

Создание приложения простого калькулятора с помощью библиотеки kivy на языке программирования Python

Языков Кирилл Александрович

Приамурский государственный университет имени Шолом-Алейхема

Студент

Аннотация

В статье рассматривается процесс разработки простого графического калькулятора с использованием библиотеки Kivy на языке программирования Python. Основное внимание уделяется созданию пользовательского интерфейса с визуальными элементами, такими как кнопки цифр и арифметических операций, строка ввода и окно истории вычислений. Приводится структура приложения, включая организацию виджетов с использованием контейнеров BoxLayout и GridLayout, а также реализацию обработки событий нажатия клавиш и взаимодействия с клавиатурой. Особое внимание уделяется обеспечению интерактивности, предотвращению ошибок ввода, корректной работе арифметических операций и сохранению истории вычислений. Статья демонстрирует практический пример интеграции графического интерфейса и логики вычислений, что делает её полезной для начинающих разработчиков, изучающих создание настольных приложений на Python с графическим интерфейсом.

Ключевые слова: Python, Kivy, графический интерфейс, калькулятор, интерактивное приложение.

Creating a simple calculator app using the kivy library in the Python programming language

Yazykov Kirill Alexandrovich

Sholom-Aleichem Priamursky State University

Student

Abstract

The article examines the process of developing a simple graphical calculator using the Kivy library in the Python programming language. The main focus is on creating a user interface with visual elements such as digit and arithmetic operation buttons, an input field, and a calculation history window. The structure of the application is presented, including the organization of widgets using BoxLayout and GridLayout containers, as well as the implementation of event handling for key presses and keyboard interaction. Special attention is given to ensuring interactivity, preventing input errors, correct operation of arithmetic functions, and maintaining a history of calculations. The article demonstrates a practical example of integrating a graphical

interface with computational logic, making it useful for beginner developers learning to create desktop applications in Python with a graphical user interface.

Keywords: Python, Kivy, graphical interface, calculator, interactive application.

1 Введение

1.1 Актуальность

Калькулятор является классическим примером обучающего проекта для изучения основ программирования, разработки пользовательского интерфейса и обработки событий, что делает его удобной практической задачей для начинающих разработчиков. Использование библиотеки Kivy позволяет создавать кроссплатформенные графические приложения с современным интерфейсом, что особенно актуально в условиях растущей популярности мобильных и настольных приложений. разработка интерактивного калькулятора с функционалом ввода с клавиатуры, визуализацией вычислений и сохранением истории способствует пониманию принципов интеграции логики программы и графического интерфейса.

1.2 Обзор исследований

Для ознакомления с языком программирования Python можно познакомиться с различными исследованиями. Г.С. Сейдаметов продемонстрировал особенности использования языка программирования python в подготовке будущих инженеров-программистов [2]. Э.А. Усеинов продемонстрировал использование объектно-ориентированного программирования в рамках дисциплины «язык программирования python» [3]. Д.В. Шкодырев продемонстрировал использование языка программирования python и системы компьютерной алгебры sympy на факультативных занятиях по теории чисел» [4]. В.В. Бурков и И.В. Бантыш показали использование языка программирования python для решения задач математического моделирования электромеханических систем [5].

1.3 Цель исследования

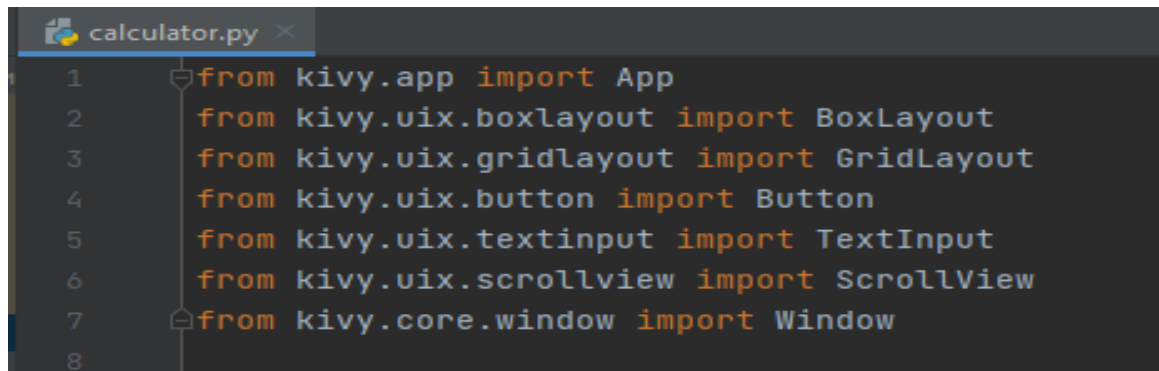
Цель исследования – разработать простое интерактивное приложение калькулятора с графическим интерфейсом на языке программирования Python с использованием библиотеки Kivy.

