Guida Kivy

Indice

Che cos’è

Kivy è una libreria open-source, la si utilizza per lo sviluppo di interfaccie grafiche utenti(GUI) e applicazioni multipiattaforma, si possono sviluppare applicazioni che funzionano sui seguenti dispositivi: **Windows**, **macOS**, **Linux**, **Android** e **iOS**.

Caratteristiche principali

1. **Multipiattaforma**

   Supporta diverse piattaforme(desktop e mobile)

2. **Touch-friendly**

   Progettato per applicazioni touch-screen, supportando gesture come pinch, zoom, scroll e swipe

3. **Moderno e flessibile**

   Basato su OpenGL ES 2, offre un’interfaccia utente reattiva e fluida, con animazioni e transizioni fluide

4. **Orientato agli eventi**

   Utilizza un loop degli eventi simile a quello di altre librerie GUI come tkinter, ma con un sistema avanzato di gestione degli eventi

5. **Facitle personalizzazione**

   Supporta un linguaggio di markup chiamato ‘KV Language’ per definire facilmente il layout delle interfaccie

Introduzione

Come Installare kivy, si possono verificare dei scenari nei quali è complesso installare kivy e riscontrare degli errori fastidiosi, per evitare ciò è sufficente creare un ambiente virtuale con python, nel seguente modo

Su Windows

#scarica ed installa python dal seguente link oppure dal MS Store

https://www.python.org/

#creazione ambiente

‘’’python3 –m venv <nome_ambiente>’’’

#attivare l’ambiente virtuale

CMD = “.\<nome_ambiente>\Scripts\activate.bat”

PowerSheel = “per evitare bug inutili usa il CMD ;) oppure cambia sistema operativo”

Commento: a questo punto tutti i file e i pacchetti verrano eseguiti dentro l’ambiente virtuale, che a tua scelta e piacimento puoi clonare da un’altra parte

#installazione kivy

pip3 install kivy

#setup_completato

Su Linux

```
#installazione dei pacchetti
#Ubuntu/Debian
    sudo apt update #aggiorna pkg e repo
    sudo apt install -y python3 python3-pip python3-venv
#Arch
    sudo pacman -Syu #aggiorna pkg e repo
    sudo pacman -S python python-pip
#Fedora
    sudo dnf upgrade --refresh  #aggiorna pkg e repo
    sudo dnf install -y python3 python3-pip
#creazione dell'ambiente virtuale
python3 –m venv <nome_ambiente>
#attivazione dell'ambiente virtuale
source <nome_ambiente>/bin/activate
#installazione kivy
pip3 install kivy
#setup_completato
```

Primi Passi

Cominciamo stendendo i primi passi del codice che serviranno ad generare una semplice applicazione senza niente

