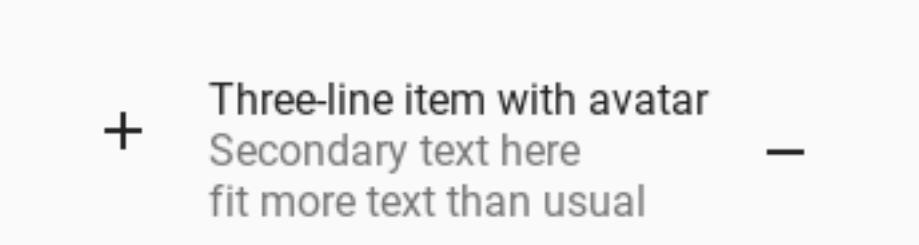
```
TwoLineAvatarIconListItem:  
    text: "Two-line item with avatar"  
    secondary_text: "Secondary text here"  
  
    IconLeftWidget:  
        icon: "plus"  
  
    IconRightWidget:  
        icon: "minus"
```



+ Two-line item with avatar -  
Secondary text here

### ThreeLineAvatarIconListItem

```
ThreeLineAvatarIconListItem:  
    text: "Three-line item with avatar"  
    secondary_text: "Secondary text here"  
    tertiary_text: "fit more text than usual"  
  
    IconLeftWidget:  
        icon: "plus"  
  
    IconRightWidget:  
        icon: "minus"
```



+ Three-line item with avatar -  
Secondary text here  
fit more text than usual

## Custom list item

```

from kivy.lang import Builder
from kivy.properties import StringProperty

from kivymd.app import MDApp
from kivymd.uix.list import IRightBodyTouch, OneLineAvatarIconListItem
from kivymd.uix.selectioncontrol import MDCheckbox
from kivymd.icon_definitions import md_icons

KV = '''
<ListItemWithCheckbox>:

    IconLeftWidget:
        icon: root.icon

    RightCheckbox:

BoxLayout:
    ScrollView:

        MDList:
            id: scroll
'''


class ListItemWithCheckbox(OneLineAvatarIconListItem):
    '''Custom list item.'''
    icon = StringProperty("android")

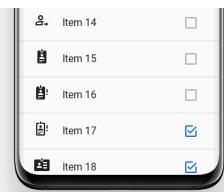

class RightCheckbox(IRightBodyTouch, MDCheckbox):
    '''Custom right container.'''
    pass


class MainApp(MDApp):
    def build(self):
        return Builder.load_string(KV)

    def on_start(self):
        icons = list(md_icons.keys())
        for i in range(30):
            self.root.ids.scroll.add_widget(
                ListItemWithCheckbox(text=f"Item {i}", icon=icons[i])
            )

MainApp().run()

```



```

from kivy.lang import Builder

from kivymd.app import MDApp
from kivymd.uix.boxlayout import MDBBoxLayout
from kivymd.uix.list import IRightBodyTouch

KV = '''
OneLineAvatarIconListItem:
    text: "One-line item with avatar"
    on_size:
        self.ids._right_container.width = container.width
        self.ids._right_container.x = container.width

    IconLeftWidget:
        icon: "cog"

    Container:
        id: container

        MDIconButton:
            icon: "minus"

        MDIconButton:
            icon: "plus"
'''


class Container(IRightBodyTouch, MDBBoxLayout):
    adaptive_width = True


class MainApp(MDApp):
    def build(self):
        return Builder.load_string(KV)

MainApp().run()

```

One-line item with avatar - +

**API - kivymd.uix.list**

```
class kivymd.uix.list.MDList(**kwargs)
```

ListItem container. Best used in conjunction with a kivy.uixScrollView.

When adding (or removing) a widget, it will resize itself to fit its children, plus top and bottom paddings as described by the *MD* spec.

```
add_widget (self, widget, index=0, canvas=None)
```

Add a new widget as a child of this widget.

**Parameters**

**widget: Widget** Widget to add to our list of children.

**index: int, defaults to 0** Index to insert the widget in the list. Notice that the default of 0 means the widget is inserted at the beginning of the list and will thus be drawn on top of other sibling widgets. For a full discussion of the index and widget hierarchy, please see the [Widgets Programming Guide](#).

New in version 1.0.5.

**canvas: str, defaults to None** Canvas to add widget's canvas to. Can be 'before', 'after' or None for the default canvas.

New in version 1.9.0.

```
>>> from kivy.uix.button import Button
>>> from kivy.uix.slider import Slider
>>> root = Widget()
>>> root.add_widget(Button())
>>> slider = Slider()
>>> root.add_widget(slider)
```

```
remove_widget (self, widget)
```

Remove a widget from the children of this widget.

**Parameters**

**widget: Widget** Widget to remove from our children list.

```
>>> from kivy.uix.button import Button
>>> root = Widget()
>>> button = Button()
>>> root.add_widget(button)
>>> root.remove_widget(button)
```

```
class kivymd.uix.list.BaseListItem(**kwargs)
```

Base class to all ListItems. Not supposed to be instantiated on its own.

**text**

Text shown in the first line.

*text* is a [StringProperty](#) and defaults to ''.

**text\_color**

Text color in rgba format used if *theme\_text\_color* is set to 'Custom'.

*text\_color* is a [ListProperty](#) and defaults to *None*.

**font\_style**

Text font style. See `kivymd.font_definitions.py`.

`font_style` is a `OptionProperty` and defaults to ‘`Subtitle1`’.

**theme\_text\_color**  
Theme text color in `rgba` format for primary text.  
`theme_text_color` is a `StringProperty` and defaults to ‘`Primary`’.

**secondary\_text**  
Text shown in the second line.  
`secondary_text` is a `StringProperty` and defaults to ‘’.

**tertiary\_text**  
The text is displayed on the third line.  
`tertiary_text` is a `StringProperty` and defaults to ‘’.

**secondary\_text\_color**  
Text color in `rgba` format used for secondary text if `secondary_theme_text_color` is set to ‘`Custom`’.  
`secondary_text_color` is a `ListProperty` and defaults to `None`.

**tertiary\_text\_color**  
Text color in `rgba` format used for tertiary text if `secondary_theme_text_color` is set to ‘`Custom`’.  
`tertiary_text_color` is a `ListProperty` and defaults to `None`.

**secondary\_theme\_text\_color**  
Theme text color for secondary text.  
`secondary_theme_text_color` is a `StringProperty` and defaults to ‘`Secondary`’.

**tertiary\_theme\_text\_color**  
Theme text color for tertiary text.  
`tertiary_theme_text_color` is a `StringProperty` and defaults to ‘`Secondary`’.

**secondary\_font\_style**  
Font style for secondary line. See `kivymd.font_definitions.py`.  
`secondary_font_style` is a `OptionProperty` and defaults to ‘`Body1`’.

**tertiary\_font\_style**  
Font style for tertiary line. See `kivymd.font_definitions.py`.  
`tertiary_font_style` is a `OptionProperty` and defaults to ‘`Body1`’.

**divider**  
Divider mode. Available options are: ‘`Full`’, ‘`Inset`’ and default to ‘`Full`’.  
`tertiary_font_style` is a `OptionProperty` and defaults to ‘`Body1`’.

**bg\_color**  
Background color for menu item.  
`bg_color` is a `ListProperty` and defaults to `[]`.

**class kivymd.uix.list.ILeftBody**  
Pseudo-interface for widgets that go in the left container for `ListItems` that support it.  
Implements nothing and requires no implementation, for annotation only.

```
class kivymd.uix.list.ILeftBodyTouch
    Same as ILeftBody, but allows the widget to receive touch events instead of triggering the ListItem's ripple effect.

class kivymd.uix.list.IRightBodyTouch
    Pseudo-interface for widgets that go in the right container for ListItems that support it.

    Implements nothing and requires no implementation, for annotation only.

class kivymd.uix.list.IRightBodyTouch
    Same as IRightBody, but allows the widget to receive touch events instead of triggering the ListItem's ripple effect

class kivymd.uix.list.ContainerSupport
    Overrides add_widget in a ListItem to include support for I*Body widgets when the appropriate containers are present.

    add_widget (self, widget, index=0)
    remove_widget (self, widget)
    on_touch_down (self, touch)
    on_touch_move (self, touch, *args)
    on_touch_up (self, touch)
    propagate_touch_to_touchable_widgets (self, touch, touch_event, *args)

class kivymd.uix.list.OneLineListItem(**kwargs)
    A one line list item.

class kivymd.uix.list.TwoLineListItem(**kwargs)
    A two line list item.

class kivymd.uix.list.ThreeLineListItem(**kwargs)
    A three line list item.

class kivymd.uix.list.OneLineAvatarListItem(**kwargs)
    Overrides add_widget in a ListItem to include support for I*Body widgets when the appropriate containers are present.

class kivymd.uix.list.TwoLineAvatarListItem(**kwargs)
    Overrides add_widget in a ListItem to include support for I*Body widgets when the appropriate containers are present.

class kivymd.uix.list.ThreeLineAvatarListItem(**kwargs)
    Overrides add_widget in a ListItem to include support for I*Body widgets when the appropriate containers are present.

class kivymd.uix.list.OneLineIconListItem(**kwargs)
    Overrides add_widget in a ListItem to include support for I*Body widgets when the appropriate containers are present.

class kivymd.uix.list.TwoLineIconListItem(**kwargs)
    Overrides add_widget in a ListItem to include support for I*Body widgets when the appropriate containers are present.

class kivymd.uix.list.ThreeLineIconListItem(**kwargs)
    Overrides add_widget in a ListItem to include support for I*Body widgets when the appropriate containers are present.
```

```
class kivymd.uix.list.OneLineRightIconListItem(**kwargs)
    Overrides add_widget in a ListItem to include support for I*Body widgets when the appropriate containers are present.

class kivymd.uix.list.TwoLineRightIconListItem(**kwargs)
    Overrides add_widget in a ListItem to include support for I*Body widgets when the appropriate containers are present.

class kivymd.uix.list.ThreeLineRightIconListItem(**kwargs)
    Overrides add_widget in a ListItem to include support for I*Body widgets when the appropriate containers are present.

class kivymd.uix.list.OneLineAvatarIconListItem(**kwargs)
    Overrides add_widget in a ListItem to include support for I*Body widgets when the appropriate containers are present.

class kivymd.uix.list.TwoLineAvatarIconListItem(**kwargs)
    Overrides add_widget in a ListItem to include support for I*Body widgets when the appropriate containers are present.

class kivymd.uix.list.ThreeLineAvatarIconListItem(**kwargs)
    Overrides add_widget in a ListItem to include support for I*Body widgets when the appropriate containers are present.

class kivymd.uix.list.ImageLeftWidget(**kwargs)
    Pseudo-interface for widgets that go in the left container for ListItems that support it.

    Implements nothing and requires no implementation, for annotation only.

class kivymd.uix.list.ImageRightWidget(**kwargs)
    Same as IRightBody, but allows the widget to receive touch events instead of triggering the ListItem's ripple effect

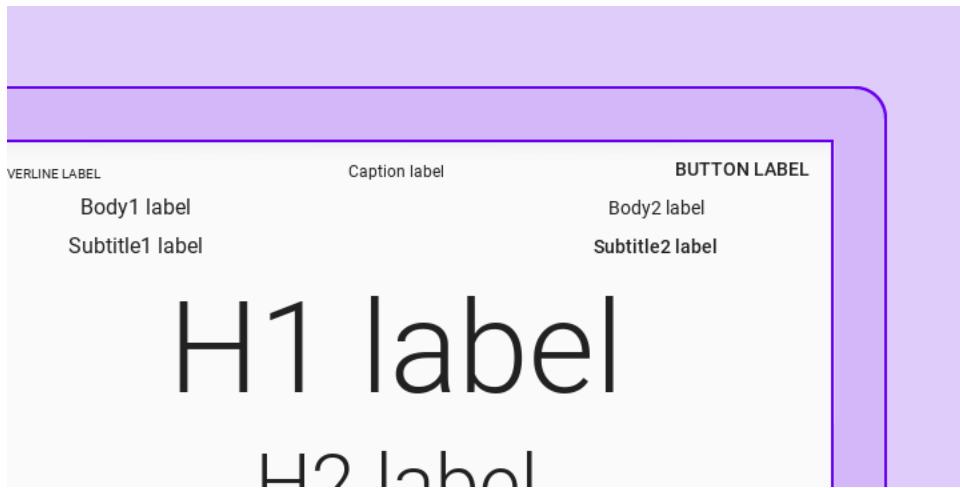
class kivymd.uix.list.IconRightWidget(**kwargs)
    Same as IRightBody, but allows the widget to receive touch events instead of triggering the ListItem's ripple effect

class kivymd.uix.list.IconLeftWidget(**kwargs)
    Same as ILeftBody, but allows the widget to receive touch events instead of triggering the ListItem's ripple effect.

class kivymd.uix.list.CheckboxLeftWidget(**kwargs)
    Same as ILeftBody, but allows the widget to receive touch events instead of triggering the ListItem's ripple effect.
```

### 2.3.26 Label

The `MDLabel` widget is for rendering text.



- `MDLabel`
- `MDIcon`

#### MDLabel

Class `MDLabel` inherited from the `Label` class but for `MDLabel` the `text_size` parameter is `(self.width, None)` and default is positioned on the left:

```
from kivy.lang import Builder

from kivymd.app import MDApp

KV = '''
Screen:

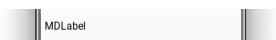
    BoxLayout:
        orientation: "vertical"

        MDToolbar:
            title: "MDLabel"

            MDLabel:
                text: "MDLabel"
'''


class Test(MDApp):
    def build(self):
        return Builder.load_string(KV)

Test().run()
```



---

**Note:** See `halign` and `valign` attributes of the `Label` class

---

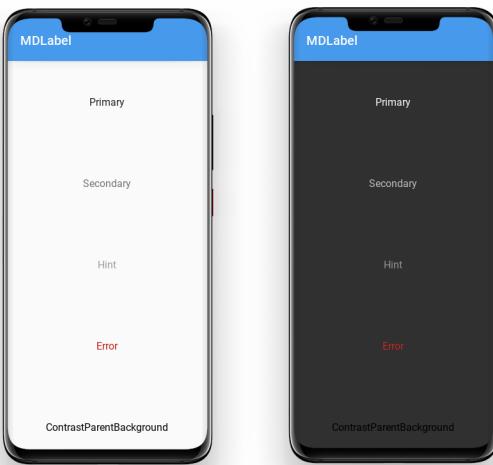
```
MDLabel:  
    text: "MDLabel"  
    halign: "center"
```



#### MDLabel color:

`MDLabel` provides standard color themes for label color management:

```
from kivy.lang import Builder  
  
from kivymd.app import MDApp  
from kivymd.uix.label import MDLabel  
  
KV = ''''  
Screen:  
  
    BoxLayout:  
        id: box  
        orientation: "vertical"  
  
        MDToolbar:  
            title: "MDLabel"  
'''  
  
  
class Test(MDApp):  
    def build(self):  
        screen = Builder.load_string(KV)  
        # Names of standard color themes.  
        for name_theme in [  
            "Primary",  
            "Secondary",  
            "Hint",  
            "Error",  
            "ContrastParentBackground",  
        ]:  
            screen.ids.box.add_widget(  
                MDLabel(  
                    text=name_theme,  
                    halign="center",  
                    theme_text_color=name_theme,  
                )  
            )  
        return screen  
  
Test().run()
```



To use a custom color for `MDLabel`, use a theme ‘*Custom*’. After that, you can specify the desired color in the `rgba` format in the `text_color` parameter:

```
MDLabel:
    text: "Custom color"
    halign: "center"
    theme_text_color: "Custom"
    text_color: 0, 0, 1, 1
```



`MDLabel` provides standard font styles for labels. To do this, specify the name of the desired style in the `font_style` parameter:

```
from kivy.lang import Builder

from kivymd.app import MDApp
from kivymd.uix.label import MDLabel
from kivymd.font_definitions import theme_fonts

KV = '''
Screen:

    BoxLayout:
        orientation: "vertical"

        MDToolbar:
            title: "MDLabel"

        ScrollView:

            MDList:
                id: box
'''


class Test(MDApp):
    def build(self):
        screen = Builder.load_string(KV)
```

(continues on next page)

(continued from previous page)

```
# Names of standard font styles.
for name_style in theme_font_styles[:-1]:
    screen.ids.box.add_widget(
        MDLabel(
            text=f'{name_style} style',
            halign='center',
            font_style=name_style,
        )
    )
return screen

Test().run()
```

## MDIcon

You can use labels to display material design icons using the `MDIcon` class.

**See also:**

[Material Design Icons](#)

[Material Design Icon Names](#)

The `MDIcon` class is inherited from `MDLabel` and has the same parameters.

**Warning:** For the `MDIcon` class, you cannot use `text` and `font_style` options!

```
MDIcon:
    halign: "center"
    icon: "language-python"
```



## API - kivymd.uix.label

`class kivymd.uix.label.MDLabel(**kwargs)`

Label class, see module documentation for more information.

### Events

`on_ref_press` Fired when the user clicks on a word referenced with a `[ref]` tag in a text markup.

### font\_style

Label font style.

Available options are: `'H1'`, `'H2'`, `'H3'`, `'H4'`, `'H5'`, `'H6'`, `'Subtitle1'`, `'Subtitle2'`, `'Body1'`, `'Body2'`, `'Button'`, `'Caption'`, `'Overline'`, `'Icon'`.

`font_style` is an `OptionProperty` and defaults to `'Body1'`.

### text

Text of the label.

**theme\_text\_color**  
Label color scheme name.  
Available options are: ‘Primary’, ‘Secondary’, ‘Hint’, ‘Error’, ‘Custom’, ‘ContrastParentBackground’.  
`theme_text_color` is an `OptionProperty` and defaults to `None`.

**text\_color**  
Label text color in `rgba` format.  
`text_color` is an `ListProperty` and defaults to `None`.

**parent\_background**

**can\_capitalize**

**update\_font\_style** (`self, *args`)

**on\_theme\_text\_color** (`self, instance, value`)

**on\_text\_color** (`self, *args`)

**on\_opposite\_colors** (`self, instance, value`)

**class** `kivymd.uix.label.MDIcon(**kwargs)`  
Label class, see module documentation for more information.

**Events**

**on\_ref\_press** Fired when the user clicks on a word referenced with a `[ref]` tag in a text markup.

**icon**  
Label icon name.  
`icon` is an `StringProperty` and defaults to ‘android’.

**source**  
Path to icon.  
`source` is an `StringProperty` and defaults to `None`.

### 2.3.27 Card

#### See also:

Material Design spec, Cards

**Cards contain content and actions about a single subject.**

KivyMD provides the following card classes for use:

- `MDCard`
- `MDCardSwipe`

---

**Note:** `MDCard` inherited from `BoxLayout`. You can use all parameters and attributes of the `BoxLayout` class in the `MDCard` class.

---

## MDCard

```
from kivy.lang import Builder

from kivymd.app import MDApp

KV = '''
Screen:

    MDCard:
        size_hint: None, None
        size: "280dp", "180dp"
        pos_hint: {"center_x": .5, "center_y": .5}
'''


class TestCard(MDApp):
    def build(self):
        return Builder.load_string(KV)

TestCard().run()
```



## Add content to card:

```
from kivy.lang import Builder

from kivymd.app import MDApp

KV = '''
Screen:

    MDCard:
        orientation: "vertical"
        padding: "8dp"
        size_hint: None, None
        size: "280dp", "180dp"
        pos_hint: {"center_x": .5, "center_y": .5}

        MDLabel:
            text: "Title"
            theme_text_color: "Secondary"
            size_hint_y: None
            height: self.texture_size[1]

        MDSeparator:
            height: "1dp"

        MDLabel:
'''
```

(continues on next page)

(continued from previous page)

```

        text: "Body"
    ...

class TestCard(MDApp):
    def build(self):
        return Builder.load_string(KV)

TestCard().run()

```



## MDCardSwipe

To create a card with *swipe-to-delete* behavior, you must create a new class that inherits from the `MDCardSwipe` class:

```

<SwipeToDeleteItem>:
    size_hint_y: None
    height: content.height

MDCardSwipeLayerBox:

MDCardSwipeFrontBox:

    OneLineListItem:
        id: content
        text: root.text
        _no_ripple_effect: True

```

```

class SwipeToDeleteItem(MDCardSwipe):
    text = StringProperty()

```



**End full code**

```
from kivy.lang import Builder
from kivy.properties import StringProperty

from kivymd.app import MDApp
from kivymd.uix.card import MDCardSwipe

KV = '''
<SwipeToDeleteItem>:
    size_hint_y: None
    height: content.height

    MDCardSwipeLayerBox:
        # Content under the card.

    MDCardSwipeFrontBox:

        # Content of card.

        OneLineListItem:
            id: content
            text: root.text
            _no_ripple_effect: True

Screen:

    BoxLayout:
        orientation: "vertical"
        spacing: "10dp"

        MDToolbar:
            elevation: 10
            title: "MDCardSwipe"

        ScrollView:
            scroll_timeout : 100

            MDList:
                id: md_list
                padding: 0
    '''

class SwipeToDeleteItem(MDCardSwipe):
    '''Card with `swipe-to-delete` behavior.'''

    text = StringProperty()

class TestCard(MDApp):
    def __init__(self, **kwargs):
        super().__init__(**kwargs)
        self.screen = Builder.load_string(KV)

    def build(self):
        return self.screen
```

(continues on next page)

(continued from previous page)

```
def on_start(self):
    '''Creates a list of cards.'''

    for i in range(20):
        self.screen.ids.md_list.add_widget(
            SwipeToDeleteItem(text=f"One-line item {i}")
    )

TestCard().run()
```

### Binding a swipe to one of the sides of the screen

```
<SwipeToDeleteItem>:
    # By default, the parameter is "left"
    anchor: "right"
```

### Swipe behavior

```
<SwipeToDeleteItem>:
    # By default, the parameter is "hand"
    type_swipe: "hand"
```

```
<SwipeToDeleteItem>:
    type_swipe: "auto"
```

### Removing an item using the `type_swipe = "auto"` parameter

The map provides the `MDCardSwipe.on_swipe_complete` event. You can use this event to remove items from a list:

```
<SwipeToDeleteItem>:
    on_swipe_complete: app.on_swipe_complete(root)
```

```
def on_swipe_complete(self, instance):
    self.screen.ids.md_list.remove_widget(instance)
```

**End full code**

```
from kivy.lang import Builder
from kivy.properties import StringProperty

from kivymd.app import MDApp
from kivymd.uix.card import MDCardSwipe

KV = '''
<SwipeToDeleteItem>:
    size_hint_y: None
    height: content.height
    type_swipe: "auto"
    on_swipe_complete: app.on_swipe_complete(root)

    MDCardSwipeLayerBox:

        MDCardSwipeFrontBox:

            OneLineListItem:
                id: content
                text: root.text
                _no_ripple_effect: True

Screen:

    BoxLayout:
        orientation: "vertical"
        spacing: "10dp"

        MDToolbar:
            elevation: 10
            title: "MDCardSwipe"

        ScrollView:

            MDList:
                id: md_list
                padding: 0
    '''

class SwipeToDeleteItem(MDCardSwipe):
    text = StringProperty()

class TestCard(MDApp):
    def __init__(self, **kwargs):
        super().__init__(**kwargs)
        self.screen = Builder.load_string(KV)

    def build(self):
        return self.screen

    def on_swipe_complete(self, instance):
        self.screen.ids.md_list.remove_widget(instance)
```

(continues on next page)

(continued from previous page)

```

def on_start(self):
    for i in range(20):
        self.screen.ids.md_list.add_widget(
            SwipeToDeleteItem(text=f"One-line item {i}")
    )

TestCard().run()

```

### Add content to the bottom layer of the card

To add content to the bottom layer of the card, use the `MDCardSwipeLayerBox` class.

```

<SwipeToDeleteItem>:

    MDCardSwipeLayerBox:
        padding: "8dp"

        MDIconButton:
            icon: "trash-can"
            pos_hint: {"center_y": .5}
            on_release: app.remove_item(root)

```

### End full code

```

from kivy.lang import Builder
from kivy.properties import StringProperty

from kivymd.app import MDApp
from kivymd.uix.card import MDCardSwipe

KV = '''
<SwipeToDeleteItem>:
    size_hint_y: None
    height: content.height

    MDCardSwipeLayerBox:
        padding: "8dp"

        MDIconButton:
            icon: "trash-can"
            pos_hint: {"center_y": .5}
            on_release: app.remove_item(root)

    MDCardSwipeFrontBox:

        OneLineListItem:
            id: content
            text: root.text
            _no_ripple_effect: True

```

(continues on next page)

(continued from previous page)

Screen:

```
BoxLayout:
    orientation: "vertical"
    spacing: "10dp"

    MDToolbar:
        elevation: 10
        title: "MDCardSwipe"

    ScrollView:

        MDList:
            id: md_list
            padding: 0
    ...

class SwipeToDeleteItem(MDCardSwipe):
    text = StringProperty()

class TestCard(MDApp):
    def __init__(self, **kwargs):
        super().__init__(**kwargs)
        self.screen = Builder.load_string(KV)

    def build(self):
        return self.screen

    def remove_item(self, instance):
        self.screen.ids.md_list.remove_widget(instance)

    def on_start(self):
        for i in range(20):
            self.screen.ids.md_list.add_widget(
                SwipeToDeleteItem(text=f"One-line item {i}")
            )

TestCard().run()
```

## Focus behavior

```
MDCard:
    focus_behavior: True
```

## Ripple behavior

```
MDCard:
    ripple_behavior: True
```

## End full code

```
from kivy.lang import Builder

from kivymd.app import MDApp

KV = '''
<StarButton@MDIconButton>
    icon: "star"
    on_release: self.icon = "star-outline" if self.icon == "star" else "star"

Screen:

    MDCard:
        orientation: "vertical"
        size_hint: .5, None
        height: box_top.height + box_bottom.height
        focus_behavior: True
        ripple_behavior: True
        pos_hint: {"center_x": .5, "center_y": .5}

        MDBoxLayout:
            id: box_top
            spacing: "20dp"
            adaptive_height: True

            FitImage:
                source: "/Users/macbookair/album.jpeg"
                size_hint: .3, None
                height: text_box.height

        MDBoxLayout:
            id: text_box
            orientation: "vertical"
            adaptive_height: True
            spacing: "10dp"
            padding: 0, "10dp", "10dp", "10dp"
```

(continues on next page)

(continued from previous page)

```

MDLabel:
    text: "Ride the Lightning"
    theme_text_color: "Primary"
    font_style: "H5"
    bold: True
    size_hint_y: None
    height: self.texture_size[1]

MDLabel:
    text: "July 27, 1984"
    size_hint_y: None
    height: self.texture_size[1]
    theme_text_color: "Primary"

MDSeparator:

MDBoxLayout:
    id: box_bottom
    adaptive_height: True
    padding: "10dp", 0, 0, 0

MDLabel:
    text: "Rate this album"
    size_hint_y: None
    height: self.texture_size[1]
    pos_hint: {"center_y": .5}
    theme_text_color: "Primary"

StarButton:
StarButton:
StarButton:
StarButton:
StarButton:
StarButton:
...
.

class Test(MDApp):
    def build(self):
        self.theme_cls.theme_style = "Dark"
        return Builder.load_string(KV)

Test().run()

```

**API - kivymd.uix.card**

```

class kivymd.uix.card.MDSeparator(**kwargs)
A separator line.

color
    Separator color in rgba format.

    color is a ListProperty and defaults to [].

on_orientation(self, *args)

```

```
class kivymd.uix.card.MDCard(**kwargs)
```

Widget class. See module documentation for more information.

#### Events

**on\_touch\_down:** (*touch*, ) Fired when a new touch event occurs. *touch* is the touch object.

**on\_touch\_move:** (*touch*, ) Fired when an existing touch moves. *touch* is the touch object.

**on\_touch\_up:** (*touch*, ) Fired when an existing touch disappears. *touch* is the touch object.

**on\_kv\_post:** (*base\_widget*, ) Fired after all the kv rules associated with the widget and all other widgets that are in any of those rules have had all their kv rules applied. *base\_widget* is the base-most widget whose instantiation triggered the kv rules (i.e. the widget instantiated from Python, e.g. `MyWidget()`).

Changed in version 1.11.0.

**Warning:** Adding a `__del__` method to a class derived from Widget with Python prior to 3.4 will disable automatic garbage collection for instances of that class. This is because the Widget class creates reference cycles, thereby preventing garbage collection.

Changed in version 1.0.9: Everything related to event properties has been moved to the `EventDispatcher`. Event properties can now be used when contructing a simple class without subclassing `Widget`.

Changed in version 1.5.0: The constructor now accepts `on_*` arguments to automatically bind callbacks to properties or events, as in the Kv language.

#### background

Background image path.

`background` is a `StringProperty` and defaults to ''.

#### focus\_behavior

Using focus when hovering over a card.

`focus_behavior` is a `BooleanProperty` and defaults to `False`.

#### ripple\_behavior

Use ripple effect for card.

`ripple_behavior` is a `BooleanProperty` and defaults to `False`.

#### elevation

Elevation value.

`elevation` is an `NumericProperty` and defaults to 1.

#### on\_radius(*self, instance, value*)

```
class kivymd.uix.card.MDCardSwipe(**kw)
```

#### Events

**on\_swipe\_complete** Called when a swipe of card is completed.

#### open\_progress

Percent of visible part of side panel. The percent is specified as a floating point number in the range 0-1. 0.0 if panel is closed and 1.0 if panel is opened.

`open_progress` is a `NumericProperty` and defaults to 0.0.

**opening\_transition**

The name of the animation transition type to use when animating to the `state` ‘opened’.

`opening_transition` is a `StringProperty` and defaults to ‘out\_cubic’.

**closing\_transition**

The name of the animation transition type to use when animating to the `state` ‘closed’.

`closing_transition` is a `StringProperty` and defaults to ‘out\_sine’.

**anchor**

Anchoring screen edge for card. Available options are: ‘left’, ‘right’.

`anchor` is a `OptionProperty` and defaults to `left`.

**swipe\_distance**

The distance of the swipe with which the movement of navigation drawer begins.

`swipe_distance` is a `NumericProperty` and defaults to `50`.

**opening\_time**

The time taken for the card to slide to the `state` ‘open’.

`opening_time` is a `NumericProperty` and defaults to `0.2`.

**state**

Detailed state. Sets before `state`. Bind to `state` instead of status. Available options are: ‘closed’, ‘opened’.

`status` is a `OptionProperty` and defaults to ‘closed’.

**max\_swipe\_x**

If, after the events of `on_touch_up` card position exceeds this value - will automatically execute the method `open_card`, and if not - will automatically be `close_card` method.

`max_swipe_x` is a `NumericProperty` and defaults to `0.3`.

**max\_opened\_x**

The value of the position the card shifts to when `type_swipe` s set to ‘hand’.

`max_opened_x` is a `NumericProperty` and defaults to `100dp`.

**type\_swipe**

Type of card opening when swipe. Shift the card to the edge or to a set position `max_opened_x`. Available options are: ‘auto’, ‘hand’.

`type_swipe` is a `OptionProperty` and defaults to `auto`.

**add\_widget (self, widget, index=0, canvas=None)**

Add a new widget as a child of this widget.

**Parameters**

**widget: Widget** Widget to add to our list of children.

**index: int, defaults to 0** Index to insert the widget in the list. Notice that the default of 0 means the widget is inserted at the beginning of the list and will thus be drawn on top of other sibling widgets. For a full discussion of the index and widget hierarchy, please see the [Widgets Programming Guide](#).

New in version 1.0.5.

**canvas: str, defaults to None** Canvas to add widget’s canvas to. Can be ‘before’, ‘after’ or None for the default canvas.

New in version 1.9.0.

```
>>> from kivy.uix.button import Button
>>> from kivy.uix.slider import Slider
>>> root = Widget()
>>> root.add_widget(Button())
>>> slider = Slider()
>>> root.add_widget(slider)
```

**on\_swipe\_complete (self, \*args)**

Called when a swipe of card is completed.

**on\_anchor (self, instance, value)****on\_open\_progress (self, instance, value)****on\_touch\_move (self, touch)**

Receive a touch move event. The touch is in parent coordinates.

See [on\\_touch\\_down \(\)](#) for more information.

**on\_touch\_up (self, touch)**

Receive a touch up event. The touch is in parent coordinates.

See [on\\_touch\\_down \(\)](#) for more information.

**on\_touch\_down (self, touch)**

Receive a touch down event.

**Parameters**

**touch: MotionEvent class** Touch received. The touch is in parent coordinates. See [relativeLayout](#) for a discussion on coordinate systems.

**Returns** bool If True, the dispatching of the touch event will stop. If False, the event will continue to be dispatched to the rest of the widget tree.

**complete\_swipe (self)****open\_card (self)****close\_card (self)****class kivymd.uix.card.MDCardSwipeFrontBox (\*\*kwargs)**

Widget class. See module documentation for more information.

**Events**

**on\_touch\_down: (touch, )** Fired when a new touch event occurs. *touch* is the touch object.

**on\_touch\_move: (touch, )** Fired when an existing touch moves. *touch* is the touch object.

**on\_touch\_up: (touch, )** Fired when an existing touch disappears. *touch* is the touch object.

**on\_kv\_post: (base\_widget, )** Fired after all the kv rules associated with the widget and all other widgets that are in any of those rules have had all their kv rules applied. *base\_widget* is the base-most widget whose instantiation triggered the kv rules (i.e. the widget instantiated from Python, e.g. `MyWidget ()`).

Changed in version 1.11.0.

**Warning:** Adding a `__del__` method to a class derived from Widget with Python prior to 3.4 will disable automatic garbage collection for instances of that class. This is because the Widget class creates reference cycles, thereby preventing garbage collection.

Changed in version 1.0.9: Everything related to event properties has been moved to the `EventDispatcher`. Event properties can now be used when contructing a simple class without subclassing `Widget`.

Changed in version 1.5.0: The constructor now accepts `on_*` arguments to automatically bind callbacks to properties or events, as in the Kv language.

