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 №4.7
дисциплины «Объектно-ориентированное программирование»

Выполнил:
Магомедов Имран Борисович
3 курс, группа «Программная инженерия»,
направленность (профиль) «Разработка
и сопровождение программного
обеспечения», очная форма обучения

_____
(подпись)

Руководитель практики:
Воронкин Р.А., доцент департамента
цифровых, робототехнических систем и
электроники

_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

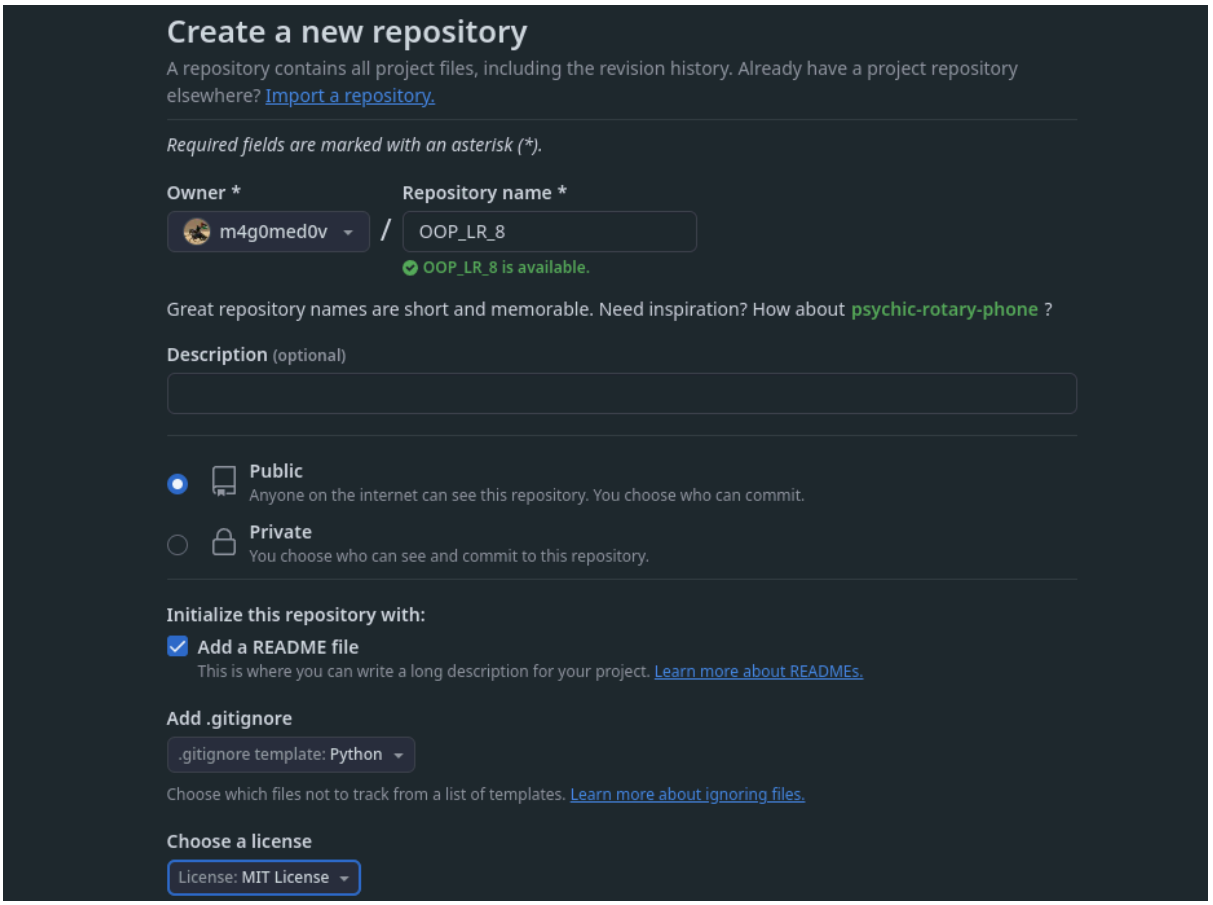
Ставрополь, 2024 г.

Тема: Основы работы с Tkinter.

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

Методика выполнения

1. Изучить теоретический материал работы.
2. Создать общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT и язык программирования Python.