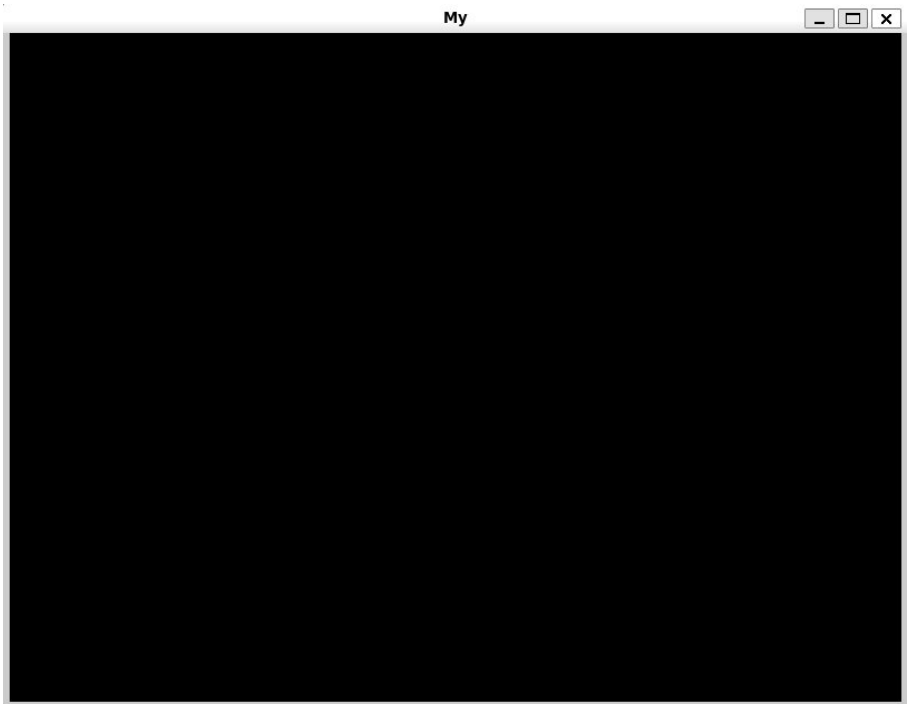
main.py

‘’’

```
from kivy.uix.gridlayout import GridLayout #importiamo il tipo di Layout

from kivy.app import App  #importiamo il fulcro dell’applicazione che la fa funzionare


class MyApp(App):  #creiamo una classe con l’elemento precedentemente importato

        def builld(self): #questa è una classe che permette la costruzione della schermata

                return GridLayout() #definiamo un layout a griglia e lo ritorniamo al main


MyApp().run() #la funzione .run() è una funzione dell’elemento App serve per avviare l’app
```

‘’’

Questo è l’output



Elementi

Layout in questa guida affronteremo solo 2 dei layout predisposti in kivy, partiamo intanto con il ***GridLayout*** la caratteristica principale di questo layout è che è come una griglia, della quale noi possiamo aumentare o diminuire le colonne a nostro piacimento, con il seguente attributo ***cols***

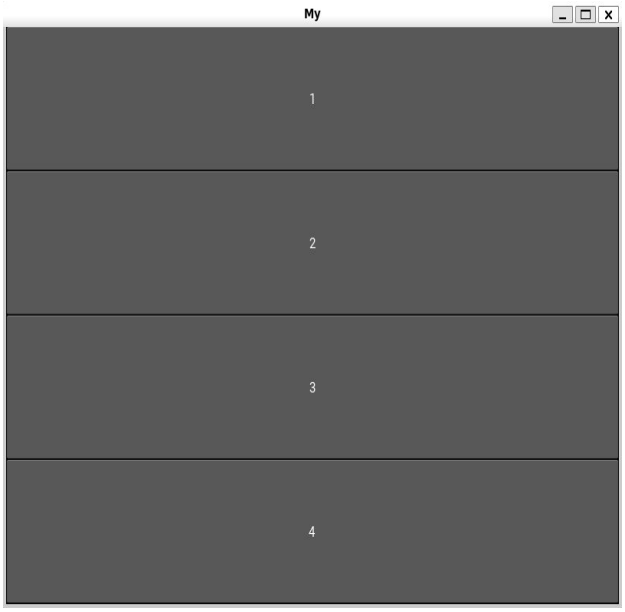
codice

‘’’

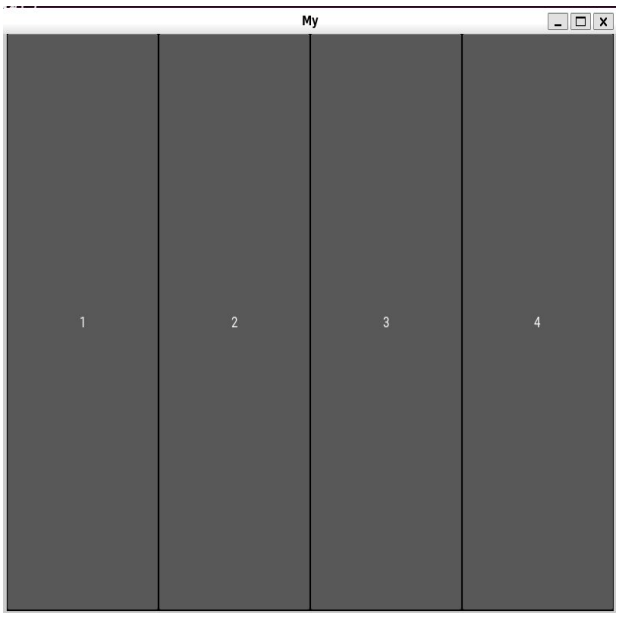
```
from kivy.uix.gridlayout import GridLayout #importiamo il tipo di Layout
from kivy.uix.button import Button #importiamo il bottone basilare
from kivy.app import App  #importiamo il fulcro dell’applicazione che la fa funzionare

class MyApp(App):  #creiamo una classe con l’elemento precedentemente importato
      def build(self): #questa è una classe che permette la costruzione della schermata
         self.main = GridLayout(cols=4, size=(1920, 1080)) #definiamo un layout a griglia e lo ritorniamo al main
         botton1 = Button(text="1")# definiamo un bottone bottone
         botton2 = Button(text="2")# definiamo un bottone bottone
         botton3 = Button(text="3")# definiamo un bottone bottone
         botton4 = Button(text="4")# definiamo un bottone bottone

         self.main.add_widget(botton1)#aggiungamo il bottone 1 al layout
         self.main.add_widget(botton2)#aggiungamo il bottone 2 al layout
         self.main.add_widget(botton3)#aggiungamo il bottone 3 al layout
         self.main.add_widget(botton4)#aggiungamo il bottone 4 al layout
         return self.main#ritorniamo il nostro layout
 MyApp().run() #la funzione .run() è una funzione dell’elemento App serve per avviare l’app
```