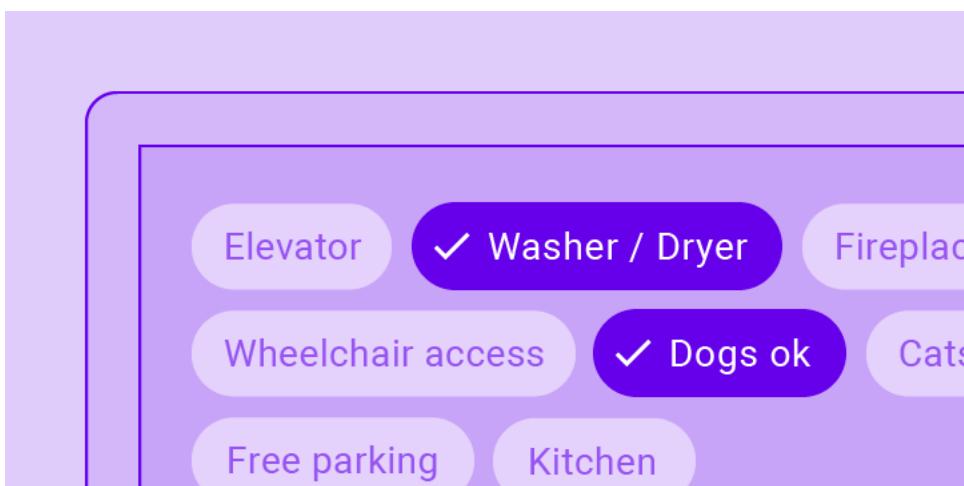
```
class kivymd.uix.card.MDCardSwipeLayerBox(**kwargs)
    Box layout class. See module documentation for more information.
```

## 2.3.28 Chip

See also:

Material Design spec, Chips

**Chips are compact elements that represent an input, attribute, or action.**

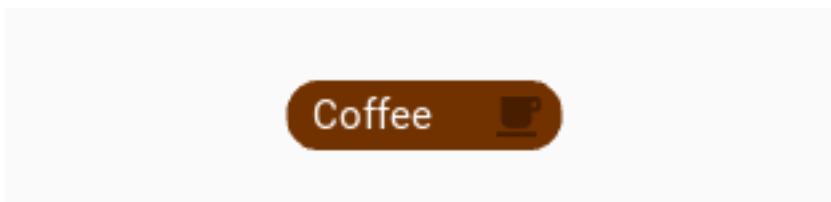


## Usage

```
MDChip:
    label: 'Coffee'
    color: .4470588235118, .1960787254902, 0, 1
    icon: 'coffee'
    callback: app.callback_for_menu_items
```

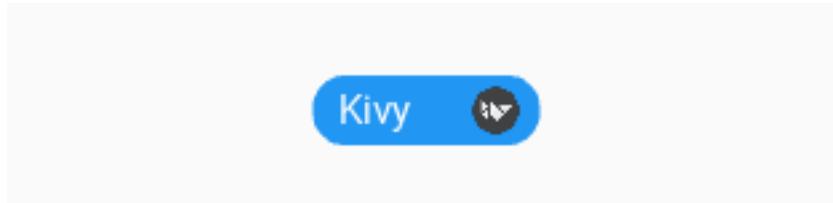
The user function takes two arguments - the object and the text of the chip:

```
def callback_for_menu_items(self, instance, value):
    print(instance, value)
```



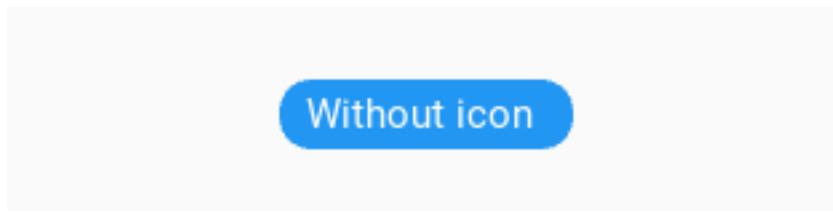
## Use custom icon

```
MDChip:
    label: 'Kivy'
    icon: 'data/logo/kivy-icon-256.png'
```



## Use without icon

```
MDChip:
    label: 'Without icon'
    icon: ''
```



## Chips with check

```
MDChip:
    label: 'Check with icon'
    icon: 'city'
    check: True
```

## Choose chip

```
MDChooseChip:

    MDChip:
        label: 'Earth'
        icon: 'earth'
        selected_chip_color: .21176470535294, .098039627451, 1, 1

    MDChip:
        label: 'Face'
        icon: 'face'
        selected_chip_color: .21176470535294, .098039627451, 1, 1

    MDChip:
```

(continues on next page)

(continued from previous page)

```
label: 'Facebook'  
icon: 'facebook'  
selected_chip_color: .21176470535294, .098039627451, 1, 1
```

---

**Note:** See full example

---

## API - kivymd.uix.chip

**class** kivymd.uix.chip.**MDChip**(\*\*kwargs)

Box layout class. See module documentation for more information.

**label**

Chip text.

*label* is an `StringProperty` and defaults to ''.

**icon**

Chip icon.

*icon* is an `StringProperty` and defaults to 'checkbox-blank-circle'.

**color**

Chip color in `rgba` format.

*color* is an `ListProperty` and defaults to [].

**text\_color**

Chip's text color in `rgba` format.

*text\_color* is an `ListProperty` and defaults to [].

**check**

If True, a checkmark is added to the left when touch to the chip.

*check* is an `BooleanProperty` and defaults to *False*.

**callback**

Custom method.

*callback* is an `ObjectProperty` and defaults to *None*.

**radius**

Corner radius values.

*radius* is an `NumericProperty` and defaults to '12dp'.

**selected\_chip\_color**

The color of the chip that is currently selected in `rgba` format.

*selected\_chip\_color* is an `ListProperty` and defaults to [].

**on\_icon**(*self, instance, value*)

**on\_touch\_down**(*self, touch*)

Receive a touch down event.

### Parameters

**touch: MotionEvent class** Touch received. The touch is in parent coordinates. See `relativelayout` for a discussion on coordinate systems.

**Returns** bool If True, the dispatching of the touch event will stop. If False, the event will continue to be dispatched to the rest of the widget tree.

```
class kivymd.uix.chip.MDChooseChip(**kwargs)
```

Stack layout class. See module documentation for more information.

```
add_widget(self, widget, index=0, canvas=None)
```

Add a new widget as a child of this widget.

#### Parameters

**widget: Widget** Widget to add to our list of children.

**index: int, defaults to 0** Index to insert the widget in the list. Notice that the default of 0 means the widget is inserted at the beginning of the list and will thus be drawn on top of other sibling widgets. For a full discussion of the index and widget hierarchy, please see the [Widgets Programming Guide](#).

New in version 1.0.5.

**canvas: str, defaults to None** Canvas to add widget's canvas to. Can be 'before', 'after' or None for the default canvas.

New in version 1.9.0.

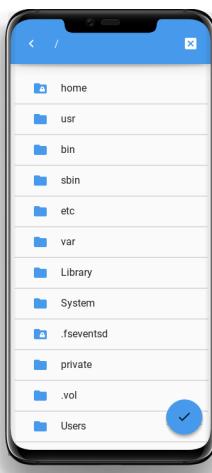
```
>>> from kivy.uix.button import Button
>>> from kivy.uix.slider import Slider
>>> root = Widget()
>>> root.add_widget(Button())
>>> slider = Slider()
>>> root.add_widget(slider)
```

## 2.3.29 File Manager

A simple manager for selecting directories and files.

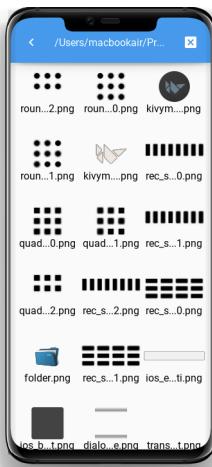
### Usage

```
path = '/' # path to the directory that will be opened in the file manager
file_manager = MDFileManager(
    exit_manager=self.exit_manager, # function called when the user reaches_
    ↪directory tree root
    select_path=self.select_path, # function called when selecting a file/directory
)
file_manager.show(path)
```



Or with preview mode:

```
file_manager = MDFFileManager(  
    exit_manager=self.exit_manager,  
    select_path=self.select_path,  
    preview=True,  
)
```



**Warning:** The *preview* mode is intended only for viewing images and will not display other types of files.

## Example

```
from kivy.core.window import Window  
from kivy.lang import Builder  
  
from kivymd.app import MDApp  
from kivymd.uix.filemanager import MDFFileManager  
from kivymd.toast import toast
```

(continues on next page)

(continued from previous page)

```

KV = '''
BoxLayout:
    orientation: 'vertical'

    MDToolbar:
        title: "MDFileManager"
        left_action_items: [['menu', lambda x: None]]
        elevation: 10

    FloatLayout:

        MDRoundFlatButton:
            text: "Open manager"
            icon: "folder"
            pos_hint: {'center_x': .5, 'center_y': .6}
            on_release: app.file_manager_open()
'''

class Example(MDApp):
    def __init__(self, **kwargs):
        super().__init__(**kwargs)
        Window.bind(on_keyboard=self.events)
        self.manager_open = False
        self.file_manager = MDFileManager(
            exit_manager=self.exit_manager,
            select_path=self.select_path,
            preview=True,
        )

    def build(self):
        return Builder.load_string(KV)

    def file_manager_open(self):
        self.file_manager.show('/') # output manager to the screen
        self.manager_open = True

    def select_path(self, path):
        '''It will be called when you click on the file name
        or the catalog selection button.

        :type path: str;
        :param path: path to the selected directory or file;
        '''

        self.exit_manager()
        toast(path)

    def exit_manager(self, *args):
        '''Called when the user reaches the root of the directory tree.'''

        self.manager_open = False
        self.file_manager.close()

    def events(self, instance, keyboard, keycode, text, modifiers):
        '''Called when buttons are pressed on the mobile device.'''

```

(continues on next page)

(continued from previous page)

```
if keyboard in (1001, 27):
    if self.manager_open:
        self.file_manager.back()
return True
```

```
Example().run()
```

## API - kivymd.uix.filemanager

**class** kivymd.uix.filemanager.**MDFFileManager**(\*\*kwargs)

Float layout class. See module documentation for more information.

### **icon**

The icon that will be used on the directory selection button.

*icon* is an `StringProperty` and defaults to `check`.

### **icon\_folder**

The icon that will be used for folder icons when using `preview = True`.

*icon* is an `StringProperty` and defaults to `check`.

### **exit\_manager**

Function called when the user reaches directory tree root.

*exit\_manager* is an `ObjectProperty` and defaults to `lambda x: None`.

### **select\_path**

Function, called when selecting a file/directory.

*select\_path* is an `ObjectProperty` and defaults to `lambda x: None`.

### **ext**

List of file extensions to be displayed in the manager. For example, `['.py', '.kv']` - will filter out all files, except python scripts and Kv Language.

*ext* is an `ListProperty` and defaults to `[]`.

### **search**

It can take the values ‘all’ ‘dirs’ ‘files’ - display only directories or only files or both them. By default, it displays folders, and files. Available options are: ‘all’, ‘dirs’, ‘files’.

*search* is an `OptionProperty` and defaults to `all`.

### **current\_path**

Current directory.

*current\_path* is an `StringProperty` and defaults to `/`.

### **use\_access**

Show access to files and directories.

*use\_access* is an `BooleanProperty` and defaults to `True`.

### **preview**

Shows only image previews.

*preview* is an `BooleanProperty` and defaults to `False`.

**show\_hidden\_files**

Shows hidden files.

`show_hidden_files` is an `BooleanProperty` and defaults to `False`.

**sort\_by**

It can take the values ‘nothing’ ‘name’ ‘date’ ‘size’ ‘type’ - sorts files by option By default, sort by name.  
Available options are: ‘*nothing*’, ‘*name*’, ‘*date*’, ‘*size*’, ‘*type*’.

`sort_by` is an `OptionProperty` and defaults to `name`.

**sort\_by\_desc**

Sort by descending.

`sort_by_desc` is an `BooleanProperty` and defaults to `False`.

**show(self, path)**

Forms the body of a directory tree.

**Parameters** `path` – The path to the directory that will be opened in the file manager.

**get\_access\_string(self, path)****get\_content(self)**

Returns a list of the type [[Folder List], [file list]].

**close(self)**

Closes the file manager window.

**select\_dir\_or\_file(self, path)**

Called by tap on the name of the directory or file.

**back(self)**

Returning to the branch down in the directory tree.

**select\_directory\_on\_press\_button(self, \*args)**

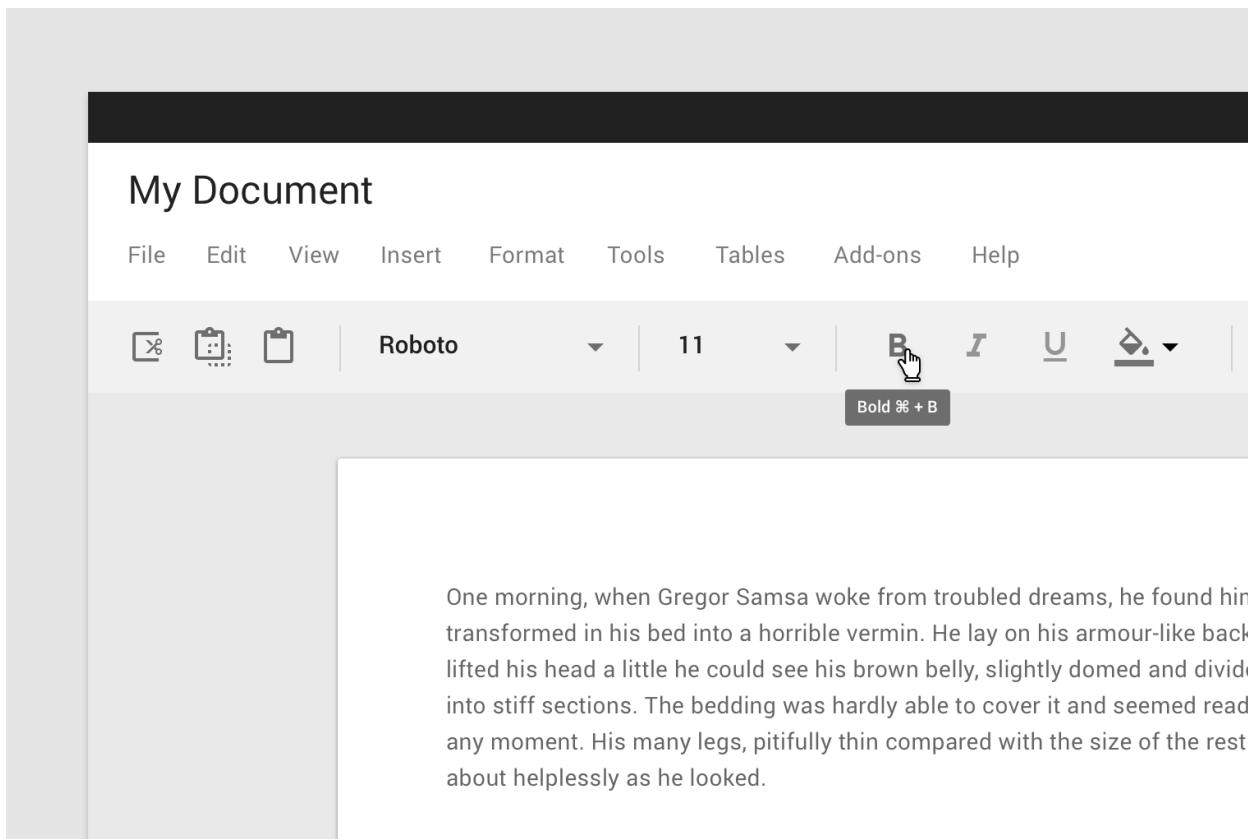
Called when a click on a floating button.

### 2.3.30 Tooltip

**See also:**

Material Design spec, Tooltips

**Tooltips display informative text when users hover over, focus on, or tap an element.**



To use the `MDTooltip` class, you must create a new class inherited from the `MDTooltip` class:

In Kv-language:

```
<TooltipMDIconButton@MDIconButton+MDTooltip>
```

In Python code:

```
class TooltipMDIconButton(MDIconButton, MDTooltip):
    pass
```

**Warning:** `MDTooltip` only works correctly with button and label classes.

```
from kivy.lang import Builder

from kivymd.app import MDApp

KV = '''
<TooltipMDIconButton@MDIconButton+MDTooltip>

Screen:

    TooltipMDIconButton:
```

(continues on next page)

(continued from previous page)

```

icon: "language-python"
tooltip_text: self.icon
pos_hint: {"center_x": .5, "center_y": .5}
...

class Test(MDApp):
    def build(self):
        return Builder.load_string(KV)

Test().run()

```

---

**Note:** The behavior of tooltips on desktop and mobile devices is different. For more detailed information, [click here](#).

---

### API - kivymd.uix.tooltip

**class** kivymd.uix.tooltip.MDTooltip(\*\*kwargs)

#### Events

**on\_enter** Call when mouse enter the bbox of the widget.

**on\_leave** Call when the mouse exit the widget.

#### tooltip\_bg\_color

Tooltip background color in rgba format.

*tooltip\_bg\_color* is an `ListProperty` and defaults to `[]`.

#### tooltip\_text\_color

Tooltip text color in rgba format.

*tooltip\_text\_color* is an `ListProperty` and defaults to `[]`.

#### tooltip\_text

Tooltip text.

*tooltip\_text* is an `StringProperty` and defaults to ''.

#### padding

**delete\_clock**(self, widget, touch, \*args)

**adjust\_tooltip\_position**(self, x, y)

Returns the coordinates of the tooltip that fit into the borders of the screen.

**display\_tooltip**(self, interval)

**animation\_tooltip\_show**(self, interval)

**remove\_tooltip**(self, \*args)

**on\_long\_touch**(self, touch, \*args)

Called when the widget is pressed for a long time.

**on\_enter** (*self*, \**args*)  
See *on\_enter* method in *HoverBehavior* class.

**on\_leave** (*self*)  
See *on\_leave* method in *HoverBehavior* class.

**class** kivymd.uix.tooltip.MDTooltipViewClass(\*\**kwargs*)  
Box layout class. See module documentation for more information.

**tooltip\_bg\_color**  
See *tooltip\_bg\_color*.

**tooltip\_text\_color**  
See *tooltip\_text\_color*.

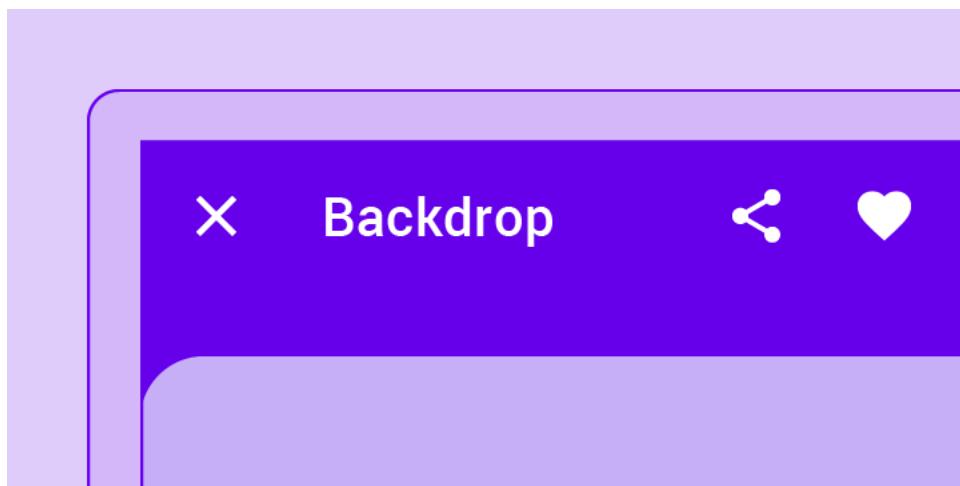
**tooltip\_text**  
See *tooltip\_text*.

### 2.3.31 Backdrop

See also:

Material Design spec, Backdrop

Skeleton layout for using **MDBackdrop**:



#### Usage

```
<Root>:  
  
    MDBackdrop:  
  
        MDBackdropBackLayer:  
  
            ContentForBackdropBackLayer:  
  
        MDBackdropFrontLayer:
```

(continues on next page)

(continued from previous page)

**ContentForBackdropFrontLayer:****Example**

```

from kivy.lang import Builder
from kivy.uix.screenmanager import Screen

from kivymd.app import MDApp

# Your layouts.
Builder.load_string(
    '''
#:import Window kivy.core.window.Window
#:import IconLeftWidget kivymd.uix.list.IconLeftWidget
#:import images_path kivymd.images_path

<ItemBackdropFrontLayer@TwoLineAvatarListItem>
    icon: "android"

    IconLeftWidget:
        icon: root.icon


<MyBackdropFrontLayer@ItemBackdropFrontLayer>
    backdrop: None
    text: "Lower the front layer"
    secondary_text: " by 50 %"
    icon: "transfer-down"
    on_press: root.backdrop.open(-Window.height / 2)
    pos_hint: {"top": 1}
    _no_ripple_effect: True


<MyBackdropBackLayer@Image>
    size_hint: .8, .8
    source: f"{images_path}/kivymd.png"
    pos_hint: {"center_x": .5, "center_y": .6}
    ...
)

# Usage example of MDBackdrop.
Builder.load_string(
    '''
<ExampleBackdrop>

    MDBackdrop:
        id: backdrop
        left_action_items: [['menu', lambda x: self.open()]]
        title: "Example Backdrop"
        header_text: "Menu:"


        MDBackdropBackLayer:

```

(continues on next page)

(continued from previous page)

```
MyBackdropBackLayer:  
    id: backlayer  
  
    MDBackdropFrontLayer:  
        MyBackdropFrontLayer:  
            backdrop: backdrop  
...  
)  
  
class ExampleBackdrop(Screen):  
    pass  
  
class TestBackdrop(MDApp):  
    def __init__(self, **kwargs):  
        super().__init__(**kwargs)  
  
    def build(self):  
        return ExampleBackdrop()  
  
TestBackdrop().run()
```

---

**Note:** See full example

---

## API - kivymd.uix.backdrop

**class** kivymd.uix.backdrop.**MDBackdrop**(*\*\*kwargs*)

### Events

**on\_open** When the front layer drops.

**on\_close** When the front layer rises.

### **padding**

Padding for contents of the front layer.

*padding* is an `ListProperty` and defaults to `[0, 0, 0, 0]`.

### **left\_action\_items**

The icons and methods left of the `kivymd.uix.toolbar.MDToolbar` in back layer. For more information, see the `kivymd.uix.toolbar.MDToolbar` module and `left_action_items` parameter.

`left_action_items` is an `ListProperty` and defaults to `[]`.

### **right\_action\_items**

Works the same way as `left_action_items`.

`right_action_items` is an `ListProperty` and defaults to `[]`.

**title**

See the `kivymd.uix.toolbar.MDToolbar.title` parameter.

`title` is an `StringProperty` and defaults to ''.

**background\_color**

Background color of back layer.

`background_color` is an `ListProperty` and defaults to `[]`.

**radius**

The value of the rounding radius of the upper left corner of the front layer.

`radius` is an `NumericProperty` and defaults to 25.

**header**

Whether to use a header above the contents of the front layer.

`header` is an `BooleanProperty` and defaults to `True`.

**header\_text**

Text of header.

`header_text` is an `StringProperty` and defaults to '`Header`'.

**close\_icon**

The name of the icon that will be installed on the toolbar on the left when opening the front layer.

`close_icon` is an `StringProperty` and defaults to '`close`'.

**on\_open(self)**

When the front layer drops.

**on\_close(self)**

When the front layer rises.

**on\_left\_action\_items(self, instance, value)****on\_header(self, instance, value)****open(self, open\_up\_to=0)**

Opens the front layer.

**Open\_up\_to** the height to which the front screen will be lowered; if equal to zero - falls to the bottom of the screen;

**close(self)**

Opens the front layer.

**animtion\_icon\_menu(self)****animtion\_icon\_close(self, instance\_animation, instance\_icon\_menu)****add\_widget(self, widget, index=0, canvas=None)**

Add a new widget as a child of this widget.

### Parameters

**widget: Widget** Widget to add to our list of children.

**index: int, defaults to 0** Index to insert the widget in the list. Notice that the default of 0 means the widget is inserted at the beginning of the list and will thus be drawn on top of other sibling widgets. For a full discussion of the index and widget hierarchy, please see the [Widgets Programming Guide](#).

New in version 1.0.5.

**canvas: str, defaults to None** Canvas to add widget's canvas to. Can be 'before', 'after' or None for the default canvas.

New in version 1.9.0.

```
>>> from kivy.uix.button import Button
>>> from kivy.uix.slider import Slider
>>> root = Widget()
>>> root.add_widget(Button())
>>> slider = Slider()
>>> root.add_widget(slider)
```

```
class kivymd.uix.backdrop.MDBackdropToolbar(**kwargs)
```

#### Events

**on\_action\_button** Method for the button used for the MDBottomAppBar class.

```
class kivymd.uix.backdrop.MDBackdropFrontLayer(**kwargs)
```

Box layout class. See module documentation for more information.

```
class kivymd.uix.backdrop.MDBackdropBackLayer(**kwargs)
```

Box layout class. See module documentation for more information.

### 2.3.32 RelativeLayout

RelativeLayout class equivalent. Simplifies working with some widget properties. For example:

#### RelativeLayout

```
RelativeLayout:
    canvas:
        Color:
            rgba: app.theme_cls.primary_color
        RoundedRectangle:
            pos: (0, 0)
            size: self.size
            radius: [25, ]
```

#### MDRelativeLayout

```
MDRelativeLayout:
    radius: [25, ]
    md_bg_color: app.theme_cls.primary_color
```

**API - kivymd.uix.relativelayout**

```
class kivymd.uix.relativelayout.MDRelativeLayout (**kw)
```

RelativeLayout class, see module documentation for more information.

### 2.3.33 StackLayout

StackLayout class equivalent. Simplifies working with some widget properties. For example:

**StackLayout**

```
StackLayout:
    size_hint_y: None
    height: self.minimum_height

    canvas:
        Color:
            rgba: app.theme_cls.primary_color
        Rectangle:
            pos: self.pos
            size: self.size
```

**MDStackLayout**

```
MDStackLayout:
    adaptive_height: True
    md_bg_color: app.theme_cls.primary_color
```

Available options are:

- *adaptive\_height*
- *adaptive\_width*
- *adaptive\_size*

**adaptive\_height**

```
adaptive_height: True
```

Equivalent

```
size_hint_y: None
height: self.minimum_height
```

### adaptive\_width

```
adaptive_width: True
```

Equivalent

```
size_hint_x: None  
height: self.minimum_width
```

### adaptive\_size

```
adaptive_size: True
```

Equivalent

```
size_hint: None, None  
size: self.minimum_size
```

## API - kivymd.uix.stacklayout

```
class kivymd.uix.stacklayout.MDStackLayout(**kwargs)  
Stack layout class. See module documentation for more information.
```

### 2.3.34 Screen

Screen class equivalent. Simplifies working with some widget properties. For example:

#### Screen

```
Screen:  
    canvas:  
        Color:  
            rgba: app.theme_cls.primary_color  
        RoundedRectangle:  
            pos: self.pos  
            size: self.size  
            radius: [25, 0, 0, 0]
```

#### MDScreen

```
MDScreen:  
    radius: [25, 0, 0, 0]  
    md_bg_color: app.theme_cls.primary_color
```

**API - kivymd.uix.screen**

```
class kivymd.uix.screen.MDScreen(**kw)
```

Screen is an element intended to be used with a ScreenManager. Check module documentation for more information.

**Events**

**on\_pre\_enter:** () Event fired when the screen is about to be used: the entering animation is started.

**on\_enter:** () Event fired when the screen is displayed: the entering animation is complete.

**on\_pre\_leave:** () Event fired when the screen is about to be removed: the leaving animation is started.

**on\_leave:** () Event fired when the screen is removed: the leaving animation is finished.

Changed in version 1.6.0: Events *on\_pre\_enter*, *on\_enter*, *on\_pre\_leave* and *on\_leave* were added.

### 2.3.35 DataTables

**See also:**

Material Design spec, DataTables

**Data tables display sets of data across rows and columns.**

<input type="checkbox"/>	Online	Astrid: NE shared mail
<input checked="" type="checkbox"/>	Offline	Cosmo: prod shared account
<input checked="" type="checkbox"/>	Online	Phoenix: prod shared I
<input type="checkbox"/>	Online	Sirius: prod shared acc

**Warning:** Data tables are still far from perfect. Errors are possible and we hope you inform us about them.

**API - kivymd.uix.datatables**

```
class kivymd.uix.datatables.MDDDataTable(**kwargs)
```

**Events**

**on\_row\_press** Called when a table row is clicked.

**on\_check\_press** Called when the check box in the table row is checked.

**Use events as follows**

```
from kivy.metrics import dp

from kivymd.app import MDApp
from kivymd.uix.datatables import MDDDataTable

class Example(MDApp):
    def build(self):
        self.data_tables = MDDDataTable(
            size_hint=(0.9, 0.6),
            use_pagination=True,
            check=True,
            column_data=[
                ("No.", dp(30)),
                ("Column 1", dp(30)),
                ("Column 2", dp(30)),
                ("Column 3", dp(30)),
                ("Column 4", dp(30)),
                ("Column 5", dp(30)),
            ],
            row_data=[
                (f"{i + 1}", "2.23", "3.65", "44.1", "0.45", "62.5")
                for i in range(50)
            ],
        )
        self.data_tables.bind(on_row_press=self.on_row_press)
        self.data_tables.bind(on_check_press=self.on_check_press)

    def on_start(self):
        self.data_tables.open()

    def on_row_press(self, instance_table, instance_row):
        '''Called when a table row is clicked.'''
        print(instance_table, instance_row)

    def on_check_press(self, instance_table, current_row):
        '''Called when the check box in the table row is checked.'''
        print(instance_table, current_row)

Example().run()
```

**column\_data**

Data for header columns.

```

from kivy.metrics import dp

from kivymd.app import MDApp
from kivymd.uix.datatables import MDDDataTable


class Example(MDApp):
    def build(self):
        self.data_tables = MDDDataTable(
            size_hint=(0.9, 0.6),
            # name column, width column
            column_data=[
                ("Column 1", dp(30)),
                ("Column 2", dp(30)),
                ("Column 3", dp(30)),
                ("Column 4", dp(30)),
                ("Column 5", dp(30)),
                ("Column 6", dp(30)),
            ],
        )

    def on_start(self):
        self.data_tables.open()

Example().run()

```

Column 1	Column 2	Column 3	Column 4	Column 5	Column 6

`column_data` is an `ListProperty` and defaults to `[]`.

#### row\_data

Data for rows.

```

from kivy.metrics import dp

from kivymd.app import MDApp
from kivymd.uix.datatables import MDDDataTable


class Example(MDApp):

```

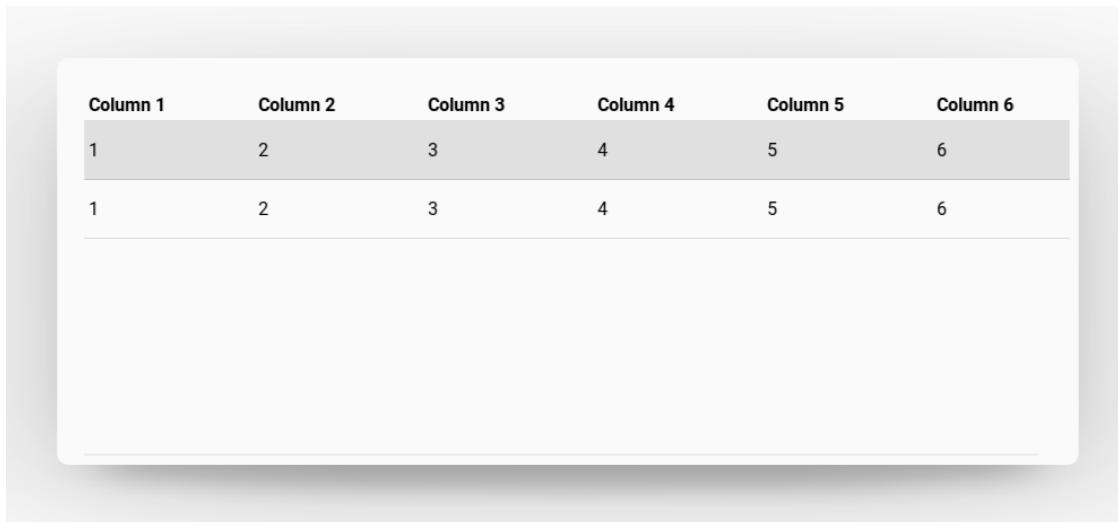
(continues on next page)

(continued from previous page)

```
def build(self):
    self.data_tables = MDDataTable(
        size_hint=(0.9, 0.6),
        column_data=[
            ("Column 1", dp(30)),
            ("Column 2", dp(30)),
            ("Column 3", dp(30)),
            ("Column 4", dp(30)),
            ("Column 5", dp(30)),
            ("Column 6", dp(30)),
        ],
        row_data=[
            # The number of elements must match the length
            # of the `column_data` list.
            ("1", "2", "3", "4", "5", "6"),
            ("1", "2", "3", "4", "5", "6"),
        ],
    )

    def on_start(self):
        self.data_tables.open()
```

Example().run()



The screenshot shows a KivyMD MDDataTable component. It has a header row with columns labeled 'Column 1' through 'Column 6'. Below the header, there are two data rows, each containing the same values: '1', '2', '3', '4', '5', and '6' respectively. The table is contained within a light gray rounded rectangular frame.

Column 1	Column 2	Column 3	Column 4	Column 5	Column 6
1	2	3	4	5	6
1	2	3	4	5	6

`row_data` is an `ListProperty` and defaults to `[]`.

**sort**

Whether to display buttons for sorting table items.

`sort` is an `BooleanProperty` and defaults to `False`.

**check**

Use or not use checkboxes for rows.

`check` is an `BooleanProperty` and defaults to `False`.

**use\_pagination**

Use page pagination for table or not.

```

from kivymd.app import MDApp
from kivymd.uix.datatables import MDDDataTable

class Example(MDApp):
    def build(self):
        self.data_tables = MDDDataTable(
            size_hint=(0.9, 0.6),
            use_pagination=True,
            column_data=[
                ("No.", dp(30)),
                ("Column 1", dp(30)),
                ("Column 2", dp(30)),
                ("Column 3", dp(30)),
                ("Column 4", dp(30)),
                ("Column 5", dp(30)),
            ],
            row_data=[
                (f"{i + 1}", "1", "2", "3", "4", "5") for i in range(50)
            ],
        )

    def on_start(self):
        self.data_tables.open()

Example().run()

```

`use_pagination` is an `BooleanProperty` and defaults to `False`.

#### `rows_num`

The number of rows displayed on one page of the table.

`rows_num` is an `NumericProperty` and defaults to `10`.

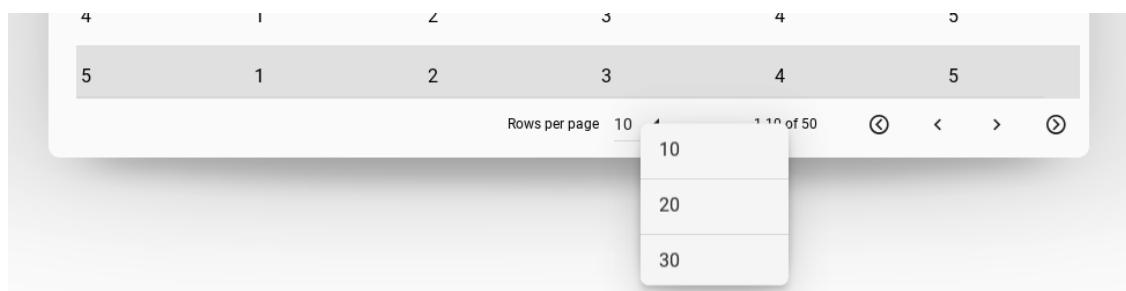
#### `pagination_menu_pos`

Menu position for selecting the number of displayed rows. Available options are ‘center’, ‘auto’.