2 Материалы и методы

В данном исследовании для разработки приложения калькулятора использовалась интегрированная среда разработки PyCharm. Язык программирования Python был выбран благодаря своей простоте, читаемости и широким возможностям для создания приложений. Для создания графического интерфейса использовалась библиотека Kivy [1]. Она предоставляет инструменты для создания кроссплатформенных приложений с гибким пользовательским интерфейсом.

Основным методом исследования является практическая реализация приложения с использованием библиотеки Kivy. В рамках исследования были

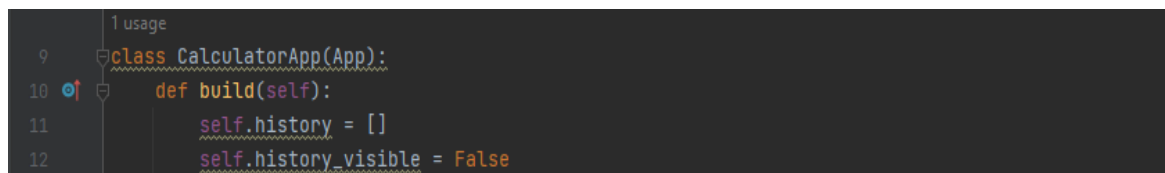
Теперь следует создать новый файл, например, `calculator.py` в главном проекте IDE. Код начинается с импорта `kivy` некоторых модулей (рис. 3).



```
calculator.py x
1  from kivy.app import App
2  from kivy.ui.boxlayout import BoxLayout
3  from kivy.ui.gridlayout import GridLayout
4  from kivy.ui.button import Button
5  from kivy.ui.textinput import TextInput
6  from kivy.ui.scrollview import ScrollView
7  from kivy.core.window import Window
8
```

Рисунок 3 – Импорт модулей `kivy`

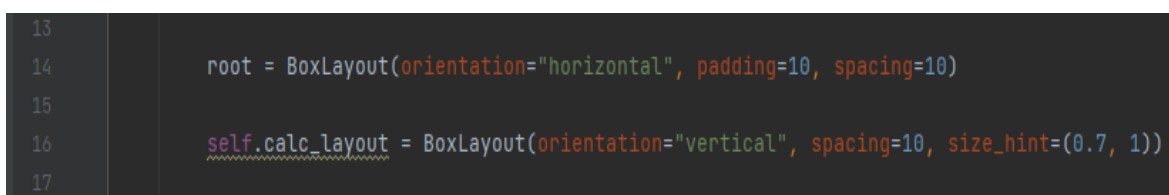
Следующим этапом необходимо создать класс. В данном классе будут размещены методы, которые будут вызываться при запуске приложения и хранить историю калькулятора и его отображение (рис. 4).



```
1 usage
9  class CalculatorApp(App):
10     def build(self):
11         self.history = []
12         self.history_visible = False
```

Рисунок 4 – Создание класса `CalculatorApp`

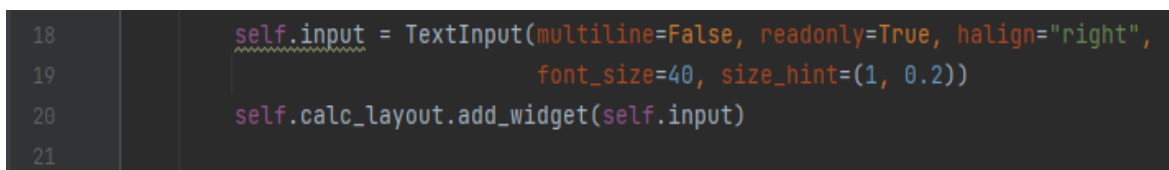
В следующей строке кода будет указан контейнер, в котором прописаны правила размещения кнопок, а также их размер (рис. 5).



```
13
14     root = BoxLayout(orientation="horizontal", padding=10, spacing=10)
15
16     self.calc_layout = BoxLayout(orientation="vertical", spacing=10, size_hint=(0.7, 1))
17
```

