L1.1 Einrichten eines ersten Projektes

1.1.1 Basisinformationen zu Kivy

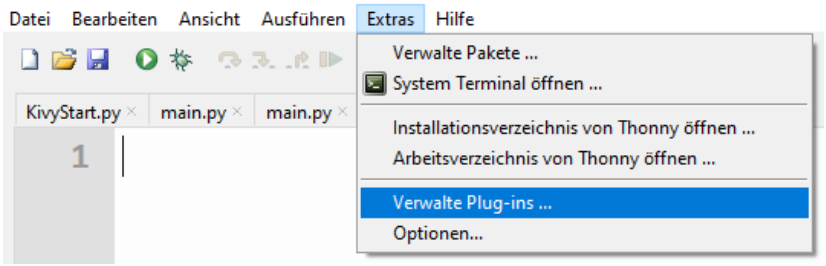
Kivy ist ein modernes Toolkit zur Erstellung von graphischen Benutzeroberflächen. Aus mehreren Gründen ist Kivy für Softwareentwickler äußerst attraktiv:

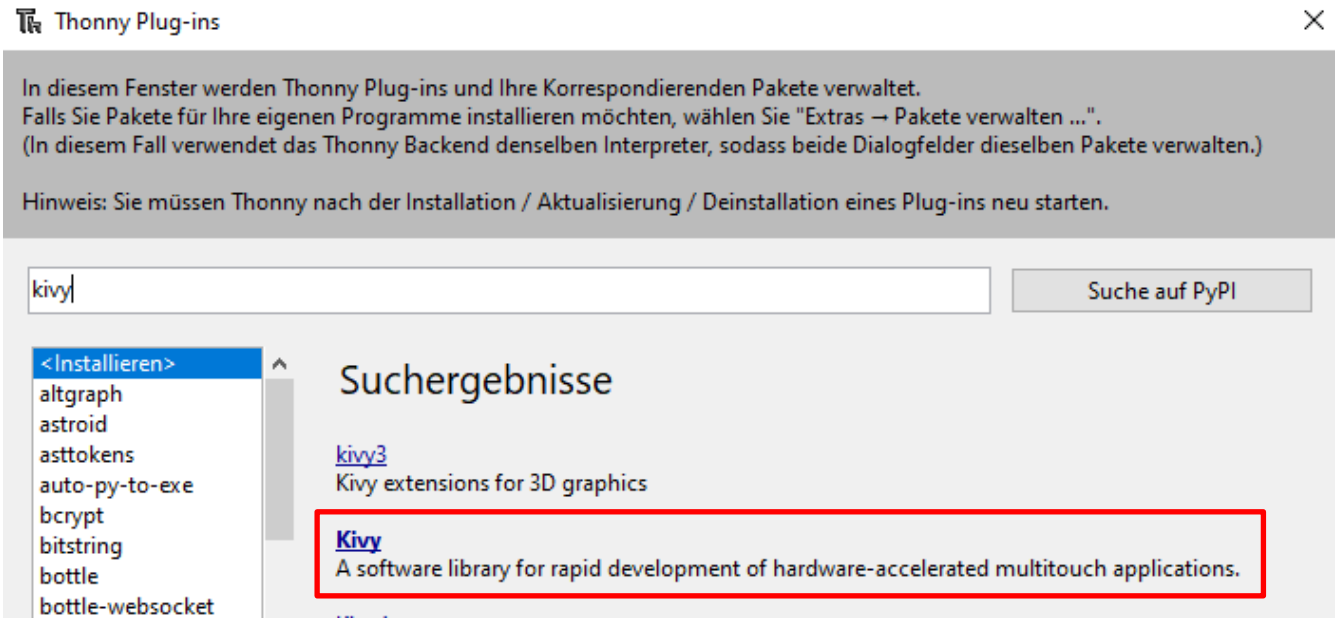
- Es bietet eine elegante Unterstützung für Multitouch-Geräte
- Es bietet den aktuell einzigen Weg, Apps in Python zu kodieren
- Kivy erlaubt es, eine einzelne Applikation auf zahlreiche Betriebssystemen (Android, F-Droid, iOS, …) auszuführen.