## Center



## Auto

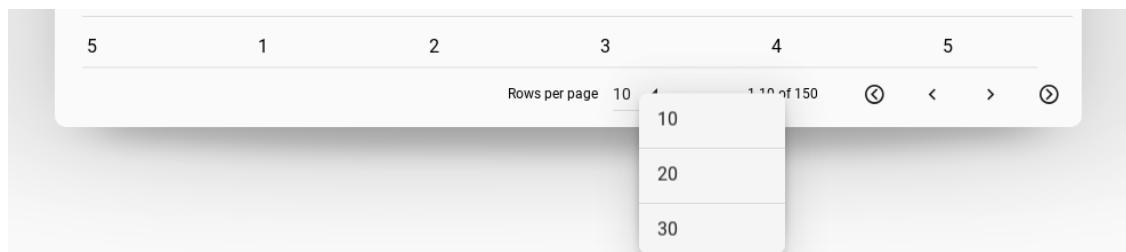


`pagination_menu_pos` is an OptionProperty and defaults to ‘center’.

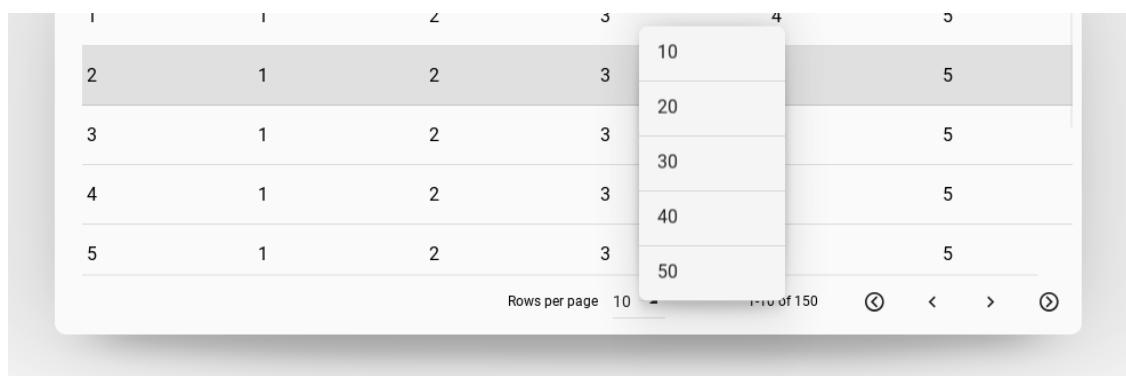
### `pagination_menu_height`

Menu height for selecting the number of displayed rows.

**140dp**



**240dp**



*pagination\_menu\_height* is an `NumericProperty` and defaults to ‘140dp’.

#### `background_color`

Background color in the format (r, g, b, a). See `background_color`.

`background_color` is a `ListProperty` and defaults to [0, 0, 0, .7].

#### `on_row_press (self, *args)`

Called when a table row is clicked.

#### `on_check_press (self, *args)`

Called when the check box in the table row is checked.

#### `create_pagination_menu (self, interval)`

### 2.3.36 TapTargetView

See also:

[TapTargetView, GitHub](#)

[TapTargetView, Material archive](#)

**Provide value and improve engagement by introducing users to new features and functionality at relevant moments.**

### Usage

```
from kivy.lang import Builder

from kivymd.app import MDApp
from kivymd.uix.taptargetview import MDTapTargetView

KV = '''
Screen:

    MDFloatingActionButton:
        id: button
        icon: "plus"
        pos: 10, 10
        on_release: app.tap_target_start()
'''


class TapTargetViewDemo(MDApp):
    def build(self):
        screen = Builder.load_string(KV)
        self.tap_target_view = MDTapTargetView(
            widget=screen.ids.button,
            title_text="This is an add button",
            description_text="This is a description of the button",
            widget_position="left_bottom",
        )
```

(continues on next page)

(continued from previous page)

```
return screen

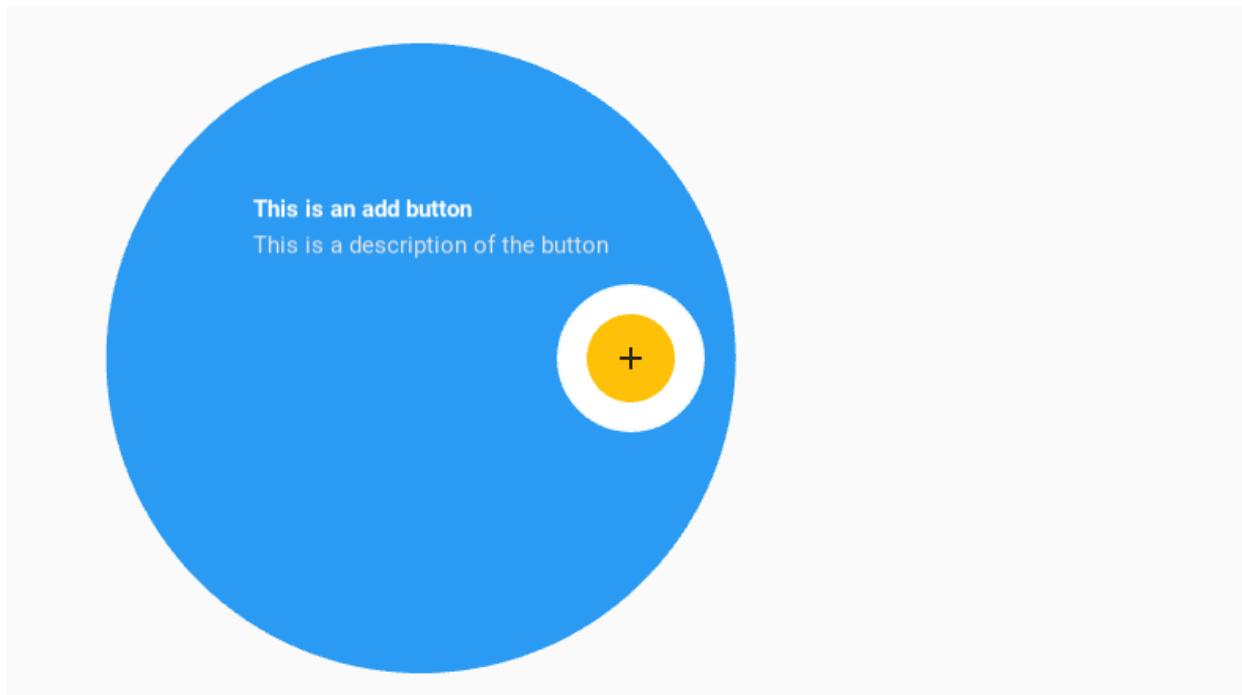
def tap_target_start(self):
    if self.tap_target_view.state == "close":
        self.tap_target_view.start()
    else:
        self.tap_target_view.stop()

TapTargetViewDemo().run()
```

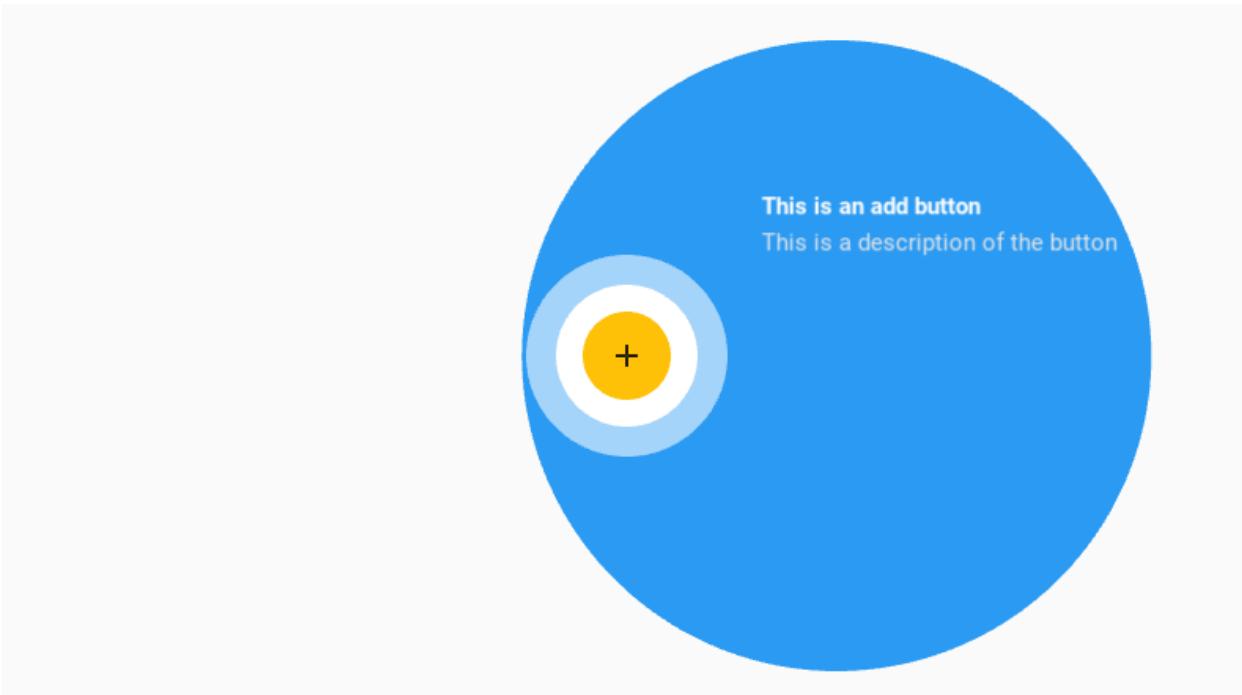
## Widget position

Sets the position of the widget relative to the floating circle.

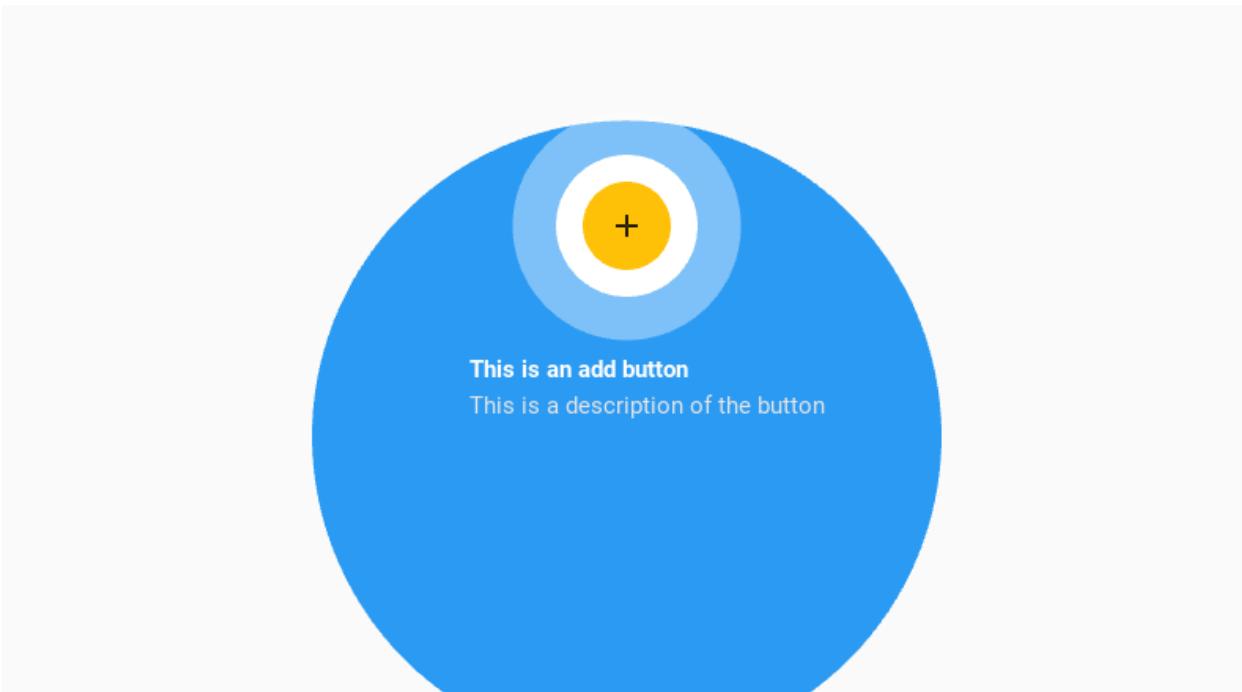
```
self.tap_target_view = MDTapTargetView(
    ...
    widget_position="right",
)
```



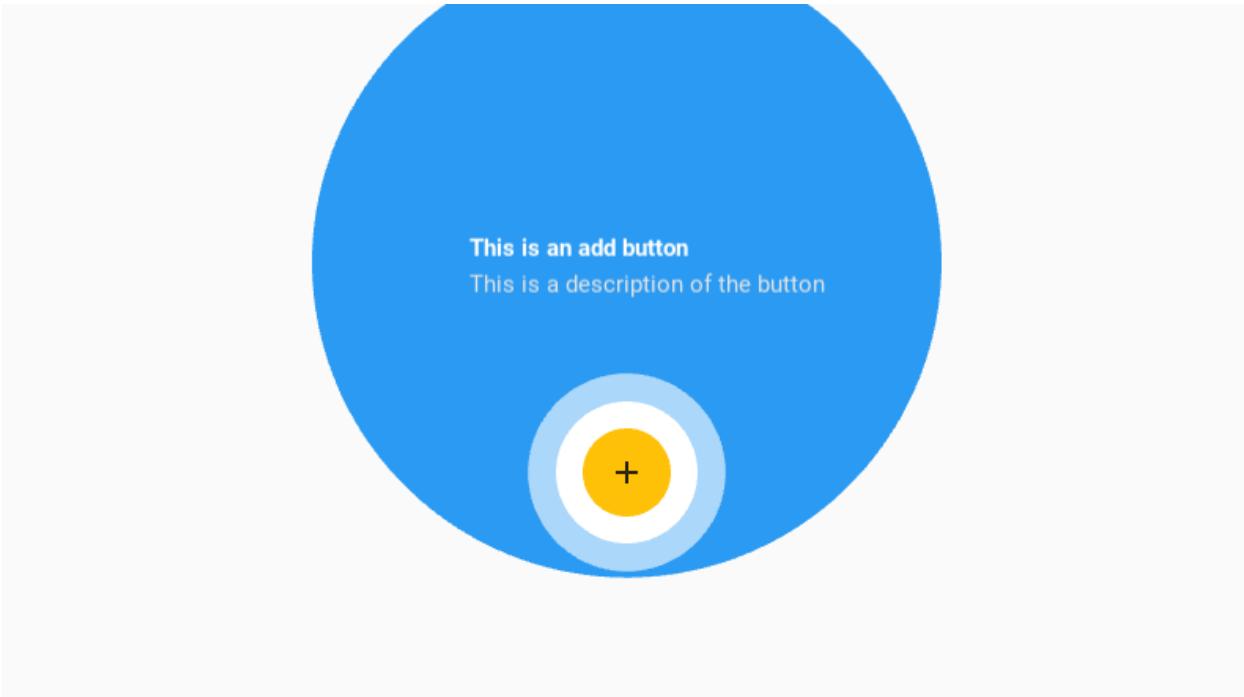
```
self.tap_target_view = MDTapTargetView(
    ...
    widget_position="left",
)
```



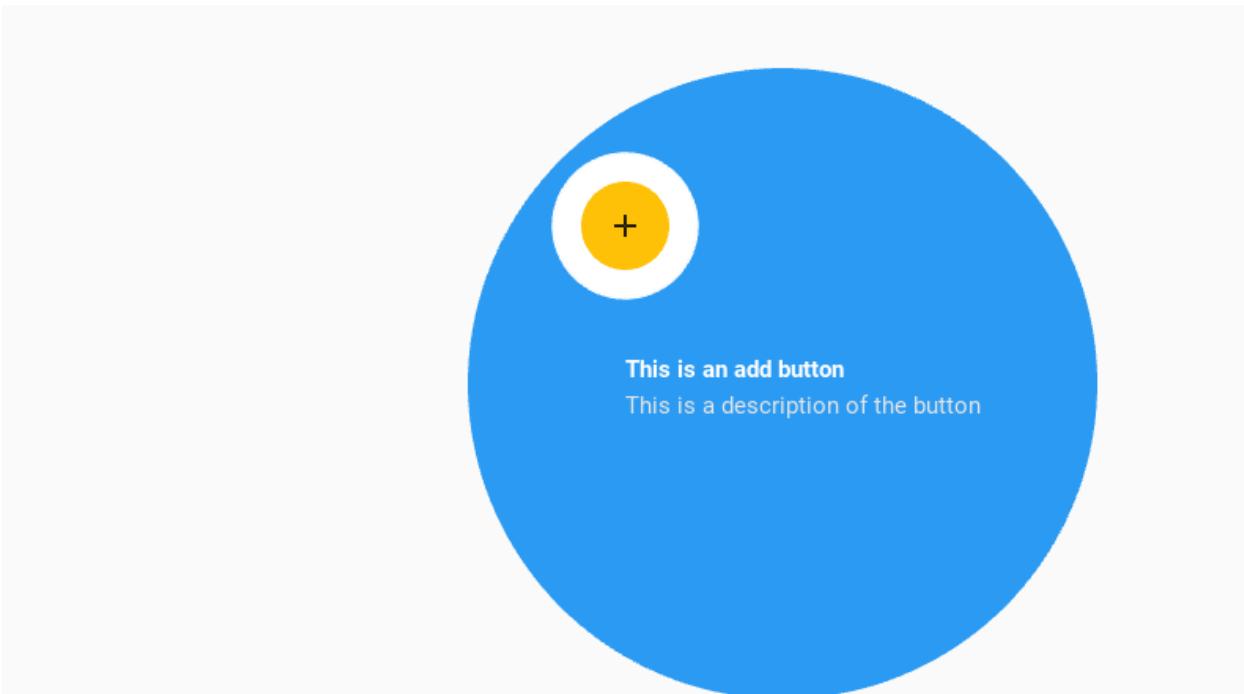
```
self.tap_target_view = MDTapTargetView(  
    ...  
    widget_position="top",  
)
```



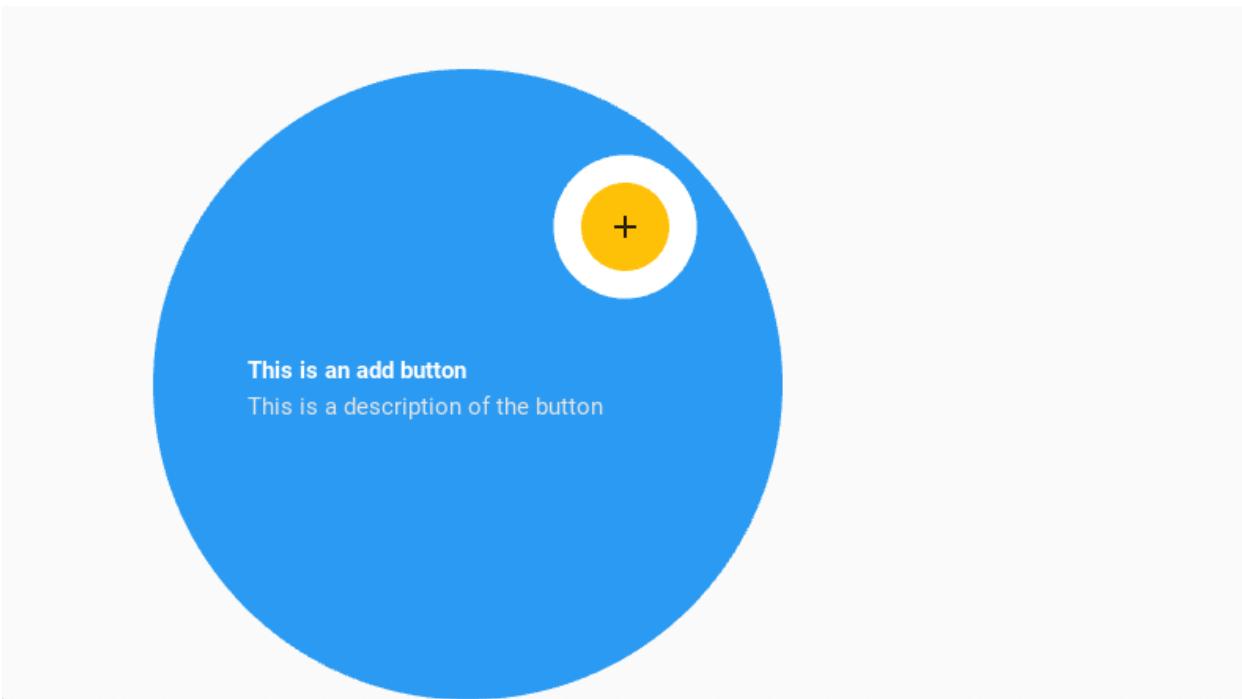
```
self.tap_target_view = MDTapTargetView(  
    ...  
    widget_position="bottom",  
)
```



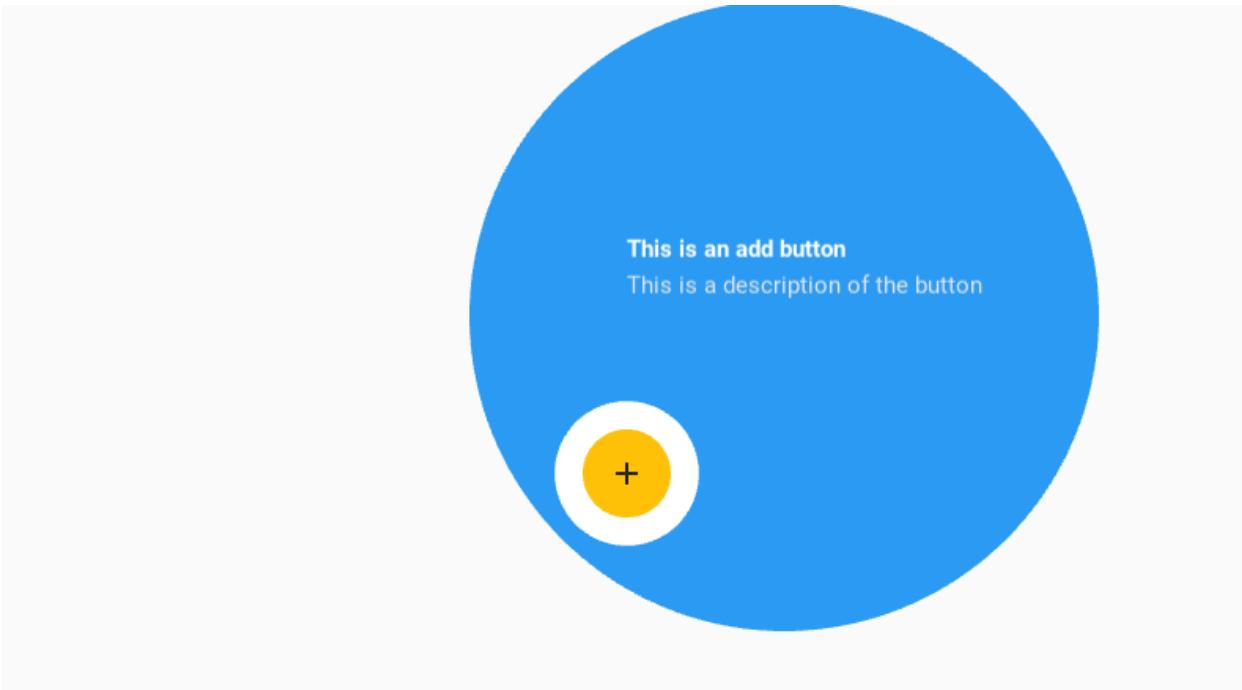
```
self.tap_target_view = MDTapTargetView(  
    ...  
    widget_position="left_top",  
)
```



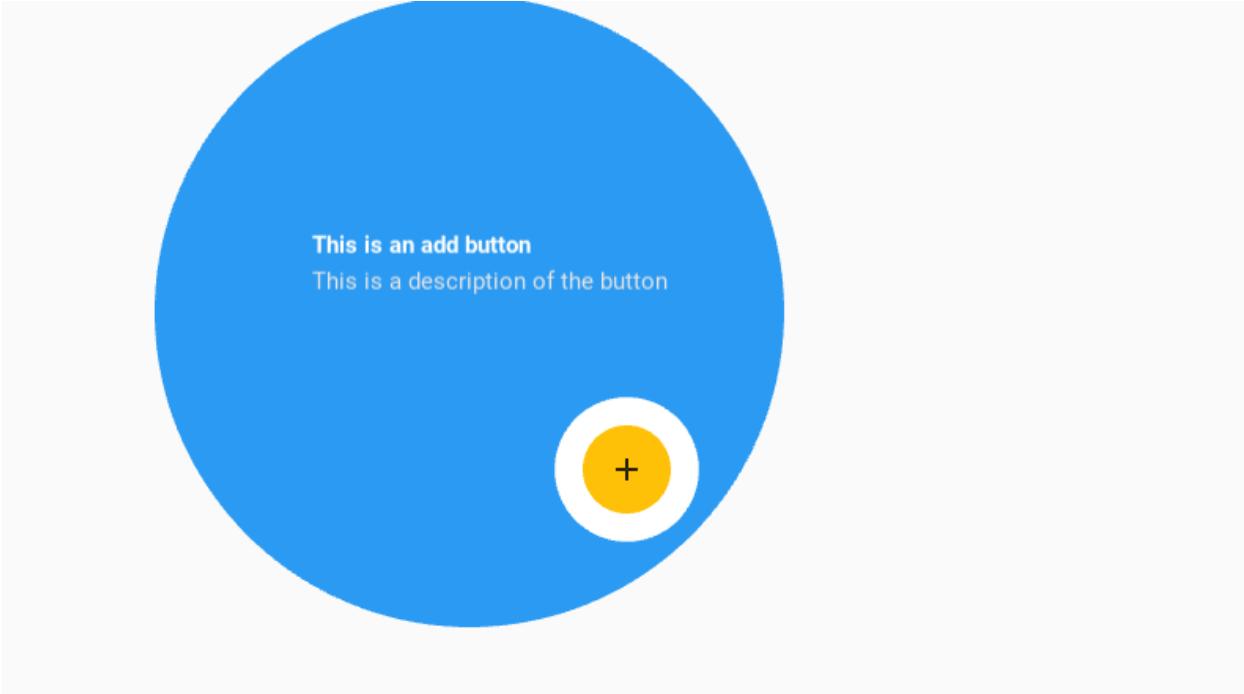
```
self.tap_target_view = MDTapTargetView(  
    ...  
    widget_position="right_top",  
)
```



```
self.tap_target_view = MDTapTargetView(  
    ...  
    widget_position="left_bottom",  
)
```

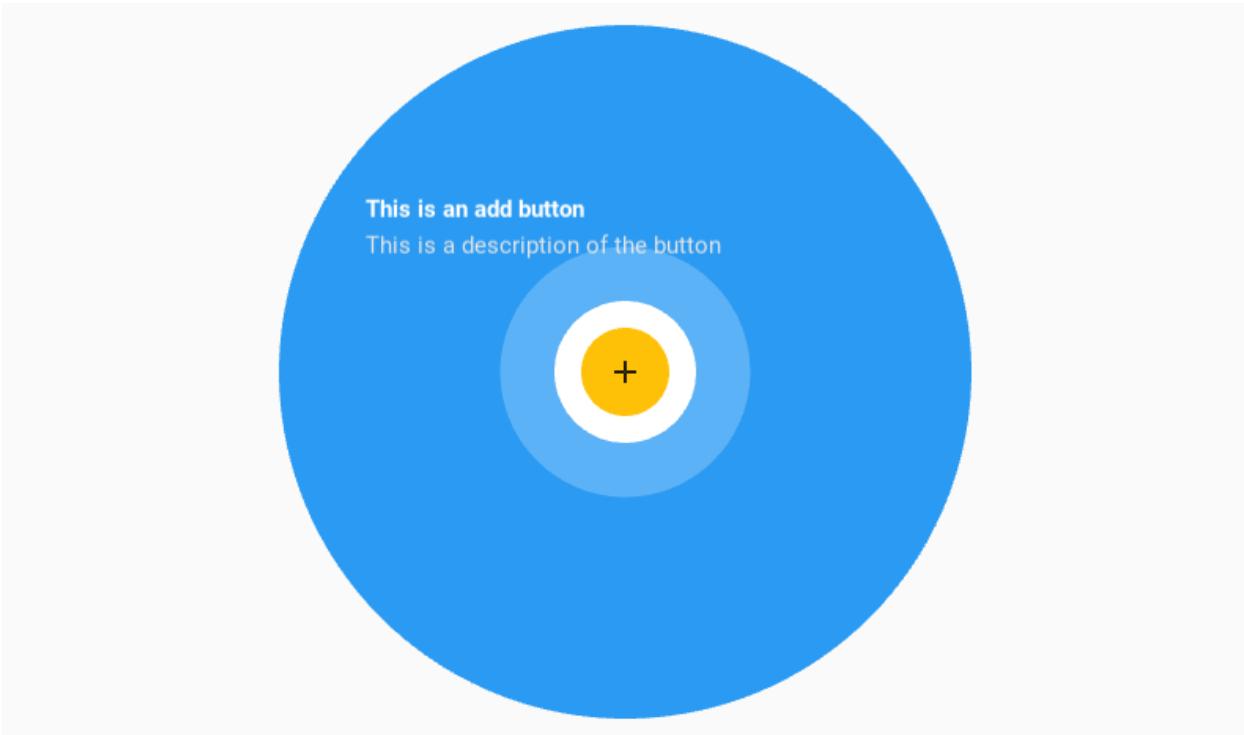


```
self.tap_target_view = MDTapTargetView(  
    ...  
    widget_position="right_bottom",  
)
```



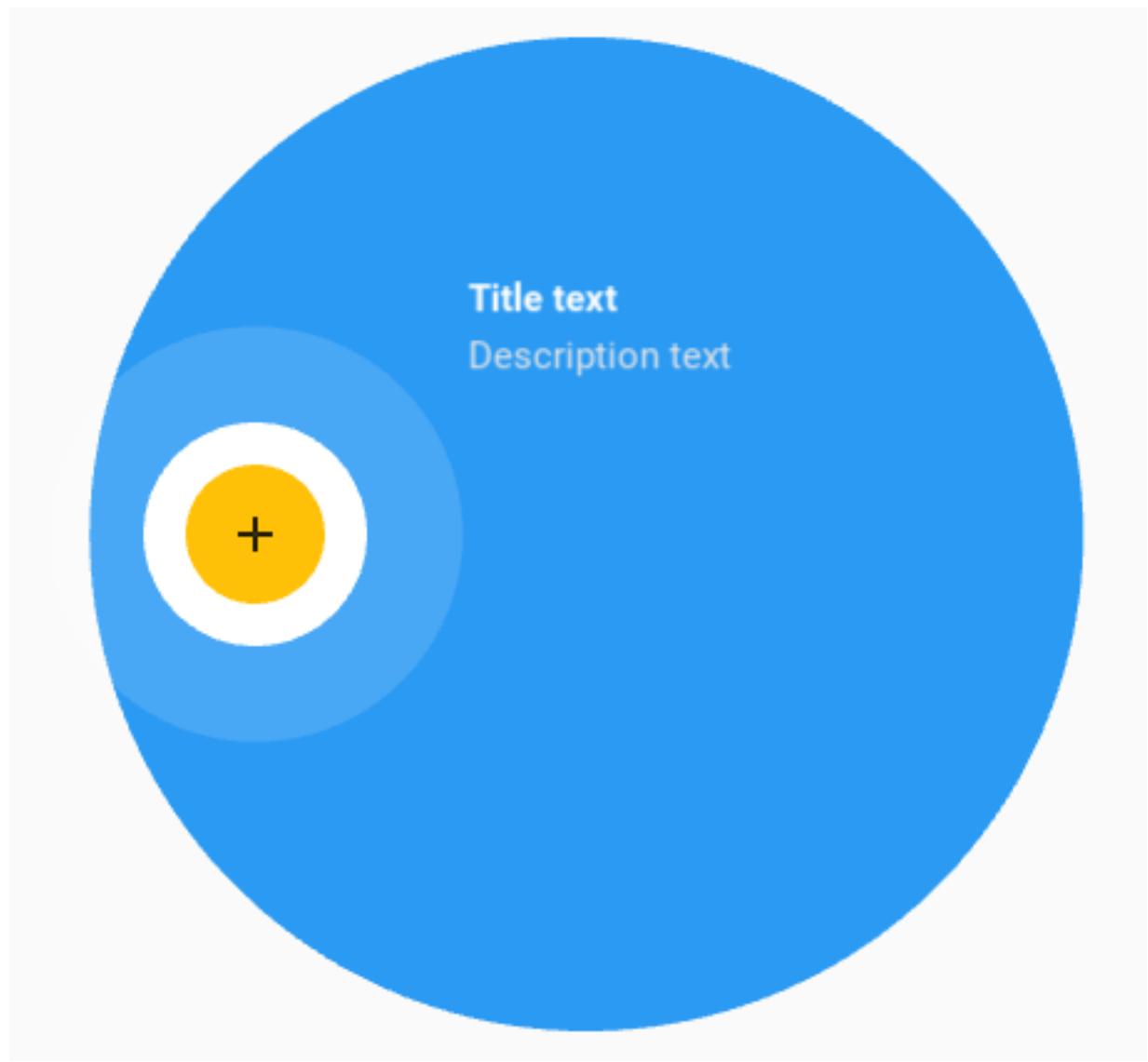
If you use the `widget_position = "center"` parameter then you must definitely specify the `title_position`.

```
self.tap_target_view = MDTapTargetView(  
    ...  
    widget_position="center",  
    title_position="left_top",  
)
```



## Text options

```
self.tap_target_view = MDTapTargetView(  
    ...  
    title_text="Title text",  
    description_text="Description text",  
)
```



You can use the following options to control font size, color, and boldness:

- `title_text_size`
- `title_text_color`
- `title_text_bold`
- `description_text_size`
- `description_text_color`
- `description_text_bold`

```
self.tap_target_view = MDTapTargetView(  
    ...  
    title_text="Title text",  
    title_text_size="36sp",  
    description_text="Description text",  
    description_text_color=[1, 0, 0, 1]  
)
```



But you can also use markup to set these values.

```
self.tap_target_view = MDTapTargetView(  
    ...  
    title_text=" [size=36]Title text[/size]",  
    description_text=" [color=#ff0000ff]Description text[/color]",  
)
```

## Events control

```
self.tap_target_view.bind(on_open=self.on_open, on_close=self.on_close)
```

```
def on_open(self, instance_tap_target_view):
    '''Called at the time of the start of the widget opening animation.'''
    print("Open", instance_tap_target_view)

def on_close(self, instance_tap_target_view):
    '''Called at the time of the start of the widget closed animation.'''
    print("Close", instance_tap_target_view)
```

**Note:** See other parameters in the `MDTapTargetView` class.

## API - kivymd.uix.taptargetview

**class** `kivymd.uix.taptargetview.MDTapTargetView(**kwargs)`  
Rough try to mimic the working of Android's TapTargetView.

### Events

**on\_open** Called at the time of the start of the widget opening animation.

**on\_close** Called at the time of the start of the widget closed animation.

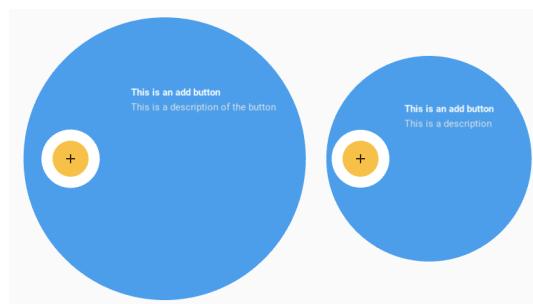
### widget

Widget to add TapTargetView upon.

`widget` is an `ObjectProperty` and defaults to `None`.