‘’’

cols=1



cols=4



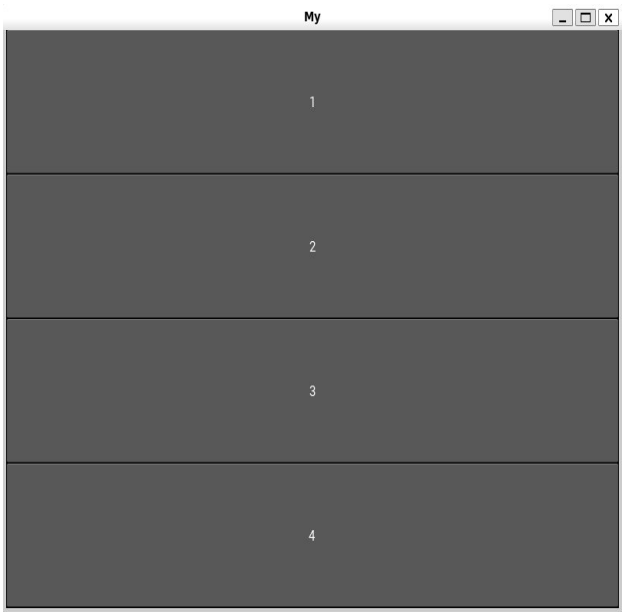
BoxLayout questo layout lavora sull'orientazione degli elementi quindi o verticale o orizontale con orientation="vertical" oppure "horizontal"

‘’’

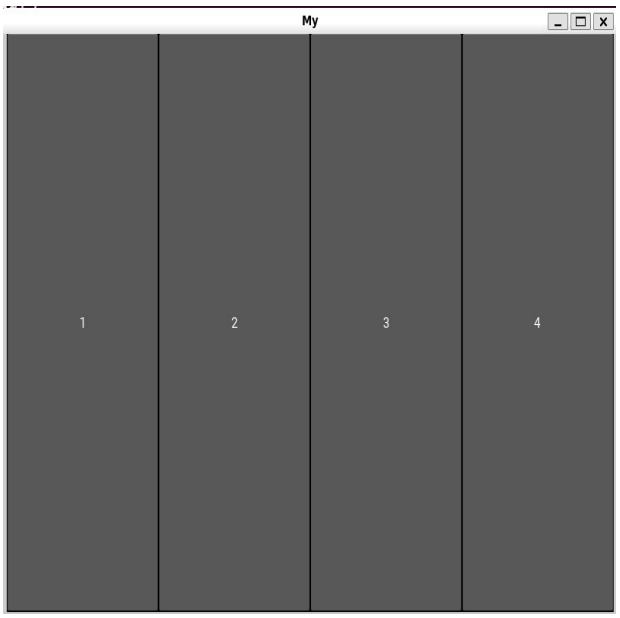
```
from kivy.uix.boxlayout import BoxLayout #importiamo il tipo di Layout
from kivy.uix.button import Button #importiamo il bottone basilare
from kivy.uix.label import Label #importiamo l'etichetta principale
from kivy.app import App  #importiamo il fulcro dell'applicazione che la fa funzionare
class MyApp(App):  #creiamo una classe con l'elemento precedentemente importato
        def build(self): #questa è una classe che permette la costruzione della schermata
                self.main = BoxLayout(orientation="horizontal", size=(1920, 1080)) #definiamo un layout a box

                botton1 = Button(text="1")# definiamo un bottone bottone
                botton2 = Button(text="2")# definiamo un bottone bottone
                botton3 = Button(text="3")# definiamo un bottone bottone
                botton4 = Button(text="4")# definiamo un bottone bottone

                self.main.add_widget(botton1)#aggiungamo il bottone 1 al layout
                self.main.add_widget(botton2)#aggiungamo il bottone 2 al layout
                self.main.add_widget(botton3)#aggiungamo il bottone 3 al layout
                self.main.add_widget(botton4)#aggiungamo il bottone 4 al layout

                return self.main#ritorniamo il nostro layout
MyApp().run() #la funzione .run() è una funzione dell'elemento App serve per avviare l'app
```

