# Kivy

[YTB](https://www.youtube.com/watch?v=LRXo0juuTrw&list=PLhTjy8cBISEoQQLZ9IBlVlr4WjVoStmy-)

[code](https://github.com/attreyabhatt/KivyMD-Basics)

# Installation

* Pip install kivy kivymd

# Basics

## Labels and Text Styles

from kivymd.app import MDApp  
from kivymd.uix.label import MDLabel, MDIcon  
  
  
class DemoApp(MDApp):  
 def build(self):  
 label = MDLabel(text="Hello world", halign='center', theme\_text\_color='Custom',  
 # text\_color=(R, G, B, T)  
 text\_color=(236 / 255.0, 98 / 255.0, 81 / 255.0, 1),  
 font\_style='H1')  
 icon\_label = MDIcon(icon='language-python', halign='center')  
 return icon\_label  
  
  
DemoApp().run()

## Buttons

from kivymd.app import MDApp  
from kivymd.uix.screen import Screen  
from kivymd.uix.button import MDFlatButton, MDRectangleFlatButton, MDIconButton, MDFloatingActionButton  
  
  
class DemoApp(MDApp):  
  
 def build(self):  
 screen = Screen()  
 btn\_flat = MDRectangleFlatButton(text="Hello world",  
 pos\_hint={'center\_x': 0.5, 'center\_y': 0.5})  
 btn\_icon = MDFloatingActionButton(icon='android',  
 pos\_hint={'center\_x': 0.5, 'center\_y': 0.5})  
 screen.add\_widget(btn\_icon)  
 return screen  
  
  
DemoApp().run()

## Themes and Color Palettes

from kivymd.app import MDApp  
from kivymd.uix.screen import Screen  
from kivymd.uix.button import MDFlatButton, MDRectangleFlatButton, MDIconButton, MDFloatingActionButton  
  
  
class DemoApp(MDApp):  
  
 def build(self):  
 self.theme\_cls.primary\_palette = 'Yellow'  
 self.theme\_cls.primary\_hue = 'A700'  
 self.theme\_cls.theme\_style = 'Dark'  
 screen = Screen()  
 btn\_flat = MDRectangleFlatButton(text="Hello world",  
 pos\_hint={'center\_x': 0.5, 'center\_y': 0.5})  
  
 screen.add\_widget(btn\_flat)  
 return screen  
  
  
DemoApp().run()

## User Input with Text Field

from kivymd.app import MDApp  
from kivymd.uix.screen import Screen  
from kivymd.uix.textfield import MDTextField  
from kivy.lang import Builder  
  
username\_helper = """  
MDTextField:  
 hint\_text: "Enter username"  
 helper\_text: "or click on forgot username"  
 helper\_text\_mode: "on\_focus"  
 icon\_right: 'android'  
 icon\_right\_color: 'app.theme\_cls.primary\_color'  
 pos\_hint: {'center\_x': 0.5, 'center\_y': 0.5}  
 size\_hint\_x: None   
 width: 300  
"""  
  
  
class DemoApp(MDApp):  
  
 def build(self):  
 screen = Screen()  
 self.theme\_cls.primary\_palette = 'Green'  
 # username = MDTextField(text='Enter username',  
 # pos\_hint={'center\_x': 0.5, 'center\_y': 0.5},  
 # size\_hint\_x=None, width=300)  
 username = Builder.load\_string(username\_helper)  
  
 screen.add\_widget(username)  
 return screen  
  
  
DemoApp().run()

## Binding Input and Button

from kivymd.app import MDApp  
from kivymd.uix.screen import Screen  
from kivy.lang import Builder  
from kivymd.uix.button import MDRectangleFlatButton  
from helpers import username\_helper  
  
  
class DemoApp(MDApp):  
  
 def build(self):  
 screen = Screen()  
 self.theme\_cls.primary\_palette = 'Green'  
 self.username = Builder.load\_string(username\_helper)  
 button = MDRectangleFlatButton(text='Show',  
 pos\_hint={'center\_x': 0.5, 'center\_y': 0.4},  
 on\_release=self.show\_data)  
 screen.add\_widget(self.username)  
 screen.add\_widget(button)  
 return screen  
  
 def show\_data(self, obj):  
 print(self.username.text)  
  
