


# Building Android Apps With Python: Part -1

Step by Step Guide to Build Android Apps using Python



Kaustubh Gupta

Aug 5, 2020 · 7 min read

## Introduction



Photo by [Hitesh Choudhary](#) on [Unsplash](#)

Are you curious about developing android apps but Java is not your companion? Android universe is mainly build using Java, Kotlin, Flutter, and Corona using Lua Scripting Language (mainly gaming engine, used in games like angry birds) but in recent times, Python has made its way into every domain and android is no different. In this series of articles, we will look at how to set-up the required environment, the basics of developing an android app, referencing the documentation, and how to move ahead with your projects.

## Kivy — The Godfather

Android development in Python has been made possible only because of an open-source Python library for developing mobile apps and other multi-touch application software that is **Kivy**. Its initial release was in 2011 and a stable one in 2019! Kivy not only supports android application development but its applications can be run on IOS, Linux, OS X, Windows, and Android.

It is written in Python and Cython, and most of the core developers are from Russia.

We will use Kivy a lot for the front-end of the application but with another package and why we require that package will be covered shortly.

. . .

## Setting the environment

It’s usually a good practice to set-up a new environment for new projects as:

- 1. It **helps in maintaining the different versions** of different libraries.  
For example, ML flow requires a lower version of Numpy and when you try to install ML flow in the base directory, it conflicts with the pre-installed libraries and makes it difficult to manage different versions.
- 2. It **helps in isolating** custom codes and makes it easier while deploying your application on any platform.

I use the Conda package manager for creating and managing my environments. You can use any other package manager but to follow along with me, you can use Conda (see this guide to set up Conda and anaconda). Open up your Conda terminal and type:

```
conda create -n name-of-env python=version
```

Replace ‘name-of-env’ with your custom name and ‘version’ of your choice but greater than 3.5. I will use Python 3.7. To list all the Conda environments, type:

```
conda info --envs
```

The output will be similar to this:

