Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт перспективной инженерии
Департамент цифровых, робототехнических систем и электроники

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ № 7
дисциплины
«Объектно-ориентированное программирование»
Вариант 15

Выполнил:
Степанов Леонид Викторович
3 курс, группа ИВТ-б-о-22-1,
09.03.01 «Информатика и вычислительная техника»,
направленность (профиль) «Программное обеспечение средств вычислительной техники и автоматизированных систем», очная форма обучения

_____
(подпись)

Проверил:
Богданов Сергей Сергеевич,
Ассистент департамента цифровых, робототехнических систем и электроники

_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2024 г.

Тема: Основы работы с Tkinter

Цель: приобретение навыков построения графического интерфейса пользователя GUI с помощью пакета Tkinter языка программирования Python версии 3.x.

Порядок выполнения работы:

Решите задачу: напишите простейший калькулятор, состоящий из двух текстовых полей, куда пользователь вводит числа, и четырех кнопок "+", "-", "*", "/". Результат вычисления должен отображаться в метке. Если арифметическое действие выполнить невозможно (например, если были введены буквы, а не числа), то в метке должно появляться слово "ошибка".

```
def calculate(operation):
    try:
        num1 = float(entry1.get())
        num2 = float(entry2.get())

        if operation == "+":
            result = num1 + num2
        elif operation == "-":
            result = num1 - num2
        elif operation == "*":
            result = num1 * num2
        elif operation == "/":
            result = num1 / num2 if num2 != 0 else "ошибка"

        label_result.config(text=str(result))
    except ValueError:
        label_result.config(text="ошибка")

# Создание основного окна
root = tk.Tk()
root.title("Простейший калькулятор")

# Поля ввода
entry1 = tk.Entry(root)
entry1.pack()

entry2 = tk.Entry(root)
entry2.pack()

# Кнопки операций
button_add = tk.Button(root, text="+", command=lambda: calculate("+"), width=20)
button_add.pack()

button_subtract = tk.Button(root, text="-", command=lambda: calculate("-"), width=20)
button_subtract.pack()

button_multiply = tk.Button(root, text="*", command=lambda: calculate("*"), width=20)
button_multiply.pack()

button_divide = tk.Button(root, text="/", command=lambda: calculate("/"), width=20)
button_divide.pack()

# Метка для отображения результата
label_result = tk.Label(root, text="")
label_result.pack()
```

Рисунок 1 – Код калькулятора

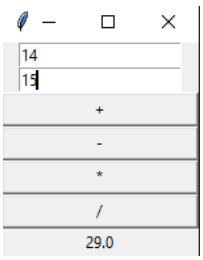

Рисунок 2 – Результат работы

Решите задачу: напишите программу, состоящую из семи кнопок, цвета которых соответствуют цветам радуги. При нажатии на ту или иную кнопку в текстовое поле должен вставляться код цвета, а в метку – название цвета.

```
def calculate(operation):
    try:
        print(operation)
        if operation == "red":
            result = '#ff0000'
        elif operation == "orange":
            result = '#ff7d00'
        elif operation == "yellow":
            result = '#ffff00'
        elif operation == "green":
            result = '#00ff00'
        elif operation == "blue":
            result = '#007dff'
        elif operation == "blue2":
            result = '#0000ff'
        elif operation == "phioll":
            result = '#7d00ff'

        entry1.config(text=str(result))
    except ValueError:
        entry1.config(text="ошибка")
```

Рисунок 3 – Код вывода цветов радуги

```
# Создание основного окна
root = tk.Tk()
root.title("Простейший калькулятор")
root.withdraw = 100

# Поля ввода
entry1 = tk.Label(root)
entry1.pack()

# Кнопки операций
button_red = tk.Button(root, command=lambda: calculate("red"), width=20, background='#ff0000')
button_red.pack()

button_orange = tk.Button(root, command=lambda: calculate("orange"), width=20, background='#ff7d00')
button_orange.pack()

button_yellow = tk.Button(root, command=lambda: calculate("yellow"), width=20, background='#ffff00')
button_yellow.pack()

button_green = tk.Button(root, command=lambda: calculate("green"), width=20, background='#00ff00')
button_green.pack()

button_blue = tk.Button(root, command=lambda: calculate("blue"), width=20, background='#007dff')
button_blue.pack()

button_blue2 = tk.Button(root, command=lambda: calculate("blue2"), width=20, background='#0000ff')
button_blue2.pack()

button_ = tk.Button(root, command=lambda: calculate("phioll"), width=20, background='#7d00ff')
button_.pack()


# Запуск основного цикла
root.mainloop()
```