### outer\_radius

Radius for outer circle.



`outer_radius` is an `NumericProperty` and defaults to `dp(200)`.

### outer\_circle\_color

Color for the outer circle in `rgb` format.

```
self.tap_target_view = MDTapTargetView(
    ...
    outer_circle_color=(1, 0, 0)
)
```



`outer_circle_color` is an `ListProperty` and defaults to `theme_cls.primary_color`.

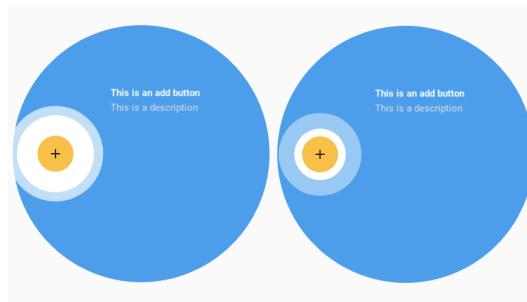
**outer\_circle\_alpha**

Alpha value for outer circle.

`outer_circle_alpha` is an `NumericProperty` and defaults to `0.96`.

**target\_radius**

Radius for target circle.

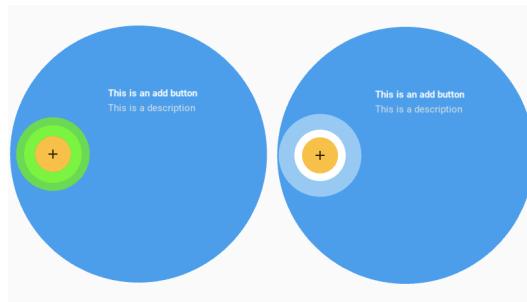


`target_radius` is an `NumericProperty` and defaults to `dp(45)`.

#### `target_circle_color`

Color for target circle in `rgb` format.

```
self.tap_target_view = MDTapTargetView(
    ...
    target_circle_color=(1, 0, 0)
)
```



`target_circle_color` is an `ListProperty` and defaults to `[1, 1, 1]`.

#### `title_text`

Title to be shown on the view.

`title_text` is an `StringProperty` and defaults to ''.

#### `title_text_size`

Text size for title.

`title_text_size` is an `NumericProperty` and defaults to `dp(25)`.

#### `title_text_color`

Text color for title.

`title_text_color` is an `ListProperty` and defaults to `[1, 1, 1, 1]`.

#### `title_text_bold`

Whether title should be bold.

`title_text_bold` is an `BooleanProperty` and defaults to `True`.

#### `description_text`

Description to be shown below the title (keep it short).

`description_text` is an `StringProperty` and defaults to ''.

#### `description_text_size`

Text size for description text.

`description_text_size` is an `NumericProperty` and defaults to `dp(20)`.

**description\_text\_color**

Text size for description text.

*description\_text\_color* is an `ListProperty` and defaults to `[0.9, 0.9, 0.9, 1]`.

**description\_text\_bold**

Whether description should be bold.

*description\_text\_bold* is an `BooleanProperty` and defaults to `False`.

**draw\_shadow**

Whether to show shadow.

*draw\_shadow* is an `BooleanProperty` and defaults to `False`.

**cancelable**

Whether clicking outside the outer circle dismisses the view.

*cancelable* is an `BooleanProperty` and defaults to `False`.

**widget\_position**

Sets the position of the widget on the `outer_circle`. Available options are `'left'`, `'right'`, `'top'`, `'bottom'`, `'left_top'`, `'right_top'`, `'left_bottom'`, `'right_bottom'`, `'center'`.

*widget\_position* is an `OptionProperty` and defaults to `'left'`.

**title\_position**

Sets the position of `:attr`~title_text`` on the outer circle. Only works if `:attr`~widget_position`` is set to `'center'`. In all other cases, it calculates the `:attr`~title_position`` itself. Must be set to other than `'auto'` when `:attr`~widget_position`` is set to `'center'`.

Available options are `'auto'`, `'left'`, `'right'`, `'top'`, `'bottom'`, `'left_top'`, `'right_top'`, `'left_bottom'`, `'right_bottom'`, `'center'`.

*title\_position* is an `OptionProperty` and defaults to `'auto'`.

**stop\_on\_outer\_touch**

Whether clicking on outer circle stops the animation.

*stop\_on\_outer\_touch* is an `BooleanProperty` and defaults to `False`.

**stop\_on\_target\_touch**

Whether clicking on target circle should stop the animation.

*stop\_on\_target\_touch* is an `BooleanProperty` and defaults to `True`.

**state**

State of `MDTapTargetView`.

*state* is an `OptionProperty` and defaults to `'close'`.

**stop (self, \*args)**

Starts widget close animation.

**start (self, \*args)**

Starts widget opening animation.

**on\_open (self, \*args)**

Called at the time of the start of the widget opening animation.

**on\_close (self, \*args)**

Called at the time of the start of the widget closed animation.

**on\_draw\_shadow (self, instance, value)****on\_description\_text (self, instance, value)**

```
on_description_text_size(self, instance, value)
on_description_text_bold(self, instance, value)
on_title_text(self, instance, value)
on_title_text_size(self, instance, value)
on_title_text_bold(self, instance, value)
on_outer_radius(self, instance, value)
on_target_radius(self, instance, value)
on_target_touch(self)
on_outer_touch(self)
on_outside_click(self)
```

## 2.4 Behaviors

### 2.4.1 Touch

Provides easy access to events.

The following events are available:

- on\_long\_touch
- on\_double\_tap
- on\_triple\_tap

#### Usage

```
from kivy.lang import Builder

from kivymd.app import MDApp
from kivymd.uix.behaviors import TouchBehavior
from kivymd.uix.button import MDRaisedButton

KV = '''
Screen:

    MyButton:
        text: "PRESS ME"
        pos_hint: {"center_x": .5, "center_y": .5}
'''


class MyButton(MDRaisedButton, TouchBehavior):
    def on_long_touch(self, *args):
        print("<on_long_touch> event")

    def on_double_tap(self, *args):
        print("<on_double_tap> event")
```

(continues on next page)

(continued from previous page)

```
def on_triple_tap(self, *args):
    print("<on_triple_tap> event")

class MainApp(MDApp):
    def build(self):
        return Builder.load_string(KV)

MainApp().run()
```

#### API - kivymd.uix.behaviors.touch\_behavior

```
class kivymd.uix.behaviors.touch_behavior.TouchBehavior(**kwargs)

duration_long_touch
    Time for a long touch.

    duration_long_touch is an NumericProperty and defaults to 0.4.

create_clock(self, widget, touch, *args)
delete_clock(self, widget, touch, *args)
on_long_touch(self, touch, *args)
    Called when the widget is pressed for a long time.

on_double_tap(self, touch, *args)
    Called by double clicking on the widget.

on_triple_tap(self, touch, *args)
    Called by triple clicking on the widget.
```

## 2.4.2 Hover

### Changing when the mouse is on the widget.

To apply hover behavior, you must create a new class that is inherited from the widget to which you apply the behavior and from the *HoverBehavior* class.

In *KV file*:

```
<HoverItem@MDBoxLayout+ThemableBehavior+HoverBehavior>
```

In *python file*:

```
class HoverItem(MDBoxLayout, ThemableBehavior, HoverBehavior):
    '''Custom item implementing hover behavior.'''

```

After creating a class, you must define two methods for it: *HoverBehavior.on\_enter* and *HoverBehavior.on\_leave*, which will be automatically called when the mouse cursor is over the widget and when the mouse cursor goes beyond the widget.

```

from kivy.lang import Builder

from kivymd.app import MDApp
from kivymd.uix.behaviors import HoverBehavior
from kivymd.uix.boxlayout import MDBBoxLayout
from kivymd.theming import ThemableBehavior

KV = '''
Screen

    MDBBoxLayout:
        id: box
        pos_hint: {'center_x': .5, 'center_y': .5}
        size_hint: .8, .8
        md_bg_color: app.theme_cls.bg_darkest
'''

class HoverItem(MDBBoxLayout, ThemableBehavior, HoverBehavior):
    '''Custom item implementing hover behavior.'''

    def on_enter(self, *args):
        '''The method will be called when the mouse cursor
        is within the borders of the current widget.'''
        self.md_bg_color = (1, 1, 1, 1)

    def on_leave(self, *args):
        '''The method will be called when the mouse cursor goes beyond
        the borders of the current widget.'''
        self.md_bg_color = self.theme_cls.bg_darkest

class Test(MDApp):
    def build(self):
        self.screen = Builder.load_string(KV)
        for i in range(5):
            self.screen.ids.box.add_widget(HoverItem())
        return self.screen

Test().run()

```

## API - kivymd.uix.behaviors.hover\_behavior

**class** kivymd.uix.behaviors.hover\_behavior.**HoverBehavior**(\*\*kwargs)

### Events

**on\_enter** Call when mouse enter the bbox of the widget.

**on\_leave** Call when the mouse exit the widget.

**hovered**

*True*, if the mouse cursor is within the borders of the widget.

*hovered* is an `BooleanProperty` and defaults to *False*.

**border\_point**

Contains the last relevant point received by the Hoverable. This can be used in `on_enter` or `on_leave` in order to know where was dispatched the event.

*border\_point* is an `ObjectProperty` and defaults to *None*.

**on\_mouse\_pos** (*self*, \**args*)

**on\_enter** (*self*)

Call when mouse enter the bbox of the widget.

**on\_leave** (*self*)

Call when the mouse exit the widget.

## 2.4.3 Focus

### Changing the background color when the mouse is on the widget.

To apply focus behavior, you must create a new class that is inherited from the widget to which you apply the behavior and from the `FocusBehavior` class.

### Usage

```
from kivy.lang import Builder

from kivymd.app import MDApp
from kivymd.uix.behaviors import RectangularElevationBehavior, FocusBehavior
from kivymd.uix.boxlayout import MDBBoxLayout

KV = '''
MDScreen:
    md_bg_color: 1, 1, 1, 1

    FocusWidget:
        size_hint: .5, .3
        pos_hint: {"center_x": .5, "center_y": .5}
        md_bg_color: app.theme_cls.bg_light

        MDLabel:
            text: "Label"
            theme_text_color: "Primary"
            pos_hint: {"center_y": .5}
            halign: "center"
    ...

class FocusWidget(MDBoxLayout, RectangularElevationBehavior, FocusBehavior):
    pass

class Test(MDApp):
    pass
```

(continues on next page)

(continued from previous page)

```
def build(self):
    self.theme_cls.theme_style = "Dark"
    return Builder.load_string(KV)

Test().run()
```

Color change at focus/defocus

```
FocusWidget:
    focus_color: 1, 0, 1, 1
    unfocus_color: 0, 0, 1, 1
```

## API - kivymd.uix.behaviors.focus\_behavior

```
class kivymd.uix.behaviors.focus_behavior.FocusBehavior(**kwargs)
```

### Events

**on\_enter** Call when mouse enter the bbox of the widget.

**on\_leave** Call when the mouse exit the widget.

### focus\_behavior

Using focus when hovering over a widget.

*focus\_behavior* is a `BooleanProperty` and defaults to `False`.

### focus\_color

The color of the widget when the mouse enters the bbox of the widget.

*focus\_color* is a `ListProperty` and defaults to `[]`.

### unfocus\_color

The color of the widget when the mouse exits the bbox widget.

*unfocus\_color* is a `ListProperty` and defaults to `[]`.

### on\_enter(*self*)

Called when mouse enter the bbox of the widget.

### on\_leave(*self*)

Called when the mouse exit the widget.

## 2.4.4 Ripple

Classes implements a circular and rectangular ripple effects.

To create a widget with circular ripple effect, you must create a new class that inherits from the `CircularRippleBehavior` class.

For example, let's create an image button with a circular ripple effect:

```
from kivy.lang import Builder
from kivy.uix.behaviors import ButtonBehavior
from kivy.uix.image import Image

from kivymd.app import MDApp
from kivymd.uix.behaviors import CircularRippleBehavior

KV = '''
#:import images_path kivymd.images_path

Screen:

    CircularRippleButton:
        source: f"{images_path}/kivymd.png"
        size_hint: None, None
        size: "250dp", "250dp"
        pos_hint: {"center_x": .5, "center_y": .5}
'''


class CircularRippleButton(CircularRippleBehavior, ButtonBehavior, Image):
    def __init__(self, **kwargs):
        self.ripple_scale = 0.85
        super().__init__(**kwargs)

class Example(MDApp):
    def build(self):
        self.theme_cls.theme_style = "Dark"
        return Builder.load_string(KV)

Example().run()
```

To create a widget with rectangular ripple effect, you must create a new class that inherits from the `RectangularRippleBehavior` class:

```
from kivy.lang import Builder
from kivy.uix.behaviors import ButtonBehavior

from kivymd.app import MDApp
from kivymd.uix.behaviors import RectangularRippleBehavior, BackgroundColorBehavior

KV = '''
Screen:
```

(continues on next page)

(continued from previous page)

```

RectangularRippleButton:
    size_hint: None, None
    size: "250dp", "50dp"
    pos_hint: {"center_x": .5, "center_y": .5}
    ...

class RectangularRippleButton(
    RectangularRippleBehavior, ButtonBehavior, BackgroundColorBehavior
):
    md_bg_color = [0, 0, 1, 1]

class Example(MDApp):
    def build(self):
        self.theme_cls.theme_style = "Dark"
        return Builder.load_string(KV)

Example().run()

```

**API - kivymd.uix.behaviors.ripplebehavior****class kivymd.uix.behaviors.ripplebehavior.CommonRipple**

Base class for ripple effect.

**ripple\_rad\_default**

Default value of the ripple effect radius.

*ripple\_rad\_default* is an `NumericProperty` and defaults to *1*.**ripple\_color**Ripple color in `rgba` format.*ripple\_color* is an `ListProperty` and defaults to `[]`.**ripple\_alpha**

Alpha channel values for ripple effect.

*ripple\_alpha* is an `NumericProperty` and defaults to *0.5*.**ripple\_scale**

Ripple effect scale.

*ripple\_scale* is an `NumericProperty` and defaults to *None*.**ripple\_duration\_in\_fast**

Ripple duration when touching to widget.

*ripple\_duration\_in\_fast* is an `NumericProperty` and defaults to *0.3*.**ripple\_duration\_in\_slow**

Ripple duration when long touching to widget.

*ripple\_duration\_in\_slow* is an `NumericProperty` and defaults to *2*.

**ripple\_duration\_out**

The duration of the disappearance of the wave effect.

*ripple\_duration\_out* is an `NumericProperty` and defaults to `0.3`.

**ripple\_func\_in**

Type of animation for ripple in effect.

*ripple\_func\_in* is an `StringProperty` and defaults to '`out_quad`'.

**ripple\_func\_out**

Type of animation for ripple out effect.

*ripple\_func\_in* is an `StringProperty` and defaults to '`ripple_func_out`'.

**abstract lay\_canvas\_instructions (self)****start\_ripple (self)****finish\_ripple (self)****fade\_out (self, \*args)****anim\_complete (self, \*args)****on\_touch\_down (self, touch)****on\_touch\_move (self, touch, \*args)****on\_touch\_up (self, touch)****class kivymd.uix.behaviors.ripplebehavior.RectangularRippleBehavior**

Class implements a rectangular ripple effect.

**ripple\_scale**

See *ripple\_scale*.

*ripple\_scale* is an `NumericProperty` and defaults to `2.75`.

**lay\_canvas\_instructions (self)****class kivymd.uix.behaviors.ripplebehavior.CircularRippleBehavior**

Class implements a circular ripple effect.

**ripple\_scale**

See *ripple\_scale*.

*ripple\_scale* is an `NumericProperty` and defaults to `1`.

**lay\_canvas\_instructions (self)**

## 2.4.5 Magic

### Magical effects for buttons.

**Warning:** Magic effects do not work correctly with *KivyMD* buttons!

To apply magic effects, you must create a new class that is inherited from the widget to which you apply the effect and from the `MagicBehavior` class.

In *KV file*:

```
<MagicButton@MagicBehavior+MDRectangleFlatButton>
```

In *python* file:

```
class MagicButton(MagicBehavior, MDRectangleFlatButton):
    pass
```

**The `MagicBehavior` class provides five effects:**

- `MagicBehavior.wobble`
- `MagicBehavior.grow`
- `MagicBehavior.shake`
- `MagicBehavior.twist`
- `MagicBehavior.shrink`

Example:

```
from kivymd.app import MDApp
from kivy.lang import Builder

KV = '''
#:import MagicBehavior kivymd.uix.behaviors.MagicBehavior

<MagicButton@MagicBehavior+MDRectangleFlatButton>

FloatLayout:

    MagicButton:
        text: "WOBBLE EFFECT"
        on_release: self.wobble()
        pos_hint: {"center_x": .5, "center_y": .3}

    MagicButton:
        text: "GROW EFFECT"
        on_release: self.grow()
        pos_hint: {"center_x": .5, "center_y": .4}

    MagicButton:
        text: "SHAKE EFFECT"
        on_release: self.shake()
        pos_hint: {"center_x": .5, "center_y": .5}

    MagicButton:
        text: "TWIST EFFECT"
        on_release: self.twist()
        pos_hint: {"center_x": .5, "center_y": .6}

    MagicButton:
        text: "SHRINK EFFECT"
        on_release: self.shrink()
        pos_hint: {"center_x": .5, "center_y": .7}
'''
```

(continues on next page)

(continued from previous page)

```
class Example(MDApp):
    def build(self):
        return Builder.load_string(KV)

Example().run()
```

#### API - kivymd.uix.behaviors.magic\_behavior

```
class kivymd.uix.behaviors.magic_behavior.MagicBehavior

grow(self)
    Grow effect animation.

shake(self)
    Shake effect animation.

wobble(self)
    Wobble effect animation.

twist(self)
    Twist effect animation.

shrink(self)
    Shrink effect animation.
```

#### 2.4.6 Background Color

---

**Note:** The following classes are intended for in-house use of the library.

---

#### API - kivymd.uix.behaviors.backgroundcolorbehavior

```
class kivymd.uix.behaviors.backgroundcolorbehavior.BackgroundColorBehavior(**kwargs)
Widget class. See module documentation for more information.
```

##### Events

**on\_touch\_down:** (*touch*, ) Fired when a new touch event occurs. *touch* is the touch object.  
**on\_touch\_move:** (*touch*, ) Fired when an existing touch moves. *touch* is the touch object.  
**on\_touch\_up:** (*touch*, ) Fired when an existing touch disappears. *touch* is the touch object.  
**on\_kv\_post:** (*base\_widget*, ) Fired after all the kv rules associated with the widget and all other widgets that are in any of those rules have had all their kv rules applied. *base\_widget* is the base-most widget whose instantiation triggered the kv rules (i.e. the widget instantiated from Python, e.g. `MyWidget()`).

Changed in version 1.11.0.

**Warning:** Adding a `__del__` method to a class derived from Widget with Python prior to 3.4 will disable automatic garbage collection for instances of that class. This is because the Widget class creates reference cycles, thereby preventing garbage collection.

Changed in version 1.0.9: Everything related to event properties has been moved to the `EventDispatcher`. Event properties can now be used when contructing a simple class without subclassing `Widget`.

Changed in version 1.5.0: The constructor now accepts `on_*` arguments to automatically bind callbacks to properties or events, as in the Kv language.

**r**

The value of red in the rgba palette.

`r` is an `BoundedNumericProperty` and defaults to `1.0`.

**g**

The value of green in the rgba palette.

`g` is an `BoundedNumericProperty` and defaults to `1.0`.

**b**

The value of blue in the rgba palette.

`b` is an `BoundedNumericProperty` and defaults to `1.0`.

**a**

The value of alpha channel in the rgba palette.

`a` is an `BoundedNumericProperty` and defaults to `0.0`.

### radius

Canvas radius.

```
# Top left corner slice.
MDBoxLayout:
    md_bg_color: app.theme_cls.primary_color
    radius: [25, 0, 0, 0]
```

`radius` is an `ListProperty` and defaults to `[0, 0, 0, 0]`.

### md\_bg\_color

The background color of the widget (`Widget`) that will be inherited from the `BackgroundColorBehavior` class.

For example:

```
Widget:
    canvas:
        Color:
            rgba: 0, 1, 1, 1
        Rectangle:
            size: self.size
            pos: self.pos
```

similar to code:

```
<MyWidget@BackgroundColorBehavior>
    md_bg_color: 0, 1, 1, 1
```

`md_bg_color` is an `ReferenceListProperty` and defaults to `r, g, b, a`.

**class** kivymd.uix.behaviors.backgroundcolorbehavior.**SpecificBackgroundColorBehavior** (\*\*kwargs)  
Widget class. See module documentation for more information.

#### Events

**on\_touch\_down:** (*touch*, ) Fired when a new touch event occurs. *touch* is the touch object.

**on\_touch\_move:** (*touch*, ) Fired when an existing touch moves. *touch* is the touch object.

**on\_touch\_up:** (*touch*, ) Fired when an existing touch disappears. *touch* is the touch object.

**on\_kv\_post:** (*base\_widget*, ) Fired after all the kv rules associated with the widget and all other widgets that are in any of those rules have had all their kv rules applied. *base\_widget* is the base-most widget whose instantiation triggered the kv rules (i.e. the widget instantiated from Python, e.g. `MyWidget()`).

Changed in version 1.11.0.

**Warning:** Adding a `__del__` method to a class derived from Widget with Python prior to 3.4 will disable automatic garbage collection for instances of that class. This is because the Widget class creates reference cycles, thereby preventing garbage collection.

Changed in version 1.0.9: Everything related to event properties has been moved to the `EventDispatcher`. Event properties can now be used when contructing a simple class without subclassing `Widget`.

Changed in version 1.5.0: The constructor now accepts `on_*` arguments to automatically bind callbacks to properties or events, as in the Kv language.

#### **background\_palette**

See `kivymd.color_definitions.palette`.

`background_palette` is an `OptionProperty` and defaults to ‘Primary’.

#### **background\_hue**

See `kivymd.color_definitions.hue`.

`background_hue` is an `OptionProperty` and defaults to ‘500’.

#### **specific\_text\_color**

`specific_text_color` is an `ListProperty` and defaults to `[0, 0, 0, 0.87]`.

#### **specific\_secondary\_text\_color**

`specific_secondary_text_color` is an `:class:`~kivy.properties.ListProperty`` and defaults to `[0, 0, 0, 0.87]`.

## 2.4.7 Elevation

Classes implements a circular and rectangular elevation effects.

To create a widget with rectangular or circular elevation effect, you must create a new class that inherits from the `RectangularElevationBehavior` or `CircularElevationBehavior` class.

For example, let’s create an button with a rectangular elevation effect:

```
from kivy.lang import Builder
from kivy.uix.behaviors import ButtonBehavior

from kivymd.app import MDApp
```

(continues on next page)

(continued from previous page)

```

from kivymd.uix.behaviors import (
    RectangularRippleBehavior,
    BackgroundColorBehavior,
    RectangularElevationBehavior,
)

KV = '''
<RectangularElevationButton>:
    size_hint: None, None
    size: "250dp", "50dp"

Screen:

    # With elevation effect
    RectangularElevationButton:
        pos_hint: {"center_x": .5, "center_y": .6}
        elevation: 11

    # Without elevation effect
    RectangularElevationButton:
        pos_hint: {"center_x": .5, "center_y": .4}
'''


class RectangularElevationButton(
    RectangularRippleBehavior,
    RectangularElevationBehavior,
    ButtonBehavior,
    BackgroundColorBehavior,
):
    md_bg_color = [0, 0, 1, 1]

class Example(MDApp):
    def build(self):
        return Builder.load_string(KV)

Example().run()

```

Similarly, create a button with a circular elevation effect:

```

from kivy.lang import Builder
from kivy.uix.image import Image
from kivy.uix.behaviors import ButtonBehavior

from kivymd.app import MDApp
from kivymd.uix.behaviors import (
    CircularRippleBehavior,
    CircularElevationBehavior,
)

KV = '''
#:import images_path kivymd.images_path

```

(continues on next page)

(continued from previous page)

```
<CircularElevationButton>:  
    size_hint: None, None  
    size: "100dp", "100dp"  
    source: f"{images_path}/kivymd.png"  
  
Screen:  
  
    # With elevation effect  
    CircularElevationButton:  
        pos_hint: {"center_x": .5, "center_y": .6}  
        elevation: 5  
  
    # Without elevation effect  
    CircularElevationButton:  
        pos_hint: {"center_x": .5, "center_y": .4}  
        elevation: 0  
    ...  
  
  
class CircularElevationButton(  
    CircularRippleBehavior,  
    CircularElevationBehavior,  
    ButtonBehavior,  
    Image,  
):  
    md_bg_color = [0, 0, 1, 1]  
  
class Example(MDApp):  
    def build(self):  
        return Builder.load_string(KV)  
  
Example().run()
```

---

### API - kivymd.uix.behaviors.elevation

```
class kivymd.uix.behaviors.elevation.CommonElevationBehavior(**kwargs)  
    Common base class for rectangular and circular elevation behavior.  
  
    elevation  
        Elevation value.  
        elevation is an NumericProperty and defaults to 1.  
  
class kivymd.uix.behaviors.elevation.RectangularElevationBehavior(**kwargs)  
    Base class for rectangular elevation behavior. Controls the size and position of the shadow.  
  
class kivymd.uix.behaviors.elevation.CircularElevationBehavior(**kwargs)  
    Base class for circular elevation behavior. Controls the size and position of the shadow.
```

## 2.4.8 ToggleButton

This behavior must always be inherited after the button's Widget class since it works with the inherited properties of the button class.

example:

```
class MyToggleButtonWidget(MDFlatButton, MDToggleButton):
    # [...]
    pass

from kivy.lang import Builder

from kivymd.app import MDApp
from kivymd.uix.behaviors.toggle_behavior import MDToggleButton
from kivymd.uix.button import MDRectangleFlatButton

KV = '''
Screen:

    MDBBoxLayout:
        adaptive_size: True
        pos_hint: {"center_x": .5, "center_y": .5}

        MyToggleButton:
            text: "Show ads"
            group: "x"

        MyToggleButton:
            text: "Do not show ads"
            group: "x"

        MyToggleButton:
            text: "Does not matter"
            group: "x"
    '''

class MyToggleButton(MDRectangleFlatButton, MDToggleButton):
    def __init__(self, **kwargs):
        super().__init__(**kwargs)
        self.background_down = self.theme_cls.primary_light

class Test(MDApp):
    def build(self):
        return Builder.load_string(KV)

Test().run()
```

```
class MyToggleButton(MDFillRoundFlatButton, MDToggleButton):
    def __init__(self, **kwargs):
        self.background_down = MDApp.get_running_app().theme_cls.primary_dark
        super().__init__(**kwargs)
```

You can inherit the `MyToggleButton` class only from the following classes

- `MDRaisedButton`
- `MDFlatButton`
- `MDRectangleFlatButton`
- `MDRectangleFlatIconButton`
- `MDRoundFlatButton`
- `MDRoundFlatIconButton`
- `MDFillRoundFlatButton`
- `MDFillRoundFlatIconButton`

#### API - `kivymd.uix.behaviors.toggle_behavior`

`class kivymd.uix.behaviors.toggle_behavior.MDToggleButton(**kwargs)`

This `mixin` class provides `togglebutton` behavior. Please see the `togglebutton` behaviors module documentation for more information.

New in version 1.8.0.

##### `background_normal`

Color of the button in `rgba` format for the ‘normal’ state.

`background_normal` is a `ListProperty` and is defaults to `[]`.

##### `background_down`

Color of the button in `rgba` format for the ‘down’ state.

`background_down` is a `ListProperty` and is defaults to `[]`.

##### `font_color_normal`

Color of the font’s button in `rgba` format for the ‘normal’ state.

`font_color_normal` is a `ListProperty` and is defaults to `[]`.

##### `font_color_down`

Color of the font’s button in `rgba` format for the ‘down’ state.

`font_color_down` is a `ListProperty` and is defaults to `[1, 1, 1, 1]`.

## 2.5 Change Log

### 2.5.1 Unreleased

See on GitHub: [branch master](#) | [compare 0.104.1/master](#)

```
pip install https://github.com/kivymd/KivyMD/archive/master.zip
```

- Bug fixes and other minor improvements.
- Add `HotReloadViewer` class

- Added features to *Snackbar* class: use padding, set custom button color, elevation
- Add *MDToggleButton* class
- Change to *Material Design Baseline* dark theme spec
- Fix *ReferenceError: weakly-referenced object no longer exists* when start demo application
- Changed the default value for the *theme\_text\_color* parameter in the *BaseButton* class (to the value “*Primary*”)
- Fix setting of the *text\_color\_normal* and *text\_color\_active* parameters - earlier their values did not affect anything
- Fixed the length of the right edge of the border in relation to the hint text when the *MDTextField* is in the *rectangle* mode
- Add *get\_tab\_list* method to *MDTabs* class
- Add hover behavior when using *MDDropdownMenu* class
- Added the feature to use the *FitImage* component to download images from the network
- The *elevation* value for *RectangularElevationBehavior* and *CircularElevationBehavior* classes after pressing was always set to 2 - fixed
- Methods that implement the ripple effect have always been called twice - fixed
- The *SmartTile* class now uses the *FitImage* class to display images instead of the *Image* class
- Removed dependency on *PIL* library
- Add *hint\_bg\_color*, *hint\_text\_color*, *hint\_radius* attributes to *MDSlider* class
- Delete *progressloader.py*
- Delete *context\_menu.py*
- Added the feature to control the properties of menu items during creation in *MDDropdownMenu* class
- Added the feature to change the number of buttons after creating the *MDFloatingActionButtonSpeedDial* object
- Added the feature to set the *font\_name* property for the *MDTabsLabel* class
- Add *MDCarousel* class
- Delete *kivymd/uix/useranimationcard.py*

## 2.5.2 0.104.1

See on GitHub: [tag 0.104.1](#) | [compare 0.104.0/0.104.1](#)

```
pip install kivymd==0.104.1
```

- Bug fixes and other minor improvements.
- Added *MDGridLayout* and *MDBBoxLayout* classes
- Add *TouchBehavior* class
- Add *radius* parameter to *BackgroundColorBehavior* class
- Add *MDScreen* class
- Add *MDFloatLayout* class
- Added a *MDTextField* with *fill* mode

- Added a shadow, increased speed of opening, added the feature to control the position of the *MDDropdownMenu* class
- The *MDDropDownItem* class is now a regular element, such as a button
- Added the ability to use the texture of the icon on the right in any *MDTextField* classes
- Added the feature to use ripple and focus behavior in *MDCard* class
- *MDDialogs* class redesigned to meet material design requirements
- Added *MDDataTable* class

### 2.5.3 0.104.0

See on GitHub: [tag 0.104.0](#) | compare 0.103.0/0.104.0

```
pip install kivymd==0.104.0
```

- Fixed bug in *kivymd.uix.expansionpanel.MDExpansionPanel* if, with the panel open, without closing it, try to open another panel, then the chevron of the first panel remained open.
- The *kivymd.uix.textfield.MDTextFieldRound* class is now directly inherited from the *kivy.uix.textinput.TextInput* class.
- Removed *kivymd.uix.textfield.MDTextFieldClear* class.
- *kivymd.uix.navigationdrawer.NavigationLayout* allowed to add *kivymd.uix.toolbar.MDToolbar* class.
- Added feature to control range of dates to be active in *kivymd.uix.picker.MDDatePicker* class.
- Updated *kivymd.uix.navigationdrawer.MDNavigationDrawer* realization.
- Removed *kivymd.uix.card.MDCardPost* class.
- Added *kivymd.uix.card.MDCardSwipe* class.
- Added *switch\_tab* method for switching tabs to *kivymd.uix.bottomnavigation.MDBottomNavigation* class.
- Added feature to use panel type in the *kivymd.uix.expansionpanel.MDExpansionPanel* class: *kivymd.uix.expansionpanel.MDExpansionPanelOneLine*, *kivymd.uix.expansionpanel.MDExpansionPanelTwoLine* or *kivymd.uix.expansionpanel.MDExpansionPanelThreeLine*.
- Fixed panel opening animation in the *kivymd.uix.expansionpanel.MDExpansionPanel* class.
- Delete *kivymd.uix.managerswiper.py*
- Add *MDFloatingActionButtonSpeedDial* class
- Added the feature to create text on tabs using markup, thereby triggering the *on\_ref\_press* event in the *MDTabLabel* class
- Added *color\_indicator* attribute to set custom indicator color in the *MDTabs* class
- Added the feature to change the background color of menu items in the *BaseListItem* class
- Add *MDTapTargetView* class

## 2.5.4 0.103.0

See on GitHub: [tag 0.103.0](#) | [compare 0.102.1/0.103.0](#)

```
pip install kivymd==0.103.0
```

- Fix *MDSwitch* size according to *material design* guides
- Fix *MDSwitch*'s thumb position when size changes
- Fix position of the icon relative to the right edge of the *MDChip* class on mobile devices
- Updated *MDBottomAppBar* class.
- Updated *navigationdrawer.py*
- Added *on\_tab\_switch* method that is called when switching tabs (*MDTabs* class)
- Added *FpsMonitor* class
- Added *fitimage.py* - feature to automatically crop a *Kivy* image to fit your layout
- Added animation when changing the action button position mode in *MDBottomAppBar* class
- Delete *fanscreenmanager.py*
- Bug fixes and other minor improvements.

## 2.5.5 0.102.1

See on GitHub: [tag 0.102.1](#) | [compare 0.102.0/0.102.1](#)

```
pip install kivymd==0.102.1
```

- Implemented the ability [Backdrop](<https://material.io/components/backdrop>)
- Added *MDApp* class. Now app object should be inherited from *kivymd.app.MDApp*.
- Added *MDRoundImageButton* class.
- Added *MDDoltip* class.
- Added *MDBanner* class.
- Added hook for *PyInstaller* (add *hookspath=[kivymd.hooks\_path]*).
- Added examples of *spec* files for building [Kitchen Sink demo]([https://github.com/kivymd/KivyMD/tree/master/demos/kitchen\\_sink](https://github.com/kivymd/KivyMD/tree/master/demos/kitchen_sink)).
- Added some features to *MDProgressLoader*.
- Added feature to preview the current value of *MDSlider*.
- Added feature to use custom screens for dialog in *MDBottomSheet* class.
- Removed *MDPopupScreen*.
- Added [studies]([https://github.com/kivymd/KivyMD/tree/master/demos/kitchen\\_sink/studies](https://github.com/kivymd/KivyMD/tree/master/demos/kitchen_sink/studies)) directory for demos in Material Design.
- Bug fixes and other minor improvements.

## 2.5.6 0.102.0

See on GitHub: [tag 0.102.0](#) | compare 0.101.8/0.102.0

```
pip install kivymd==0.102.0
```

- Moved *kivymd.behaviors* to *kivymd.uix.behaviors*.
- Updated [Iconic font]([https://github.com/Templarian/MaterialDesign-Webfont]) (v4.5.95).
- Added *blank* icon to *icon\_definitions*.
- Bug fixes and other minor improvements.

## 2.5.7 0.101.8

See on GitHub: [tag 0.101.8](#) | compare 0.101.7/0.101.8

```
pip install https://github.com/kivymd/KivyMD/archive/0.101.8.zip
```

- Added *uix* and *behaviors* folder to *package\_data*.