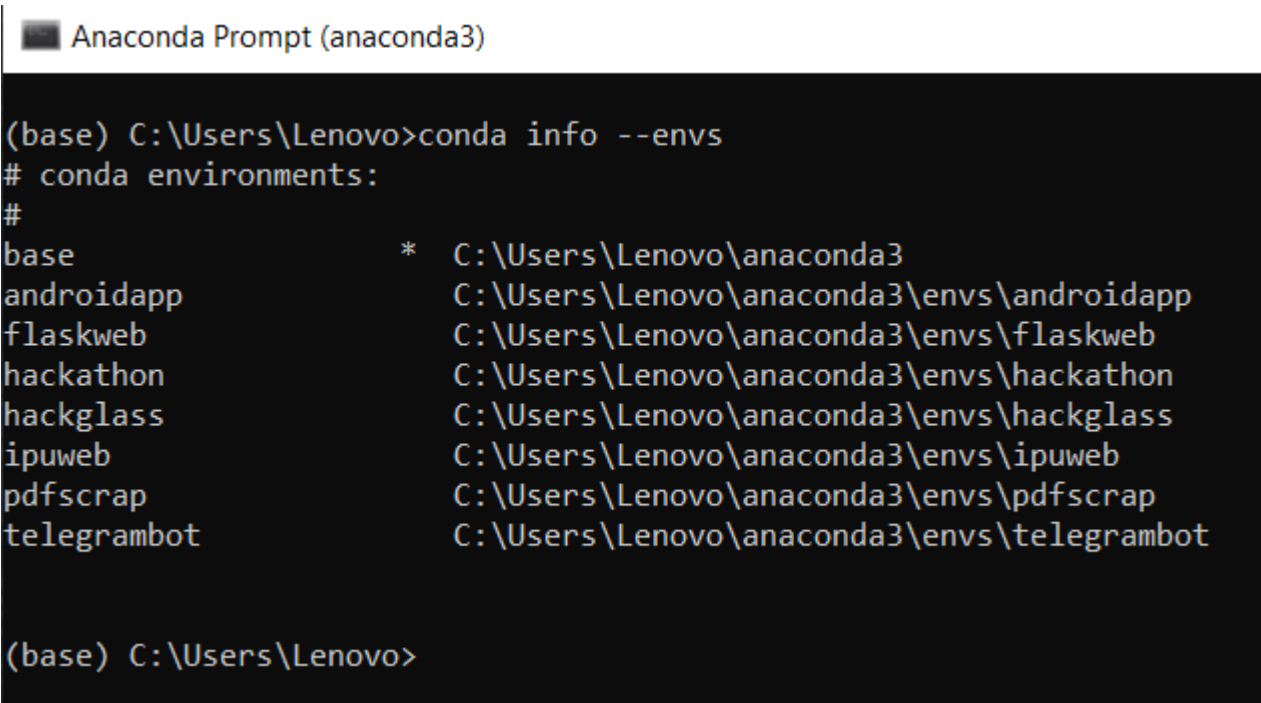


Image by author

Here is the link to the cheat-sheet of Conda in-case you are interested in exploring more about this. Now, after checking the name here, activate the environment like this:

```
conda activate nameofenv
```

We are ready to install the required libraries. As we are using python, pip is a great way to install and manage python packages. To install Kivy and its dependencies, type the following command one-by-one:

```
pip install kivy
pip install kivy-deps.angle
pip install kivy-deps.glew
pip install kivy-deps.gstreamer
pip install kivy-deps.sdl2
```

**A bonus tip:** Make a file called requirements.txt, copy the above lines in the file, place the file in a known location and terminal run:

```
pip install requirements.txt
```

It will install all the dependencies at one go!

We are ready to develop some awesome applications, but there is one problem here. In the beginning, I told you we need an additional package to be used with Kivy. Just install it for now and we will discuss the why part later in the article.

```
pip install kivymd
```

. . .

## Let’s code!

The part you have been waiting for so long is here. I am using Pycharm IDE for coding because it’s easy to code for me, but you can use VSCode, Sublime, or spyder as per your wish. Before I start, we need to understand some points here:

1. An android app has a **front-end (UI/UX) or the interactive part** where the user interacts with your application and all the inputs are given via this layer.

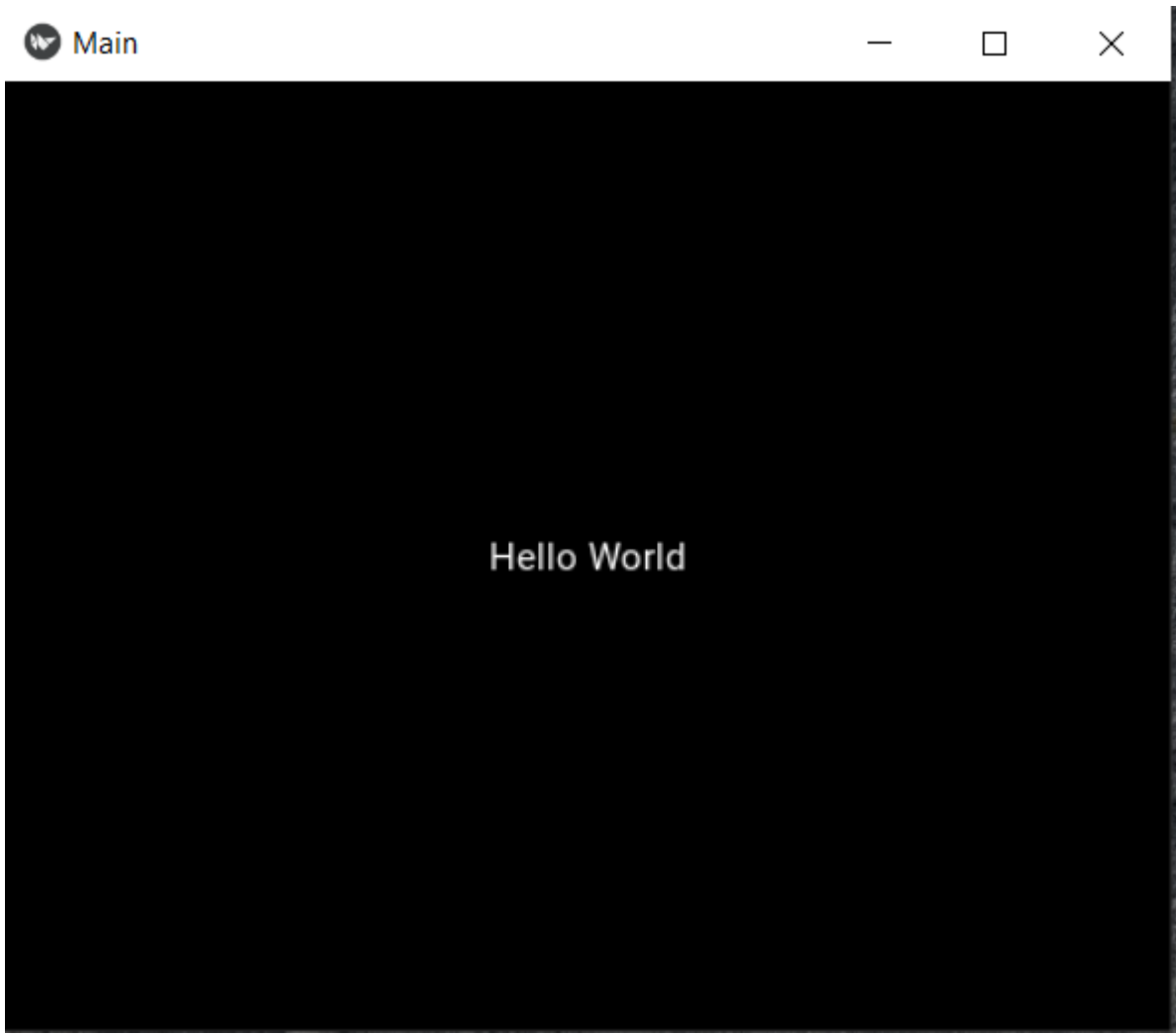
- 2. The inputs are **transferred to the backend layer**, which is our python code. This backend layer controls the flow, processes the outputs, and the content to be displayed on the screen.
- 3. Here, **Object-Oriented Programming** is highly used and most of the programming will be done using this concept so if you are lacking in this then I would suggest you follow this [video by Keith Galli on OOP](#).