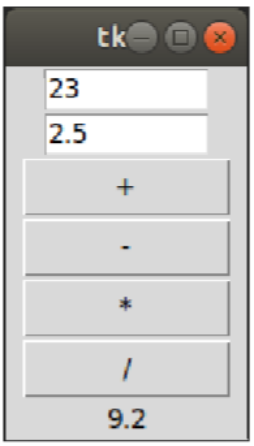
3. Выполните клонирование созданного репозитория.

```
~/ncfu git clone https://github.com/m4g0med0v/OOP_LR_8.git
Клонирование в «OOP_LR_8»...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Получение объектов: 100% (5/5), готово.
```

4. Организуйте свой репозиторий в соответствие с моделью ветвления git-flow.

```
~/n/OOP_LR_8 main git branch develop
~/n/OOP_LR_8 main git switch develop
Переключились на ветку «develop»
```

5. 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".

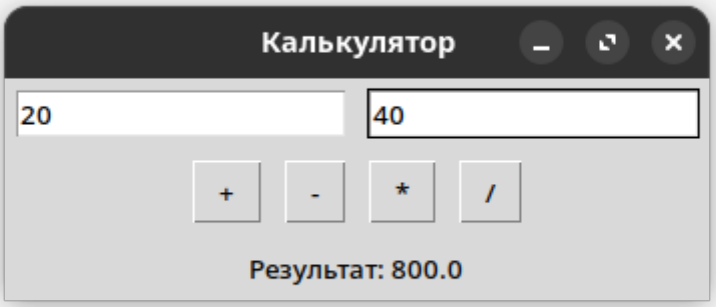


Решение:

```python
import tkinter as tk


def calculate(operation):
    try:
        # Преобразуем введенные значения в числа
        num1 = float(entry1.get())
        num2 = float(entry2.get())
        # Выполняем соответствующую операцию
        if operation == "+":
            result = num1 + num2
        elif operation == "-":
            result = num1 - num2
        elif operation == "*":
            result = num1 * num2
        elif operation == "/":
            result = num1 / num2
        # Отображаем результат
        label_result.config(text=f"Результат: {result}")
    except (ValueError, ZeroDivisionError):
        # Обрабатываем ошибки ввода или деления на ноль
        label_result.config(text="Ошибка")
```

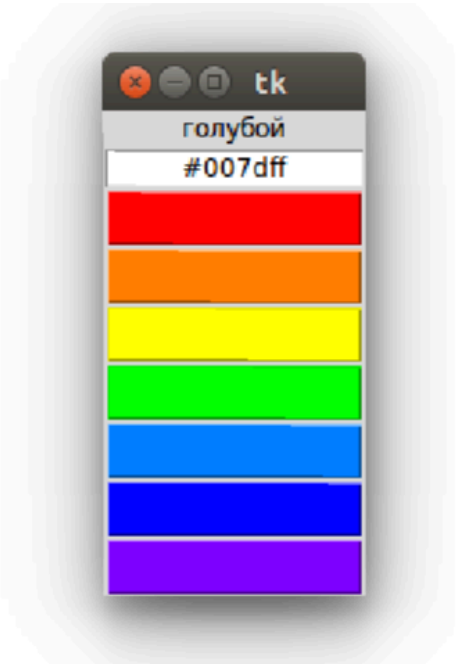
```python
# Создаем главное окно
window = tk.Tk()
window.title("Калькулятор")

# Создаем рамки для группировки элементов
top_frame = tk.Frame(window)
middle_frame = tk.Frame(window)
bottom_frame = tk.Frame(window)

# Поля ввода для чисел
entry1 = tk.Entry(top_frame)
entry1.grid(row=0, column=0, padx=5, pady=5)

entry2 = tk.Entry(top_frame)
entry2.grid(row=0, column=2, padx=5, pady=5)

# Кнопки для выполнения операций
button_add = tk.Button(middle_frame, text="+", command=lambda: calculate("+"))
button_add.grid(row=1, column=0, padx=5, pady=5)

button_sub = tk.Button(middle_frame, text="-", command=lambda: calculate("-"))
button_sub.grid(row=1, column=1, padx=5, pady=5)
```

```python
button_div = tk.Button(middle_frame, text="/", command=lambda: calculate("/"))
button_div.grid(row=1, column=3, padx=5, pady=5)

# Метка для отображения результата
label_result = tk.Label(bottom_frame, text="Результат: ")
label_result.grid(row=2, column=0, columnspan=4, padx=5, pady=5)

# Размещаем рамки в основном окне
top_frame.pack()
middle_frame.pack()
bottom_frame.pack()

if __name__ == "__main__":
    # Запускаем главный цикл приложения
    window.mainloop()
```



6. 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. Коды цветов в шестнадцатеричной кодировке: #ff0000 – красный, #ff7d00 – оранжевый, #ffff00 – желтый, #00ff00 – зеленый, #007dff – голубой, #0000ff – синий, #7d00ff – фиолетовый.