## 2.5.8 0.101.7

See on GitHub: [tag 0.101.7](#) | compare 0.101.6/0.101.7

```
pip install https://github.com/kivymd/KivyMD/archive/0.101.7.zip
```

- Fixed colors and position of the buttons in the *Buttons* demo screen ([Kitchen Sink demo]([https://github.com/kivymd/KivyMD/tree/master/demos/kitchen\_sink])).
- Displaying percent of loading kv-files ([Kitchen Sink demo]([https://github.com/kivymd/KivyMD/tree/master/demos/kitchen\_sink])).

## 2.5.9 0.101.6

See on GitHub: [tag 0.101.6](#) | compare 0.101.5/0.101.6

```
pip install https://github.com/kivymd/KivyMD/archive/0.101.6.zip
```

- Fixed *NameError: name 'MDThemePicker' is not defined*.

## 2.5.10 0.101.5

See on GitHub: [tag 0.101.5](#) | compare 0.101.4/0.101.5

```
pip install https://github.com/kivymd/KivyMD/archive/0.101.5.zip
```

- Added feature to see source code of current example ([Kitchen Sink demo]([https://github.com/kivymd/KivyMD/tree/master/demos/kitchen\_sink])).
- Added names of authors of this fork ([Kitchen Sink demo]([https://github.com/kivymd/KivyMD/tree/master/demos/kitchen\_sink])).
- Bug fixes and other minor improvements.

## 2.5.11 0.101.4

See on GitHub: [tag 0.101.4](#) | [compare 0.101.3/0.101.4](#)

```
pip install https://github.com/kivymd/KivyMD/archive/0.101.4.zip
```

- Bug fixes and other minor improvements.

## 2.5.12 0.101.3

See on GitHub: [tag 0.101.3](#) | [compare 0.101.2/0.101.3](#)

```
pip install https://github.com/kivymd/KivyMD/archive/0.101.3.zip
```

- Bug fixes and other minor improvements.

## 2.5.13 0.101.2

See on GitHub: [tag 0.101.2](#) | [compare 0.101.1/0.101.2](#)

```
pip install https://github.com/kivymd/KivyMD/archive/0.101.2.zip
```

- Bug fixes and other minor improvements.

## 2.5.14 0.101.1

See on GitHub: [tag 0.101.1](#) | [compare 0.101.0/0.101.1](#)

```
pip install https://github.com/kivymd/KivyMD/archive/0.101.1.zip
```

- Bug fixes and other minor improvements.

## 2.5.15 0.101.0

See on GitHub: [tag 0.101.0](#) | [compare 0.100.2/0.101.0](#)

```
pip install https://github.com/kivymd/KivyMD/archive/0.101.0.zip
```

- Added *MDContextMenu* class.
- Added *MDExpansionPanel* class.
- Removed *MDAccordion* and *MDAccordionListItem*. Use *MDExpansionPanel* instead.
- Added *HoverBehavior* class by [Olivier POYEN](<https://gist.github.com/opqopq/15c707dc4cffc2b6455f>).
- Added markup support for buttons.
- Added *duration* property to *Toast*.
- Added *TextInput*'s events and properties to *MDTextFieldRound*.
- Added feature to resize text field
- Added color property to *MDSeparator* class

- Added [tool]([https://github.com/kivymd/KivyMD/blob/master/kivymd/tools/update\\_icons.py](https://github.com/kivymd/KivyMD/blob/master/kivymd/tools/update_icons.py)) for updating [Iconic font](<https://github.com/Templarian/MaterialDesign-Webfont>).
- Updated [Iconic font](<https://github.com/Templarian/MaterialDesign-Webfont>) (v4.3.95).
- Added new examples for [Kitchen Sink demo]([https://github.com/kivymd/KivyMD/tree/master/demos/kitchen\\_sink](https://github.com/kivymd/KivyMD/tree/master/demos/kitchen_sink)).
- Bug fixes and other minor improvements.

## 2.5.16 0.100.2

See on GitHub: [tag 0.100.2](#) | [compare 0.100.1/0.100.2](#)

```
pip install https://github.com/kivymd/KivyMD/archive/0.100.2.zip
```

- [Black](<https://github.com/psf/black>) formatting.

## 2.5.17 0.100.1

See on GitHub: [tag 0.100.1](#) | [compare 0.100.0/0.100.1](#)

```
pip install https://github.com/kivymd/KivyMD/archive/0.100.1.zip
```

- *MDUserAnimationCard* uses *Image* instead of *AsyncImage*.

## 2.5.18 0.100.0

See on GitHub: [tag 0.100.0](#) | [compare 0.99.99/0.100.0](#)

```
pip install https://github.com/kivymd/KivyMD/archive/0.100.0.zip
```

- Added feature to change color for *MDStackFloatingButtons*.

## 2.5.19 0.99.99.01

See on GitHub: [tag 0.99.99.01](#) | [compare 0.99.98/0.99.99.01](#)

```
pip install https://github.com/kivymd/KivyMD/archive/0.99.99.01.zip
```

- Fixed *MNavigationDrawer:use\_logo*.

## 2.5.20 0.99.99

See on GitHub: [tag 0.99.99](#) | [compare 0.99.99.01/0.99.99](#)

```
pip install https://github.com/kivymd/KivyMD/archive/0.99.99.zip
```

- Added *icon\_color* property for *NavigationDrawerIconButton*.

## 2.5.21 0.99.98

See on GitHub: [tag 0.99.98](#) | [compare 0.99.97/0.99.98](#)

```
pip install https://github.com/kivymd/KivyMD/archive/0.99.98.zip
```

- Added *MDFillRoundFlatButton* class.

## 2.5.22 0.99.97

See on GitHub: [tag 0.99.97](#) | [compare 0.99.96/0.99.97](#)

```
pip install https://github.com/kivymd/KivyMD/archive/0.99.97.zip
```

- Fixed *Spinner* animation.

## 2.5.23 0.99.96

See on GitHub: [tag 0.99.96](#) | [compare 0.99.95/0.99.96](#)

```
pip install https://github.com/kivymd/KivyMD/archive/0.99.96.zip
```

- Added *asynckivy* module by [Nattōsai Mitō](<https://github.com/gottadiveintopython/asynckivy>).

## 2.5.24 0.99.95

See on GitHub: [tag 0.99.95](#) | [compare 0.99.94/0.99.95](#)

```
pip install https://github.com/kivymd/KivyMD/archive/0.99.95.zip
```

- Added function to create a round image in *kivymd/utils/cropimage.py* module.
- Added *MDCustomRoundIconButton* class.
- Added demo application [Account Page](<https://www.youtube.com/watch?v=dfUOwqtYoYg>) for [Kitchen Sink demo]([https://github.com/kivymd/KivyMD/tree/master/demos/kitchen\\_sink](https://github.com/kivymd/KivyMD/tree/master/demos/kitchen_sink)).

## 2.5.25 0.99.94

See on GitHub: [tag 0.99.94](#) | [compare 0.99.93/0.99.94](#)

```
pip install https://github.com/kivymd/KivyMD/archive/0.99.94.zip
```

- Added *\_no\_ripple\_effect* property to *BaseListItem* class.
- Added check to use *ripple effect* in *RectangularRippleBehavior* class.
- [Disabled]([https://www.youtube.com/watch?v=P\\_9oSx0Pz\\_U](https://www.youtube.com/watch?v=P_9oSx0Pz_U)) using *ripple effect* in *MADAccordionListItem* class.

## 2.5.26 0.99.93

See on GitHub: [tag 0.99.93](#) | [compare 0.99.92/0.99.93](#)

```
pip install https://github.com/kivymd/KivyMD/archive/0.99.93.zip
```

- Updated [Iconic font](<https://github.com/Templarian/MaterialDesign-Webfont>) (v3.6.95).

## 2.5.27 0.99.92

See on GitHub: [tag 0.99.92](#) | [compare 0.99.91/0.99.92](#)

```
pip install https://github.com/kivymd/KivyMD/archive/0.99.92.zip
```

- Removed automatic change of text field length in *MTextFieldRound* class.

## 2.6 About

### 2.6.1 License

Refer to [LICENSE](#).

#### MIT License

Copyright (c) 2015 Andrés Rodríguez and other contributors - KivyMD library up to  
↳version 0.1.2  
Copyright (c) 2020 KivyMD Team and other contributors - KivyMD library version 0.1.3  
↳and higher

Other libraries used in the project:

Copyright (c) 2010–2020 Kivy Team and other contributors  
Copyright (c) 2013 Brian Knapp - Androidoast library  
Copyright (c) 2014 LogicalDash - stiffscroll library  
Copyright (c) 2015 Davide Depau - circularTimePicker, circleLayout libraries  
Copyright (c) 2015 Kivy Garden - tabs module  
Copyright (c) 2020 Nattōsai Mitō - asynckivy module  
Copyright (c) 2020 tshirtman - magic\_behavior module  
Copyright (c) 2020 shashi278 - taptargetview module  
Copyright (c) 2020 Benedikt Zwölfer - fitimage module  
Hoverable Behaviour (changing when the mouse is on the widget by O. Poyen, License:  
↳GPL) - hover\_behavior module

Permission is hereby granted, free of charge, to any person obtaining a copy  
of this software and associated documentation files (the "Software"), to deal  
in the Software without restriction, including without limitation the rights  
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell  
copies of the Software, and to permit persons to whom the Software is  
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all  
copies or substantial portions of the Software.

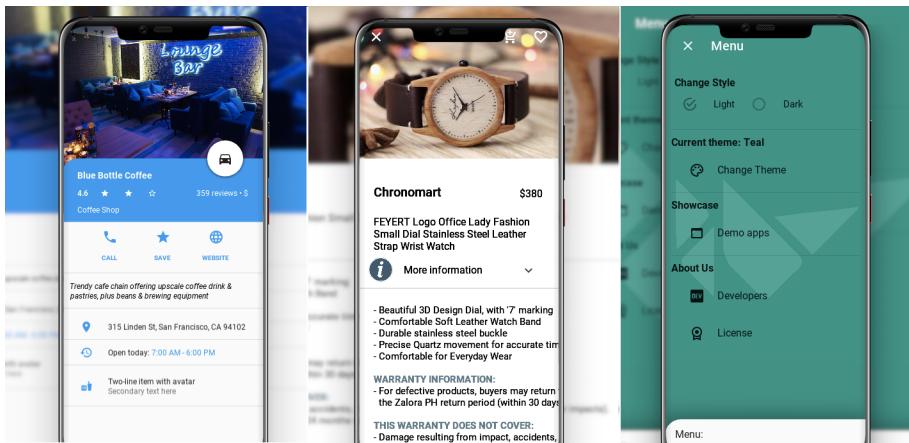
THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR

(continues on next page)

(continued from previous page)

IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

## 2.7 KivyMD



Is a collection of Material Design compliant widgets for use with, Kivy cross-platform graphical framework a framework for cross-platform, touch-enabled graphical applications. The project's goal is to approximate Google's [Material Design spec](#) as close as possible without sacrificing ease of use or application performance.

This library is a fork of the [KivyMD project](#) the author of which stopped supporting this project three years ago. We found the strength and brought this project to a new level. Currently we're in **beta** status, so things are changing all the time and we cannot promise any kind of API stability. However it is safe to vendor now and make use of what's currently available.

Join the project! Just fork the project, branch out and submit a pull request when your patch is ready. If any changes are necessary, we'll guide you through the steps that need to be done via PR comments or access to your for may be requested to outright submit them. If you wish to become a project developer (permission to create branches on the project without forking for easier collaboration), have at least one PR approved and ask for it. If you contribute regularly to the project the role may be offered to you without asking too.

### 2.7.1 API - kivymd

```
kivymd.release = False
kivymd.path
    Path to KivyMD package directory.
kivymd.fonts_path
    Path to fonts directory.
kivymd.images_path
    Path to images directory.
```

## 2.7.2 Submodules

### Register KivyMD widgets to use without import

Register KivyMD widgets to use without import

#### API - `kivymd.factory_registers`

```
kivymd.factory_registers.r
```

### Material Resources

#### API - `kivymd.material_resources`

```
kivymd.material_resources.dp
kivymd.material_resources.DEVICE_IOS
kivymd.material_resources.DEVICE_TYPE = desktop
kivymd.material_resources.MAX_NAV_DRAWER_WIDTH
kivymd.material_resources.TOUCH_TARGET_HEIGHT
```

### Theming Dynamic Text

Two implementations. The first is based on color brightness obtained from- <https://www.w3.org/TR/AERT#color-contrast> The second is based on relative luminance calculation for sRGB obtained from- <https://www.w3.org/TR/2008/REC-WCAG20-20081211/#relativeluminancedef> and contrast ratio calculation obtained from- <https://www.w3.org/TR/2008/REC-WCAG20-20081211/#contrast-ratiodef>

Preliminary testing suggests color brightness more closely matches the *Material Design spec* suggested text colors, but the alternative implementation is both newer and the current ‘correct’ recommendation, so is included here as an option.

#### API - `kivymd.theming_dynamic_text`

```
kivymd.theming_dynamic_text.get_contrast_text_color(color,
                                                    use_color_brightness=True)
kivymd.theming_dynamic_text.color
```

### Stiff Scroll Effect

An Effect to be used with ScrollView to prevent scrolling beyond the bounds, but politely.

A ScrollView constructed with StiffScrollEffect, eg. `ScrollView(effect_cls=StiffScrollEffect)`, will get harder to scroll as you get nearer to its edges. You can scroll all the way to the edge if you want to, but it will take more finger-movement than usual.

Unlike DampedScrollEffect, it is impossible to overscroll with StiffScrollEffect. That means you cannot push the contents of the ScrollView far enough to see what’s beneath them. This is appropriate if the ScrollView contains, eg., a background image, like a desktop wallpaper. Overscrolling may give the impression that there is some reason to overscroll, even if just to take a peek beneath, and that impression may be misleading.

StiffScrollEffect was written by Zachary Spector. His other stuff is at: <https://github.com/LogicalDash/> He can be reached, and possibly hired, at: [zacharyspector@gmail.com](mailto:zacharyspector@gmail.com)

## API - kivymd.stiffscroll

```
class kivymd.stiffscroll.StiffScrollEffect(**kwargs)
```

Kinetic effect class. See module documentation for more information.

### **drag\_threshold**

Minimum distance to travel before the movement is considered as a drag.

*drag\_threshold* is an `NumericProperty` and defaults to '20sp'.

### **min**

Minimum boundary to stop the scrolling at.

*min* is an `NumericProperty` and defaults to 0.

### **max**

Maximum boundary to stop the scrolling at.

*max* is an `NumericProperty` and defaults to 0.

### **max\_friction**

How hard should it be to scroll, at the worst?

*max\_friction* is an `NumericProperty` and defaults to 1.

### **body**

Proportion of the range in which you can scroll unimpeded.

*body* is an `NumericProperty` and defaults to 0.7.

### **scroll**

Computed value for scrolling

*scroll* is an `NumericProperty` and defaults to 0.0.

### **transition\_min**

The AnimationTransition function to use when adjusting the friction near the minimum end of the effect.

*transition\_min* is an `ObjectProperty` and defaults to `kivy.animation.AnimationTransition`.

### **transition\_max**

The AnimationTransition function to use when adjusting the friction near the maximum end of the effect.

*transition\_max* is an `ObjectProperty` and defaults to `kivy.animation.AnimationTransition`.

### **target\_widget**

The widget to apply the effect to.

*target\_widget* is an `ObjectProperty` and defaults to None.

### **displacement**

The absolute distance moved in either direction.

*displacement* is an `NumericProperty` and defaults to 0.

### **update\_velocity(self, dt)**

Before actually updating my velocity, meddle with `self.friction` to make it appropriate to where I'm at, currently.

```
on_value (self, *args)
    Prevent moving beyond my bounds, and update self.scroll

start (self, val, t=None)
    Start movement with self.friction = self.base_friction

update (self, val, t=None)
    Reduce the impact of whatever change has been made to me, in proportion with my current friction.

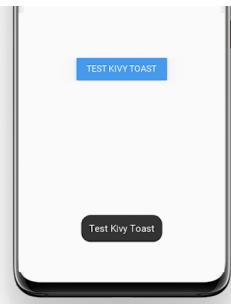
stop (self, val, t=None)
    Work out whether I've been flung.
```

### kivymd.toast

#### API - kivymd.toast

##### Submodules

##### Toast for Android device



#### API - kivymd.toast.androidtoast

##### Submodules

##### AndroidToast

##### Native implementation of toast for Android devices.

```
from kivymd.app import MDApp
# Will be automatically used native implementation of the toast
# if your application is running on an Android device.
# Otherwise, will be used toast implementation
# from the kivymd/toast/kivytoast package.
from kivymd.toast import toast

KV = '''
BoxLayout:
    orientation:'vertical'

    MDToolbar:
```

(continues on next page)

(continued from previous page)

```

id: toolbar
    title: 'Test Toast'
    md_bg_color: app.theme_cls.primary_color
    left_action_items: [['menu', lambda x: '']]

    FloatLayout:

        MDRaisedButton:
            text: 'TEST KIVY TOAST'
            on_release: app.show_toast()
            pos_hint: {'center_x': .5, 'center_y': .5}

    ...

class Test(MDApp):
    def show_toast(self):
        '''Displays a toast on the screen.'''
        toast('Test Kivy Toast')

    def build(self):
        return Builder.load_string(KV)

Test().run()

```

**API - kivymd.toast.androidtoast.androidtoast**

kivymd.toast.androidtoast.androidtoast.**toast**(text, length\_long=False)  
Displays a toast.

**Length\_long** The amount of time (in seconds) that the toast is visible on the screen.

**kivymd.toast.kivytoast****API - kivymd.toast.kivytoast****Submodules****KivyToast****Implementation of toasts for desktop.**

```

from kivy.lang import Builder

from kivymd.app import MDApp
from kivymd.toast import toast

KV = '''
BoxLayout:
    orientation:'vertical'

```

(continues on next page)

(continued from previous page)

```
MDToolbar:  
    id: toolbar  
    title: 'Test Toast'  
    md_bg_color: app.theme_cls.primary_color  
    left_action_items: [['menu', lambda x: '']]  
  
FloatLayout:  
  
    MDRaisedButton:  
        text: 'TEST KIVY TOAST'  
        on_release: app.show_toast()  
        pos_hint: {'center_x': .5, 'center_y': .5}  
  
'''  
  
class Test(MDApp):  
    def show_toast(self):  
        '''Displays a toast on the screen.'''  
  
        toast('Test Kivy Toast')  
  
    def build(self):  
        return Builder.load_string(KV)  
  
Test().run()
```

## API - `kivymd.toast.kivytoast.kivytoast`

**class** kivymd.toast.kivytoast.kivytoast.Toast(\*\*kwargs)  
ModalView class. See module documentation for more information.

### Events

**on\_pre\_open:** Fired before the ModalView is opened. When this event is fired ModalView is not yet added to window.

**on\_open:** Fired when the ModalView is opened.

**on\_pre\_dismiss:** Fired before the ModalView is closed.

**on\_dismiss:** Fired when the ModalView is closed. If the callback returns True, the dismiss will be canceled.

Changed in version 1.11.0: Added events `on_pre_open` and `on_pre_dismiss`.

### duration

The amount of time (in seconds) that the toast is visible on the screen.

`duration` is an `NumericProperty` and defaults to 2.5.

**label\_check\_texture\_size**(self, instance, texture\_size)

**toast**(self, text\_toast)

**on\_open**(self)

**fade\_in**(self)

```
fade_out(self, interval)
on_touch_down(self, touch)
    Receive a touch down event.
```

#### Parameters

**touch: MotionEvent class** Touch received. The touch is in parent coordinates. See `relativelayout` for a discussion on coordinate systems.

**Returns** bool If True, the dispatching of the touch event will stop. If False, the event will continue to be dispatched to the rest of the widget tree.

`kivymd.toast.kivytoast.kivytoast.toast(text: str, duration=2.5)`

Displays a toast.

**Duration** The amount of time (in seconds) that the toast is visible on the screen.

## kivymd.tools

### API - kivymd.tools

#### Submodules

### kivymd.tools.packaging

#### API - kivymd.tools.packaging

#### Submodules

### PyInstaller hooks

Add `hookspath=[kivymd.hooks_path]` to your .spec file.

#### Example of .spec file

```
# -*- mode: python ; coding: utf-8 -*-

import sys
import os

from kivy_deps import sdl2, glew

from kivymd import hooks_path as kivymd_hooks_path

path = os.path.abspath(".")

a = Analysis(
    ["main.py"],
    pathex=[path],
    hookspath=[kivymd_hooks_path],
    win_no_prefer_redirects=False,
    win_private_assemblies=False,
    cipher=None,
```

(continues on next page)

(continued from previous page)

```
        noarchive=False,
    )
pyz = PYZ(a.pure, a.zipped_data, cipher=None)

exe = EXE(
    pyz,
    a.scripts,
    a.binaries,
    a.zipfiles,
    a.datas,
    *[Tree(p) for p in (sd12.dep_bins + glew.dep_bins)],
    debug=False,
    strip=False,
    upx=True,
    name="app_name",
    console=True,
)
```

### API - `kivymd.tools.packaging.pyinstaller`

`kivymd.tools.packaging.pyinstaller.hooks_path`

Path to hook directory to use with PyInstaller. See `kivymd.tools.packaging.pyinstaller` for more information.

`kivymd.tools.packaging.pyinstaller.datas = [None, None]`

`kivymd.tools.packaging.pyinstaller.hiddenimports = ['PIL']`

### Submodules

#### `kivymd.tools.packaging.pyinstaller.hook-kivymd`

#### API - `kivymd.tools.packaging.pyinstaller.hook-kivymd`

#### `kivymd.tools.release`

#### API - `kivymd.tools.release`

### Submodules

#### `kivymd.tools.release.argument_parser`

**API - kivymd.tools.release.argument\_parser**

```
class kivymd.tools.release.argument_parser.ArgumentParserWithHelp(prog=None,  

    us-  

    age=None,  

    descrip-  

    tion=None,  

    epi-  

    log=None,  

    parents=[],  

    format-  

    ter_class=HelpFormatter,  

    prefix_chars='-'  

    ',', from-  

    file_prefix_chars=None,  

    argu-  

    ment_default=None,  

    con-  

    flict_handler='error',  

    add_help=True,  

    al-  

    low_abbrev=True)
```

Object for parsing command line strings into Python objects.

**Keyword Arguments:**

- prog – The name of the program (default: sys.argv[0])
- usage – A usage message (default: auto-generated from arguments)
- description – A description of what the program does
- epilog – Text following the argument descriptions
- parents – Parsers whose arguments should be copied into this one
- formatter\_class – HelpFormatter class for printing help messages
- prefix\_chars – Characters that prefix optional arguments
- **fromfile\_prefix\_chars – Characters that prefix files containing additional arguments**
- argument\_default – The default value for all arguments
- conflict\_handler – String indicating how to handle conflicts
- add\_help – Add a -h/-help option
- allow\_abbrev – Allow long options to be abbreviated unambiguously

**parse\_args** (self, args=None, namespace=None)

**error** (self, message)  
 error(message: string)

Prints a usage message incorporating the message to stderr and exits.

If you override this in a subclass, it should not return – it should either exit or raise an exception.

**format\_help** (self)

## kivymd.tools.release.git\_commands

### API - kivymd.tools.release.git\_commands

`kivymd.tools.release.git_commands.command(cmd: list, capture_output: bool = False) → str`

Run system command.

`kivymd.tools.release.git_commands.get_previous_version() → str`

Returns latest tag in git.

`kivymd.tools.release.git_commands.git_clean(ask: bool = True)`

Clean git repository from untracked and changed files.

`kivymd.tools.release.git_commands.git_commit(message: str, allow_error: bool = False, add_files: list = None)`

Make commit.

`kivymd.tools.release.git_commands.git_tag(name: str)`

Create tag.

`kivymd.tools.release.git_commands.git_push(branches_to_push: list, ask: bool = True, push: bool = False)`

Push all changes.

## Script to make release

Run this script before release (before deploying).

What this script does:

- Undo all local changes in repository
- Update version in `__init__.py`, `README`
- Format files
- Rename file “`unreleased.rst`” to `version`, add to `index.rst`
- Commit “Version ...”
- Create tag
- Add “`unreleased.rst`” to Change Log, add to `index.rst`
- Commit
- Git push

### API - kivymd.tools.release.make\_release

`kivymd.tools.release.make_release.run_pre_commit()`

Run pre-commit.

`kivymd.tools.release.make_release.replace_in_file(pattern, repl, file)`

Replace one `pattern` match to `repl` in file `file`.

`kivymd.tools.release.make_release.update_init_py(version, is_release, test: bool = False)`

Change version in `kivymd/__init__.py`.

```
kivymd.tools.release.make_release.update_readme(previous_version, version, test: bool = False)
```

Change version in README.

```
kivymd.tools.release.make_release.move_changelog(index_file, unreleased_file, previous_version, version_file, version, test: bool = False)
```

Edit unreleased.rst and rename to <version>.rst.

```
kivymd.tools.release.make_release.create_unreleased_changelog(index_file, unreleased_file, version, ask: bool = True, test: bool = False)
```

Create unreleased.rst by template.

```
kivymd.tools.release.make_release.main()
```

```
kivymd.tools.release.make_release.create_argument_parser()
```

## Tool for updating Iconic font

Downloads archive from <https://github.com/Templarian/MaterialDesign-Webfont> and updates font file with icon\_definitions.

### API - kivymd.tools.release.update\_icons

```
kivymd.tools.release.update_icons.kivymd_path
```

```
kivymd.tools.release.update_icons.font_path
```

```
kivymd.tools.release.update_icons.icon_definitions_path
```

```
kivymd.tools.release.update_icons.font_version = master
```

```
kivymd.tools.release.update_icons.url
```

```
kivymd.tools.release.update_icons.temp_path
```

```
kivymd.tools.release.update_icons.temp_repo_path
```

```
kivymd.tools.release.update_icons.temp_font_path
```

```
kivymd.tools.release.update_icons.temp_preview_path
```

```
kivymd.tools.release.update_icons.re_icons_json
```

```
kivymd.tools.release.update_icons.re_additional_icons
```

```
kivymd.tools.release.update_icons.re_version
```

```
kivymd.tools.release.update_icons.re_quote_keys
```

```
kivymd.tools.release.update_icons.re_icon_definitions
```

```
kivymd.tools.release.update_icons.re_version_in_file
```

```
kivymd.tools.release.update_icons.download_file(url, path)
```

```
kivymd.tools.release.update_icons.unzip_archive(archive_path, dir_path)
```

```
kivymd.tools.release.update_icons.get_icons_list()
```

```
kivymd.tools.release.update_icons.make_icon_definitions(icons)
```

```
kivymd.tools.release.update_icons.export_icon_definitions(icon_definitions,      ver-  
sion)  
kivymd.tools.release.update_icons.update_icons(make_commit: bool = False)  
kivymd.tools.release.update_icons.main()
```

## kivymd.uix

### API - kivymd.uix

```
class kivymd.uix.MDAdaptiveWidget(**kwargs)  
Widget class. See module documentation for more information.
```

#### Events

**on\_touch\_down:** (*touch*, ) Fired when a new touch event occurs. *touch* is the touch object.  
**on\_touch\_move:** (*touch*, ) Fired when an existing touch moves. *touch* is the touch object.  
**on\_touch\_up:** (*touch*, ) Fired when an existing touch disappears. *touch* is the touch object.  
**on\_kv\_post:** (*base\_widget*, ) Fired after all the kv rules associated with the widget and all other widgets that are in any of those rules have had all their kv rules applied. *base\_widget* is the base-most widget whose instantiation triggered the kv rules (i.e. the widget instantiated from Python, e.g. `MyWidget()`).

Changed in version 1.11.0.

**Warning:** Adding a `__del__` method to a class derived from Widget with Python prior to 3.4 will disable automatic garbage collection for instances of that class. This is because the Widget class creates reference cycles, thereby preventing garbage collection.

Changed in version 1.0.9: Everything related to event properties has been moved to the `EventDispatcher`. Event properties can now be used when contructing a simple class without subclassing `Widget`.

Changed in version 1.5.0: The constructor now accepts `on_*` arguments to automatically bind callbacks to properties or events, as in the Kv language.

#### adaptive\_height

If `True`, the following properties will be applied to the widget:

```
size_hint_y: None  
height: self.minimum_height
```

`adaptive_height` is an `BooleanProperty` and defaults to `False`.

#### adaptive\_width

If `True`, the following properties will be applied to the widget:

```
size_hint_x: None  
width: self.minimum_width
```

`adaptive_width` is an `BooleanProperty` and defaults to `False`.

#### adaptive\_size

If `True`, the following properties will be applied to the widget:

```
size_hint: None, None
size: self.minimum_size
```

`adaptive_size` is an `BooleanProperty` and defaults to `False`.

```
on_adaptive_height(self, instance, value)
on_adaptive_width(self, instance, value)
on_adaptive_size(self, instance, value)
```

## Submodules

### kivymd.uix.carousel

#### API - kivymd.uix.carousel

```
class kivymd.uix.carousel.MDCarousel(**kwargs)
Carousel class. See module documentation for more information.

on_slide_progress(self, *args)
on_slide_complete(self, *args)
```

## Behaviors

Modules and classes implementing various behaviors for buttons etc.

#### API - kivymd.uix.behaviors

## Submodules

### kivymd.utils

#### API - kivymd.utils

## Submodules

### asynckivy

Copyright (c) 2019 Nattōsai Mitō

**GitHub** - <https://github.com/gottadiveintopython>

**GitHub Gist** - <https://gist.github.com/gottadiveintopython/5f4a775849f9277081c396de65dc57c1>

### API - kivymd.utils.asynckivy

```
kivymd.utils.asynckivy.start(coro)
kivymd.utils.asynckivy.sleep(duration)
class kivymd.utils.asynckivy.event(ed, name)

bind(self, step_coro)
callback(self, *args, **kwargs)
```

### Crop Image

#### API - kivymd.utils.cropimage

```
kivymd.utils.cropimage.crop_image(cutting_size, path_to_image, path_to_save_crop_image,
corner=0, blur=0, corner_mode='all')
```

Call functions of cropping/blurring/rounding image.

cutting\_size: size to which the image will be cropped; path\_to\_image: path to origin image; path\_to\_save\_crop\_image: path to new image; corner: value of rounding corners; blur: blur value; corner\_mode: 'all'/'top'/'bottom' - indicates which corners to round out;

```
kivymd.utils.cropimage.add.blur(im, mode)
kivymd.utils.cropimage.add.corners(im, corner, corner_mode)
kivymd.utils.cropimage.prepare.mask(size, antialias=2)
kivymd.utils.cropimage.crop.round.image(cutting_size, path_to_image, path_to_new_image)
```

### Fit Image

Feature to automatically crop a Kivy image to fit your layout Write by Benedikt Zwölfer

Referene - <https://gist.github.com/benni12er/95a45eb168fc33a4fc2d545af692dad>

#### Example:

```
BoxLayout:
    size_hint_y: None
    height: "200dp"
    orientation: 'vertical'

    FitImage:
        size_hint_y: 3
        source: 'images/img1.jpg'

    FitImage:
        size_hint_y: 1
        source: 'images/img2.jpg'
```

**Example with round corners:**

```
from kivy.uix.modalview import ModalView
from kivy.lang import Builder

from kivymd import images_path
from kivymd.app import MDApp
from kivymd.uix.card import MDCard

Builder.load_string(
    '''
<Card>:
    elevation: 10
```

(continues on next page)

(continued from previous page)

```
radius: [36, ]  
  
FitImage:  
    id: bg_image  
    source: "images/bg.png"  
    size_hint_y: .35  
    pos_hint: {"top": 1}  
    radius: [36, 36, 0, 0, ]  
'''')  
  
class Card(MDCard):  
    pass  
  
class Example(MDApp):  
    def build(self):  
        modal = ModalView(  
            size_hint=(0.4, 0.8),  
            background=f"images_path)/transparent.png",  
            overlay_color=(0, 0, 0, 0),  
        )  
        modal.add_widget(Card())  
        modal.open()  
  
Example().run()
```

## API - kivymd.utils.fitimage

```
class kivymd.utils.fitimage.FitImage(**kwargs)  
    Box layout class. See module documentation for more information.  
  
    source  
    container  
    radius
```

## Monitor module

The Monitor module is a toolbar that shows the activity of your current application :

- FPS

### API - kivymd.utils.fpsmonitor

```
class kivymd.utils.fpsmonitor.FpsMonitor(**kwargs)
```

Label class, see module documentation for more information.

#### Events

**on\_ref\_press** Fired when the user clicks on a word referenced with a [ref] tag in a text markup.

**updated\_interval**

FPS refresh rate.

**start(self)**

**update\_fps(self, \*args)**

## HotReloadViewer

---

**Note:** The `HotReloadViewer` class is based on the `KvViewerApp` class

---

`HotReloadViewer`, for KV-Viewer, is a simple tool allowing you to dynamically display a KV file, taking its changes into account (thanks to watchdog). The idea is to facilitate design using the KV language.

## Usage

```
from kivy.lang import Builder

from kivymd.app import MDApp

KV = '''
#:import KivyLexer kivy.extras.highlight.KivyLexer
#:import HotReloadViewer kivymd.utils.hot_reload_viewer.HotReloadViewer

BoxLayout:

    CodeInput:
        lexer: KivyLexer()
        style_name: "native"
        on_text: app.update_kv_file(self.text)
        size_hint_x: .7

    HotReloadViewer:
        size_hint_x: .3
        path: app.path_to_kv_file
        errors: True
```

(continues on next page)

(continued from previous page)

```
errors_text_color: 1, 1, 0, 1
errors_background_color: app.theme_cls.bg_dark
...

class Example(MDApp):
    path_to_kv_file = "kv_file.kv"

    def build(self):
        self.theme_cls.theme_style = "Dark"
        return Builder.load_string(KV)

    def update_kv_file(self, text):
        with open(self.path_to_kv_file, "w") as kv_file:
            kv_file.write(text)

Example().run()
```

This will display the test.kv and automatically update the display when the file changes.

**This scripts uses watchdog to listen for file changes. To install watchdog.**

```
pip install watchdog
```

#### API - kivymd.utils.hot\_reload\_viewer

**class** kivymd.utils.hot\_reload\_viewer.**HotReloadErrorText** (\*\*kwargs)  
ScrollView class. See module documentation for more information.

##### Events

*on\_scroll\_start* Generic event fired when scrolling starts from touch.  
*on\_scroll\_move* Generic event fired when scrolling move from touch.  
*on\_scroll\_stop* Generic event fired when scrolling stops from touch.

Changed in version 1.9.0: *on\_scroll\_start*, *on\_scroll\_move* and *on\_scroll\_stop* events are now dispatched when scrolling to handle nested ScrollViews.

Changed in version 1.7.0: *auto\_scroll*, *scroll\_friction*, *scroll\_moves*, *scroll\_stoptime*' has been deprecated, use *:attr:`effect\_cls`* instead.

##### text

Text errors.

*text* is an *StringProperty* and defaults to ''.

##### errors\_text\_color

Error text color.

*errors\_text\_color* is an *ListProperty* and defaults to [].

**class** kivymd.utils.hot\_reload\_viewer.**HotReloadHandler** (callback, target, \*\*kwargs)

##### on\_any\_event (self, event)

```
class kivymd.utils.hot_reload_viewer.HotReloadViewer(**kwargs)
```

### Events

**on\_error** Called when an error occurs in the KV-file that the user is editing.

#### path

Path to KV file.