Рисунок 4 – Код создания окна

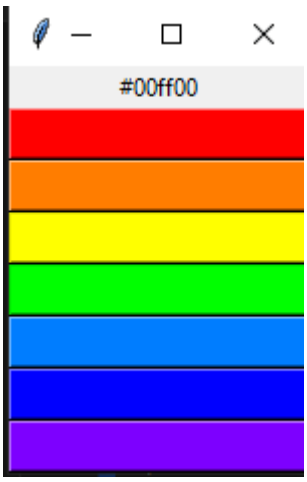

Рисунок 5– Результат работы

Решите задачу: перепишите программу из задачи выше так, чтобы интерфейс выглядел примерно следующим образом (рис. 6):

Рисунок 6 – Условие задачи

```
# Кнопки операций
button_red = tk.Button(root, command=lambda: calculate("red"), width=5, background='#ff0000')
button_red.pack(side=tk.LEFT)

button_orange = tk.Button(root, command=lambda: calculate("orange"), width=5, background='#ff7d00')
button_orange.pack(side=tk.LEFT)

button_yellow = tk.Button(root, command=lambda: calculate("yellow"), width=5, background='#ffff00')
button_yellow.pack(side=tk.LEFT)

button_green = tk.Button(root, command=lambda: calculate("green"), width=5, background='#00ff00')
button_green.pack(side=tk.LEFT)

button_blue = tk.Button(root, command=lambda: calculate("blue"), width=5, background='#007dff')
button_blue.pack(side=tk.LEFT)

button_blue2 = tk.Button(root, command=lambda: calculate("blue2"), width=5, background='#0000ff')
button_blue2.pack(side=tk.LEFT)

button_ = tk.Button(root, command=lambda: calculate("phioll"), width=5, background='#7d00ff')
button_.pack(side=tk.LEFT)
```

Рисунок 7 – Фрагмент исправленного кода

Рисунок 8 – Результат

Решите задачу: напишите программу, состоящую из однострочного и многострочного текстовых полей и двух кнопок "Открыть" и "Сохранить". При клике на первую должен открываться на чтение файл, чье имя указано в поле класса Entry, а содержимое файла должно загружаться в поле типа Text .

```
def open_file():
    filename = entry_filename.get()
    try:
        with open(filename, 'r', encoding='utf-8') as file:
            content = file.read()
            text_area.delete(1.0, tk.END)  # Очистка текстового поля
            text_area.insert(tk.END, content)  # Вставка содержимого файла
    except FileNotFoundError:
        messagebox.showerror("Ошибка", "Файл не найден.")
    except Exception as e:
        messagebox.showerror("Ошибка", str(e))
```

Рисунок 9 – Команда открытия файла

```
def save_file():
    filename = entry_filename.get()
    try:
        with open(filename, 'w', encoding='utf-8') as file:
            content = text_area.get(1.0, tk.END)  # Получение текста из текстового поля
            file.write(content.strip())  # Сохранение текста в файл
    except Exception as e:
        messagebox.showerror("Ошибка", str(e))
```