  
DemoApp().run()

## Creating Dialog Boxes

from kivymd.app import MDApp  
from kivymd.uix.screen import Screen  
from kivy.lang import Builder  
from kivymd.uix.dialog import MDDialog  
from kivymd.uix.button import MDRectangleFlatButton, MDFlatButton  
from helpers import username\_helper  
  
  
class DemoApp(MDApp):  
  
 def build(self):  
 screen = Screen()  
 self.theme\_cls.primary\_palette = 'Green'  
 self.username = Builder.load\_string(username\_helper)  
 button = MDRectangleFlatButton(text='Show',  
 pos\_hint={'center\_x': 0.5, 'center\_y': 0.4},  
 on\_release=self.show\_data)  
 screen.add\_widget(self.username)  
 screen.add\_widget(button)  
 return screen  
  
 def show\_data(self, obj):  
 if self.username.text is "":  
 check\_string = 'Please enter a username'  
 else:  
 check\_string = self.username.text + " does not exist"  
 btn\_close = MDFlatButton(text="Close", on\_release=self.close\_dialog)  
 btn\_more = MDFlatButton(text="More")  
 self.dialog = MDDialog(title="Username check", text=check\_string,  
 size\_hint=(0.7, 1),  
 buttons=[btn\_close, btn\_more])  
 self.dialog.open()  
  
 def close\_dialog(self, obj):  
 self.dialog.dismiss()  
  
  
DemoApp().run()

* Helpers.py

username\_helper = """  
MDTextField:  
 hint\_text: "Enter username"  
 helper\_text: "or click on forgot username"  
 helper\_text\_mode: "on\_focus"  
 icon\_right: 'android'  
 icon\_right\_color: 'app.theme\_cls.primary\_color'  
 pos\_hint: {'center\_x': 0.5, 'center\_y': 0.5}  
 size\_hint\_x: None   
 width: 300  
"""

## list item

from kivymd.app import MDApp  
from kivymd.uix.screen import Screen  
from kivymd.uix.list import MDList, OneLineListItem, ThreeLineListItem  
from kivy.uix.scrollview import ScrollView  
  
  
class DemoApp(MDApp):  
  
 def build(self):  
 screen = Screen()  
 scroll = ScrollView()  
 list\_view = MDList()  
 scroll.add\_widget(list\_view)  
  
 for i in range(20):  
 items = ThreeLineListItem(text="Item %d" % i, secondary\_text="Hello world",  
 tertiary\_text="sub text")  
 list\_view.add\_widget(items)  
  
 screen.add\_widget(scroll)  
 return screen  
  
  
DemoApp().run()

## Threelineitem

from kivymd.app import MDApp  
from kivymd.uix.screen import Screen  
from kivymd.uix.list import MDList, ThreeLineListItem, ThreeLineAvatarListItem  
from kivy.uix.scrollview import ScrollView  
from kivymd.uix.list import IconLeftWidget, ImageLeftWidget  
  
  
class DemoApp(MDApp):  
  
 def build(self):  
 screen = Screen()  
 scroll = ScrollView()  
 list\_view = MDList()  
 scroll.add\_widget(list\_view)  
  
 for i in range(20):  
 icon = IconLeftWidget(icon="android")  
 image = ImageLeftWidget(source="fb\_icon.png")  
 items = ThreeLineAvatarListItem(text="Item %d" % i, secondary\_text="Hello world",  
 tertiary\_text="sub text")  
 items.add\_widget(image)  
 list\_view.add\_widget(items)  
  
 screen.add\_widget(scroll)  
 return screen  
  
  
DemoApp().run()