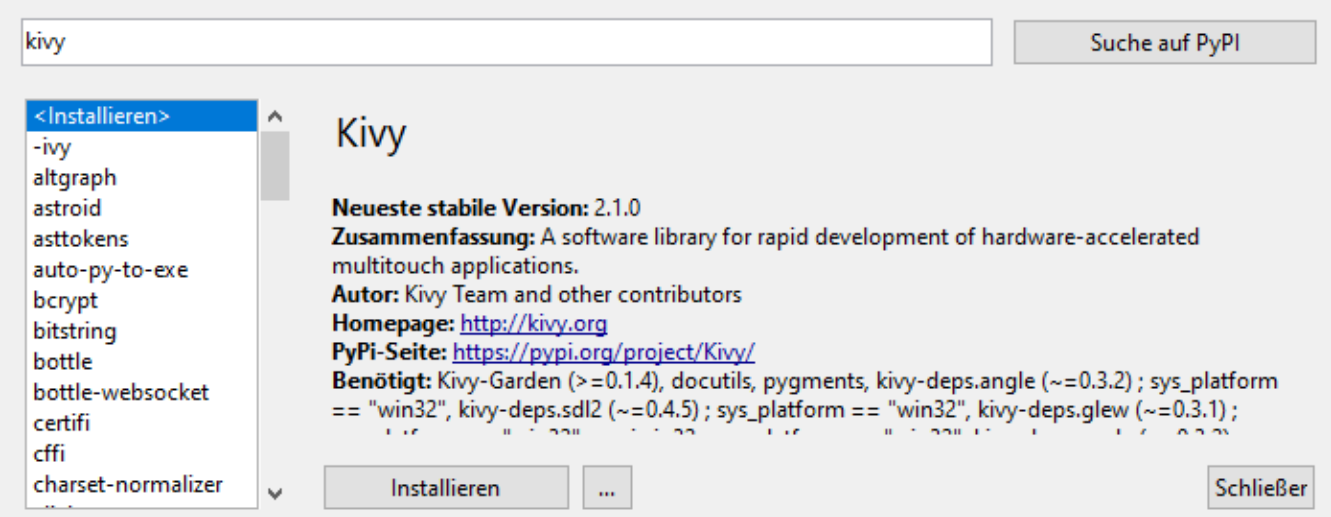
Eine Übersicht über Kivy finden Sie hier: https://kivy.org/
Die offizielle Dokumentation finden Sie hier: https://kivy.org/doc/stable/

1.1.2 Installation von Kivy

Um Kivy nutzen zu können, ist zuerst die Bibliothek von Kivy innerhalb der Entwicklungsumgebung (z. B. Thonny) zu installieren.



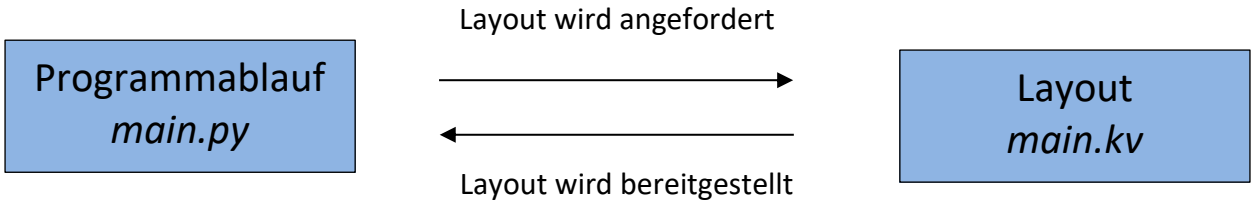




Das gesamte Material bezieht sich auf die Kivy-Version 2.1.0.

1.1.3 Einfacher Aufbau moderner Softwareprojekte

In moderner Softwareentwicklung sind Projekte üblicherweise so aufgebaut, dass klar zwischen dem Layout und dem Programmablauf getrennt wird.
Das Layout wird in einer Auszeichnungssprache namens KV gepflegt.
Der Programmablauf wird in Python beschrieben. In der Python-Datei wird auch ein Bezug zur Datei hergestellt, in der das Layout liegt.



1.1.4 Einrichtung eines ersten Projektes

Aufbau der Datei main.py:

```
1  from kivy.app import App
2  from kivy.lang import Builder
3
4  class MainApp(App):
5      def build(self):
6          return Builder.load_file("main.kv")
7
8  if __name__ == '__main__':
9      app = MainApp()
10     app.run()
```

main.py

Zeile 1: Dieser Import bezieht sich auf die App-Klasse aus dem Kivy-Framework, die für den Aufbau und das Ausführen der Anwendung verantwortlich ist.
Zeile 2: Importiert den Builder aus kivy.lang, der zum Laden und Analysieren von .kv-Dateien verwendet wird.
Zeilen 4-6:

- `MainApp` ist eine Unterklasse von App, was bedeutet, dass sie von App erbt und deren Funktionalität nutzen kann.
- Die Methode `build(self)` ist eine spezielle Methode in Kivy-Apps, die aufgerufen wird, um die Haupt-UI-Komponente zu erstellen, die dann angezeigt wird.
- Diese Zeile lädt die `.kv`-Datei namens `main.kv` mithilfe des Builder und gibt sie als Haupt-Widget zurück. Die Benutzeroberfläche wird durch die `main.kv`-Datei definiert und in die App integriert.