Рисунок 5 – Создание контейнера для размера кнопок и отступов

Следующим этапом следует уделить внимание строке, в которую будут вводиться значения, чтобы она была во всю ширину приложения (рис. 6).



```
18     self.input = TextInput(multiline=False, readonly=True, halign="right",
19                           font_size=40, size_hint=(1, 0.2))
20     self.calc_layout.add_widget(self.input)
21
```

Рисунок 6 – Код для строки ввода калькулятора

В данном коде прописаны выравнивание по правому краю, размер шрифта и процент занимаемый от всего приложения.

Далее создадим четыре кнопки: удаление одного символа, полная очистка содержимого строки ввода, история вычислений и вычислить. Каждая из них содержит обработчик нажатия и располагаться они будут в главном контейнере (рис. 7).

```

22 control_layout = GridLayout(cols=4, spacing=10, size_hint=(1, 0.1))
23 for label in ['←', 'C', 'M', '=']:
24     control_layout.add_widget(Button(text=label, font_size=28, on_press=self.on_control_press))
25 self.calc_layout.add_widget(control_layout)
26

```

Рисунок 7 – Создание 4 кнопок

На следующем этапе следует создать основную сетку кнопок калькулятора: цифры и арифметические операции. Как и для других кнопок нужно прописать размеры и добавить обработчик нажатия (рис. 8).

```

27 buttons = [
28     ['7', '8', '9', '*'],
29     ['4', '5', '6', '/'],
30     ['1', '2', '3', '+'],
31     ['.', '0', '00', '-']
32 ]
33 button_layout = GridLayout(cols=4, spacing=10, size_hint=(1, 0.7))
34 for row in buttons:
35     for label in row:
36         button_layout.add_widget(Button(text=label, font_size=32, on_press=self.on_button_press))
37 self.calc_layout.add_widget(button_layout)

```

Рисунок 8 – Создание основной сетки кнопок

Так как ранее была создана кнопка истории, следует разделить калькулятор на правую и левую часть, чтобы можно было скрывать ее, нужно поместить `self.calc_layout` в основной контейнер `root` (рис. 9).

```

39 root.add_widget(self.calc_layout)
40

```

Рисунок 9 – Разделение калькулятора на основную часть и с хранением истории

Далее необходимо немного описать контейнер истории: размеры скролл истории, сохранение записи (рис. 10).

```

41 self.history_layout = BoxLayout(orientation="vertical", size_hint=(0, 1))
42 self.history_scroll = ScrollView()
43 self.history_grid = GridLayout(cols=1, spacing=5, size_hint_y=None)
44 self.history_grid.bind(minimum_height=self.history_grid.setter('height'))
45 self.history_scroll.add_widget(self.history_grid)
46 self.history_layout.add_widget(self.history_scroll)
47 root.add_widget(self.history_layout)

```

Рисунок 10 – Код для описания размеров и характеристик контейнера истории

На следующем этапе по желанию можно привязать клавиатуру к калькулятору, чтобы не использовать только нажатия мыши, `on_key_down` будет вызываться при каждом нажатии клавиши (рис. 11).

```
49         Window.bind(on_key_down=self.on_key_down)
50
51     return root
52
53     1 usage
54     def on_button_press(self, instance):
55         self.add_char(instance.text)
56
57     1 usage
58     def on_control_press(self, instance):
59         if instance.text == '←':
60             self.input.text = self.input.text[:-1]
61         elif instance.text == 'C':
62             self.input.text = ''
63         elif instance.text == 'M':
64             self.toggle_history()
65         elif instance.text == '=':
66             self.calculate()
67
68     5 usages
69     def add_char(self, char):
70         if char in '+-*/':
71             if self.input.text and self.input.text[-1] in '+-*/':
72                 self.input.text = self.input.text[:-1] + char
73             elif self.input.text:
74                 self.input.text += char
75         else:
76             self.input.text += char
```