## List with kivy builder

from kivymd.app import MDApp  
from kivy.lang import Builder  
from kivymd.uix.list import OneLineListItem  
  
list\_helper = """  
Screen:   
 ScrollView:  
 MDList:  
 id: container  
"""  
  
  
class DemoApp(MDApp):  
  
 def build(self):  
 screen = Builder.load\_string(list\_helper)  
 return screen  
  
 def on\_start(self):  
 for i in range(20):  
 item = OneLineListItem(text="Item " + str(i))  
 self.root.ids.container.add\_widget(item)  
  
  
DemoApp().run()

## Creating DataTable

from kivymd.app import MDApp  
from kivymd.uix.screen import Screen  
from kivymd.uix.datatables import MDDataTable  
from kivy.metrics import dp  
  
  
class DemoApp(MDApp):  
  
 def build(self):  
 screen = Screen()  
 table = MDDataTable(pos\_hint={'center\_x': 0.5, 'center\_y': 0.5},  
 size\_hint=(0.9, 0.6),  
 check=True,  
 rows\_num=10,  
 column\_data=[  
 ('No.', dp(20)),  
 ('Food', dp(30)),  
 ("Calories", dp(20))  
 ],  
 row\_data=[  
 ("1", "Burger", "300"),  
 ("2", "Oats", "150"),  
 ]  
 )  
 table.bind(on\_check\_press=self.check\_press)  
 table.bind(on\_row\_press=self.row\_press)  
 screen.add\_widget(table)  
 return screen  
  
 def check\_press(self, instance\_table, current\_row):  
 print(instance\_table, current\_row)  
  
 def row\_press(self, instance\_table, current\_row):  
 print(instance\_table, current\_row)  
  
  
DemoApp().run()

## Creating Toolbars

from kivymd.app import MDApp  
from kivy.lang import Builder  
  
# only for testing, remove this line on production  
from kivy.core.window import Window  
  
Window.size = (300, 500)  
  
screen\_helper = """  
Screen:   
 BoxLayout:  
 orientation: 'vertical'  
 MDTopAppBar:   
 title: 'Demo'  
 left\_action\_items: [['menu', lambda x: app.navigation\_draw()]]  
 right\_action\_items: [['clock', lambda x: app.navigation\_draw()]]  
 elevation: 10  
 MDLabel:  
 text: 'Hello World'  
 halign: 'center'   
 MDBottomAppBar:  
 MDTopAppBar:  
 icon: "git"  
 type: "bottom"  
 left\_action\_items: [["coffee", lambda x: app.navigation\_draw()]]  
 mode: 'end'   
 on\_action\_button: app.navigation\_draw()  
"""  
  
  
class DemoApp(MDApp):  
  
 def build(self):  
 self.theme\_cls.primary\_palette = "Red"  
 screen = Builder.load\_string(screen\_helper)  
  
 return screen  
  
 def navigation\_draw(self):  
 print("Navigation")  
  
  
DemoApp().run()

## Navigation Drawer

from kivymd.app import MDApp  
from kivy.lang import Builder  
  
# only for testing, remove this line on production  
from kivy.core.window import Window  
  
Window.size = (300, 500)  
  
screen\_helper = """  
MDScreen:   
 MDNavigationLayout:  
 ScreenManager:  
 MDScreen:  
 BoxLayout:  
 orientation: 'vertical'  
 MDTopAppBar:  
 title: 'Demo'  
 left\_action\_items: [['menu', lambda x: nav\_drawer.set\_state('toggle')]]  
 elevation: 10  
 Widget:  
   
 MDNavigationDrawer:  
 id: nav\_drawer  
 BoxLayout:  
 orientation: 'vertical'  
 spacing: '8dp'  
 padding: '8dp'  
 Image:  
 source: 'fb\_icon.png'  
   
 MDLabel:  
 text: 'Abhijith M'  
 font\_style: 'Subtitle1'  
 size\_hint\_y: None  
 height: self.texture\_size[1]  
   
 MDLabel:  
 text: 'abhijithm2447@gmail.com'  
 font\_style: 'Caption'  
 size\_hint\_y: None  
 height: self.texture\_size[1]  
   