‘’’

orientatio="vertical”



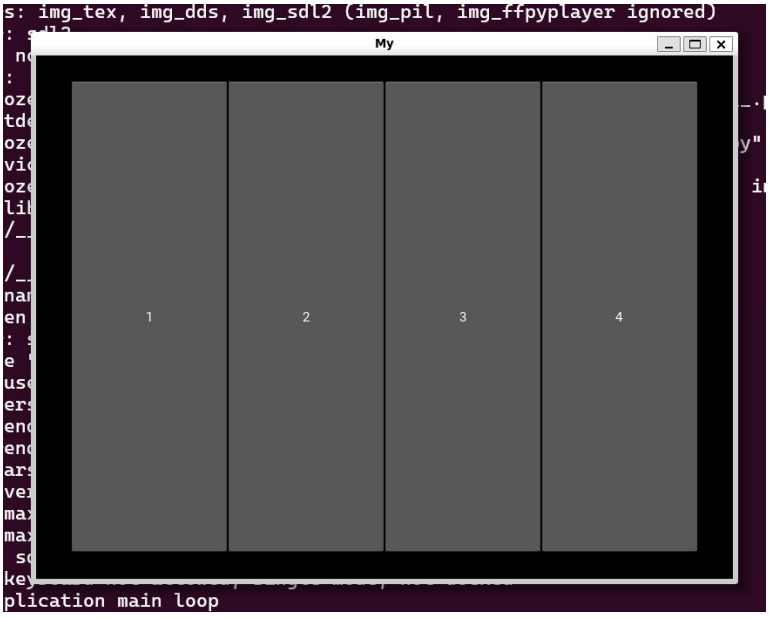
orientatio="horizontal”



Caratteristica comune di entrambi è che possiamo aggiungerli dei margini ed anche cambiarli le posizioni, all'interno dell'applicazione, con i seguenti metodi

Dimensione con .size_hint
Poszione con .pos_hint
‘’’

```
from kivy.uix.boxlayout import BoxLayout #importiamo il tipo di Layout
from kivy.uix.button import Button #importiamo il bottone basilare
from kivy.uix.label import Label #importiamo l'etichetta principale
from kivy.app import App  #importiamo il fulcro dell'applicazione che la fa funzionare
class MyApp(App):  #creiamo una classe con l'elemento precedentemente importato
def build(self): #questa è una classe che permette la costruzione della schermata
self.main = BoxLayout(orientation="horizontal", size=(1920, 1080)) #definiamo un layout a box
self.main.size_hint = (.9, .9)
self.main.pos_hint = {"center_x":.5, "center_y":.5}

botton1 = Button(text="1")# definiamo un bottone bottone
botton2 = Button(text="2")# definiamo un bottone bottone
botton3 = Button(text="3")# definiamo un bottone bottone
botton4 = Button(text="4")# definiamo un bottone bottone

self.main.add_widget(botton1)#aggiungamo il bottone 1 al layout
self.main.add_widget(botton2)#aggiungamo il bottone 2 al layout
self.main.add_widget(botton3)#aggiungamo il bottone 3 al layout
self.main.add_widget(botton4)#aggiungamo il bottone 4 al layout

return self.main#ritorniamo il nostro layout

MyApp().run() #la funzione .run() è una funzione dell'elemento App serve per avviare l'app
```
‘’’



Button

Il **Button** è uno dei widget più utilizzati in Kivy. Viene utilizzato per creare bottoni interattivi che possono rispondere agli input dell'utente (come i clic).

