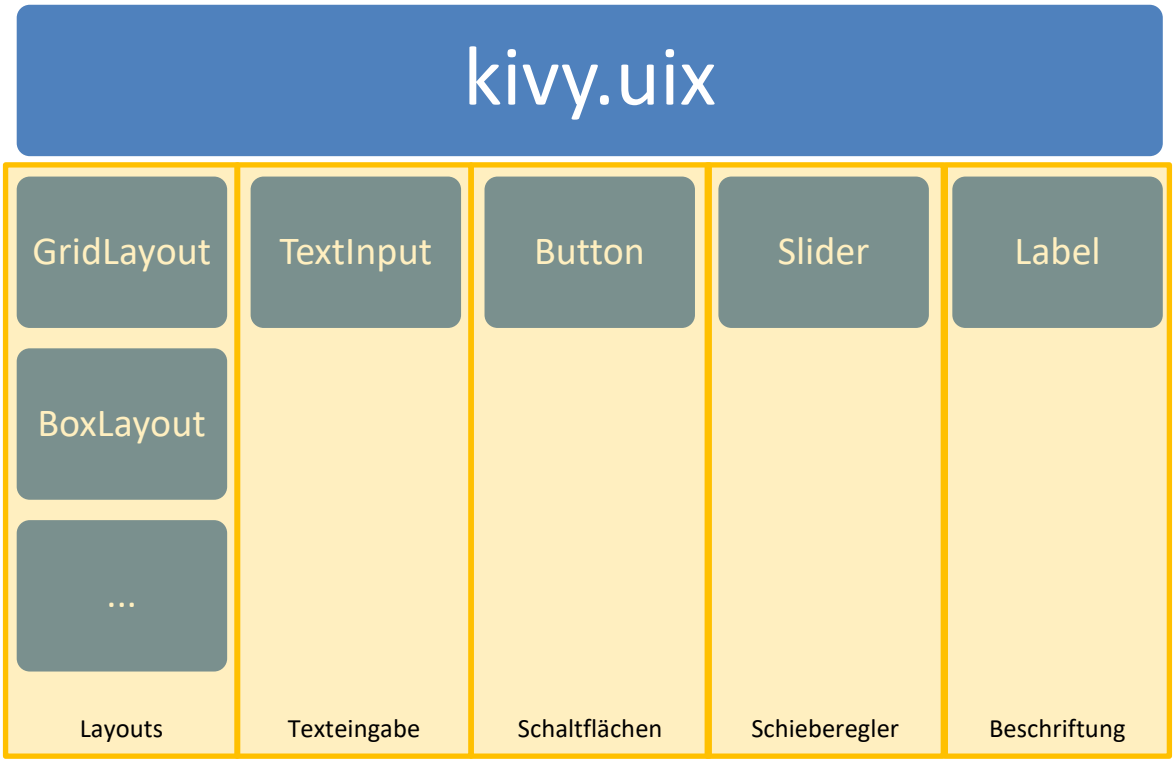
L2.1 Grid-Layout und Erste Widgets

2.1.1 Grundlegender Aufbau eines Projektes

In Kivy wird ein Layout verwendet, um die Position und Anordnung der Widgets auf dem Bildschirm einer App zu steuern. Kivy bietet eine Vielzahl von Layout-Klassen, die helfen, Benutzeroberflächen flexibel und anpassbar zu gestalten. Diese Layout-Klassen steuern, wie die einzelnen Widgets innerhalb eines Containers oder einer Oberfläche positioniert, skaliert und verteilt werden. Wir konzentrieren uns zunächst auf das Grid-Layout. Dieses ordnet Widgets in einem festen Raster von Zeilen und Spalten an.

2.1.2 Erste Widgets

Kivy verwendet das Wort „Widget“, um einzelne Elemente in der Benutzeroberfläche zu beschreiben. In Kivy gibt es ein Modul kivy.uix, welches zahlreiche Widgets enthält.



Es wird unterschieden zwischen Layouts (GridLayout, BoxLayout, …), einfachen Widgets (TextInput, Button, Slider, Label, …), komplexen Widgets (DropDown, Spinner, …) und weiteren hier nicht relevanten Kategorien. Eine Übersicht ist hier zu finden: https://kivy.org/doc/stable/api-kivy.uix.html

2.1.3 Bezeichnungsfelder und Schaltflächen in Kivy

Bezeichnungsfelder (Labels) sind in Kivy mit Hilfe des Widgets *Label* möglich. Schaltflächen (Buttons) sind in Kivy mit Hilfe des Widgets *Button* möglich.

	
Label	Button

2.1.4 Datei für den Programmablauf

Aufbau der Datei main.py:

```
1  from kivy.app import App
2  from kivy.uix.gridlayout import GridLayout
3  from kivy.lang import Builder
4
5  # interface
6  class MyGridLayout(GridLayout):
7      pass
8
9  class MainApp(App):
10     def build(self):
11         return Builder.load_file("main.kv")
12
13 if __name__ == "__main__":
14     MainApp().run()
```

main.py

Zeile 2: Importiert die GridLayout-Klasse, ein Layout-Widget, das Widgets in einem Raster anordnet (in Zeilen und Spalten).
Zeilen 5-7: Hier wird eine Klasse namens MyGridLayout definiert, die von GridLayout erbt. Sie wird später in der `main.kv`-Datei verwendet, um das Layout der App zu bestimmen.
pass: Da derzeit keine speziellen Eigenschaften oder Methoden für `MyGridLayout` definiert sind, wird `pass` verwendet, um eine leere Klasse zu erstellen.