Рисунок 11 – Код для привязки клавиатуры к калькулятору

Чтобы интерактивно работать с историей: загружать прошлые вычисления, добавление новых вычислений в историю, вывод ошибки (например, деление на 0), раскрытие и скрытие окна истории, меняя ширину контейнера, нужно добавить следующий код (рис. 12).

```
2 usages
75 def calculate(self):
76     try:
77         result = str(eval(self.input.text))
78         self.history.append(f"{self.input.text} = {result}")
79         self.input.text = result
80         self.update_history()
81     except Exception:
82         self.input.text = "Ошибка"
83
2 usages
84 def toggle_history(self):
85     if self.history_visible:
86         self.history_layout.size_hint_x = 0
87         self.history_visible = False
88     else:
89         self.history_layout.size_hint_x = 0.3
90         self.history_visible = True
91
1 usage
92 def update_history(self):
93     self.history_grid.clear_widgets()
94     for item in reversed(self.history):
95         btn = Button(text=item, size_hint_y=None, height=40)
96         btn.bind(on_press=self.load_history)
97         self.history_grid.add_widget(btn)
98
1 usage
99 def load_history(self, instance):
100     expr = instance.text.split('=')[0].strip()
101     self.input.text = expr
```

Рисунок 12 – Код для обработки истории калькулятора

Последний этап создания калькулятора – прописать все цифры и арифметические знаки, которые расположены на нумрад и горячие клавиши для первых 4 кнопок, которые создавались ранее (рис. 13).

```
102
103     1 usage
104     def on_key_down(self, window, key, scancode, codepoint, modifiers):
105         if 48 <= key <= 57:
106             self.add_char(chr(key))
107         if codepoint is not None and codepoint in '+-*/':
108             self.add_char(codepoint)
109         if key == 8:
110             self.input.text = self.input.text[:-1]
111         if key == 13:
112             self.calculate()
113         if codepoint is not None and codepoint in 'mM':
114             self.toggle_history()
115         if codepoint is not None and codepoint in 'cC':
116             self.input.text = ''
117
118         numpad_digits = {
119             256: '0',
120             257: '1',
121             258: '2',
122             259: '3',
123             260: '4',
124             261: '5',
125             262: '6',
126             263: '7',
127             264: '8',
128             265: '9'
129         }
130         if key in numpad_digits:
131             self.add_char(numpad_digits[key])
132
133         numpad_ops = {
134             270: '+',
135             269: '-',
136             268: '*',
137             267: '/',
138             271: '.',
139         }
140         if key in numpad_ops:
141             self.add_char(numpad_ops[key])
142
143     if __name__ == "__main__":
144         CalculatorApp().run()
```

Рисунок 13 – Код для обработки истории калькулятора

Приложение калькулятор готово к запуску (рис. 14).