Esempio di utilizzo base:

```
from kivy.uix.button import Button
from kivy.app import App
class MyApp(App):
        def build(self): # Creiamo un bottone con un testo e dimensioni predefinite
                btn = Button(text="Cliccami!", font_size=24) # Aggiungiamo un'azione da eseguire
        quando il bottone viene premuto
                btn.bind(on_press=self.on_button_press)
                return btn
        def on_button_press(self, instance):
                print("Bottone premuto!")
MyApp().run()
```

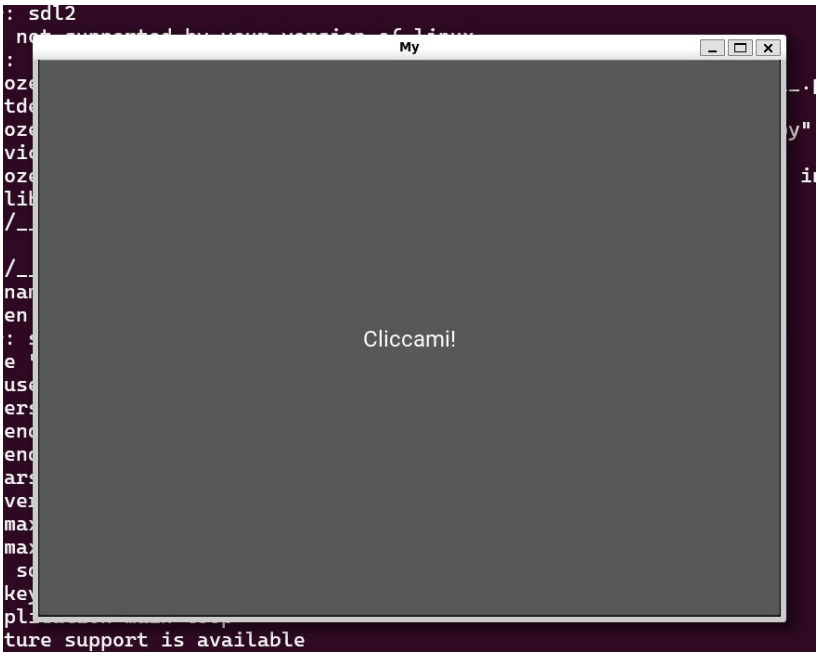
Spiegazione:

- **text**: Definisce il testo mostrato sul bottone.
- **font_size**: Permette di impostare la dimensione del font.
- **bind(on_press=...)**: Associa una funzione che verrà chiamata quando il bottone viene premuto.

Proprietà utili dei bottoni:

- **background_color**: Cambia il colore di sfondo.
- **color**: Cambia il colore del testo.
- **size_hint e pos_hint**: Modificano dimensioni e posizioni.

```
btn = Button(text="Colorato”, background_color=(0, 1, 0, 1),  # Verde     color=(1, 1, 1, 1),  # Testo bianco
        size_hint=(.5, .5),
        pos_hint={'center_x': 0.5, 'center_y': 0.5}
 )
```



Label

Il **Label** è un widget utilizzato per mostrare del testo statico.

Esempio di utilizzo:

```
from kivy.uix.label import Label
from kivy.app import App
class MyApp(App):
        def build(self):
                return Label(text="Benvenuto in Kivy!", font_size='32sp')
MyApp().run()
```

Proprietà utili:

- **text**: Contenuto da mostrare.
- **font_size**: Dimensione del testo.
- **color**: Colore del testo (in formato RGBA).
- **halign e valign**: Allineamento orizzontale e verticale.