Zeilen 8-10:

- Diese Zeilen stellen sicher, dass der Code nur ausgeführt wird, wenn das Skript direkt gestartet wird und nicht beim Import als Modul.
- `app = MainApp()` erstellt eine Instanz der `MainApp`-Klasse.
- `app.run()` startet die Kivy-Anwendung und zeigt das Hauptfenster an.

1.1.5 Ein erstes Widget

Kivy verwendet den Begriff „Widget“ für einzelne Elemente in der Benutzeroberfläche. In Kivy gibt es ein Modul kivy.uix, welches zahlreiche Widgets enthält.

Im nächsten Schritt lässt sich beispielhaft ein Bezeichnungsfeld (Label) als erstes Widget einrichten und testen.

Aufbau der Datei main.kv innerhalb der App:

```
1 Label:
2     text: 'Hello World!'
3     font_size: 72
```

main.kv

Zeile 1:

Label: Das Label ist ein Widget, das Text anzeigt. Hier wird ein einzelnes Label-Widget ohne weitere Einbettung direkt auf der Oberfläche platziert.

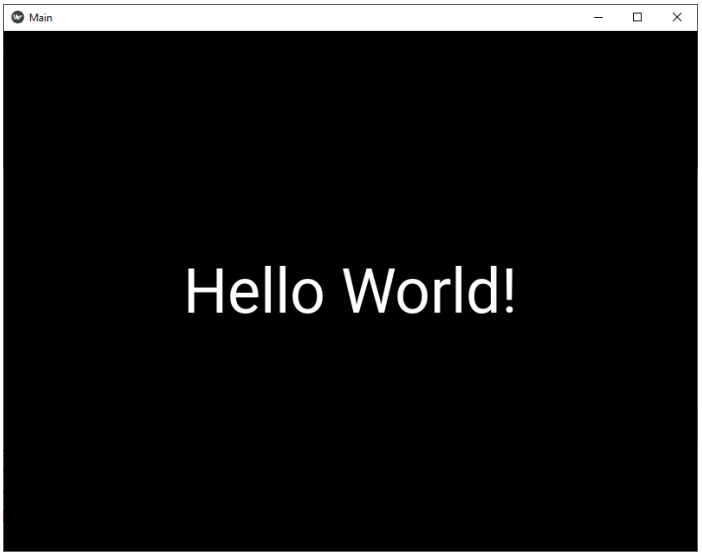
Zeile 2:

Das text-Attribut definiert den Text, der auf dem Label angezeigt wird. In diesem Fall zeigt das Label "Hello World!" an.

Zeile 3:

Das font_size-Attribut gibt die Schriftgröße des Textes im Label an. Mit 72 wird eine relativ große Schrift verwendet.

Beim Start der App erhalten wir nun als Ergebnis ein Fenster mit einem Begrüßungstext.



1.1.6 Attribute für Labels

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

Name	**Beschreibung**	**Beispiel**
text	Beschriftung des Widgets	`text: "Gut"`
font_family	Schriftart	`font_family = "Arial"`
font_size	Schriftgröße	`font_size = '72dp'`
markup	Wenn dieses Attribut gesetzt wird, dann kann der Schriftstil durch HTML-ähnlichen Code angepasst werden (https://kivy.org/doc/stable/api-kivy.core.text.markup.html).	`markup = True`

Anmerkungen:

- Sollte ein Label mit Textinhalt befüllt werden, so ist zu beachten, dass hier ausschließlich der Datentyp `str` akzeptiert wird. Zahlen müssten also zunächst mit Hilfe der Funktion `str( )` umgewandelt werden.
- Hintergrundinfo zur Einheit „dp“: Ein Display-Pixel (dp) ist ein von der Bildschirmauflösung unabhängiger Wert, der in etwa der Größe eines Pixels auf einem Laptop mit 72 Dots je Inch entspricht. Auf einem schlechteren/älteren Bildschirm entspricht ein Display-Pixel einem gewöhnlichen Pixel. Auf neueren High-End-Bildschirmen kann ein Display-Pixel auch 3 oder 4 Pixel breit sein. Ein Display-Pixel hat in etwa die gleiche relative Größe auf allen Endgeräten.