`path` is an `StringProperty` and defaults to ''.

#### errors

Show errors while editing KV-file.

`errors` is an `BooleanProperty` and defaults to `False`.

#### errors\_background\_color

Error background color.

`errors_background_color` is an `ListProperty` and defaults to `[]`.

#### errors\_text\_color

Error text color.

`errors_text_color` is an `ListProperty` and defaults to `[]`.

#### update(self, \*args)

Updates and displays the KV-file that the user edits.

#### show\_error(self, error)

Displays text with a current error.

#### on\_error(self, \*args)

Called when an error occurs in the KV-file that the user is editing.

#### on\_errors\_text\_color(self, instance, value)

#### on\_path(self, instance, value)

## kivymd.vendor

### API - kivymd.vendor

#### Submodules

#### CircularLayout

CircularLayout is a special layout that places widgets around a circle.

## **size\_hint**

`size_hint_x` is used as an angle-quota hint (widget with higher `size_hint_x` will be farther from each other, and vice versa), while `size_hint_y` is used as a widget size hint (widgets with a higher size hint will be bigger). `size_hint_x` cannot be `None`.

Widgets are all squares, unless you set `size_hint_y` to `None` (in that case you'll be able to specify your own size), and their size is the difference between the outer and the inner circle's radii. To make the widgets bigger you can just decrease `inner_radius_hint`.

## **API - `kivymd.vendor.circleLayout`**

**class** `kivymd.vendor.circleLayout.CircularLayout (**kwargs)`

Circular layout class. See module documentation for more information.

### **padding**

Padding between the layout box and its children: [`padding_left`, `padding_top`, `padding_right`, `padding_bottom`].

`padding` also accepts a two argument form [`padding_horizontal`, `padding_vertical`] and a one argument form [`padding`].

`padding` is a `VariableListProperty` and defaults to [0, 0, 0, 0].

### **start\_angle**

Angle (in degrees) at which the first widget will be placed. Start counting angles from the X axis, going counterclockwise.

`start_angle` is a `NumericProperty` and defaults to 0 (start from the right).

### **circle\_quota**

Size (in degrees) of the part of the circumference that will actually be used to place widgets.

`circle_quota` is a `BoundedNumericProperty` and defaults to 360 (all the circumference).

### **direction**

Direction of widgets in the circle.

`direction` is an `OptionProperty` and defaults to 'ccw'. Can be 'ccw' (counterclockwise) or 'cw' (clockwise).

### **outer\_radius\_hint**

Sets the size of the outer circle. A number greater than 1 will make the widgets larger than the actual widget, a number smaller than 1 will leave a gap.

`outer_radius_hint` is a `NumericProperty` and defaults to 1.

### **inner\_radius\_hint**

Sets the size of the inner circle. A number greater than `outer_radius_hint` will cause glitches. The closest it is to `outer_radius_hint`, the smallest will be the widget in the layout.

`outer_radius_hint` is a `NumericProperty` and defaults to 1.

### **radius\_hint**

Combined `outer_radius_hint` and `inner_radius_hint` in a list for convenience. See their documentation for more details.

`radius_hint` is a `ReferenceListProperty`.

### **delta\_radii**

**do\_layout (self, \*largs)**

This function is called when a layout is called by a trigger. If you are writing a new Layout subclass, don't call this function directly but use `_trigger_layout ()` instead.

The function is by default called *before* the next frame, therefore the layout isn't updated immediately. Anything depending on the positions of e.g. children should be scheduled for the next frame.

New in version 1.0.8.

## Circular Date & Time Picker for Kivy

(currently only time, date coming soon)

Based on [CircularLayout](<https://github.com/kivy-garden/garden.circularlayout>). The main aim is to provide a date and time selector similar to the one found in Android KitKat+.

### Simple usage

Import the widget with

```
from kivy.garden.circulardatetimepicker import CircularTimePicker
```

then use it! That's it!

```
from kivymd.app import MDApp
from kivymd.uix.screen import MDScreen

class Example(MDApp):
    def build(self):
        box = MDScreen(md_bg_color=self.theme_cls.bg_darkest)
        box.add_widget(CircularTimePicker())
        return box

Example().run()
```

in Kv language:

```
<TimeChooserPopup@Popup>:

    MDBBoxLayout:
        orientation: "vertical"

        CircularTimePicker:

            Button:
                text: "Dismiss"
                size_hint_y: None
                height: "40dp"
                on_release: root.dismiss()
```

## API - kivymd.vendor.circularTimePicker

kivymd.vendor.circularTimePicker.xrange(first=None, second=None, third=None)

kivymd.vendor.circularTimePicker.map\_number(x, in\_min, in\_max, out\_min, out\_max)

kivymd.vendor.circularTimePicker.rgb\_to\_hex(\*color)

**class** kivymd.vendor.circularTimePicker.Number(\*\*kwargs)

The class used to show the numbers in the selector.

### size\_factor

Font size scale.

`size_factor` is a [NumericProperty](#) and defaults to 0.5.

**class** kivymd.vendor.circularTimePicker.CircularNumberPicker(\*\*kw)

A circular number picker based on CircularLayout. A selector will help you pick a number. You can also set `multiples_of` to make it show only some numbers and use the space in between for the other numbers.

### min

The first value of the range.

`min` is a [NumericProperty](#) and defaults to 0.

### max

The last value of the range. Note that it behaves like xrange, so the actual last displayed value will be `max` - 1.

`max` is a [NumericProperty](#) and defaults to 0.

### range

Packs `min` and `max` into a list for convenience. See their documentation for further information.

`range` is a [ReferenceListProperty](#).

### multiples\_of

Only show numbers that are multiples of this number. The other numbers will be selectable, but won't have their own label.

`multiples_of` is a [NumericProperty](#) and defaults to 1.

### selector\_color

Color of the number selector. RGB.

`selector_color` is a [ListProperty](#) and defaults to [.337, .439, .490] (material green).

### color

Color of the number labels and of the center dot. RGB.

`color` is a [ListProperty](#) and defaults to [1, 1, 1] (white).

### selector\_alpha

Alpha value for the transparent parts of the selector.

`selector_alpha` is a [BoundedNumericProperty](#) and defaults to 0.3 (min=0, max=1).

### selected

Currently selected number.

`selected` is a [NumericProperty](#) and defaults to `min`.

### number\_size\_factor

Font size scale factor for the `Number`.

`number_size_factor` is a [NumericProperty](#) and defaults to 0.5.

**number\_format\_string**

String that will be formatted with the selected number as the first argument. Can be anything supported by `str.format()` (es. “{:02d}”).

`number_format_string` is a `StringProperty` and defaults to “{}”.

**scale**

Canvas scale factor. Used in `CircularTimePicker` transitions.

`scale` is a `NumericProperty` and defaults to 1.

**items****shown\_items****dot\_is\_none(self, \*args)****on\_touch\_down(self, touch)**

Receive a touch down event.

**Parameters**

**touch: MotionEvent class** Touch received. The touch is in parent coordinates. See `RelativeLayout` for a discussion on coordinate systems.

**Returns** bool If True, the dispatching of the touch event will stop. If False, the event will continue to be dispatched to the rest of the widget tree.

**on\_touch\_move(self, touch)**

Receive a touch move event. The touch is in parent coordinates.

See `on_touch_down()` for more information.

**on\_touch\_up(self, touch)**

Receive a touch up event. The touch is in parent coordinates.

See `on_touch_down()` for more information.

**on\_selected(self, \*a)****pos\_for\_number(self, n)**

Returns the center x, y coordinates for a given number.

**number\_at\_pos(self, x, y)**

Returns the number at a given x, y position. The number is found using the widget’s center as a starting point for angle calculations.

Not thoroughly tested, may yield wrong results.

**class kivymd.vendor.circularTimePicker.CircularMinutePicker(\*\*kw)**  
`CircularNumberPicker` implementation for minutes.

**class kivymd.vendor.circularTimePicker.CircularHourPicker(\*\*kw)**  
`CircularNumberPicker` implementation for hours.

**class kivymd.vendor.circularTimePicker.CircularTimePicker(\*\*kw)**  
Widget that makes use of `CircularHourPicker` and `CircularMinutePicker` to create a user-friendly, animated time picker like the one seen on Android.

See module documentation for more details.

**primary\_dark****hours**

The hours, in military format (0-23).

`hours` is a `NumericProperty` and defaults to 0 (12am).

**minutes**

The minutes.

`minutes` is a `NumericProperty` and defaults to 0.

**time\_list**

Packs `hours` and `minutes` in a list for convenience.

`time_list` is a `ReferenceListProperty`.

**time\_format**

String that will be formatted with the time and shown in the time label. Can be anything supported by `str.format()`. Make sure you don't remove the refs. See the default for the arguments passed to format. `time_format` is a `StringProperty` and defaults to “[color={hours\_color}][ref=hours]{hours}{/ref}[:color]:[color={minutes\_color}][ref=minutes] {minutes:02d}[/ref][:color]”.

**ampm\_format**

String that will be formatted and shown in the AM/PM label. Can be anything supported by `str.format()`. Make sure you don't remove the refs. See the default for the arguments passed to format.

`ampm_format` is a `StringProperty` and defaults to “[color={am\_color}][ref=am]AM[/ref] [/color] [color={pm\_color}][ref=pm]PM[/ref] [/color]”.

**picker**

Currently shown time picker. Can be one of “minutes”, “hours”.

`picker` is a `OptionProperty` and defaults to “hours”.

**selector\_color**

Color of the number selector and of the highlighted text. RGB.

`selector_color` is a `ListProperty` and defaults to [.337, .439, .490] (material green).

**color**

Color of the number labels and of the center dot. RGB.

`color` is a `ListProperty` and defaults to [1, 1, 1] (white).

**selector\_alpha**

Alpha value for the transparent parts of the selector.

`selector_alpha` is a `BoundedNumericProperty` and defaults to 0.3 (min=0, max=1).

**time**

Selected time as a `datetime.time` object.

`time` is an `AliasProperty`.

**time\_text**

**ampm\_text**

`set_time(self, dt)`

`on_ref_press(self, ign, ref)`

`on_selected(self, *a)`

`on_time_list(self, *a)`

`on_ampm(self, *a)`

`is_animating(self, *args)`

```
is_not_animating(self, *args)
on_touch_down(self, touch)
    Receive a touch down event.
```

#### Parameters

**touch: MotionEvent class** Touch received. The touch is in parent coordinates. See `relativelayout` for a discussion on coordinate systems.

**Returns** bool If True, the dispatching of the touch event will stop. If False, the event will continue to be dispatched to the rest of the widget tree.

```
on_touch_up(self, touch)
    Receive a touch up event. The touch is in parent coordinates.
```

See `on_touch_down()` for more information.

```
class kivymd.vendor.circularTimePicker.Example(**kwargs)
```

Application class, see module documentation for more information.

#### Events

**on\_start:** Fired when the application is being started (before the `runTouchApp()` call).

**on\_stop:** Fired when the application stops.

**on\_pause:** Fired when the application is paused by the OS.

**on\_resume:** Fired when the application is resumed from pause by the OS. Beware: you have no guarantee that this event will be fired after the `on_pause` event has been called.

Changed in version 1.7.0: Parameter `kv_file` added.

Changed in version 1.8.0: Parameters `kv_file` and `kv_directory` are now properties of App.

```
build(self)
```

Initializes the application; it will be called only once. If this method returns a widget (tree), it will be used as the root widget and added to the window.

**Returns** None or a root `Widget` instance if no `self.root` exists.



---

CHAPTER  
**THREE**

---

## **INDICES AND TABLES**

- genindex
- modindex
- search



## PYTHON MODULE INDEX

### k

kivymd, 251  
kivymd.app, 16  
kivymd.color\_definitions, 18  
kivymd.factory\_registers, 252  
kivymd.font\_definitions, 23  
kivymd.icon\_definitions, 21  
kivymd.material\_resources, 252  
kivymd.stiffscroll, 252  
kivymd.theming, 6  
kivymd.theming\_dynamic\_text, 252  
kivymd.toast, 254  
kivymd.toast.androidtoast, 254  
kivymd.toast.androidtoast.androidtoast, 254  
kivymd.toast.kivytoast, 255  
kivymd.toast.kivytoast.kivytoast, 255  
kivymd.tools, 257  
kivymd.tools.packaging, 257  
kivymd.tools.packaging.pyinstaller, 257  
kivymd.tools.packaging.pyinstaller.hook-  
    258  
kivymd.tools.release, 258  
kivymd.tools.release.argument\_parser,  
    258  
kivymd.tools.release.git\_commands, 260  
kivymd.tools.release.make\_release, 260  
kivymd.tools.release.update\_icons, 261  
kivymd.uix, 262  
kivymd.uix.backdrop, 202  
kivymd.uix.banner, 37  
kivymd.uix.behaviors, 263  
kivymd.uix.behaviors.backgroundcolorbehav-  
    236  
kivymd.uix.behaviors.elevation, 238  
kivymd.uix.behaviors.focus\_behavior, 230  
kivymd.uix.behaviors.hover\_behavior, 228  
kivymd.uix.behaviors.magic\_behavior, 234  
kivymd.uix.behaviors.ripplebehavior, 232  
kivymd.uix.behaviors.toggle\_behavior,  
    241  
kivymd.uix.behaviors.touch\_behavior, 227  
kivymd.uix.bottomnavigation, 26  
kivymd.uix.bottomsheet, 59  
kivymd.uix.boxlayout, 135  
kivymd.uix.button, 121  
kivymd.uix.card, 179  
kivymd.uix.carousel, 263  
kivymd.uix.chip, 192  
kivymd.uix.datatables, 209  
kivymd.uix.dialog, 72  
kivymd.uix.dropdownitem, 52  
kivymd.uix.expansionpanel, 94  
kivymd.uix.filemanager, 195  
kivymd.uix.floatlayout, 119  
kivymd.uix.gridlayout, 120  
kivymd.uix.imagelist, 141  
kivymd.uix.label, 175  
kivymd.uix.list, 162  
kivymd.uix.menu, 105  
kivymd.uix.navigationdrawer, 86  
kivymd.uix.picker, 53  
kivymd.uix.progressbar, 67  
kivymd.uix.refreshlayout, 145  
kivymd.uix.relativelayout, 206  
kivymd.uix.screen, 208  
kivymd.uix.selectioncontrol, 137  
kivymd.uix.slider, 159  
kivymd.uix.snackbar, 32  
kivymd.uix.spinner, 24  
kivymd.uix.stacklayout, 207  
kivymd.uix.tab, 41  
kivymd.uix.taptargetview, 215  
kivymd.uix.textfield, 147  
kivymd.uix.toolbar, 98  
kivymd.uix.tooltip, 199  
kivymd.uix.useranimationcard, 83  
kivymd.utils, 263  
kivymd.utils.asynckivy, 263  
kivymd.utils.cropimage, 264  
kivymd.utils.fitimage, 264  
kivymd.utils.fpsmonitor, 267  
kivymd.utils.hot\_reload\_viewer, 267  
kivymd.vendor, 269

`kivymd.vendor.circleLayout`, [269](#)  
`kivymd.vendor.circularTimePicker`, [271](#)

# INDEX

## A

a (*kivymd.uix.behaviors.backgroundcolorbehavior.BackgroundColorAttribute*), 237  
accent\_color (*kivymd.theming.ThemeManager attribute*), 10  
accent\_dark (*kivymd.theming.ThemeManager attribute*), 10  
accent\_dark\_hue (*kivymd.theming.ThemeManager attribute*), 10  
accent\_hue (*kivymd.theming.ThemeManager attribute*), 10  
accent\_light (*kivymd.theming.ThemeManager attribute*), 10  
accent\_light\_hue (*kivymd.theming.ThemeManager attribute*), 10  
accent\_palette (*kivymd.theming.ThemeManager attribute*), 10  
active (*kivymd.uix.selectioncontrol.MDCheckbox attribute*), 140  
active (*kivymd.uix.selectioncontrol.MDSwitch attribute*), 141  
active (*kivymd.uix.slider.MDSlider attribute*), 161  
active (*kivymd.uix.spinner.MDSpinner attribute*), 26  
active\_line (*kivymd.uix.textfield.MDTextField attribute*), 157  
adaptive\_height (*kivymd.uix.MDAdaptiveWidget attribute*), 262  
adaptive\_size (*kivymd.uix.MDAdaptiveWidget attribute*), 262  
adaptive\_width (*kivymd.uix.MDAdaptiveWidget attribute*), 262  
add\_actions\_buttons () (*kivymd.uix.bannerMDBanner method*), 40  
add\_banner\_to\_container () (*kivymd.uix.bannerMDBanner method*), 40  
add.blur () (*in module kivymd.utils.cropimage*), 264  
add\_corners () (*in module kivymd.utils.cropimage*), 264  
add\_item () (*kivymd.uix.bottomsheet.MDGridBottomSheet method*), 67  
add\_item () (*kivymd.uix.bottomsheet.MDListBottomSheet method*), 66  
add\_scrim () (*kivymd.uix.navigationdrawer.NavigationLayout method*), 91  
add\_widget () (*kivymd.uix.backdrop.MDBackdrop method*), 205  
add\_widget () (*kivymd.uix.bottomnavigation.MDBottomNavigation method*), 31  
add\_widget () (*kivymd.uix.bottomsheet.MDBottomSheet method*), 65  
add\_widget () (*kivymd.uix.card.MDCardSwipe method*), 190  
add\_widget () (*kivymd.uix.chip.MDChooseChip method*), 195  
add\_widget () (*kivymd.uix.expansionpanel.MDExpansionPanel method*), 97  
add\_widget () (*kivymd.uix.list.ContainerSupport method*), 173  
add\_widget () (*kivymd.uix.list.MDList method*), 171  
add\_widget () (*kivymd.uix.navigationdrawer.NavigationLayout method*), 91  
add\_widget () (*kivymd.uix.tab.MDTabs method*), 50  
add\_widget () (*kivymd.uix.toolbar.MDBottomAppBar method*), 104  
adjust\_tooltip\_position () (*kivymd.uix.tooltip.MDTooltip method*), 201  
allow\_stretch (*kivymd.uix.tab.MDTabs attribute*), 50  
ampm\_format (*kivymd.vendor.circularTimePicker.CircularTimePicker attribute*), 274  
ampm\_text (*kivymd.vendor.circularTimePicker.CircularTimePicker attribute*), 274  
anchor (*kivymd.uix.button.MDFloatingActionButtonSpeedDial attribute*), 133  
anchor (*kivymd.uix.card.MDCardSwipe attribute*), 190  
anchor (*kivymd.uix.navigationdrawer.MDNavigationDrawer attribute*), 92  
anchor\_title (*kivymd.uix.toolbar.MDToolbar attribute*), 103  
anim\_complete () (*kivymd.uix.behaviors.ripplebehavior.CommonRipple method*), 234  
anim\_duration (*kivymd.uix.tab.MDTabs attribute*),

50  
 anim\_rect () (kivymd.uix.textfield.MDTextFieldRect  
     method), 155  
 anim\_threshold (kivymd.uix.tab.MDTabs attribute),  
     50  
 animation (kivymd.uix.bottomsheetMDBottomSheet  
     attribute), 64  
 animation\_display\_banner ()  
     (kivymd.uix.bannerMDBanner  
     method), 41  
 animation\_label ()  
     (kivymd.uix.button.MDTextButton  
     method), 131  
 animation\_to\_bottom ()  
     (kivymd.uix.useranimationcard.MDUserAnimationCard  
     method), 85  
 animation\_to\_top ()  
     (kivymd.uix.useranimationcard.MDUserAnimationCard  
     method), 85  
 animation\_tooltip\_show ()  
     (kivymd.uix.tooltip.MDTooltip method), 201  
 animtion\_icon\_close ()  
     (kivymd.uix.backdrop.MDBackdrop  
     method), 205  
 animtion\_icon\_menu ()  
     (kivymd.uix.backdrop.MDBackdrop  
     method), 205  
 ArgumentParserWithHelp (class  
     in kivymd.tools.release.argument\_parser), 259

**B**

b (kivymd.uix.behaviors.backgroundcolorbehavior.BackgroundColorBehavior  
     attribute), 237  
 back () (kivymd.uix.filemanager.MDFileManager  
     method), 199  
 background (kivymd.uix.bottomsheetMDBottomSheet  
     attribute), 64  
 background (kivymd.uix.card.MDCard attribute), 189  
 background\_color (kivymd.uix.backdrop.MDBackdrop  
     attribute), 205  
 background\_color (kivymd.uix.datatables.MDDDataTable  
     attribute), 215  
 background\_color (kivymd.uix.menu.MDDropdownMenu  
     attribute), 118  
 background\_color (kivymd.uix.picker.MDDatePicker  
     attribute), 58  
 background\_color (kivymd.uix.tab.MDTabs  
     attribute), 50  
 background\_down (kivymd.uix.behaviors.toggle\_behavior.MDToggleButton  
     attribute), 242  
 background\_hue (kivymd.uix.behaviors.backgroundcolorbehavior.SpecificBackgroundColorBehavior  
     attribute), 238  
 background\_normal  
     (kivymd.uix.behaviors.toggle\_behavior.MDToggleButton  
     tribute), 58

attribute), 242  
 background\_palette  
     (kivymd.uix.behaviors.backgroundcolorbehavior.SpecificBackgroundColorBehavior  
     attribute), 238  
 background\_palette  
     (kivymd.uix.button.MDFloatingActionButton  
     attribute), 131  
 BackgroundColorBehavior (class  
     in kivymd.uix.behaviors.backgroundcolorbehavior),  
     236  
 BaseListItem (class in kivymd.uix.list), 171  
 bg\_color (kivymd.uix.bottomsheetMDBottomSheet  
     attribute), 64  
 bg\_color (kivymd.uix.list.BaseListItem attribute), 172  
 bg\_color (kivymd.uix.button.MDFloatingActionButtonSpeedDial  
     attribute), 134  
 bg\_color\_root\_button  
     (kivymd.uix.button.MDFloatingActionButtonSpeedDial  
     attribute), 134  
 bg\_color\_stack\_button  
     (kivymd.uix.button.MDFloatingActionButtonSpeedDial  
     attribute), 134  
 bg\_dark (kivymd.theming.ThemeManager attribute), 12  
 bg\_darkest (kivymd.theming.ThemeManager  
     attribute), 11  
 bg\_hint\_color (kivymd.uix.button.MDFloatingActionButtonSpeedDial  
     attribute), 134  
 bg\_light (kivymd.theming.ThemeManager attribute),  
     13  
 bg\_normal (kivymd.theming.ThemeManager attribute),  
     12  
 bind () (kivymd.utils.asynckivy.event method), 264  
 body (kivymd.stiffscroll.StiffScrollEffect attribute), 253  
 border\_color (kivymd.uix.menu.MDDropdownMenu  
     attribute), 118  
 border\_point (kivymd.uix.behaviors.hover\_behavior.HoverBehavior  
     attribute), 230  
 box\_color (kivymd.uix.imagelist.SmartTile attribute),  
     144  
 box\_content (kivymd.uix.useranimationcard.MDUserAnimationCard  
     attribute), 84  
 box\_position (kivymd.uix.imagelist.SmartTile  
     attribute), 144  
 build () (kivymd.vendor.circularTimePicker.Example  
     method), 275  
 button\_callback (kivymd.uix.snackbar.Snackbar  
     attribute), 36  
 button\_color (kivymd.uix.snackbar.Snackbar  
     attribute), 36  
 button\_text (kivymd.uix.snackbar.Snackbar  
     tribute), 36  
 buttons (kivymd.uix.dialog.MDDialog attribute), 76

**C**

cal\_layout (kivymd.uix.picker.MDDatePicker  
     tribute), 58

cal\_list (*kivymd.uix.picker.MDDatePicker* attribute), [58](#) *kivymd.vendor.circularTimePicker*, [273](#)  
 callback (*kivymd.uix.button.MDFloatingActionButtonSpeedDial* attribute), [133](#) *CircularNumberPicker* (class) [in](#)  
*kivymd.vendor.circularTimePicker*, [272](#)  
 callback (*kivymd.uix.chip.MDChip* attribute), [194](#) *CircularRippleBehavior* (class) [in](#)  
*kivymd.uix.behaviors.ripplebehavior*, [234](#)  
 callback (*kivymd.uix.picker.MDDatePicker* attribute), [58](#) *CircularTimePicker* (class) [in](#)  
*kivymd.vendor.circularTimePicker*, [273](#)  
 callback (*kivymd.uix.useranimationcard.MDUserAnimationCard* attribute), [84](#) *close()* (*kivymd.uix.backdrop.MDBackdrop* method), [205](#)  
 callback () (*kivymd.utils.asynckivy.event* method), [264](#) *close\_cancel()* (*kivymd.uix.picker.MDTimePicker* method), [59](#)  
 caller (*kivymd.uix.menu.MDDropdownMenu* attribute), [118](#) *close\_card()* (*kivymd.uix.card.MDCardSwipe* method), [191](#)  
 can\_capitalize (*kivymd.uix.label.MDLabel* attribute), [179](#) *cancelable* (*kivymd.uix.taptargetview.MDTapTargetView* attribute), [226](#)  
*kivymd.uix.backdrop.MDBackdrop* attribute), [205](#)  
 caption (*kivymd.uix.bottomsheet.GridBottomSheetItem* attribute), [66](#) *close\_ok()* (*kivymd.uix.picker.MDTimePicker* method), [59](#)  
 catching\_duration (*kivymd.uix.progressbar.MDProgressBar* attribute), [71](#) *close\_on\_click* (*kivymd.uix.navigationdrawer.MDNavigationDrawer* attribute), [92](#)  
 catching\_transition (*kivymd.uix.progressbar.MDProgressBar* attribute), [71](#) *close\_panel()* (*kivymd.uix.expansionpanel.MDExpansionPanel* method), [97](#)  
 catching\_up () (*kivymd.uix.progressbar.MDProgressBar* method), [71](#) *close\_stack()* (*kivymd.uix.button.MDFloatingActionButtonSpeedDial* method), [135](#)  
 change\_month () (*kivymd.uix.picker.MDDatePicker* method), [58](#) *losing\_time* (*kivymd.uix.button.MDFloatingActionButtonSpeedDial* attribute), [134](#)  
 check (*kivymd.uix.chip.MDChip* attribute), [194](#) *closing\_time* (*kivymd.uix.expansionpanel.MDExpansionPanel* attribute), [97](#)  
 check (*kivymd.uix.datatables.MDDDataTable* attribute), [212](#) *closing\_time* (*kivymd.uix.navigationdrawer.MDNavigationDrawer* attribute), [93](#)  
 check\_open\_panel () (*kivymd.uix.expansionpanel.MDExpansionPanel* method), [97](#) *closing\_time\_button\_rotation* (*kivymd.uix.button.MDFloatingActionButtonSpeedDial* attribute), [134](#)  
 check\_position\_caller () (*kivymd.uix.menu.MDDropdownMenu* method), [118](#) *closing\_transition* (*kivymd.uix.button.MDFloatingActionButtonSpeedDial* attribute), [133](#)  
 checkbox\_icon\_down (*kivymd.uix.selectioncontrol.MDCheckbox* attribute), [140](#) *closing\_transition* (*kivymd.uix.card.MDCardSwipe* attribute), [190](#)  
 checkbox\_icon\_normal (*kivymd.uix.selectioncontrol.MDCheckbox* attribute), [140](#) *closing\_transition* (*kivymd.uix.expansionpanel.MDExpansionPanel* attribute), [96](#)  
 CheckboxLeftWidget (*class* in *kivymd.uix.list*), [174](#) *closing\_transition* (*kivymd.uix.navigationdrawer.MDNavigationDrawer* attribute), [93](#)  
 circle\_quota (*kivymd.vendor.circleLayout.CircularLayout* attribute), [270](#) *closing\_transition\_button\_rotation* (*kivymd.uix.button.MDFloatingActionButtonSpeedDial* attribute), [134](#)  
 CircularElevationBehavior (class *kivymd.uix.behaviors.elevation*), [240](#) *color* (*in module kivymd.theming\_dynamic\_text*), [252](#)  
 CircularHourPicker (class *kivymd.vendor.circularTimePicker*), [273](#) *color* (*kivymd.uix.card.MDSeparator* attribute), [188](#)  
 CircularLayout (class *kivymd.vendor.circleLayout*), [270](#) *color* (*kivymd.uix.chip.MDChip* attribute), [194](#)  
 CircularMinutePicker (class *kivymd.uix.progressbar.MDProgressBar* attribute), [71](#)

color (*kivymd.uix.spinner.MDSpinner* attribute), 26  
color (*kivymd.vendor.circularTimePicker.CircularNumberPicker* attribute), 272  
color (*kivymd.vendor.circularTimePicker.CircularTimePicker* attribute), 274  
color\_active (*kivymd.uix.textfield.MDTextFieldRound* attribute), 158  
color\_icon\_root\_button (*kivymd.uix.button.MDFloatingActionButtonSpeedDial* attribute), 134  
color\_icon\_stack\_button (*kivymd.uix.button.MDFloatingActionButtonSpeedDial* attribute), 134  
color\_indicator (*kivymd.uix.tab.MDTabs* attribute), 50  
color\_mode (*kivymd.uix.textfield.MDTextField* attribute), 156  
colors (in module *kivymd.color\_definitions*), 18  
column\_data (*kivymd.uix.datatables.MDDDataTable* attribute), 210  
command() (in module *kivymd.tools.release.git\_commands*), 260  
CommonElevationBehavior (class in *kivymd.uix.behaviors.elevation*), 240  
CommonRipple (class in *kivymd.uix.behaviors.ripplebehavior*), 233  
complete\_swipe() (*kivymd.uix.card.MDCardSwipe* method), 191  
container (*kivymd.utils.fitimage.FitImage* attribute), 266  
ContainerSupport (class in *kivymd.uix.list*), 173  
content (*kivymd.uix.expansionpanel.MDExpansionPanel* attribute), 96  
content\_cls (*kivymd.uix.dialog.MDDialog* attribute), 80  
create\_argument\_parser() (in module *kivymd.tools.release.make\_release*), 261  
create\_buttons() (*kivymd.uix.dialog.MDDialog* method), 82  
create\_clock() (*kivymd.uix.behaviors.touch\_behavior.TouchBehavior* method), 228  
create\_items() (*kivymd.uix.dialog.MDDialog* method), 82  
create\_menu\_items() (*kivymd.uix.menu.MDDropdownMenu* method), 118  
create\_pagination\_menu() (*kivymd.uix.datatables.MDDDataTable* method), 215  
create\_unreleased\_changelog() (in module *kivymd.tools.release.make\_release*), 261  
crop\_image() (in module *kivymd.utils.cropimage*), 264  
crop\_round\_image() (in module *kivymd.utils.cropimage*), 264  
current\_item (*kivymd.uix.dropdownitem.MDDropDownItem* attribute), 53  
current\_path (*kivymd.uix.filemanager.MDFileManager* attribute), 198  
custom\_color (*kivymd.uix.button.MDTextButton* attribute), 131

**D**

data (*kivymd.uix.button.MDFloatingActionButtonSpeedDial* attribute), 133  
datas (in module *kivymd.tools.packaging.pyinstaller*), 258  
day (*kivymd.uix.picker.MDDatePicker* attribute), 58  
default\_tab (*kivymd.uix.tab.MDTabs* attribute), 49  
delete\_clock() (*kivymd.uix.behaviors.touch\_behavior.TouchBehavior* method), 228  
delete\_clock() (*kivymd.uix.tooltip.MDTooltip* method), 201  
delta\_radii (*kivymd.vendor.circleLayout.CircularLayout* attribute), 270  
description\_text (*kivymd.uix.taptargetview.MDTapTargetView* attribute), 225  
description\_text\_bold (*kivymd.uix.taptargetview.MDTapTargetView* attribute), 226  
description\_text\_color (*kivymd.uix.taptargetview.MDTapTargetView* attribute), 225  
description\_text\_size (*kivymd.uix.taptargetview.MDTapTargetView* attribute), 225  
determinate (*kivymd.uix.spinner.MDSpinner* attribute), 26  
DETERMINE\_BEFORE\_TIME (*kivymd.uix.spinner.MDSpinner* attribute), 26  
DEVICE\_IOS (in module *kivymd.material\_resources*), 252  
device\_ios (*kivymd.theming.ThemableBehavior* attribute), 16  
device\_orientation (*kivymd.theming.ThemeManager* attribute), 14  
DEVICE\_TYPE (in module *kivymd.material\_resources*), 252  
direction (*kivymd.vendor.circleLayout.CircularLayout* attribute), 270  
disabled\_color (*kivymd.uix.selectioncontrol.MDCheckbox* attribute), 141

disabled\_hint\_text\_color  
     (*kivymd.theming.ThemeManager* attribute), 14

dismiss()  
     (*kivymd.uix.menu.MDDropdownMenu*  
     method), 119

displacement (*kivymd.stiffscroll.StiffScrollEffect* attribute), 253

display\_tooltip()  
     (*kivymd.uix.tooltip.MDTooltip*  
     method), 201

divider (*kivymd.uix.list.BaseListItem* attribute), 172

divider\_color (*kivymd.theming.ThemeManager* attribute), 13

do\_animation\_open\_stack()  
     (*kivymd.uix.button.MDFloatingActionButtonSpeedDial*  
     method), 135

do\_layout()  
     (*kivymd.vendor.circleLayout.CircularLayout*  
     method), 270

dot\_is\_none()  
     (*kivymd.vendor.circularTimePicker.CircularTimePicker*  
     method), 273

download\_file()  
     (in module  
     *kivymd.tools.release.update\_icons*), 261

dp (in module *kivymd.material\_resources*), 252

drag\_threshold (*kivymd.stiffscroll.StiffScrollEffect* attribute), 253

draw\_shadow (*kivymd.uix.taptargetview.MDTapTargetView* attribute), 226

duration (*kivymd.toast.kivytoast.kivytoast.Toast* attribute), 256

duration (*kivymd.uix.snackbar.Snackbar* attribute), 36

duration\_long\_touch  
     (*kivymd.uix.behaviors.touch\_behavior.TouchBehavior*  
     attribute), 228

duration\_opening (*kivymd.uix.bottomsheet.MDBottomSheet* attribute), 64

## E

edit\_padding\_for\_item()  
     (*kivymd.uix.dialog.MDDialog* method), 82

elevation (*kivymd.uix.behaviors.elevation.CommonElevationBehavior* attribute), 240

elevation (*kivymd.uix.card.MDCard* attribute), 189

elevation (*kivymd.uix.tab.MDTabs* attribute), 50

elevation (*kivymd.uix.toolbar.MDToolbar* attribute), 103

enable\_swiping (*kivymd.uix.navigationdrawer.MDNavigationDrawer* attribute), 92

error (*kivymd.uix.textfield.MDTextField* attribute), 157

error()  
     (*kivymd.tools.release.argument\_parser.ArgumentParserWithHelp*  
     method), 259

error\_color (*kivymd.theming.ThemeManager* attribute), 14

error\_color (*kivymd.uix.textfield.MDTextField* attribute), 157

errors (*kivymd.utils.hot\_reload\_viewer.HotReloadViewer* attribute), 269

errors\_background\_color  
     (*kivymd.utils.hot\_reload\_viewer.HotReloadViewer* attribute), 269

errors\_text\_color  
     (*kivymd.utils.hot\_reload\_viewer.HotReloadErrorText* attribute), 268

errors\_text\_color  
     (*kivymd.utils.hot\_reload\_viewer.HotReloadViewer* attribute), 269

event (class in *kivymd.utils.asynckivy*), 264

Example (class in *kivymd.vendor.circularTimePicker*), 275

export\_icon\_definitions()  
     (in module  
     *kivymd.tools.release.update\_icons*), 261

filemanager.MDFileManager attribute), 198

## F

fade\_in()  
     (*kivymd.toast.kivytoast.kivytoast.Toast* method), 256

fade\_out()  
     (*kivymd.toast.kivytoast.kivytoast.Toast* method), 256

fade\_out()  
     (*kivymd.uix.behaviors.ripplebehavior.CommonRipple* method), 234

fill\_color (*kivymd.uix.textfield.MDTextField* attribute), 157

finish\_ripple()  
     (*kivymd.uix.behaviors.ripplebehavior.CommonRipple* method), 234

first\_widget (*kivymd.uix.bottomnavigation.MDBottomNavigation* attribute), 31

FitImage (class in *kivymd.utils.fitimage*), 266

fmt\_lbl\_date()  
     (*kivymd.uix.picker.MDDatePicker* method), 58

focus\_behavior (*kivymd.uix.behaviors.focus\_behavior.FocusBehavior* attribute), 231