Slider

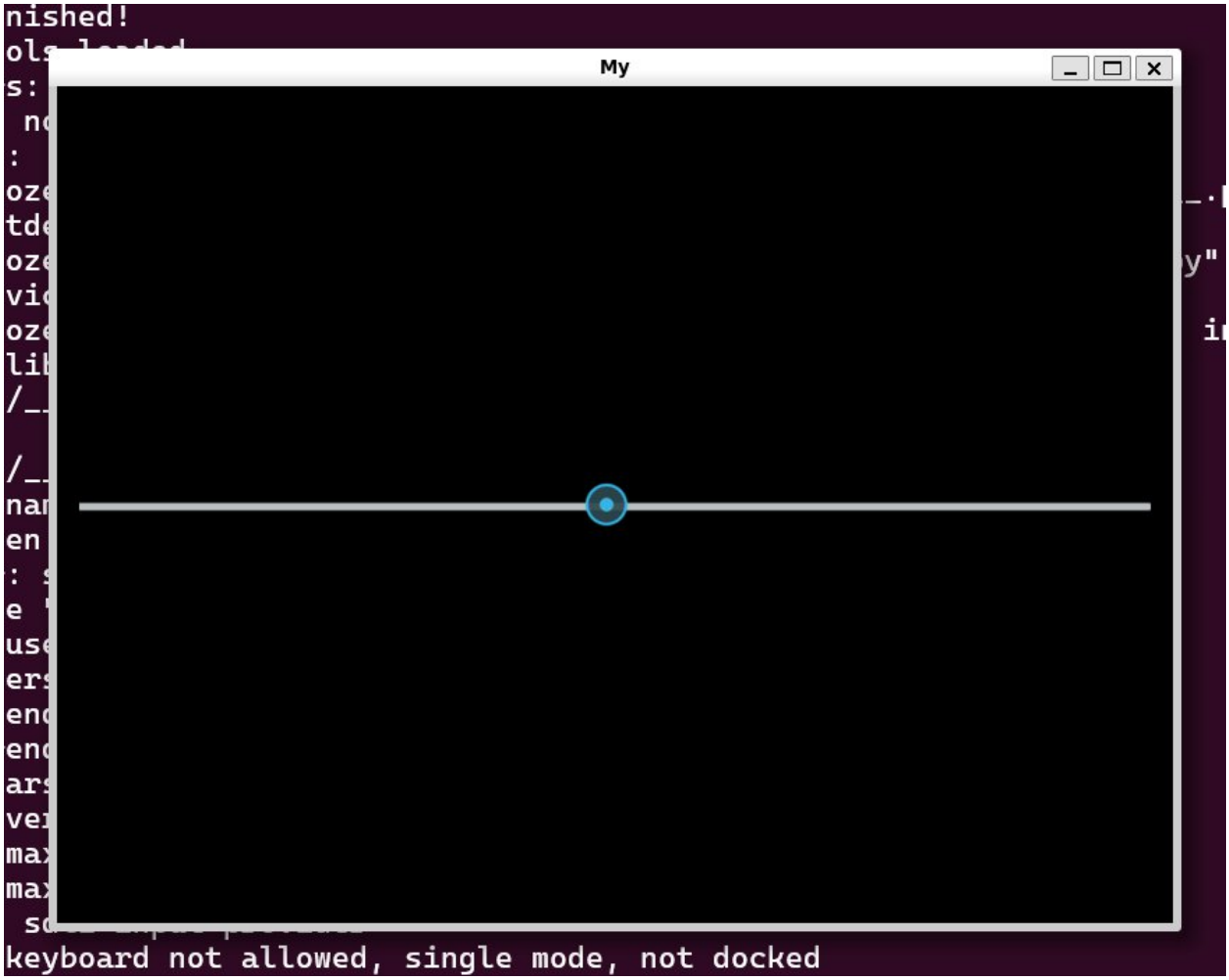
Lo **Slider** è un widget che permette agli utenti di selezionare un valore da un intervallo predefinito.

Esempio di utilizzo:

```
from kivy.uix.slider import Slider
from kivy.app import App
from kivy.uix.boxlayout import BoxLayout
class MyApp(App):
        def build(self):
                layout = BoxLayout(orientation='vertical')
                self.slider = Slider(min=0, max=100, value=50)
                layout.add_widget(self.slider)
                return layout
MyApp().run()
```

Proprietà utili:

- **min e max**: Definiscono l'intervallo dei valori.
- **value**: Il valore corrente dello slider.
- **step**: Imposta un incremento discreto.