Now quickly open up your IDE and start with this basic code of Hello World!

```
1  from kivy.app import App
2  from kivy.uix.label import Label
3
4
5  class Main(App):
6      def build(self):
7          return Label(text='Hello World')
8
9
10 Main().run()
```

base.py hosted with ❤ by GitHub view raw

After running this program you will see this output:



First Output

Let me explain this code line by line:

- 1. The first line imports the base app from the Kivy library.
- 2. Now we need to display some text on the screen and for displaying any text, we use Label functionality and as I told that these are UI/UX things, we will import them from the “kivy.uix” directory.

- 3. The third line (ignore the white-space) of the code is where our OOP concept comes into play. The App which we have imported from the “kivy.app” is the base class of the App. What it means is that the Kivy sets up all the essential things to run our app and we need to inherit this app class, change it, and build our application. The name of this class should start with capitals and it also serves as the name of the app which can be changed later on so you can name it anything you want.
- 4. def build function is the app entry point. All the things defined here will be built first and the first screen or the main screen is passed here. It returns the Label, and it has a property of text which has the value “Hello World”. Read more about labels [here](#).
- 5. Then the last line calls this main class and runs it.

. . .

## Problem with Kivy

This how we build our first app but did you notice one thing that the background is automatically black and text is white? I haven’t even mentioned this in the code. Kivy takes it by default. Now we move to the interesting part, let’s build a simple button with no enhancement in Kivy:

```
1  from kivy.app import App
2  from kivy.uix.button import Button
3
4
5  class Main(App):
6      def build(self):
7          return Button(text='Hello World',
8                          size_hint=(0.5, 0.5))
9
10
11  Main().run()
```

simpleButton.py hosted with ❤ by GitHub view raw

Its output is like this:





A simple button in Kivy

**This is a very unattractive look** and imagines you are using an app that has an interface like this. I would uninstall that app and won’t even rate it! Enhancing features in Kivy is a tedious process and requires a lot of code. Don’t believe me? Look at the code to create a rectangular flat button placed at the center with a blue border, blue text, and white background:

```
1
2  from kivy.app import App
3  from kivy.metrics import dp
4  from kivy.uix.behaviors import TouchRippleBehavior
5  from kivy.uix.button import Button
6  from kivy.lang import Builder
7
8
9  KV = """
10 <RectangleFlatButton>:
11     ripple_color: 0, 0, 0, .2
12     background_color: 0, 0, 0, 0
13     color: root.primary_color
14     canvas.before:
15         Color:
16             rgba: root.primary_color
17         Line:
18             width: 1
19             rectangle: (self.x, self.y, self.width, self.height)
20 Screen:
21     canvas:
22         Color:
23             rgba: 0.9764705882352941, 0.9764705882352941, 0.9764705882352941, 1
24         Rectangle:
25             pos: self.pos
26             size: self.size
27 """
28
29
30 class RectangleFlatButton(TouchRippleBehavior, Button):
31     primary_color = [
32         0.12941176470588237,
33         0.5882352941176471,
34         0.9529411764705882,
35         1
36     ]
37
38     def on_touch_down(self, touch):
39         collide_point = self.collide_point(touch.x, touch.y)
40         if collide_point:
41             touch.grab(self)
42             self.ripple_show(touch)
43             return True
44         return False
45
46     def on_touch_up(self, touch):
47         if touch.grab_current is self:
48             touch.ungrab(self)
49             self.ripple_fade()
50             return True
51         return False
```

```
52
53
54 class MainApp(App):
55     def build(self):
56         screen = Builder.load_string(KV)
57         screen.add_widget(
58             RectangleFlatButton(
59                 text="Hello, World",
60                 pos_hint={"center_x": 0.5, "center_y": 0.5},
61                 size_hint=(None, None),
62                 size=(dp(110), dp(35)),
63                 ripple_color=(0.8, 0.8, 0.8, 0.5),
64             )
65         )
66         return screen
67
68
69 MainApp().run()
```

rectangularFlatButtonKivy.py hosted with ❤ by GitHub

view raw

Rectangular Flat Button Using Kivy. Source: [Attrey Bhatt](#)

Don’t grasp the code as it is beyond your scope for now but just look at the output now:

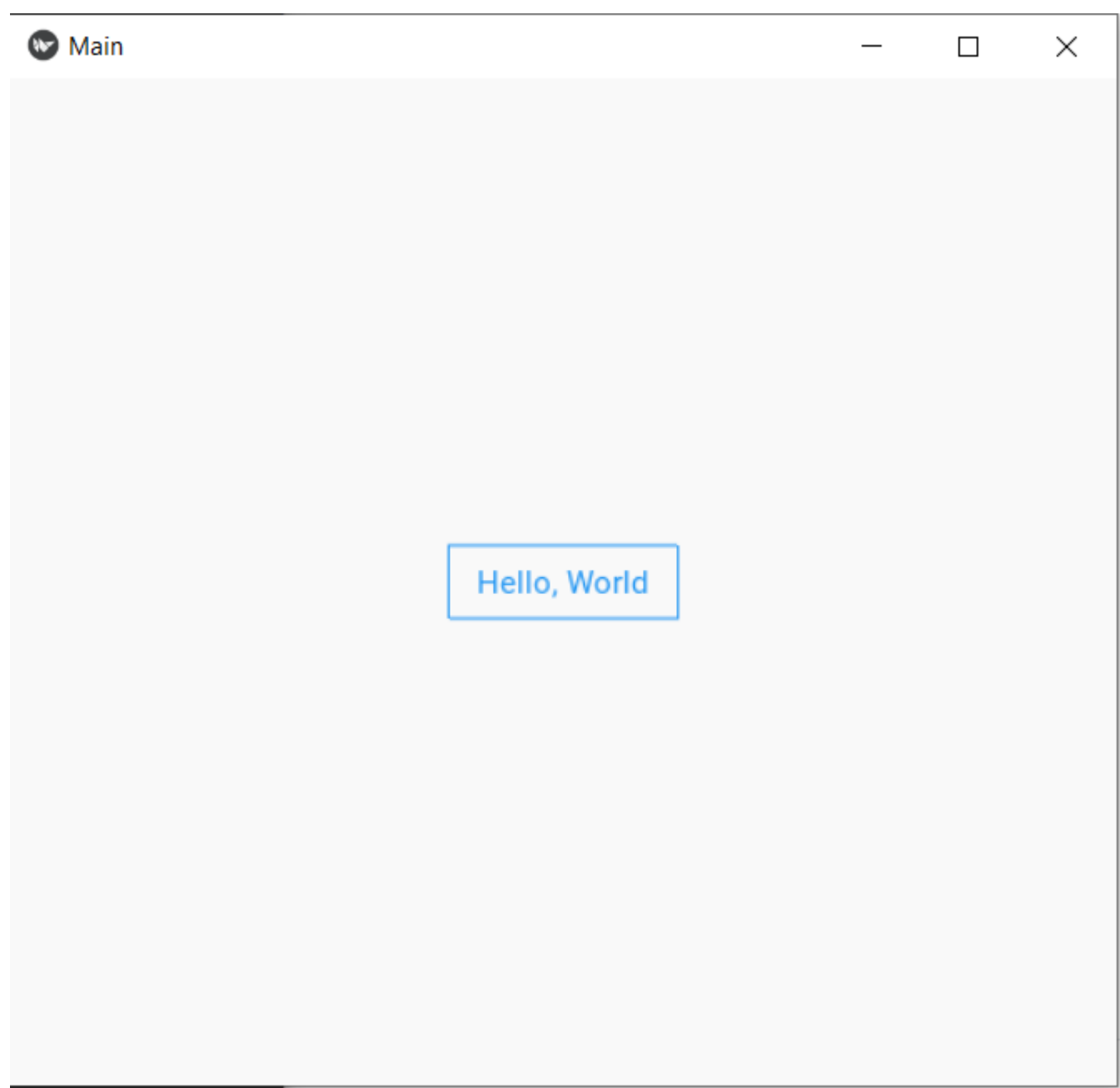


Image by author

Doesn’t it look nice now!

• • •

## Introducing Kivymd



Now we have talked a lot about Kivy and we know it provides the platform for building applications. KivyMD is a collection of Material Design compliant widgets for use with Kivy and approximately Google’s Material Design spec as close as possible without sacrificing ease of use or application performance. It is based on Kivy and is easier to code. It is very similar to Kivy and just adds MD at starting in every element and widget, plus it has a wide variety of other new elements. Now see the code in Kivymd to generate the same output button:

```
1  from kivymd.app import MDApp
2  from kivymd.uix.button import MDRectangleFlatButton
3  from kivymd.uix.screen import MDScreen
4
5
6  class Main(MDApp):
7      def build(self):
8          screen = MDScreen()
9          btn = MDRectangleFlatButton(text="Hello World",
10                                     pos_hint={'center_x': 0.5, 'center_y': 0.5}
11                                     )
12          screen.add_widget(btn)
13          return screen
14
15
16  Main().run()
```

rectangularFlatButtonKivymd.py hosted with ❤ by GitHub [view raw](#)

Try to run this script and you will see the same output as returned by long Kivy code.

. . .

## Some more things to consider

- 1. Performance Issues:** The app you will develop will work perfectly fine on your local machine, but when you try to run it in android, the animations are not so smooth. Also, as it still runs as an instance of python, it is slow.
- 2. Convert to android:** One of the major tasks is to convert the python Kivy application into an Android package (APK) and it can only be done on a Linux OS. The additional packages like python-to-android, Android SDK, bulldozer are heavy and require a lot of time to build and debug the app. All the Kivy and Kivymd libraries are converted into Cython, and these are used by the Android system. This process is usually done with high precision.
- 3. Still under development:** This is an open-source project, and still a lot of work is going on under the hood. Always try to update the packages so you don’t face any issues.

## Conclusion and what’s next



This was an introductory lesson to building android apps in python. We learned what is Kivy, why, and how to use environments, built a basic app in Kivy, compared Kivy and Kivymd with an example of a button code. In the next article, we will continue our journey and explore various other key elements in Kivymd. If you liked this article, follow me on medium so you receive notifications about upcoming parts. With that said sayonara!

You can find me here:

LinkedIn: [Link](#)


GitHub: [Link](#)

UPDATE: Part-2 is here:

**Building Android Apps With Python: Part -2**

Step by Step Guide to Build Android Apps using Python


medium.com



### Sign up for The Variable

By Towards Data Science

Every Thursday, the Variable delivers the very best of Towards Data Science: from hands-on tutorials and cutting-edge research to original features you don't want to miss. [Take a look.](#)

 Get this newsletter

You'll need to sign in or create an account to receive this newsletter.

- Python

Android

Kivy

Kivymd

Android App Development