CALCULATOR

```
from kivy.app import App
from kivy.uix.button import Button
from kivy.uix.boxlayout import BoxLayout
from kivy.uix.gridlayout import GridLayout
from kivy.uix.label import Label
from kivy.uix.scrollview import ScrollView
from kivy.core.window import Window

class MyApp(App):
 def build(self):
 self.expression = ""
 self.history_visible = False
 self.history_items = []

 self.root = BoxLayout(orientation='vertical', padding=10, spacing=10)

 # Display Area
 display = BoxLayout(orientation='vertical', size_hint_y=0.4)

 self.expression_label = Label(text="", font_size=30, halign="left", valign="top")
self.expression_label.bind(size=self.expression_label.setter("text_size"))

 self.answer_label = Label(text="", font_size=40, halign="right", valign="bottom")
self.answer_label.bind(size=self.answer_label.setter("text_size"))

 display.add_widget(self.expression_label)
 display.add_widget(self.answer_label)

 # Button Grid
 button_grid = GridLayout(cols=4, spacing=5, size_hint_y=1.2)
buttons = (
 '7', '8', '9', '/',
 '4', '5', '6', '*',
 '1', '2', '3', '-',
 '0', '.', '=', '+'
 )
 for symbol in buttons:
 btn = Button(text=symbol, font_size=28)
 btn.bind(on_press=self.on_button_press)
 button_grid.add_widget(btn)
 # Controls: Clear, Delete, Show History
 controls = BoxLayout(size_hint_y=None, height=70, spacing=10)
 clear_btn = Button(text="Clear", font_size=20)
 clear_btn.bind(on_press=self.clear_all)
 del_btn = Button(text="⌫ Delete", font_size=20)
 del_btn.bind(on_press=self.delete_last)
 self.history_btn = Button(text="�� Show History", font_size=20)
 self.history_btn.bind(on_press=self.toggle_history)

 controls.add_widget(clear_btn)
 controls.add_widget(del_btn)
 controls.add_widget(self.history_btn)

 # History Panel (Initially Hidden)
 self.history_label = Label(text="�� History", font_size=20, size_hint_y=None, height=30)
self.history_layout = BoxLayout(orientation='vertical', size_hint_y=None)
self.history_layout.bind(minimum_height=self.history_layout.setter('height'))

 self.scroll = ScrollView(size_hint=(1, 0.6))
 self.scroll.add_widget(self.history_layout)

 # Add widgets
 self.root.add_widget(display)
 self.root.add_widget(button_grid)
 self.root.add_widget(controls)

 # Keyboard input
 Window.bind(on_key_down=self.on_key_down)

 return self.root

 def on_button_press(self, instance):
 symbol = instance.text
 if symbol == '=':
 self.evaluate()
 else:
 self.expression += symbol
 self.expression_label.text = self.expression
 def evaluate(self):
 if not self.expression.strip():
 return # Do nothing if empty
 try:
 result = str(eval(self.expression))
 self.answer_label.text = result
 self.add_to_history(self.expression, result)
 self.expression = result # Keep result for further use
 self.expression_label.text = self.expression
 except Exception:
 self.answer_label.text = "Error"

 def add_to_history(self, expr, result):
 history_text = f"{expr} = {result}"
 lbl = Label(text=history_text, font_size=18, size_hint_y=None, height=30, halign="left",
valign="middle")
 lbl.bind(size=lbl.setter("text_size"))
 self.history_layout.add_widget(lbl)

 def clear_all(self, instance):
 self.expression = ""
 self.expression_label.text = ""
 self.answer_label.text = ""

 def delete_last(self, instance):
 self.expression = self.expression[:-1]
 self.expression_label.text = self.expression

 def toggle_history(self, instance):
 if self.history_visible:
 self.root.remove_widget(self.history_label)
 self.root.remove_widget(self.scroll)
 self.history_btn.text = "�� Show History"
 else:
 self.root.add_widget(self.history_label)
 self.root.add_widget(self.scroll)
 self.history_btn.text = "�� Hide History"
 self.history_visible = not self.history_visible
 def on_key_down(self, window, key, scancode, codepoint, modifiers):
if codepoint in '0123456789+-*/.=()':
 if codepoint == '=':
 self.evaluate()
 else:
 self.expression += codepoint
 self.expression_label.text = self.expression
 elif key == 13: # Enter
 self.evaluate()
 elif key == 8: # Backspace
 self.delete_last(None)
 elif codepoint.lower() == 'c': # Clear shortcut
 self.clear_all(None)

# Run it!
MyApp().run()
```