Рисунок 10 – Команда сохранения

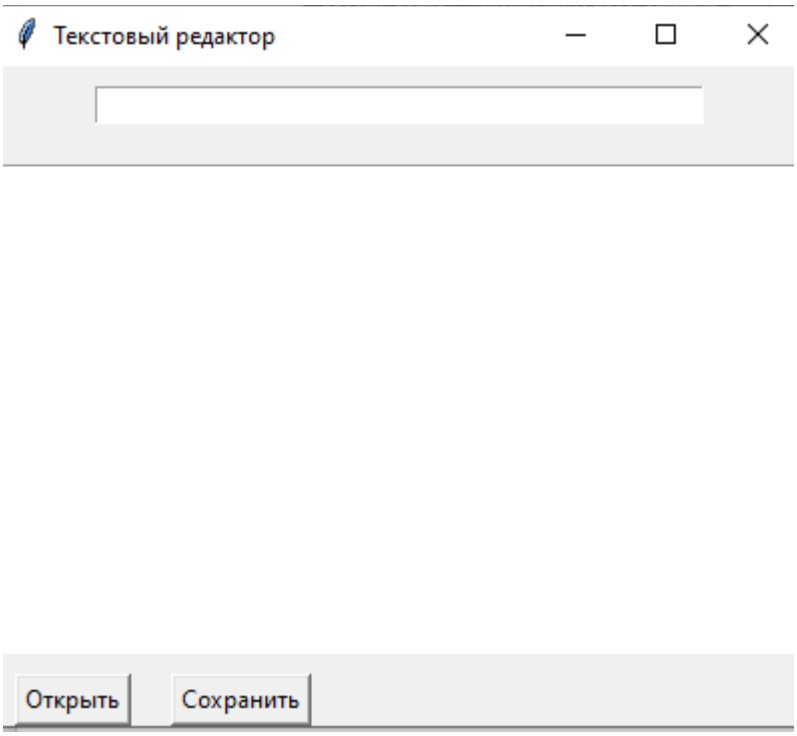

Рисунок 11 – Интерфейс

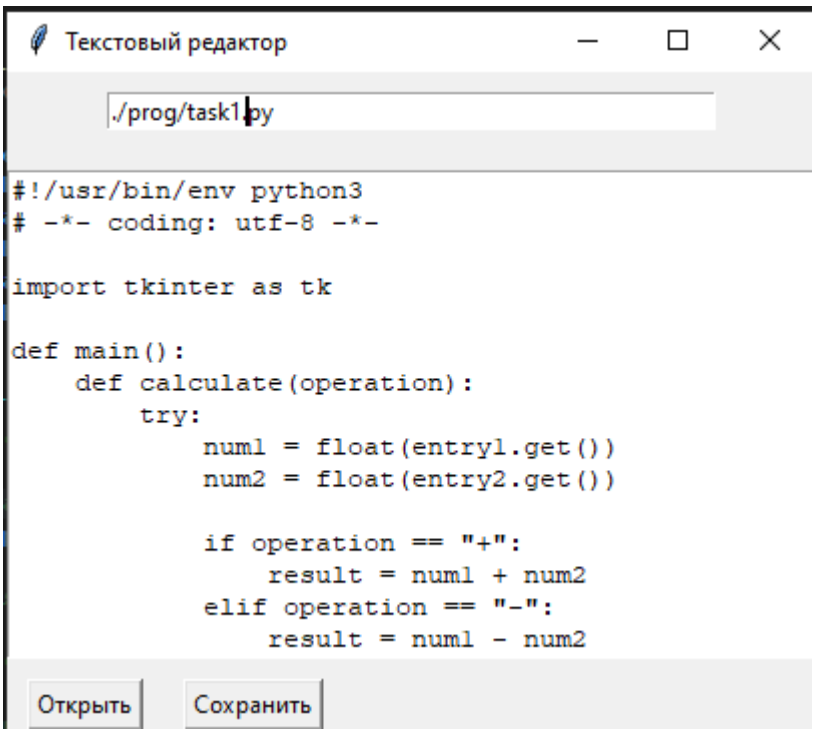

Рисунок 12 – Результат открытия файла

Решите задачу: виджеты Radiobutton и Checkbutton поддерживают большинство свойств оформления внешнего вида, которые есть у других элементов графического интерфейса. При этом у Radiobutton есть особое свойство indicatoron. По-умолчанию он равен единице, в этом случае радиокнопка выглядит как нормальная радиокнопка. Однако если присвоить этой опции ноль, то виджет Radiobutton становится похожим на обычную кнопку по внешнему виду. Но не по смыслу.

```
def update_label():
    """Обновляет текст метки в зависимости от выбранной радиокнопки."""
    name = selected_option.get()
    if name == 'Вася':
        label.config(text=f"+ 7 (985) 423-23-54")
    elif name == 'Петя':
        label.config(text=f"+ 7 (953) 411-45-54")
    elif name == 'Маша':
        label.config(text=f"+ 7 (963) 493-23-57")
```

Рисунок 13 – Реализация команды по изменению label

```python
# Создание главного окна
root = tk.Tk()
root.title("Radiobuttons with indicatoron=0")

# Переменная для отслеживания выбранной кнопки
selected_option = tk.StringVar(value="Ничего не выбрано")

# Метка для отображения информации
label = tk.Label(root, text="Ничего не выбрано", font=("Arial", 12))
label.pack(side=tk.RIGHT)

# Радиокнопки
options = ["Вася", "Петя", "Маша"]
for option in options:
    rb = tk.Radiobutton(
        root,
        text=option,
        value=option,
        variable=selected_option,
        command=update_label,
        indicatoron=0,  # Отключение индикатора
        width=15,
        padx=5,
        pady=5,
        font=15
    )
    rb.pack()

# Запуск главного цикла приложения
root.mainloop()
```

Рисунок 14 – Реализация интерфейса

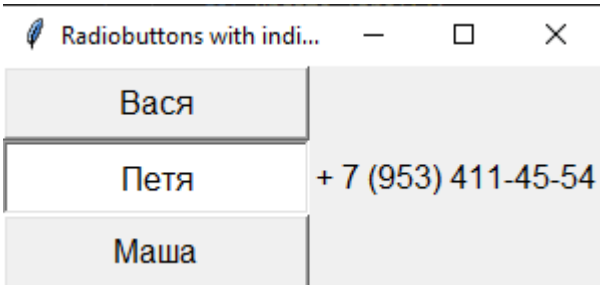

Рисунок 15 – Результат работы

Ссылка на github: https://github.com/FiaLDI/lab4.7

Контрольные вопросы

1. Какие существуют средства в стандартной библиотеке Python для построения графического интерфейса пользователя?