 ScrollView:  
 MDList:  
 OneLineIconListItem:  
 text: 'Profile'  
 IconLeftWidget:  
 icon: 'face-man-profile'  
 OneLineIconListItem:  
 text: 'Upload'  
 IconLeftWidget:  
 icon: 'file-upload'  
 OneLineIconListItem:  
 text: 'Logout'  
 IconLeftWidget:  
 icon: 'logout'  
"""  
  
  
class DemoApp(MDApp):  
  
 def build(self):  
 self.theme\_cls.primary\_palette = "Red"  
 screen = Builder.load\_string(screen\_helper)  
  
 return screen  
  
  
DemoApp().run()

## Changing Screens using ScreenManager

from kivymd.app import MDApp  
from kivy.lang.builder import Builder  
from kivy.uix.screenmanager import ScreenManager, Screen  
  
screen\_helper = """  
ScreenManager:  
 MenuScreen:  
 ProfileScreen:  
 UploadScreen:  
<MenuScreen>:  
 name: 'menu'  
 MDRectangleFlatButton:  
 text: 'Profile'  
 pos\_hint: {'center\_x':0.5,'center\_y':0.6}  
 on\_press: root.manager.current = 'profile'  
 MDRectangleFlatButton:  
 text: 'Upload'  
 pos\_hint: {'center\_x':0.5,'center\_y':0.5}  
 on\_press: root.manager.current = 'upload'  
   
<ProfileScreen>:  
 name: 'profile'  
 MDLabel:  
 text: 'Profile'  
 halign: 'center'  
 MDRectangleFlatButton:  
 text: 'Back'  
 pos\_hint: {'center\_x':0.5,'center\_y':0.1}  
 on\_press: root.manager.current = 'menu'  
   
<UploadScreen>:  
 name: 'upload'  
 MDLabel:  
 text: 'Upload'  
 halign: 'center'  
 MDRectangleFlatButton:  
 text: 'Back'  
 pos\_hint: {'center\_x':0.5,'center\_y':0.1}  
 on\_press: root.manager.current = 'menu'  
   
"""  
  
  
class MenuScreen(Screen):  
 pass  
  
  
class ProfileScreen(Screen):  
 pass  
  
  
class UploadScreen(Screen):  
 pass  
  
  
# Create the screen manager  
sm = ScreenManager()  
sm.add\_widget(MenuScreen(name='menu'))  
sm.add\_widget(ProfileScreen(name='profile'))  
sm.add\_widget(UploadScreen(name='upload'))  
  
  
class DemoApp(MDApp):  
  
 def build(self):  
 screen = Builder.load\_string(screen\_helper)  
 return screen  
  
  
DemoApp().run()

## 3 ways to convert kivy to apk

[Tutorial](https://towardsdatascience.com/3-ways-to-convert-python-app-into-apk-77f4c9cd55af)

# Advanced

## opencv + kivy

[Tutorial](https://www.youtube.com/watch?v=PwUWtfk2inQ)

from kivymd.app import MDApp  
from kivymd.uix.boxlayout import MDBoxLayout  
from kivymd.uix.button import MDRaisedButton  
from kivy.uix.image import Image  
from kivy.clock import Clock  
from kivy.graphics.texture import Texture  
import cv2  
  
  
class MainApp(MDApp):  
 def build(self):  
 layout = MDBoxLayout(orientation='vertical')  
 self.image = Image()  
 layout.add\_widget(self.image)  
 layout.add\_widget(MDRaisedButton(  
 text='CLICK HERE',  
 pos\_hint={'center\_x': .5, 'center\_y': .5},  
 size\_hint=(None, None))  
 )  
 self.capture = cv2.VideoCapture(1)  
 Clock.schedule\_interval(self.load\_video, 1.0 / 33.0)  
 return layout  
  
 def load\_video(self, \*args):  
 ret, frame = self.capture.read()  
 self.image\_frame = frame  
 buffer = cv2.flip(frame, 0).tostring()  
 texture = Texture.create(size=(frame.shape[1], frame.shape[0]), colorfmt='bgr')  
 texture.blit\_buffer(buffer, colorfmt='bgr', bufferfmt='ubyte')  
 self.image.texture = texture  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 MainApp().run()