TextInput

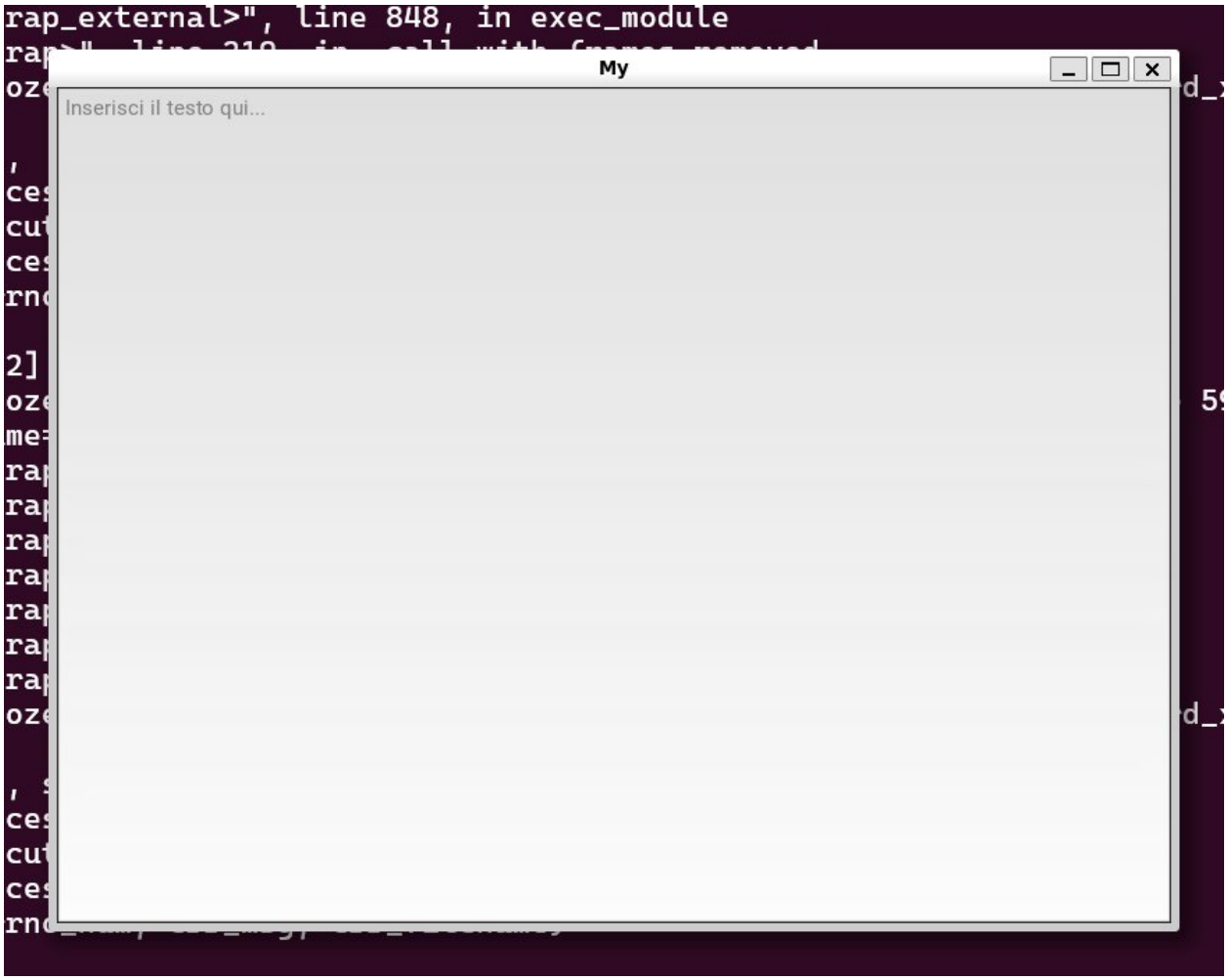
Il **TextInput** è utilizzato per permettere all'utente di inserire testo.

Esempio di utilizzo:

```
from kivy.uix.textinput import TextInput
from kivy.app import App
class MyApp(App):
        def build(self):
                return TextInput(hint_text="Inserisci il testo qui...", multiline=False)
MyApp().run()
```

Proprietà utili:

- **hint_text**: Testo di suggerimento che appare quando il campo è vuoto.
- **multiline**: Se impostato su False, consente solo una riga di testo.
- **password**: Se True, nasconde il testo (utile per password).



Conclusione

Kivy è una libreria potente e flessibile per creare applicazioni con interfacce grafiche sia su desktop che su dispositivi mobili. Utilizzando i vari elementi come **Button**, **Label**, **Slider**, e **TextInput**, è possibile costruire rapidamente applicazioni interattive e touch-friendly.