В стандартной библиотеке Python для создания графического интерфейса (GUI) можно использовать: tkinter — наиболее распространённая библиотека для создания GUI, turtle — простая графическая библиотека для обучения и рисования, IDLE — встроенная среда разработки Python, которая тоже построена с использованием Tkinter.

Для более сложных интерфейсов часто используют сторонние библиотеки, такие как PyQt, PySide, Kivy или wxPython.

2. Что такое Tkinter?

Tkinter — это стандартная библиотека Python для создания графического интерфейса пользователя. Она предоставляет обёртку над инструментарием Tk, который является кроссплатформенным GUI-фреймворком. Tkinter поддерживает создание окон, кнопок, текстовых полей, меток, списков, радиокнопок и других элементов интерфейса.

3. Какие требуется выполнить шаги для построения графического интерфейса с помощью Tkinter?

1) Импортировать модуль Tkinter: import tkinter as tk
2) Создать главное окно: root = tk.Tk()
3) Добавить виджеты (элементы интерфейса): Например, кнопки, метки, текстовые поля.
4) Разместить виджеты: Использовать методы pack(), grid() или place() для расположения элементов в окне.
5) Запустить цикл обработки событий: root.mainloop()

4. Что такое цикл обработки событий?

Цикл обработки событий — это процесс, при котором приложение ожидает действий пользователя (например, нажатия кнопки, ввода текста) и реагирует на них. В Tkinter этот цикл запускается с помощью метода mainloop(). Он обрабатывает все события, поступающие в программу.

5. Каково назначение экземпляра класса Tk при построении графического интерфейса с помощью Tkinter?

Экземпляр класса Tk представляет главное окно приложения. Он создаёт базовое окно, в котором размещаются все виджеты. Это окно управляет жизненным циклом GUI, включая отрисовку, обновление и закрытие.

6. Для чего предназначены виджеты Button, Label, Entry и Text?

Button — кнопка, на которую можно нажать для выполнения действия. Label — текстовая метка, предназначенная для отображения текста или изображений. Entry — однострочное текстовое поле для ввода данных. Text — многострочное текстовое поле для ввода и редактирования текста.

7. Каково назначение метода pack() при построении графического интерфейса пользователя?

Метод pack() используется для автоматического размещения виджетов в контейнере (например, главном окне или рамке). Он упрощает компоновку элементов, размещая их по порядку в зависимости от направления (по вертикали или горизонтали).

8. Как осуществляется управление размещением виджетов с помощью метода pack()?

Параметры управления: side — определяет сторону контейнера, к которой примыкает виджет (top, bottom, left, right). fill — позволяет виджету заполнять доступное пространство (x, y, both). expand — указывает, должен ли виджет расширяться при увеличении размера контейнера. padx и pady — добавляют отступы по горизонтали и вертикали.

9. Как осуществляется управление полосами прокрутки в виджете Text?

Для управления полосами прокрутки необходимо:

1) Создать объект Scrollbar.

2) Связать его с виджетом Text с помощью параметров yscrollcommand (для вертикальной прокрутки) или xscrollcommand (для горизонтальной).

3) Связать Scrollbar с методом прокрутки Text.

10. Для чего нужны тэги при работе с виджетом Text?

Тэги позволяют применять различные стили или действия к определённым частям текста в виджете Text. С их помощью можно: Изменять цвет, шрифт или фон текста, реализовывать обработчики событий (например, щелчок по выделенному тексту), удобно управлять форматированием.

11. Как осуществляется вставка виджетов в текстовое поле?

Для вставки виджета в текстовое поле используется метод window_create:

```
button = tk.Button(root, text="Кнопка")
text.window_create("end", window=button)
```

12. Для чего предназначены виджеты Radiobutton и Checkbutton?

Radiobutton — предоставляет возможность выбрать одну из нескольких опций (взаимоисключающий выбор). Checkbutton — позволяет выбрать несколько опций одновременно (независимые флажки).

13. Что такое переменные Tkinter и для чего они нужны?

Переменные Tkinter (StringVar, IntVar, DoubleVar, BooleanVar) — это специальные объекты, которые связывают значения виджетов с программным кодом. Изменение переменной автоматически обновляет связанный виджет, и наоборот.

14. Как осуществляется связь переменных Tkinter с виджетами Radiobutton и Checkbutton?

Radiobutton: Связывается с переменной с помощью параметра variable. Каждая кнопка имеет уникальное значение, задаваемое через value.

Вывод: в ходе выполнения работы были приобретены навыки построения графического интерфейса пользователя GUI с помощью пакета Tkinter языка программирования Python версии 3.x.