FocusBehavior  
     (*kivymd.uix.card.MDCard* attribute), 189

focus\_color (*kivymd.uix.behaviors.focus\_behavior.FocusBehavior* attribute), 231

FocusBehavior  
     (class in  
     *kivymd.uix.behaviors.focus\_behavior*), 231

font\_color\_normal

font\_name (*kivymd.uix.tab.MDTabs* attribute), 50

font\_path  
     (in module  
     *kivymd.tools.release.update\_icons*), 261

font\_size (*kivymd.uix.dropdownitem.MDDropDownItem* attribute), 53

font\_size (*kivymd.uix.snackbar.Snackbar* attribute), 36  
font\_style (*kivymd.uix.imagelist.SmartTileWithLabel* attribute), 144  
font\_style (*kivymd.uix.label.MDLabel* attribute), 178  
font\_style (*kivymd.uix.list.BaseListItem* attribute), 171  
font\_styles (*kivymd.theming.ThemeManager* attribute), 14  
font\_version (in module *kivymd.tools.release.update\_icons*), 261  
fonts (in module *kivymd.font\_definitions*), 23  
fonts\_path (in module *kivymd*), 251  
format\_help () (*kivymd.tools.release.argument\_parser.ArgumentParserWithHelp* method), 259  
FpsMonitor (class in *kivymd.utils.fpsmonitor*), 267

**G**

g (*kivymd.uix.behaviors.backgroundcolorbehavior.BackgroundColorBehavior* attribute), 237  
generate\_cal\_widgets () (*kivymd.uix.picker.MDDatePicker* method), 58  
get\_access\_string () (*kivymd.uix.filemanager.MDFileManager* method), 199  
get\_content () (*kivymd.uix.filemanager.MDFileManager* method), 199  
get\_contrast\_text\_color () (in module *kivymd.theming\_dynamic\_text*), 252  
get\_dist\_from\_side () (*kivymd.uix.navigationdrawer.MDNavigationDrawer* method), 93  
get\_icons\_list () (in module *kivymd.tools.release.update\_icons*), 261  
get\_normal\_height () (*kivymd.uix.dialog.MDDialog* method), 82  
get\_previous\_version () (in module *kivymd.tools.release.git\_commands*), 260  
get\_tab\_list () (*kivymd.uix.tab.MDTabs* method), 50  
git\_clean () (in module *kivymd.tools.release.git\_commands*), 260  
git\_commit () (in module *kivymd.tools.release.git\_commands*), 260  
git\_push () (in module *kivymd.tools.release.git\_commands*), 260  
git\_tag () (in module *kivymd.tools.release.git\_commands*), 260  
GridBottomSheetItem (class in *kivymd.uix.bottomsheet*), 66  
grow () (*kivymd.uix.behaviors.magic\_behavior.MagicBehavior* method), 236

**H**

header (*kivymd.uix.backdrop.MDBackdrop* attribute), 205  
header (*kivymd.uix.bottomnavigation.MDBottomNavigationItem* attribute), 30  
header\_text (*kivymd.uix.backdrop.MDBackdrop* attribute), 205  
helper\_text (*kivymd.uix.textfield.MDTextField* attribute), 156  
helper\_text\_mode (*kivymd.uix.textfield.MDTextField* attribute), 156  
hiddenimports (in module *kivymd.tools.packaging.pyinstaller*), 258  
hide () (*kivymd.uix.banner.MDBanner* method), 41  
hide\_animation\_spinner () (*kivymd.uix.refreshlayout.RefreshSpinner* method), 147  
hint (*kivymd.uix.slider.MDSlider* attribute), 161  
hint\_animation (*kivymd.uix.button.MDFloatingActionButtonSpeedDial* attribute), 134  
hint\_bg\_color (*kivymd.uix.slider.MDSlider* attribute), 161  
hint\_radius (*kivymd.uix.slider.MDSlider* attribute), 161  
hint\_text\_color (*kivymd.uix.slider.MDSlider* attribute), 161  
hooks\_path (in module *kivymd.tools.packaging.pyinstaller*), 258  
hor\_growth (*kivymd.uix.menu.MDDropdownMenu* attribute), 118  
horizontal\_margins (*kivymd.theming.ThemeManager* attribute), 14  
HotReloadErrorText (class in *kivymd.utils.hot\_reload\_viewer*), 268  
HotReloadHandler (class in *kivymd.utils.hot\_reload\_viewer*), 268  
HotReloadViewer (class in *kivymd.utils.hot\_reload\_viewer*), 268  
hours (*kivymd.vendor.circularTimePicker.CircularTimePicker* attribute), 273  
HoverBehavior (class in *kivymd.uix.behaviors.hover\_behavior*), 229  
hovered (*kivymd.uix.behaviors.hover\_behavior.HoverBehavior* attribute), 229  
hue (in module *kivymd.color\_definitions*), 20

**I**

icon (*kivymd.uix.banner.MDBanner* attribute), 40  
icon (*kivymd.uix.bottomnavigation.MDTab* attribute), 30  
icon (*kivymd.uix.button.MDFloatingActionButton* attribute), 131  
icon (*kivymd.uix.button.MDFloatingActionButtonSpeedDial* attribute), 132

icon (*kivymd.uix.button.MDIconButton* attribute), 131  
 icon (*kivymd.uix.chip.MDChip* attribute), 194  
 icon (*kivymd.uix.expansionpanel.MDExpansionPanel* attribute), 96  
 icon (*kivymd.uix.filemanager.MDFileManager* attribute), 198  
 icon (*kivymd.uix.label.MDIcon* attribute), 179  
 icon (*kivymd.uix.menu.RightContent* attribute), 117  
 icon (*kivymd.uix.toolbar.MDToolbar* attribute), 103  
 icon\_color (*kivymd.theming.ThemeManager* attribute), 13  
 icon\_color (*kivymd.uix.toolbar.MDToolbar* attribute), 103  
 icon\_definitions\_path (in module *kivymd.tools.release.update\_icons*), 261  
 icon\_folder (*kivymd.uix.filemanager.MDFileManager* attribute), 198  
 icon\_left (*kivymd.uix.textfield.MDTextFieldRound* attribute), 158  
 icon\_left\_color (*kivymd.uix.textfield.MDTextFieldRound* attribute), 158  
 icon\_right (*kivymd.uix.textfield.MDTextField* attribute), 157  
 icon\_right (*kivymd.uix.textfield.MDTextFieldRound* attribute), 158  
 icon\_right\_color (*kivymd.uix.textfield.MDTextField* attribute), 157  
 icon\_right\_color (*kivymd.uix.textfield.MDTextFieldRound* attribute), 158  
 icon\_size (*kivymd.uix.bottomsheet.GridBottomSheetItem* attribute), 66  
 IconLeftWidget (class in *kivymd.uix.list*), 174  
 IconRightWidget (class in *kivymd.uix.list*), 174  
 ILeftBody (class in *kivymd.uix.list*), 172  
 ILeftBodyTouch (class in *kivymd.uix.list*), 172  
 ImageLeftWidget (class in *kivymd.uix.list*), 174  
 ImageRightWidget (class in *kivymd.uix.list*), 174  
 images\_path (in module *kivymd*), 251  
 inner\_radius\_hint  
   (*kivymd.vendor.circleLayout.CircularLayout* attribute), 270  
 IRightBody (class in *kivymd.uix.list*), 173  
 IRightBodyTouch (class in *kivymd.uix.list*), 173  
 is\_animating () (*kivymd.vendor.circularTimePicker.CircularTimePicker* method), 274  
 is\_not\_animating ()  
   (*kivymd.vendor.circularTimePicker.CircularTimePicker* method), 274  
 items (*kivymd.uix.dialog.MDDialog* attribute), 76  
 items (*kivymd.uix.menu.MDDropdownMenu* attribute), 117  
 items (*kivymd.vendor.circularTimePicker.CircularNumberPicker* attribute), 273

**K**

kivymd  
   module, 1, 251  
 kivymd.app  
   module, 16  
 kivymd.color\_definitions  
   module, 18  
 kivymd.factory\_registers  
   module, 252  
 kivymd.font\_definitions  
   module, 23  
 kivymd.icon\_definitions  
   module, 21  
 kivymd.material\_resources  
   module, 252  
 kivymd.stiffscroll  
   module, 252  
 kivymd.theming  
   module, 6  
 kivymd.theming\_dynamic\_text  
   module, 252  
 kivymd.toast  
   module, 254  
 kivymd.toast.androidtoast  
   module, 254  
 kivymd.toast.androidtoast.androidtoast  
   module, 254  
 kivymd.toast.kivytoast  
   module, 255  
 kivymd.toast.kivytoast.kivytoast  
   module, 255  
 kivymd.tools  
   module, 257  
 kivymd.tools.packaging  
   module, 257  
 kivymd.tools.packaging.pyinstaller  
   module, 257  
 kivymd.tools.packaging.pyinstaller.hook-kivymd  
   module, 258  
 kivymd.tools.release  
   module, 258  
 kivymd.tools.release.argument\_parser  
   module, 258  
*kivymd.tools.release.git\_commands*  
   module, 260  
 kivymd.tools.release.make\_release  
   module, 260  
 kivymd.tools.release.update\_icons  
   module, 261  
 kivymd.uix  
   module, 262  
*kivymd.uix.backdrop*  
   module, 202  
 kivymd.uix.banner

module, 37  
kivymd.uix.behaviors  
    module, 263  
kivymd.uix.behaviors.backgroundcolorbehavior  
    module, 236  
kivymd.uix.behaviors.elevation  
    module, 238  
kivymd.uix.behaviors.focus\_behavior  
    module, 230  
kivymd.uix.behaviors.hover\_behavior  
    module, 228  
kivymd.uix.behaviors.magic\_behavior  
    module, 234  
kivymd.uix.behaviors.ripplebehavior  
    module, 232  
kivymd.uix.behaviors.toggle\_behavior  
    module, 241  
kivymd.uix.behaviors.touch\_behavior  
    module, 227  
kivymd.uix.bottomnavigation  
    module, 26  
kivymd.uix.bottomsheet  
    module, 59  
kivymd.uix.boxlayout  
    module, 135  
kivymd.uix.button  
    module, 121  
kivymd.uix.card  
    module, 179  
kivymd.uix.carousel  
    module, 263  
kivymd.uix.chip  
    module, 192  
kivymd.uix.datatables  
    module, 209  
kivymd.uix.dialog  
    module, 72  
kivymd.uix.dropdownitem  
    module, 52  
kivymd.uix.expansionpanel  
    module, 94  
kivymd.uix.filemanager  
    module, 195  
kivymd.uix.floatlayout  
    module, 119  
kivymd.uix.gridlayout  
    module, 120  
kivymd.uix.imagelist  
    module, 141  
kivymd.uix.label  
    module, 175  
kivymd.uix.list  
    module, 162  
kivymd.uix.menu  
    module, 105  
kivymd.uix.navigationdrawer  
    module, 86  
kivymd.uix.picker  
    module, 53  
kivymd.uix.progressbar  
    module, 67  
kivymd.uix.refreshlayout  
    module, 145  
kivymd.uix.relativelayout  
    module, 206  
kivymd.uix.screen  
    module, 208  
kivymd.uix.selectioncontrol  
    module, 137  
kivymd.uix.slider  
    module, 159  
kivymd.uix.snackbar  
    module, 32  
kivymd.uix.spinner  
    module, 24  
kivymd.uix.stacklayout  
    module, 207  
kivymd.uix.tab  
    module, 41  
kivymd.uix.taptargetview  
    module, 215  
kivymd.uix.textfield  
    module, 147  
kivymd.uix.toolbar  
    module, 98  
kivymd.uix.tooltip  
    module, 199  
kivymd.uix.useranimationcard  
    module, 83  
kivymd.utils  
    module, 263  
kivymd.utils.asynckivy  
    module, 263  
kivymd.utils.cropimage  
    module, 264  
kivymd.utils.fitimage  
    module, 264  
kivymd.utils.fpsmonitor  
    module, 267  
kivymd.utils.hot\_reload\_viewer  
    module, 267  
kivymd.vendor  
    module, 269  
kivymd.vendor.circleLayout  
    module, 269  
kivymd.vendor.circularTimePicker  
    module, 271

kivymd\_path (in module `kivymd.tools.release.update_icons`), 261

**L**

label (`kivymd.uix.chip.MDChip` attribute), 194

label\_check\_texture\_size ()  
     (`kivymd.toast.kivytoast.kivytoast.Toast`  
     method), 256

label\_text\_color (`kivymd.uix.button.MDFloatingActionButton` attribute), 133

lay\_canvas\_instructions ()  
     (`kivymd.uix.behaviors.ripplebehavior.CircularRipple`  
     method), 234

lay\_canvas\_instructions ()  
     (`kivymd.uix.behaviors.ripplebehavior.CommonRipple`  
     method), 234

lay\_canvas\_instructions ()  
     (`kivymd.uix.behaviors.ripplebehavior.RectangularRipple`  
     method), 234

lay\_canvas\_instructions ()  
     (`kivymd.uix.button.MDRoundFlatButton`  
     method), 131

left\_action (`kivymd.uix.banner.MDBanner` attribute), 40

left\_action\_items  
     (`kivymd.uix.backdrop.MDBackdrop` attribute), 204

left\_action\_items  
     (`kivymd.uix.toolbar.MDToolbar` attribute), 103

left\_action\_items  
     (`kivymd.uix.useranimationcard.ModifiedToolbar` attribute), 85

light\_colors (in module `kivymd.color_definitions`), 20

line\_color (`kivymd.uix.button.MDRectangleFlatButton` attribute), 132

line\_color (`kivymd.uix.textfield.MDTextFieldRound` attribute), 158

line\_color\_focus (`kivymd.uix.textfield.MDTextField` attribute), 157

line\_color\_normal  
     (`kivymd.uix.textfield.MDTextField` attribute), 156

lines (`kivymd.uix.imagelist.SmartTile` attribute), 144

lock\_swiping (`kivymd.uix.tab.MDTabs` attribute), 50

**M**

MagicBehavior (class in `kivymd.uix.behaviors.magic_behavior`), 236

main () (in module `kivymd.tools.release.make_release`), 261

main () (in module `kivymd.tools.release.update_icons`), 262

make\_icon\_definitions () (in module `kivymd.tools.release.update_icons`), 261

map\_number () (in module `kivymd.vendor.circularTimePicker`), 272

max (`kivymd.stiffscroll.StiffScrollEffect` attribute), 253

max (`kivymd.vendor.circularTimePicker.CircularNumberPicker` attribute), 272

max\_friction (`kivymd.stiffscroll.StiffScrollEffect` attribute), 253

max\_height (`kivymd.uix.menu.MDDropdownMenu` attribute), 118

MDBehavior DRAWER\_WIDTH (in module `kivymd.material_resources`), 252

max\_opened\_x (`kivymd.uix.card.MDCardSwipe` attribute), 190

max\_swipe\_x (`kivymd.uix.card.MDCardSwipe` attribute), 190

MDTextBehavior length (`kivymd.uix.textfield.MDTextField` attribute), 156

md\_bg\_color (`kivymd.uix.behaviors.backgroundcolorbehavior.BackgroundColorBehavior` attribute), 237

md\_bg\_color (`kivymd.uix.toolbar.MDToolbar` attribute), 103

md\_icons (in module `kivymd.icon_definitions`), 23

MDActionBottomAppBarButton (class in `kivymd.uix.toolbar`), 103

MDAdaptiveWidget (class in `kivymd.uix`), 262

MDApp (class in `kivymd.app`), 17

MDBackdrop (class in `kivymd.uix.backdrop`), 204

MDBackdropBackLayer (class in `kivymd.uix.backdrop`), 206

MDBackdropFrontLayer (class in `kivymd.uix.backdrop`), 206

MDBackdropToolbar (class in `kivymd.uix.backdrop`), 206

MDBanner (class in `kivymd.uix.banner`), 39

MDBottomAppBar (class in `kivymd.uix.toolbar`), 104

MDBottomNavigation (class in `kivymd.uix.bottomnavigation`), 30

MDBottomNavigationItem (class in `kivymd.uix.bottomnavigation`), 30

MDBottomSheet (class in `kivymd.uix.bottomsheet`), 64

MDBoxLayout (class in `kivymd.uix.boxlayout`), 137

MDCard (class in `kivymd.uix.card`), 188

MDCardSwipe (class in `kivymd.uix.card`), 189

MDCardSwipeFrontBox (class in `kivymd.uix.card`), 191

MDCardSwipeLayerBox (class in `kivymd.uix.card`), 192

MDCarousel (class in `kivymd.uix.carousel`), 263

MDCheckbox (class in `kivymd.uix.selectioncontrol`), 140

MDChip (class in `kivymd.uix.chip`), 194

MDChooseChip (class in `kivymd.uix.chip`), 195

MDCustomBottomSheet (class in `kivymd.uix.card`), 196

*kivymd.uix.bottomsheet), 65*  
MDDataTable (*class in kivymd.uix.datatables*), 210  
MDDatePicker (*class in kivymd.uix.picker*), 58  
MDDialog (*class in kivymd.uix.dialog*), 73  
MDDropDownItem (*class in kivymd.uix.dropdownitem*), 53  
MDDropdownMenu (*class in kivymd.uix.menu*), 117  
MDExpansionPanel (*class in kivymd.uix.expansionpanel*), 96  
MDExpansionPanelOneLine (*class in kivymd.uix.expansionpanel*), 96  
MDExpansionPanelThreeLine (*class in kivymd.uix.expansionpanel*), 96  
MDExpansionPanelTwoLine (*class in kivymd.uix.expansionpanel*), 96  
MDFileManager (*class in kivymd.uix.filemanager*), 198  
MDFillRoundFlatButton (*class in kivymd.uix.button*), 132  
MDFillRoundFlatIconButton (*class in kivymd.uix.button*), 132  
MDFlatButton (*class in kivymd.uix.button*), 131  
MDFloatingActionButton (*class in kivymd.uix.button*), 131  
MDFloatingActionButtonSpeedDial (*class in kivymd.uix.button*), 132  
MDFloatLayout (*class in kivymd.uix.floatlayout*), 120  
MDGridBottomSheet (*class in kivymd.uix.bottomsheet*), 66  
MDGridLayout (*class in kivymd.uix.gridlayout*), 121  
MDIcon (*class in kivymd.uix.label*), 179  
MDIconButton (*class in kivymd.uix.button*), 131  
MDLabel (*class in kivymd.uix.label*), 178  
MDList (*class in kivymd.uix.list*), 171  
MDListBottomSheet (*class in kivymd.uix.bottomsheet*), 65  
MDNavigationDrawer (*class in kivymd.uix.navigationdrawer*), 91  
MDProgressBar (*class in kivymd.uix.progressbar*), 71  
MDRaisedButton (*class in kivymd.uix.button*), 131  
MDRectangleFlatButton (*class in kivymd.uix.button*), 131  
MDRectangleFlatIconButton (*class in kivymd.uix.button*), 132  
MDRelativeLayout (*class in kivymd.uix.relativelayout*), 207  
MDRoundFlatButton (*class in kivymd.uix.button*), 131  
MDRoundFlatIconButton (*class in kivymd.uix.button*), 132  
MDScreen (*class in kivymd.uix.screen*), 209  
MDScrollViewRefreshLayout (*class in kivymd.uix.refreshlayout*), 147  
MDSeparator (*class in kivymd.uix.card*), 188  
MDSlider (*class in kivymd.uix.slider*), 161  
MDSpinner (*class in kivymd.uix.spinner*), 26  
MDStackLayout (*class in kivymd.uix.stacklayout*), 208  
MDSwitch (*class in kivymd.uix.selectioncontrol*), 141  
MDTab (*class in kivymd.uix.bottomnavigation*), 30  
MDTabs (*class in kivymd.uix.tab*), 49  
MDTabsBase (*class in kivymd.uix.tab*), 49  
MDTapTargetView (*class in kivymd.uix.taptargetview*), 223  
MDTextButton (*class in kivymd.uix.button*), 131  
MDTextField (*class in kivymd.uix.textfield*), 155  
MDTextFieldRect (*class in kivymd.uix.textfield*), 155  
MDTextFieldRound (*class in kivymd.uix.textfield*), 157  
MDThemePicker (*class in kivymd.uix.picker*), 59  
MDTimePicker (*class in kivymd.uix.picker*), 58  
MDToggleButton (*class in kivymd.uix.behaviors.toggle\_behavior*), 242  
MDToolbar (*class in kivymd.uix.toolbar*), 103  
MDTooltip (*class in kivymd.uix.tooltip*), 201  
MDTooltipViewClass (*class in kivymd.uix.tooltip*), 202  
MDUserAnimationCard (*class in kivymd.uix.useranimationcard*), 84  
min (*kivymd.stiffscroll.StiffScrollView attribute*), 253  
min (*kivymd.vendor.circularTimePicker.CircularNumberPicker attribute*), 272  
minutes (*kivymd.vendor.circularTimePicker.CircularTimePicker attribute*), 274  
mode (*kivymd.uix.textfield.MDTextField attribute*), 156  
mode (*kivymd.uix.toolbar.MDToolbar attribute*), 103  
ModifiedToolbar (*class in kivymd.uix.useranimationcard*), 85  
module  
    kivymd, 1, 251  
    kivymd.app, 16  
    kivymd.color\_definitions, 18  
    kivymd.factory\_registers, 252  
    kivymd.font\_definitions, 23  
    kivymd.icon\_definitions, 21  
    kivymd.material\_resources, 252  
    kivymd.stiffscroll, 252  
    kivymd.theming, 6  
    kivymd.theming\_dynamic\_text, 252  
    kivymd.toast, 254  
    kivymd.toast.androidtoast, 254  
    kivymd.toast.androidtoast.androidtoast, 254  
    kivymd.toast.kivytoast, 255  
    kivymd.toast.kivytoast.kivytoast, 255  
    kivymd.tools, 257  
    kivymd.tools.packaging, 257

kivymd.tools.packaging.pyinstaller, 257  
kivymd.tools.packaging.pyinstaller.hook, 258  
kivymd.tools.release, 258  
kivymd.tools.release.argument\_parser, 258  
kivymd.tools.release.git\_commands, 260  
kivymd.tools.release.make\_release, 260  
kivymd.tools.release.update\_icons, 261  
kivymd.uix, 262  
kivymd.uix.backdrop, 202  
kivymd.uix.banner, 37  
kivymd.uix.behaviors, 263  
kivymd.uix.behaviors.backgroundcolorbehavior, 236  
kivymd.uix.behaviors.elevation, 238  
kivymd.uix.behaviors.focus\_behavior, 230  
kivymd.uix.behaviors.hover\_behavior, 228  
kivymd.uix.behaviors.magic\_behavior, 234  
kivymd.uix.behaviors.ripplebehavior, 232  
kivymd.uix.behaviors.toggle\_behavior, 241  
kivymd.uix.behaviors.touch\_behavior, 227  
kivymd.uix.bottomnavigation, 26  
kivymd.uix.bottomsheet, 59  
kivymd.uix.boxlayout, 135  
kivymd.uix.button, 121  
kivymd.uix.card, 179  
kivymd.uix.carousel, 263  
kivymd.uix.chip, 192  
kivymd.uix.datatables, 209  
kivymd.uix.dialog, 72  
kivymd.uix.dropdownitem, 52  
kivymd.uix.expansionpanel, 94  
kivymd.uix.filemanager, 195  
kivymd.uix.floatlayout, 119  
kivymd.uix.gridlayout, 120  
kivymd.uix.imagelist, 141  
kivymd.uix.label, 175  
kivymd.uix.list, 162  
kivymd.uix.menu, 105  
kivymd.uix.navigationdrawer, 86  
kivymd.uix.picker, 53  
kivymd.uix.progressbar, 67  
kivymd.uix.refreshlayout, 145  
kivymd.uix.relativelayout, 206  
kivymd.uix.screen, 208  
kivymd.uix.selectioncontrol, 137  
kivymd.uix.slider, 159  
kivymd.uix.snackbar, 32  
kivymd.uix.spinner, 24  
kivymd.uix.stacklayout, 207  
kivymd.uix.tab, 41  
kivymd.uix.taptargetview, 215  
kivymd.uix.textfield, 147  
kivymd.uix.toolbar, 98  
kivymd.uix.tooltip, 199  
kivymd.uix.useranimationcard, 83  
kivymd.utils, 263  
kivymd.utils.asynckivy, 263  
kivymd.utils.cropimage, 264  
kivymd.utils.fitimage, 264  
kivymd.utils.fpsmonitor, 267  
kivymd.utils.hot\_reload\_viewer, 267  
kivymd.vendor, 269  
kivymd.vendor.circleLayout, 269  
kivymd.vendor.circularTimePicker, 271  
month (kivymd.uix.picker.MDDDatePicker attribute), 58  
move\_changelog() (in module kivymd.tools.release.make\_release), 261  
multiples\_of (kivymd.vendor.circularTimePicker.CircularNumberPicker attribute), 272

## N

NavigationLayout (class in kivymd.uix.navigationdrawer), 91  
normal\_color (kivymd.uix.textfield.MDTextFieldRound attribute), 158  
Number (class in kivymd.vendor.circularTimePicker), 272  
number\_at\_pos() (kivymd.vendor.circularTimePicker.CircularNumberPicker method), 273  
number\_format\_string (kivymd.vendor.circularTimePicker.CircularNumberPicker attribute), 273  
number\_size\_factor (kivymd.vendor.circularTimePicker.CircularNumberPicker attribute), 272

## O

ok\_click() (kivymd.uix.picker.MDDDatePicker method), 58  
on\_hint\_text() (kivymd.uix.textfield.MDTextField method), 157  
on\_is\_off() (kivymd.uix.slider.MDSlider method), 161  
on\_rotation\_angle() (kivymd.uix.spinner.MDSpinner method),

26  
on\_action\_button() (kivymd.uix.toolbar.MDToolbar method), 104  
on\_active() (kivymd.uix.selectioncontrol.MDCheckbox method), 141  
on\_active() (kivymd.uix.slider.MDSlider method), 161  
on\_active() (kivymd.uix.spinner.MDSpinner method), 26  
on\_adaptive\_height() (kivymd.uix.MDAdaptiveWidget method), 263  
on\_adaptive\_size() (kivymd.uix.MDAdaptiveWidget method), 263  
on\_adaptive\_width() (kivymd.uix.MDAdaptiveWidget method), 263  
on\_ampm() (kivymd.vendor.circularTimePicker.CircularTimePicker method), 274  
on\_anchor() (kivymd.uix.card.MDCardSwipe method), 191  
on\_any\_event() (kivymd.utils.hot\_reload\_viewer.HotReloadHandler method), 268  
on\_bg\_color\_root\_button() (kivymd.uix.button.MDFloatingActionButtonSpeedDial method), 135  
on\_bg\_color\_stack\_button() (kivymd.uix.button.MDFloatingActionButtonSpeedDial method), 135  
on\_bg\_hint\_color() (kivymd.uix.button.MDFloatingActionButtonSpeedDial method), 135  
on\_carousel\_index() (kivymd.uix.tab.MDTabs method), 51  
on\_check\_press() (kivymd.uix.datatables.MDDatatable method), 215  
on\_close() (kivymd.uix.backdrop.MDBackdrop method), 205  
on\_close() (kivymd.uix.button.MDFloatingActionButtonSpeedDial method), 135  
on\_close() (kivymd.uix.expansionpanel.MDExpansionPanel method), 97  
on\_close() (kivymd.uix.taptargetview.MDTapTargetView method), 226  
on\_color\_active() (kivymd.uix.textfield.MDTextFieldRound method), 159  
on\_color\_icon\_root\_button() (kivymd.uix.button.MDFloatingActionButtonSpeedDial method), 135  
on\_color\_icon\_stack\_button() (kivymd.uix.button.MDFloatingActionButtonSpeedDial method), 135  
method), 135  
on\_color\_mode() (kivymd.uix.textfield.MDTextField method), 157  
on\_data() (kivymd.uix.button.MDFloatingActionButtonSpeedDial method), 135  
on\_description\_text() (kivymd.uix.taptargetview.MDTapTargetView method), 226  
on\_description\_text\_bold() (kivymd.uix.taptargetview.MDTapTargetView method), 227  
on\_description\_text\_size() (kivymd.uix.taptargetview.MDTapTargetView method), 226  
on\_disabled() (kivymd.uix.button.MDFillRoundFlatButton method), 132  
on\_disabled() (kivymd.uix.button.MDRectangleFlatButton method), 131  
on\_disabled() (kivymd.uix.button.MDTextButton method), 131  
on\_dismiss() (kivymd.uix.bottomsheet.MDBottomSheet method), 65  
on\_dismiss() (kivymd.uix.menu.MDDropdownMenu method), 119  
on\_double\_tap() (kivymd.uix.behaviors.touch\_behavior.TouchBehavior method), 228  
draw\_shadow() (kivymd.uix.taptargetview.MDTapTargetView method), 226  
on\_elevation() (kivymd.uix.button.MDFillRoundFlatButton method), 132  
on\_enter() (kivymd.uix.behaviors.focus\_behavior.FocusBehavior method), 231  
enter() (kivymd.uix.behaviors.hover\_behavior.HoverBehavior method), 230  
on\_enter() (kivymd.uix.button.MDFloatingActionButtonSpeedDial method), 135  
on\_enter() (kivymd.uix.menu.MDDropdownMenu method), 119  
on\_enter() (kivymd.uix.tooltip.MDTooltip method), 201  
enter() (kivymd.utils.hot\_reload\_viewer.HotReloadViewer method), 269  
errors\_text\_color() (kivymd.utils.hot\_reload\_viewer.HotReloadViewer method), 269  
on\_focus() (kivymd.uix.textfield.MDTextField method), 157  
on\_focus() (kivymd.uix.textfield.MDTextFieldRound method), 159  
on\_header() (kivymd.uix.backdrop.MDBackdrop method), 205  
on\_hint() (kivymd.uix.slider.MDSlider method), 161  
on\_hint\_animation() (kivymd.uix.button.MDFloatingActionButtonSpeedDial method), 161

method), 135  
 on\_icon() (kivymd.uix.button.MDFloatingActionButtonSpeedDial method), 132  
     method), 135  
 on\_icon() (kivymd.uix.chip.MDChip method), 194  
 on\_icon() (kivymd.uix.toolbar.MDToolbar method), 104  
     104  
 on\_icon\_color() (kivymd.uix.toolbar.MDToolbar method), 104  
     method), 104  
 on\_icon\_left() (kivymd.uix.textfield.MDTextFieldRound method), 159  
     method), 159  
 on\_icon\_left\_color() (kivymd.uix.textfield.MDTextFieldRound method), 159  
     method), 159  
 on\_icon\_right() (kivymd.uix.textfield.MDTextField method), 157  
     method), 157  
 on\_icon\_right\_color() (kivymd.uix.textfield.MDTextField method), 157  
     method), 157  
 on\_label\_text\_color() (kivymd.uix.button.MDFloatingActionButtonSpeedDial method), 135  
     method), 135  
 on\_leave() (kivymd.uix.behaviors.focus\_behavior.FocusBehavior method), 231  
     method), 231  
 on\_leave() (kivymd.uix.behaviors.hover\_behavior.HoverBehavior method), 230  
     method), 230  
 on\_leave() (kivymd.uix.bottomnavigationMDBottomNavigation method), 30  
     method), 30  
 on\_leave() (kivymd.uix.button.MDFloatingActionButtonSpeedDial method), 135  
     method), 135  
 on\_leave() (kivymd.uix.menu.MDDropdownMenu method), 119  
     method), 119  
 on\_leave() (kivymd.uix.tooltip.MDTooltip method), 202  
     202  
 on\_left\_action\_items() (kivymd.uix.backdrop.MDBackdrop method), 205  
     method), 205  
 on\_left\_action\_items() (kivymd.uix.toolbar.MDToolbar method), 104  
     method), 104  
 on\_left\_action\_items() (kivymd.uix.useranimationcard.ModifiedToolbar method), 85  
     method), 85  
 on\_line\_color\_focus() (kivymd.uix.textfield.MDTextField method), 157  
     method), 157  
 on\_long\_touch() (kivymd.uix.behaviors.touch\_behavior.TouchBehavior method), 228  
     method), 228  
 on\_long\_touch() (kivymd.uix.tooltip.MDTooltip method), 201  
     method), 201  
 on\_md\_bg\_color() (kivymd.uix.button.MDFillRoundFlatButton method), 132  
     method), 132  
 on\_md\_bg\_color() (kivymd.uix.button.MDFillRoundFlatIconButton method), 132  
     method), 132  
 on\_md\_bg\_color() (kivymd.uix.button.MDFloatingActionButton method), 131  
     method), 131  
 on\_md\_bg\_color() (kivymd.uix.toolbar.MDToolbar method), 104  
     method), 104  
 on\_md\_mode() (kivymd.uix.toolbar.MDToolbar method), 104  
     method), 104  
 on\_mouse\_pos() (kivymd.uix.behaviors.hover\_behavior.HoverBehavior method), 230  
     method), 230  
 on\_open() (kivymd.toast.kivytoast.KivyToast method), 256  
     method), 256  
 on\_open() (kivymd.uix.backdrop.MDBackdrop method), 205  
     method), 205  
 on\_open() (kivymd.uix.button.MDFloatingActionButtonSpeedDial method), 135  
     method), 135  
 on\_open() (kivymd.uix.dialog.MDDialog method), 82  
     method), 82  
 on\_open() (kivymd.uix.expansionpanel.MDExpansionPanel method), 97  
     method), 97  
 on\_open() (kivymd.uix.taptargetview.MDTapTargetView method), 226  
     method), 226  
 on\_open() (kivymd.uix.useranimationcard.MDUserAnimationCard method), 84  
     method), 84  
 on\_open\_progress()  
 on\_outer\_touch() (kivymd.uix.taptargetview.MDTapTargetView method), 227  
     method), 227  
 on\_outside\_click() (kivymd.uix.taptargetview.MDTapTargetView method), 227  
     method), 227  
 on\_palette() (kivymd.uix.spinner.MDSpinner method), 26  
     method), 26  
 on\_panel\_color() (kivymd.uix.bottomnavigation.MDBottomNavigation method), 31  
     method), 31  
 on\_path() (kivymd.utils.hot\_reload\_viewer.HotReloadViewer method), 269  
     method), 269  
 on\_press() (kivymd.uix.button.MDTextButton method), 131  
     method), 131  
 on\_radius() (kivymd.uix.card.MDCard method), 189  
     method), 189  
 on\_ref\_press() (kivymd.uix.tab.MDTabs method), 274  
     method), 274  
 on\_release() (kivymd.uix.menu.MDDropdownMenu