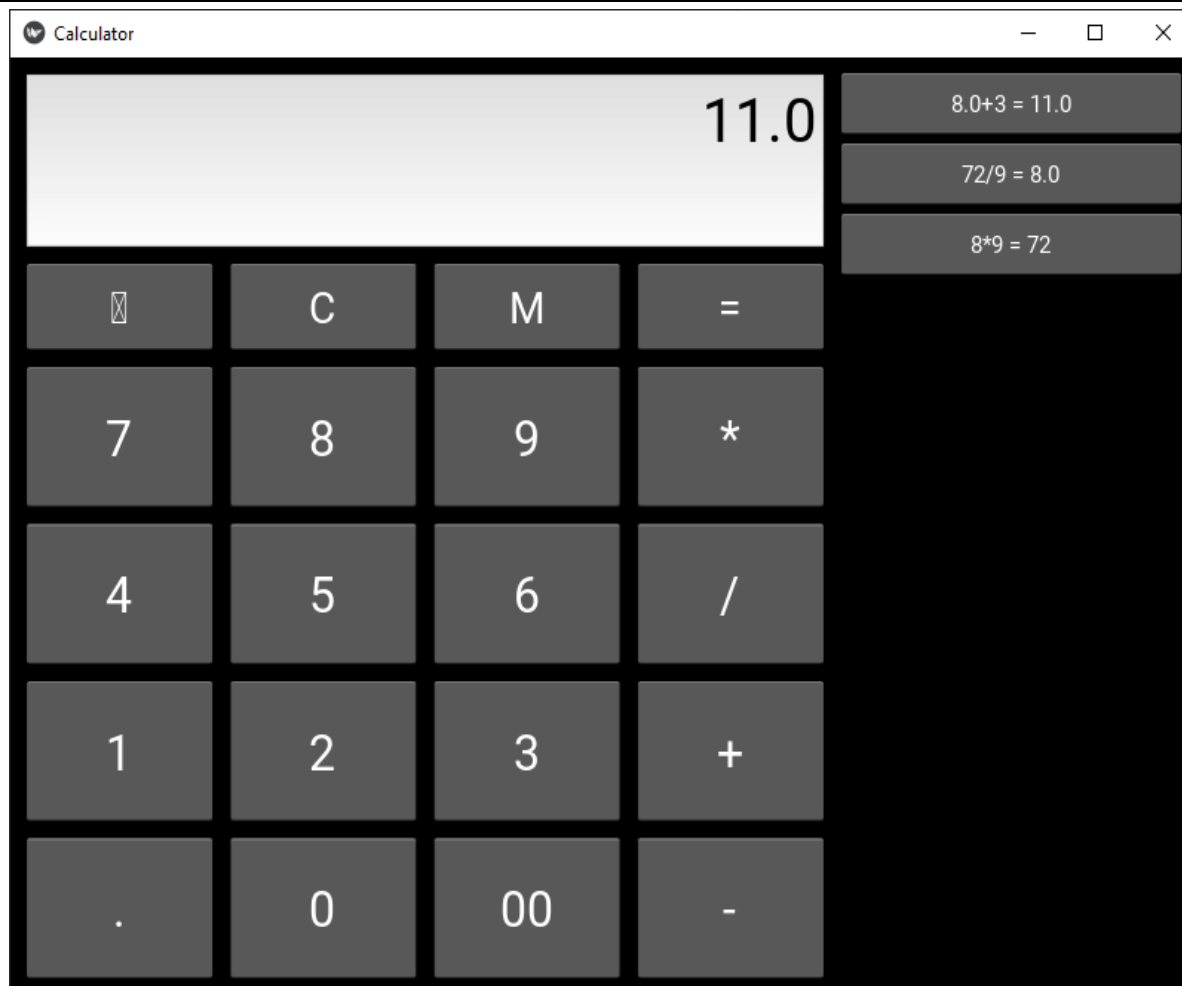


Рисунок 14 – Приложение калькулятор

4 Выводы

В результате проведённого исследования разработан простой графический калькулятор на языке Python с использованием библиотеки Kivy. Приложение демонстрирует создание удобного и интуитивно понятного интерфейса с визуальными элементами, такими как кнопки цифр и арифметических операций, строка ввода и окно истории вычислений.

Реализована корректная обработка пользовательских действий, взаимодействие с клавиатурой и сохранение истории операций.

Разработка такого приложения позволяет начинающим разработчикам получить практический опыт интеграции логики вычислений с графическим интерфейсом и служит наглядным примером создания интерактивных настольных приложений на Python.

Библиографический список

1. Фреймворк Kivy URL: <https://kivy.org> (дата обращения: 15.11.2025).
2. Сейдаметов Г. С. Особенности использования языка программирования python в подготовке будущих инженеров-программистов //International scientific review. 2015. №. 9 (10). С. 50-51.
3. Усеинов Э. А. Объектно-ориентированное программирование в рамках

- дисциплины «язык программирования Python» // //Ученые записки крымского инженерно-педагогического университета. 2012. №. 34. С. 157-160.
4. Шкодырев Д. В. Использование языка программирования Python 3 и системы компьютерной алгебры SymPy на факультативных занятиях по теории чисел // Математическое образование в школе и вузе: теория и практика (MathEdu-2015): материалы V Международной научно-практической конференции / под ред. Н. В. Тимербаевой. Пенза, 2015. С. 287–288.
 5. Бурков В. В., Бантыш И. В. Использование языка программирования Python для решения задач математического моделирования электромеханических систем // Современные проблемы автоматизации и управления в энергетике и машиностроении: сборник научных трудов международной научно-практической конференции, Пенза, 27 ноября 2015 года / Цикл «Автоматизация и управления» кафедры «Технологии машиностроения» (ТМС) Пензенского государственного технологического университета. Пенза: Пензенский государственный технологический университет, 2015. С. 226-234.