2.1.5 Datei für das Grid-Layout

In der Python-Datei wurde eine Klasse MyGridLayout erstellt, die auf einem GridLayout basiert.

Aufbau der Datei main.kv:

```
1  MyGridLayout:
2
3      cols: 2
4      rows: 1
5      row_force_default: True
6      row_default_height: 40
7
8      Label:
9          text: "Wie geht es dir?"
10
11     Button:
12         text: "Gut"
```

main.kv

Zeile 1:
Diese Zeile legt fest, dass die folgende Definition sich auf die Klasse `MyGridLayout` bezieht. Alles, was eingerückt ist, wird innerhalb dieses Layouts definiert.
MyGridLayout ist ein benutzerdefiniertes Layout, das im Python-Code als Unterklasse von GridLayout erstellt wurde.

Zeile 3: Definiert, dass das Raster aus 2 Spalten besteht.
Zeile 4: Definiert, dass das Raster aus 1 Zeile besteht.
Zeile 5: Erzwingt, dass jede Zeile eine standardmäßige Höhe hat, unabhängig vom Inhalt.
Zeile 6: Setzt die Höhe jeder Zeile auf 40 Pixel, da `row_force_default` auf `True` gesetzt ist.

Zeilen 8-12: Es folgen Widgets innerhalb des MyGridLayout.

2.1.6 Attribute im Grid-Layout

Eine ausführliche Dokumentation eines allgemeinen Widgets ist hier zu finden: https://kivy.org/doc/stable/api-kivy.uix.widget.html
Eine ausführliche Dokumentation des GridLayout ist hier zu finden: https://kivy.org/doc/stable/api-kivy.uix.gridlayout.html

Allgemeine Attribute:

Name	**Beschreibung**	**Beispiel**
cols	Anzahl Spalten	`cols: 2`
rows	Anzahl Zeilen	`rows: 2`
row_force_default	Wenn auf True gesetzt, dann wird die standardmäßig eingestellte Zeilen-höhe verwendet	`row_force_default: True`
row_default_height	Zeilenhöhe	`row_default_height: 40`
spacing	Abstand zwischen den Kacheln	`spacing: 4`
padding	Abstand zwischen der LayoutBox und ihrem Inhalt	`padding: 30`

2.1.7 Hintergrundfarbe der Zeichenfläche

Um den Hintergrund einzufärben, kann in der kv-Datei folgender Code genutzt werden:

```
canvas:
   Color:
      rgb: 0, 1, 1
   Rectangle:
      pos: [0 * coord for coord in self.size]
      size:[1 * coord for coord in self.size]
```

Zunächst muss für den Hintergrund („canvas“) eine Farbe festgelegt werden. Auch hier wird der RGB-Farbraum genutzt. Dann muss dieser Hintergrund mit einem Rechteck ausgefüllt werden.

2.1.8 Verschachtelte Grid-Layouts

Grid-Layouts lassen sich auch verschachteln. Dabei ist innerhalb der kv-Datei auf die entsprechenden Einrückungen zu achten.


```kv
<MyGridLayout>:
MyGridLayout:
    cols: 1
    rows: 2

    GridLayout:
        cols: 2
        rows: 1

        Button:
            ...

        Button:
            ...

    Button:
        ...
```

main.kv

2.1.9 Attribute für Labels und Buttons

Eine ausführliche Dokumentation eines allgemeinen Widgets ist hier zu finden:
https://kivy.org/doc/stable/api-kivy.uix.widget.html
Label: https://kivy.org/doc/stable/api-kivy.uix.label.html
Button: https://kivy.org/doc/stable/api-kivy.uix.button.html

Attribute für Widgets (z. B.: Button, Label)

Name	**Beschreibung**	**Beispiel**
id	Bezeichnung des Widgets. Über diese ID kann das Widget später im Python-Code angesteuert werden.	`id: label1`
text	Beschriftung des Widgets	`text: "Gut"`
size_hint_x	Proportion des Widgets im Verhältnis zu anderen Widgets dieser Zeile	`size_hint_x: 1`
width	Feste Breite der Widgets (wenn nicht mit size_hint_x gearbeitet wird)	`width: 100`
font_family	Schriftart	`font_family: "Arial"`
font_size	Schriftgröße	`font_size: "72dp"`

color	Textfarbe im RGBA-Farbraum (R = red, G = green, B = blue, A = alpha (Transparenz)). Jeder Wert liegt zwischen 0 und 1.	`color: 0, 1, 0, 1`
background_color	Hintergrundfarbe im RGBA-Farbraum (R = red, G = green, B = blue, A = alpha (Transparenz)). Jeder Wert liegt zwischen 0 und 1. Es ist zu beachten, dass dieses Attribut nicht für Labels anwendbar ist.	`background_color: 1, 0, 0, 0.5`

Sollte ein Label mit Textinhalt befüllt werden, so ist zu beachten, dass hier ausschließlich der Dateityp *str* akzeptiert wird. Zahlen müssten also zunächst mit Hilfe der Funktion *str( )* umgewandelt werden.