method), 119  
on\_resize () (kivymd.uix.bottomnavigation.MDBottomNavigation method), 53  
method), 31  
on\_right\_action\_items ()  
(kivymd.uix.toolbar.MDToolbar method), 104  
on\_row\_press () (kivymd.uix.datatables.MDDDataTable method), 215  
on\_selected () (kivymd.vendor.circularTimePicker.CircularNumberPicker method), 273  
on\_selected () (kivymd.vendor.circularTimePicker.CircularTimePicker method), 274  
on\_show\_off () (kivymd.uix.slider.MDSlider method), 161  
on\_size () (kivymd.uix.selectioncontrol.MDSwitch method), 141  
on\_slide\_complete ()  
(kivymd.uix.carousel.MDCarousel method), 263  
on\_slide\_progress ()  
(kivymd.uix.carousel.MDCarousel method), 263  
on\_slide\_progress () (kivymd.uix.tab.MDTabs method), 51  
on\_stars () (kivymd.uix.imagelist.SmartTileWithStar method), 145  
on\_state () (kivymd.uix.selectioncontrol.MDCheckbox method), 141  
on\_swipe\_complete ()  
(kivymd.uix.card.MDCardSwipe method), 191  
on\_tab\_press () (kivymd.uix.bottomnavigation.MDBottomNavigation method), 30  
on\_tab\_press () (kivymd.uix.bottomnavigation.MDTab method), 30  
on\_tab\_release () (kivymd.uix.bottomnavigation.MDTab method), 30  
on\_tab\_switch () (kivymd.uix.tab.MDTabs method), 51  
on\_tab\_touch\_down ()  
(kivymd.uix.bottomnavigation.MDTab method), 30  
on\_tab\_touch\_move ()  
(kivymd.uix.bottomnavigation.MDTab method), 30  
on\_tab\_touch\_up ()  
(kivymd.uix.bottomnavigation.MDTab method), 30  
on\_target\_radius ()  
(kivymd.uix.taptargetview.MDTapTargetView method), 227  
on\_target\_touch ()  
(kivymd.uix.taptargetview.MDTapTargetView method), 227  
on\_text () (kivymd.uix.dropdownitem.MDDropDownItem method), 157  
on\_text () (kivymd.uix.tab.MDTabsBase method), 49  
on\_text () (kivymd.uix.textfield.MDTextField method), 157  
on\_text\_color () (kivymd.uix.label.MDLabel method), 179  
on\_text\_color\_active ()  
on\_text\_validate ()  
(kivymd.uix.textfield.MDTextField method), 157  
on\_theme\_style () (kivymd.theming.ThemeManager method), 15  
on\_theme\_text\_color ()  
(kivymd.uix.label.MDLabel method), 179  
on\_time\_list () (kivymd.vendor.circularTimePicker.CircularTimePicker method), 274  
on\_title\_text () (kivymd.uix.taptargetview.MDTapTargetView method), 227  
on\_title\_text\_bold ()  
(kivymd.uix.taptargetview.MDTapTargetView method), 227  
on\_title\_text\_size ()  
(kivymd.uix.taptargetview.MDTapTargetView method), 227  
on\_touch\_down () (kivymd.toast.kivytoast.Toast method), 257  
on\_touch\_down () (kivymd.uix.behaviors.ripplebehavior.CommonRipple method), 234  
on\_touch\_down () (kivymd.uix.card.MDCardSwipe method), 191  
on\_touch\_down () (kivymd.uix.chip.MDChip method), 194  
on\_touch\_down () (kivymd.uix.list.ContainerSupport method), 173  
on\_touch\_down () (kivymd.uix.menu.MDDropdownMenu method), 119  
on\_touch\_down () (kivymd.uix.navigationdrawer.MDNavigationDrawer method), 93  
on\_touch\_down () (kivymd.uix.slider.MDSlider method), 161  
on\_touch\_down () (kivymd.uix.useranimationcard.MDUserAnimationCard method), 84  
on\_touch\_down () (kivymd.vendor.circularTimePicker.CircularNumberPicker method), 273  
on\_touch\_down () (kivymd.vendor.circularTimePicker.CircularTimePicker method), 275  
on\_touch\_move () (kivymd.uix.behaviors.ripplebehavior.CommonRipple method), 234

on\_touch\_move() (*kivymd.uix.card.MDCardSwipe method*), 191  
 on\_touch\_move() (*kivymd.uix.list.ContainerSupport method*), 173  
 on\_touch\_move() (*kivymd.uix.menu.MDDropdownMenu method*), 119  
 on\_touch\_move() (*kivymd.uix.navigationdrawer.MDNavigationDrawer method*), 93  
 on\_touch\_move() (*kivymd.uix.useranimationcard.MDUserAnimationCard method*), 84  
 on\_touch\_move() (*kivymd.vendor.circularTimePicker.CircularNumberPicker method*), 273  
 on\_touch\_up() (*kivymd.uix.behaviors.ripplebehavior.CommonRipple method*), 234  
 on\_touch\_up() (*kivymd.uix.card.MDCardSwipe method*), 191  
 on\_touch\_up() (*kivymd.uix.list.ContainerSupport method*), 173  
 on\_touch\_up() (*kivymd.uix.menu.MDDropdownMenu method*), 119  
 on\_touch\_up() (*kivymd.uix.navigationdrawer.MDNavigationDrawer method*), 93  
 on\_touch\_up() (*kivymd.uix.refreshlayout.MDScrollViewRefreshLayout method*), 147  
 on\_touch\_up() (*kivymd.uix.slider.MDSlider method*), 162  
 on\_touch\_up() (*kivymd.uix.useranimationcard.MDUserAnimationCard method*), 85  
 on\_touch\_up() (*kivymd.vendor.circularTimePicker.CircularNumberPicker method*), 273  
 on\_touch\_up() (*kivymd.vendor.circularTimePicker.CircularTimePicker method*), 275  
 on\_triple\_tap() (*kivymd.uix.behaviors.touch\_behavior.TouchBehavior method*), 228  
 on\_type() (*kivymd.uix.navigationdrawer.MDNavigationDrawer method*), 93  
 on\_value() (*kivymd.stiffscroll.StiffScrollEffect method*), 253  
 on\_value\_normalized() (*kivymd.uix.slider.MDSlider method*), 161  
 on\_width() (*kivymd.uix.textfield.MDTextField method*), 157  
 OneLineAvatarIconListItem (*class in kivymd.uix.list*), 174  
 OneLineAvatarListItem (*class in kivymd.uix.list*), 173  
 OneLineIconListItem (*class in kivymd.uix.list*), 173  
 OneLineListMenuItem (*class in kivymd.uix.list*), 173  
 OneLineRightIconListItem (*class in kivymd.uix.list*), 173  
 open() (*kivymd.uix.backdrop.MDBackdrop method*), 205  
 open() (*kivymd.uix.bottomsheetMDBottomSheet method*), 64  
 open() (*kivymd.uix.menu.MDDropdownMenu method*), 119  
 open\_card() (*kivymd.uix.card.MDCardSwipe method*), 191  
 open\_panel() (*kivymd.uix.expansionpanel.MDExpansionPanel method*), 97  
 open\_progress (*kivymd.uix.card.MDCardSwipe attribute*), 189  
 open\_progress (*kivymd.uix.navigationdrawer.MDNavigationDrawer attribute*), 118  
 open\_stack() (*kivymd.uix.button.MDFloatingActionButtonSpeedDial attribute*), 135  
 opening\_time (*kivymd.uix.button.MDFloatingActionButtonSpeedDial attribute*), 134  
 opening\_time (*kivymd.uix.card.MDCardSwipe attribute*), 190  
 opening\_time (*kivymd.uix.expansionpanel.MDExpansionPanel attribute*), 96  
 opening\_time (*kivymd.uix.menu.MDDropdownMenu attribute*), 118  
 opening\_time (*kivymd.uix.navigationdrawer.MDNavigationDrawer attribute*), 118  
 opening\_time\_button\_rotation (*kivymd.uix.button.MDFloatingActionButtonSpeedDial attribute*), 134  
 opening\_transition (*kivymd.uix.banner.MDBanner attribute*), 48  
 opening\_transition (*kivymd.uix.button.MDFloatingActionButtonSpeedDial attribute*), 133  
 opening\_transition (*kivymd.uix.card.MDCardSwipe attribute*), 189  
 opening\_transition (*kivymd.uix.expansionpanel.MDExpansionPanel attribute*), 96  
 opening\_transition (*kivymd.uix.menu.MDDropdownMenu attribute*), 118  
 opening\_transition (*kivymd.uix.navigationdrawer.MDNavigationDrawer attribute*), 93  
 opening\_transition\_button\_rotation (*kivymd.uix.button.MDFloatingActionButtonSpeedDial attribute*), 134  
 opposite\_bg\_dark (*kivymd.theming.ThemeManager attribute*), 12  
 opposite\_bg\_darkest (*kivymd.theming.ThemeManager attribute*), 12  
 opposite\_bg\_light (*kivymd.theming.ThemeManager attribute*), 13  
 opposite\_bg\_normal (*kivymd.theming.ThemeManager attribute*)

(*kivymd.theming.ThemeManager* attribute), 13  
opposite\_colors (*kivymd.theming.ThemableBehavior* attribute), 16  
opposite\_disabled\_hint\_text\_color (*kivymd.theming.ThemeManager* attribute), 14  
opposite\_divider\_color (*kivymd.theming.ThemeManager* attribute), 13  
opposite\_icon\_color (*kivymd.theming.ThemeManager* attribute), 14  
opposite\_secondary\_text\_color (*kivymd.theming.ThemeManager* attribute), 13  
opposite\_text\_color (*kivymd.theming.ThemeManager* attribute), 13  
orientation (*kivymd.uix.progressbar.MDProgressBar* attribute), 71  
outer\_circle\_alpha (*kivymd.uix.taptargetview.MDTapTargetView* attribute), 224  
outer\_circle\_color (*kivymd.uix.taptargetview.MDTapTargetView* attribute), 223  
outer\_radius (*kivymd.uix.taptargetview.MDTapTargetView* attribute), 223  
outer\_radius\_hint (*kivymd.vendor.circleLayout.CircularLayout* attribute), 270  
over\_widget (*kivymd.uix.bannerMDBanner* attribute), 40  
overlap (*kivymd.uix.imagelist.SmartTile* attribute), 144

**P**

padding (*kivymd.uix.backdrop.MDBackdrop* attribute), 204  
padding (*kivymd.uix.snackbar.Snackbar* attribute), 37  
padding (*kivymd.uix.tooltip.MDTooltip* attribute), 201  
padding (*kivymd.vendor.circleLayout.CircularLayout* attribute), 270  
pagination\_menu\_height (*kivymd.uix.datatables.MDDDataTable* attribute), 214  
pagination\_menu\_pos (*kivymd.uix.datatables.MDDDataTable* attribute), 213  
palette (*in module kivymd.color\_definitions*), 20  
palette (*kivymd.uix.spinner.MDSpinner* attribute), 26  
panel\_cls (*kivymd.uix.expansionpanel.MDExpansionPanel* attribute), 97  
panel\_color (*kivymd.uix.bottomnavigation.TabbedPanelBase* attribute), 30  
parent\_background (*kivymd.uix.label.MDLabel* attribute), 179  
parse\_args () (*kivymd.tools.release.argument\_parser.ArgumentParser* method), 259  
path (*in module kivymd*), 251

path (*kivymd.utils.hot\_reload\_viewer.HotReloadViewer* attribute), 269  
path\_to\_avatar (*kivymd.uix.useranimationcard.MDUserAnimationCard* attribute), 84  
path\_to\_avatar (*kivymd.uix.useranimationcard.UserAnimationCard* attribute), 85  
picker (*kivymd.vendor.circularTimePicker.CircularTimePicker* attribute), 274  
pos\_for\_number () (*kivymd.vendor.circularTimePicker.CircularNumber* method), 273  
position (*kivymd.uix.menu.MDDropdownMenu* attribute), 118  
prepare\_mask () (*in module kivymd.utils.cropimage*), 264  
preview (*kivymd.uix.filemanager.MDFFileManager* attribute), 198  
previous\_tab (*kivymd.uix.bottomnavigation.TabbedPanelBase* attribute), 30  
primary\_color (*kivymd.theming.ThemeManager* attribute), 9  
primary\_dark (*kivymd.theming.ThemeManager* attribute), 10  
primary\_dark (*kivymd.vendor.circularTimePicker.CircularTimePicker* attribute), 273  
primary\_dark\_hue (*kivymd.theming.ThemeManager* attribute), 9  
primary\_hue (*kivymd.theming.ThemeManager* attribute), 7  
primary\_light (*kivymd.theming.ThemeManager* attribute), 9  
primary\_light\_hue (*kivymd.theming.ThemeManager* attribute), 8  
primary\_palette (*kivymd.theming.ThemeManager* attribute), 7  
propagate\_touch\_to\_touchable\_widgets () (*kivymd.uix.list.ContainerSupport* method), 173

**R**

r (*in module kivymd.factory\_registers*), 252  
r (*kivymd.uix.behaviors.backgroundcolorbehavior.BackgroundColorBehavior* attribute), 237  
radio\_icon\_down (*kivymd.uix.selectioncontrol.MDCheckbox* attribute), 140  
radio\_icon\_normal (*kivymd.uix.selectioncontrol.MDCheckbox* attribute), 140  
radius (*kivymd.uix.backdrop.MDBackdrop* attribute), 205  
radius (*kivymd.uix.behaviors.backgroundcolorbehavior.BackgroundColorBehavior* attribute), 237  
radiusParas (*kivymd.uix.bottomsheet.MDBottomSheet* attribute), 64  
radius (*kivymd.uix.chip.MDChip* attribute), 194

radius (*kivymd.uix.dialog.MDDialog* attribute), 75  
 radius (*kivymd.utils.fitimage.FitImage* attribute), 266  
 radius\_from (*kivymd.uix.bottomsheetMDBottomSheet* attribute), 64  
 radius\_hint (*kivymd.vendor.circleLayout.CircularLayout* attribute), 270  
 range (*kivymd.vendor.circularTimePicker.CircularNumberPicker* attribute), 272  
 re\_additional\_icons (in module *kivymd.tools.release.update\_icons*), 261  
 re\_icon\_definitions (in module *kivymd.tools.release.update\_icons*), 261  
 re\_icons\_json (in module *kivymd.tools.release.update\_icons*), 261  
 re\_quote\_keys (in module *kivymd.tools.release.update\_icons*), 261  
 re\_version (in module *kivymd.tools.release.update\_icons*), 261  
 re\_version\_in\_file (in module *kivymd.tools.release.update\_icons*), 261  
 RectangularElevationBehavior (class in *kivymd.uix.behaviors.elevation*), 240  
 RectangularRippleBehavior (class in *kivymd.uix.behaviors.ripplebehavior*), 234  
 refresh\_done () (*kivymd.uix.refreshlayout.MDScrollViewRefreshLayout* method), 147  
 refresh\_tabs () (*kivymd.uix.bottomnavigation.MDBottomNavigation* method), 31  
 RefreshSpinner (class in *kivymd.uix.refreshlayout*), 147  
 release (in module *kivymd*), 251  
 reload() (*kivymd.uix.imagelist.SmartTile* method), 144  
 remove\_notch () (*kivymd.uix.toolbar.MDToolbar* method), 104  
 remove\_shadow () (*kivymd.uix.toolbar.MDToolbar* method), 104  
 remove\_tooltip () (*kivymd.uix.tooltip.MDTooltip* method), 201  
 remove\_widget () (*kivymd.uix.bottomnavigation.MDBottomNavigation* method), 31  
 remove\_widget () (*kivymd.uix.list.ContainerSupport* method), 173  
 remove\_widget () (*kivymd.uix.list.MDList* method), 171  
 remove\_widget () (*kivymd.uix.tab.MDTabs* method), 51  
 replace\_in\_file () (in module *kivymd.tools.release.make\_release*), 260  
 required (*kivymd.uix.textfield.MDTextField* attribute), 156  
 resize\_content\_layout () (*kivymd.uix.bottomsheetMDBottomSheet* method), 65  
 reversed (*kivymd.uix.progressbar.MDProgressBar* attribute), 71  
 rgb\_to\_hex () (in module *kivymd.vendor.circularTimePicker*), 272  
 right\_action (*kivymd.uix.banner.MDBanner* attribute), 40  
 right\_action\_items (*kivymd.uix.backdrop.MDBackdrop* attribute), 204  
 right\_action\_items (*kivymd.uix.toolbar.MDToolbar* attribute), 103  
 right\_pad (*kivymd.uix.button.MDFloatingActionButtonSpeedDial* attribute), 133  
 RightContent (class in *kivymd.uix.menu*), 117  
 ripple\_alpha (*kivymd.uix.behaviors.ripplebehavior.CommonRipple* attribute), 233  
 ripple\_behavior (*kivymd.uix.card.MDCard* attribute), 189  
 ripple\_color (*kivymd.theming.ThemeManager* attribute), 14  
 ripple\_color (*kivymd.uix.behaviors.ripplebehavior.CommonRipple* attribute), 233  
 ripple\_duration\_in\_fast  
 ripple\_duration\_in\_slow (*kivymd.uix.behaviors.ripplebehavior.CommonRipple* attribute), 233  
 ripple\_duration\_out (*kivymd.uix.behaviors.ripplebehavior.CommonRipple* attribute), 234  
 ripple\_func\_in (*kivymd.uix.behaviors.ripplebehavior.CommonRipple* attribute), 234  
 ripple\_func\_out (*kivymd.uix.behaviors.ripplebehavior.CommonRipple* attribute), 234  
 ripple\_rad\_default (*kivymd.uix.behaviors.ripplebehavior.CommonRipple* attribute), 233  
 ripple\_scale (*kivymd.uix.behaviors.ripplebehavior.CommonRipple* attribute), 233  
 ripple\_scale (*kivymd.uix.behaviors.ripplebehavior.RectangularRipple* attribute), 234  
 root\_layout (*kivymd.uix.refreshlayout.MDScrollViewRefreshLayout* attribute), 147  
 rotation\_root\_button (*kivymd.uix.button.MDFloatingActionButtonSpeedDial* attribute), 133  
 round (*kivymd.uix.toolbar.MDToolbar* attribute), 103  
 row\_data (*kivymd.uix.datatables.MDDDataTable* attribute), 211  
 rows\_num (*kivymd.uix.datatables.MDDDataTable*

attribute), 213  
run\_pre\_commit() (in module kivymd.tools.release.make\_release), 260  
running\_away() (kivymd.uix.progressbar.MDProgressBar method), 71  
running\_duration (kivymd.uix.progressbar.MDProgressBar attribute), 71  
running\_transition (kivymd.uix.progressbar.MDProgressBar attribute), 71

S

scale (kivymd.vendor.circularTimePicker.CircularNumberPicker attribute), 273  
screen (kivymd.uix.bottomsheet.MDCustomBottomSheet attribute), 65  
scrim\_alpha\_transition (kivymd.uix.navigationdrawer.MDNavigationDrawer attribute), 92  
scrim\_color (kivymd.uix.navigationdrawer.MDNavigationDrawer attribute), 92  
scroll (kivymd.stiffscroll.StiffScrollEffect attribute), 253  
search (kivymd.uix.filemanager.MDFFileManager attribute), 198  
secondary\_font\_style (kivymd.uix.list.BaseListItem attribute), 172  
secondary\_text (kivymd.uix.list.BaseListItem attribute), 172  
secondary\_text\_color (kivymd.theming.ThemeManager attribute), 13  
secondary\_text\_color (kivymd.uix.list.BaseListItem attribute), 172  
secondary\_theme\_text\_color (kivymd.uix.list.BaseListItem attribute), 172  
sel\_day (kivymd.uix.picker.MDDatePicker attribute), 58  
sel\_month (kivymd.uix.picker.MDDatePicker attribute), 58  
sel\_year (kivymd.uix.picker.MDDatePicker attribute), 58  
select\_dir\_or\_file() (kivymd.uix.filemanager.MDFFileManager method), 199  
select\_directory\_on\_press\_button() (kivymd.uix.filemanager.MDFFileManager method), 199  
select\_path (kivymd.uix.filemanager.MDFFileManager attribute), 198  
selected (kivymd.vendor.circularTimePicker.CircularNumberPicker attribute), 272  
selected\_chip\_color (kivymd.uix.chip.MDChip attribute), 194

selected\_color (kivymd.uix.menu.MDDropdownMenu attribute), 117  
selected\_color (kivymd.uix.selectioncontrol.MDCheckbox attribute), 140  
selector\_alpha (kivymd.vendor.circularTimePicker.CircularNumberPicker attribute), 272  
selector\_alpha (kivymd.vendor.circularTimePicker.CircularTimePicker attribute), 274  
selector\_color (kivymd.vendor.circularTimePicker.CircularNumberPicker attribute), 272  
selector\_color (kivymd.vendor.circularTimePicker.CircularTimePicker attribute), 274

set\_bg\_color\_items() (kivymd.uix.menu.MDDropdownMenu method), 118  
set\_chevron\_down() (kivymd.uix.expansionpanel.MDExpansionPanel method), 97  
set\_chevron\_up() (kivymd.uix.expansionpanel.MDExpansionPanel method), 97  
set\_clearcolor (kivymd.theming.ThemeManager attribute), 14  
set\_clearcolor\_by\_theme\_style() (kivymd.theming.ThemeManager method), 15  
set\_date() (kivymd.uix.picker.MDDatePicker method), 58  
set\_item() (kivymd.uix.dropdownitem.MDDropDownItem method), 53  
set\_left\_action() (kivymd.uix.banner.MDBanner method), 40  
set\_menu\_properties() (kivymd.uix.menu.MDDropdownMenu method), 118  
set\_month\_day() (kivymd.uix.picker.MDDatePicker method), 58  
set\_normal\_height() (kivymd.uix.dialog.MDDialog method), 82  
set\_notch() (kivymd.uix.toolbar.MDToolbar method), 104  
set\_objects\_labels() (kivymd.uix.textfield.MDTextField method), 157  
set\_pos\_bottom\_buttons() (kivymd.uix.button.MDFloatingActionButtonSpeedDial method), 135  
set\_pos\_labels() (kivymd.uix.button.MDFloatingActionButtonSpeedDial method), 135  
set\_pos\_root\_button() (kivymd.uix.button.MDFloatingActionButtonSpeedDial method), 135  
set\_right\_action() (kivymd.uix.banner.MDBanner method), 40

set\_selected\_widget ()  
     (*kivymd.uix.picker.MDDatePicker* method), 58

set\_shadow ()  
     (*kivymd.uix.toolbar.MDToolbar* method), 104

set\_spinner ()  
     (*kivymd.uix.refreshlayout.RefreshSpinner* method), 147

set\_state ()  
     (*kivymd.uix.navigationdrawer.MDNavigationDrawer* method), 93

set\_time ()  
     (*kivymd.uix.picker.MDTimePicker* method), 58

set\_time ()  
     (*kivymd.vendor.circularTimePicker.CircularTimePicker* method), 274

set\_type\_banner ()  
     (*kivymd.uix.bannerMDBanner* method), 40

shake ()  
     (*kivymd.uix.behaviors.magic\_behavior.MagicBehavior* method), 236

sheet\_list (*kivymd.uix.bottomsheet.MDListBottomSheet* attribute), 66

show ()  
     (*kivymd.uix.bannerMDBanner* method), 40

show ()  
     (*kivymd.uix.filemanager.MDFFileManager* method), 199

show ()  
     (*kivymd.uix.snackbar.Snackbar* method), 37

show\_error ()  
     (*kivymd.utils.hot\_reload\_viewer.HotReloadViewer* method), 269

show\_hidden\_files  
     (*kivymd.uix.filemanager.MDFFileManager* attribute), 198

show\_off (*kivymd.uix.slider.MDSlider* attribute), 161

shown\_items (*kivymd.vendor.circularTimePicker.CircularNumberPicker* attribute), 273

shrink ()  
     (*kivymd.uix.behaviors.magic\_behavior.MagicBehavior* method), 236

size\_factor (*kivymd.vendor.circularTimePicker.Number* attribute), 272

sleep ()  
     (in module *kivymd.utils.asynckivy*), 264

SmartTile (class in *kivymd.uix.imagelist*), 144

SmartTileWithLabel (class in *kivymd.uix.imagelist*), 144

SmartTileWithStar (class in *kivymd.uix.imagelist*), 145

Snackbar (class in *kivymd.uix.snackbar*), 36

sort (*kivymd.uix.datatables.MDDDataTable* attribute), 212

sort\_by (*kivymd.uix.filemanager.MDFFileManager* attribute), 199

sort\_by\_desc (*kivymd.uix.filemanager.MDFFileManager* attribute), 199

source (*kivymd.uix.bottomsheet.GridBottomSheetItem* attribute), 66

source (*kivymd.uix.imagelist.SmartTile* attribute), 144

source (*kivymd.uix.label.MDIcon* attribute), 179

source (*kivymd.utils.fitimage.FitImage* attribute), 266

specific\_secondary\_text\_color  
     (*kivymd.uix.behaviors.backgroundcolorbehavior.SpecficBackgroundattribute*), 238

specific\_text\_color  
     (*kivymd.uix.behaviors.backgroundcolorbehavior.SpecficBackgroundattribute*), 238

specificBackgroundColorBehavior (class in *kivymd.uix.behaviors.backgroundcolorbehavior*), 237

spinner\_color (*kivymd.uix.refreshlayout.RefreshSpinner* attribute), 147

standard\_increment

start ()  
     (in module *kivymd.utils.asynckivy*), 264

stars (*kivymd.uix.imagelist.SmartTileWithStar* attribute), 145

start ()  
     (*kivymd.stiffscroll.StiffScrollView* method), 254

start ()  
     (*kivymd.uix.progressbar.MDProgressBar* method), 71

start ()  
     (*kivymd.uix.taptargetview.MDTapTargetView* method), 226

start ()  
     (*kivymd.utils.fpsmonitor.FpsMonitor* method), 267

startAngle (*kivymd.vendor.circleLayout.CircularLayout* attribute), 270

start\_anim\_spinner ()  
     (*kivymd.uix.refreshlayout.RefreshSpinner* method), 147

start\_ripple ()  
     (*kivymd.uix.behaviors.ripplebehavior.CommonRipple* method), 234

state (*kivymd.uix.button.MDFloatingActionButtonSpeedDial* attribute), 134

state (*kivymd.uix.card.MDCardSwipe* attribute), 190

state (*kivymd.uix.navigationdrawer.MDNavigationDrawer* attribute), 92

state (*kivymd.uix.taptargetview.MDTapTargetView* attribute), 226

status (*kivymd.uix.navigationdrawer.MDNavigationDrawer* attribute), 92

StiffScrollView (class in *kivymd.stiffscroll*), 253

stop ()  
     (*kivymd.stiffscroll.StiffScrollView* method), 254

stop ()  
     (*kivymd.uix.progressbar.MDProgressBar* method), 71

stop ()  
     (*kivymd.uix.taptargetview.MDTapTargetView* method), 226

stop\_on\_outer\_touch  
     (*kivymd.uix.taptargetview.MDTapTargetView* attribute), 226

stop\_on\_target\_touch  
     (*kivymd.uix.taptargetview.MDTapTargetView* attribute), 226

swipe\_distance (*kivymd.uix.card.MDCardSwipe* attribute), 190

swipe\_distance (*kivymd.uix.navigationdrawer.MDNavigationDrawer* attribute), 92

attribute), 92  
swipe\_edge\_width (kivymd.uix.navigationdrawer.MDNavigationDrawer attribute), 92  
switch\_tab () (kivymd.uix.bottonnavigation.MDBottomNavigation method), 31  
switch\_tab () (kivymd.uix.tab.MDTabs method), 50

**T**

tab\_bar\_height (kivymd.uix.tab.MDTabs attribute), 49  
tab\_header (kivymd.uix.bottonnavigation.MDBottomNavigation attribute), 31  
tab\_indicator\_anim (kivymd.uix.tab.MDTabs attribute), 49  
tab\_indicator\_height (kivymd.uix.tab.MDTabs attribute), 49  
tab\_label (kivymd.uix.tab.MDTabsBase attribute), 49  
TabbedPanelBase (class in kivymd.uix.bottonnavigation), 30  
tabs (kivymd.uix.bottonnavigation.TabbedPanelBase attribute), 30  
target\_circle\_color (kivymd.uix.taptargetview.MDTapTargetView attribute), 225  
target\_radius (kivymd.uix.taptargetview.MDTapTargetView attribute), 224  
target\_widget (kivymd.stiffscroll.StiffScrollEffect attribute), 253  
temp\_font\_path (in module kivymd.tools.release.update\_icons), 261  
temp\_path (in module kivymd.tools.release.update\_icons), 261  
temp\_preview\_path (in module kivymd.tools.release.update\_icons), 261  
temp\_repo\_path (in module kivymd.tools.release.update\_icons), 261  
tertiary\_font\_style (kivymd.uix.list.BaseListItem attribute), 172  
tertiary\_text (kivymd.uix.list.BaseListItem attribute), 172  
tertiary\_text\_color (kivymd.uix.list.BaseListItem attribute), 172  
tertiary\_theme\_text\_color (kivymd.uix.list.BaseListItem attribute), 172  
text (kivymd.uix.banner.MDBanner attribute), 40  
text (kivymd.uix.bottonnavigation.MDTab attribute), 30  
text (kivymd.uix.dialog.MDDialog attribute), 74  
text (kivymd.uix.dropdownitem.MDDropDownItem attribute), 53  
text (kivymd.uix.imagelist.SmartTileWithLabel attribute), 145  
text (kivymd.uix.label.MDLabel attribute), 178  
text (kivymd.uix.list.BaseListItem attribute), 171

text (kivymd.uix.menu.RightContent attribute), 117  
text (kivymd.uix.snackbar.Snackbar attribute), 36  
text (kivymd.uix.tab.MDTabsBase attribute), 49  
text (kivymd.utils.hot\_reload\_viewer.HotReloadErrorText attribute), 268  
text\_color (kivymd.theming.ThemeManager attribute), 13  
text\_color (kivymd.uix.button.MDFillRoundFlatButton attribute), 132  
text\_color (kivymd.uix.chip.MDChip attribute), 194  
text\_color (kivymd.uix.label.MDLabel attribute), 179  
text\_color (kivymd.uix.list.BaseListItem attribute), 171  
text\_color\_active (kivymd.uix.bottonnavigation.MDBottomNavigation attribute), 31  
text\_color\_active (kivymd.uix.tab.MDTabs attribute), 50  
text\_color\_normal (kivymd.uix.bottonnavigation.MDBottomNavigation attribute), 31  
text\_color\_normal (kivymd.uix.tab.MDTabs attribute), 50  
text\_colors (in module kivymd.color\_definitions), 20  
ThemableBehavior (class in kivymd.theming), 15  
theme\_cls (kivymd.app.MDApp attribute), 17  
theme\_cls (kivymd.theming.ThemableBehavior attribute), 15  
theme\_colors (in module kivymd.color\_definitions), 20  
theme\_font\_styles (in module kivymd.font\_definitions), 23  
theme\_style (kivymd.theming.ThemeManager attribute), 10  
theme\_text\_color (kivymd.uix.label.MDLabel attribute), 178  
theme\_text\_color (kivymd.uix.list.BaseListItem attribute), 172  
ThemeManager (class in kivymd.theming), 7  
ThreeLineAvatarIconListItem (class in kivymd.uix.list), 174  
ThreeLineAvatarListItem (class in kivymd.uix.list), 173  
ThreeLineIconListItem (class in kivymd.uix.list), 173  
ThreeLineList Item (class in kivymd.uix.list), 173  
ThreeLineRightIconListItem (class in kivymd.uix.list), 174  
thumb\_color (kivymd.uix.selectioncontrol.MDSwitch attribute), 141  
thumb\_color (kivymd.uix.slider.MDSlider attribute), 161  
thumb\_color\_disabled



update\_action\_bar\_text\_colors() (kivymd.uix.useranimationcard.ModifiedToolbar method), 85  
update\_cal\_matrix() (kivymd.uix.picker.MDDatePicker method), 58  
update\_color() (kivymd.uix.selectioncontrol.MDCheckbox method), 141  
update\_font\_style() (kivymd.uix.label.MDLabel method), 179  
update\_fps() (kivymd.utils.fpsmonitor.FpsMonitor method), 267  
update\_height() (kivymd.uix.dialog.MDDialog method), 82  
update\_icon() (kivymd.uix.selectioncontrol.MDCheckbox method), 141  
update\_icons() (in module kivymd.tools.release.update\_icons), 262  
update\_init\_py() (in module kivymd.tools.release.make\_release), 260  
update\_md\_bg\_color() (kivymd.uix.button.MDFillRoundFlatButton method), 132  
update\_md\_bg\_color() (kivymd.uix.button.MDFillRoundFlatIconButton method), 132  
update\_md\_bg\_color() (kivymd.uix.button.MDRectangleFlatButton method), 131  
update\_md\_bg\_color() (kivymd.uix.button.MDRoundFlatButton method), 131  
update\_pos() (kivymd.uix.navigationdrawer.NavigationLayout method), 91  
update\_primary\_color() (kivymd.uix.selectioncontrol.MDCheckbox method), 141  
update\_readme() (in module kivymd.tools.release.make\_release), 260  
update\_scrim\_rectangle() (kivymd.uix.navigationdrawer.NavigationLayout method), 91  
update\_status() (kivymd.uix.navigationdrawer.MDNavigationDrawer method), 93  
update\_velocity() (kivymd.stiffscroll.StiffScrollEffect method), 253  
updated\_interval (kivymd.utils.fpsmonitor.FpsMonitor attribute), 267  
url (in module kivymd.tools.release.update\_icons), 261  
use\_access (kivymd.uix.filemanager.MDFileManager attribute), 198  
use\_pagination (kivymd.uix.datatables.MDDatatable attribute), 212

V

user\_name (kivymd.uix.useranimationcard.MDUserAnimationCard attribute), 84  
user\_name (kivymd.uix.useranimationcard.UserAnimationCard attribute), 85  
UserAnimationCard (class in kivymd.uix.useranimationcard), 85

W

value\_transparent (kivymd.uix.bottomsheet.MDBottomSheet attribute), 64  
ver\_growth (kivymd.uix.menu.MDDropdownMenu attribute), 118  
vertical\_pad (kivymd.uix.bannerMDBanner attribute), 40

X

widget (kivymd.uix.taptargetview.MDTapTargetView attribute), 223  
widget\_position (kivymd.uix.taptargetview.MDTapTargetView attribute), 226  
width\_mult (kivymd.uix.menu.MDDropdownMenu attribute), 117  
wobble() (kivymd.uix.behaviors.magic\_behavior.MagicBehavior method), 236

Y

xrange() (in module kivymd.vendor.circularTimePicker), 272

year (kivymd.uix.picker.MDDatePicker attribute), 58