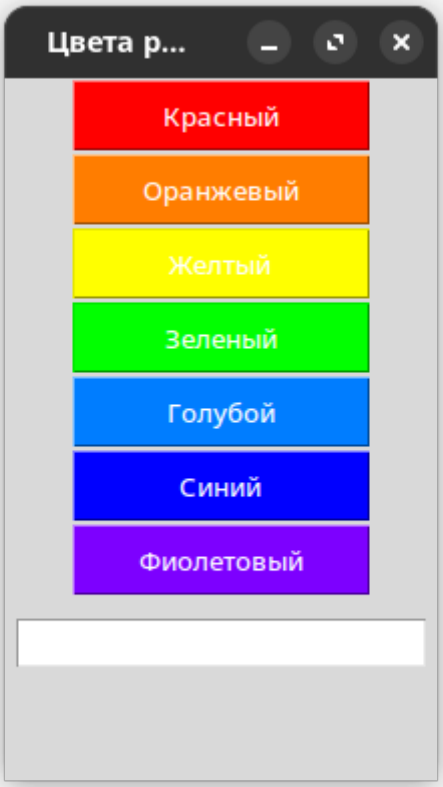
Примерно должно получиться так:



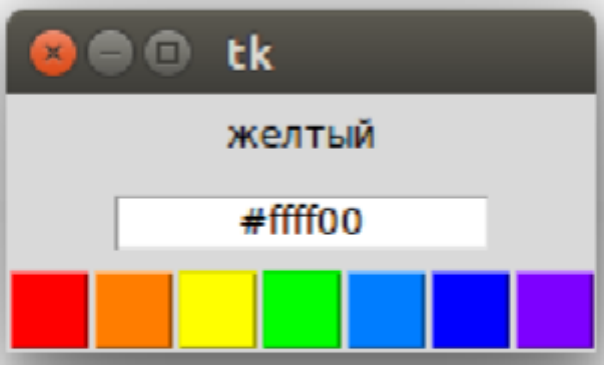
Решение:

```python
import tkinter as tk


def display_color(color_code, color_name):
    entry_color.delete(0, tk.END)
    entry_color.insert(0, color_code)
    label_color_name.config(text=color_name)


# Создание окна
window = tk.Tk()
window.title("Цвета радуги")

color_frame = tk.Frame(window)
output_frame = tk.Frame(window)

# Словарь цветов
rainbow_colors = {
    "Красный": "#ff0000",
    "Оранжевый": "#ff7d00",
    "Желтый": "#ffff00",
    "Зеленый": "#00ff00",
    "Голубой": "#007dff",
    "Синий": "#0000ff",
    "Фиолетовый": "#7d00ff",
}
```

```python
# Создание кнопок
for color_name, color_code in rainbow_colors.items():
    button = tk.Button(
        color_frame,
        text=color_name,
        bg=color_code,
        fg="white",
        command=lambda c=color_code, n=color_name: display_color(c, n),
    )
    button.pack(fill=tk.X, ipadx=20, ipady=2)

# Текстовое поле для кода цвета
entry_color = tk.Entry(output_frame, font=("Arial", 14), justify="center")
entry_color.pack(padx=5, pady=10)

# Метка для названия цвета
label_color_name = tk.Label(output_frame, text="", font=("Arial", 16))
label_color_name.pack(padx=5, pady=10)


color_frame.pack()
output_frame.pack()

if __name__ == "__main__":
    # Запуск приложения
    window.mainloop()
```



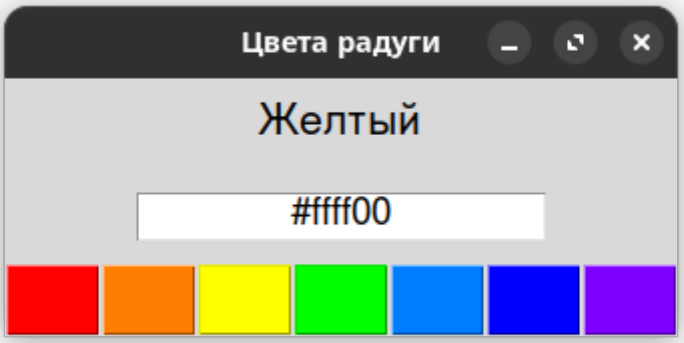
7. Решите задачу: перепишите программу из пункта 8 так, чтобы интерфейс выглядел примерно следующим образом:



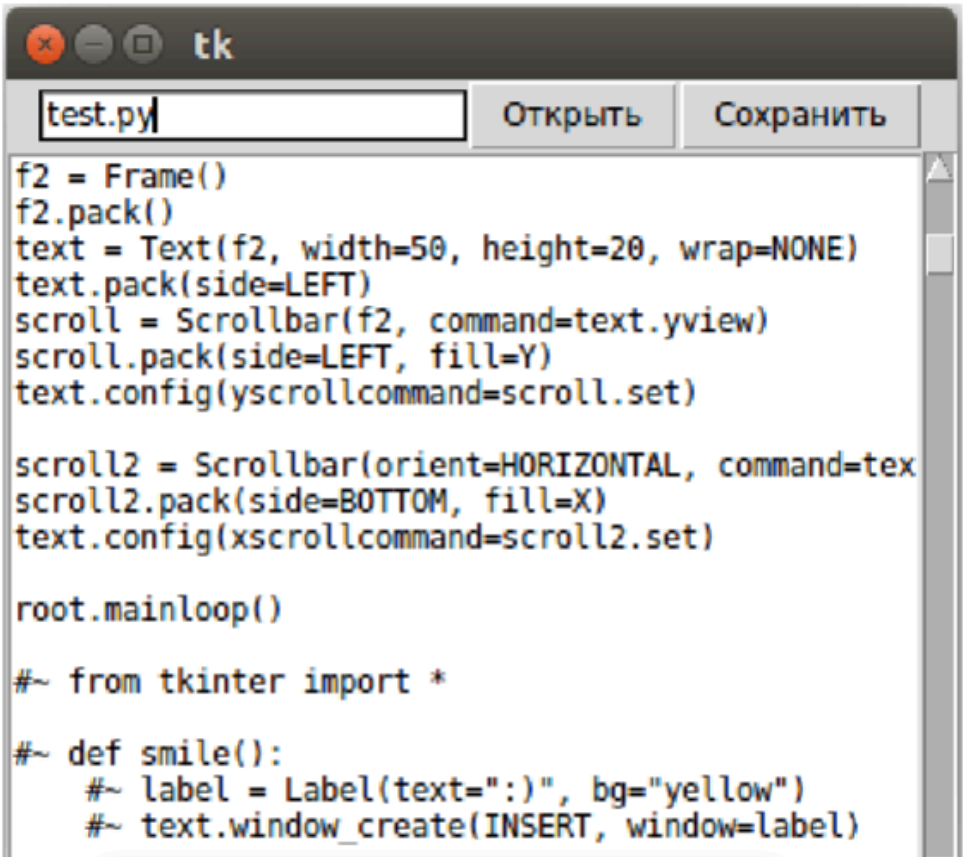
Решение:

```python
import tkinter as tk


def display_color(color_code, color_name):
    entry_color.delete(0, tk.END)
    entry_color.insert(0, color_code)
    label_color_name.config(text=color_name)


# Создание окна
window = tk.Tk()
window.title("Цвета радуги")

color_frame = tk.Frame(window)
output_frame = tk.Frame(window)

# Словарь цветов
rainbow_colors = {
    "Красный": "#ff0000",
    "Оранжевый": "#ff7d00",
    "Желтый": "#ffff00",
    "Зеленый": "#00ff00",
    "Голубой": "#007dff",
    "Синий": "#0000ff",
    "Фиолетовый": "#7d00ff",
}
```

```python
# Создание кнопок
for color_name, color_code in rainbow_colors.items():
    button = tk.Button(
        color_frame,
        bg=color_code,
        fg="white",
        command=lambda c=color_code, n=color_name: display_color(c, n),
    )
    button.pack(fill=tk.X, ipadx=10, ipady=2, side=tk.LEFT)

# Текстовое поле для кода цвета
entry_color = tk.Entry(output_frame, font=("Arial", 14), justify="center")
entry_color.pack(padx=5, pady=10, side=tk.BOTTOM)

# Метка для названия цвета
label_color_name = tk.Label(output_frame, text="", font=("Arial", 16))
label_color_name.pack(padx=5, pady=10, side=tk.BOTTOM)


color_frame.pack(side=tk.BOTTOM)
output_frame.pack(side=tk.BOTTOM)

if __name__ == "__main__":
    # Запуск приложения
    window.mainloop()
```



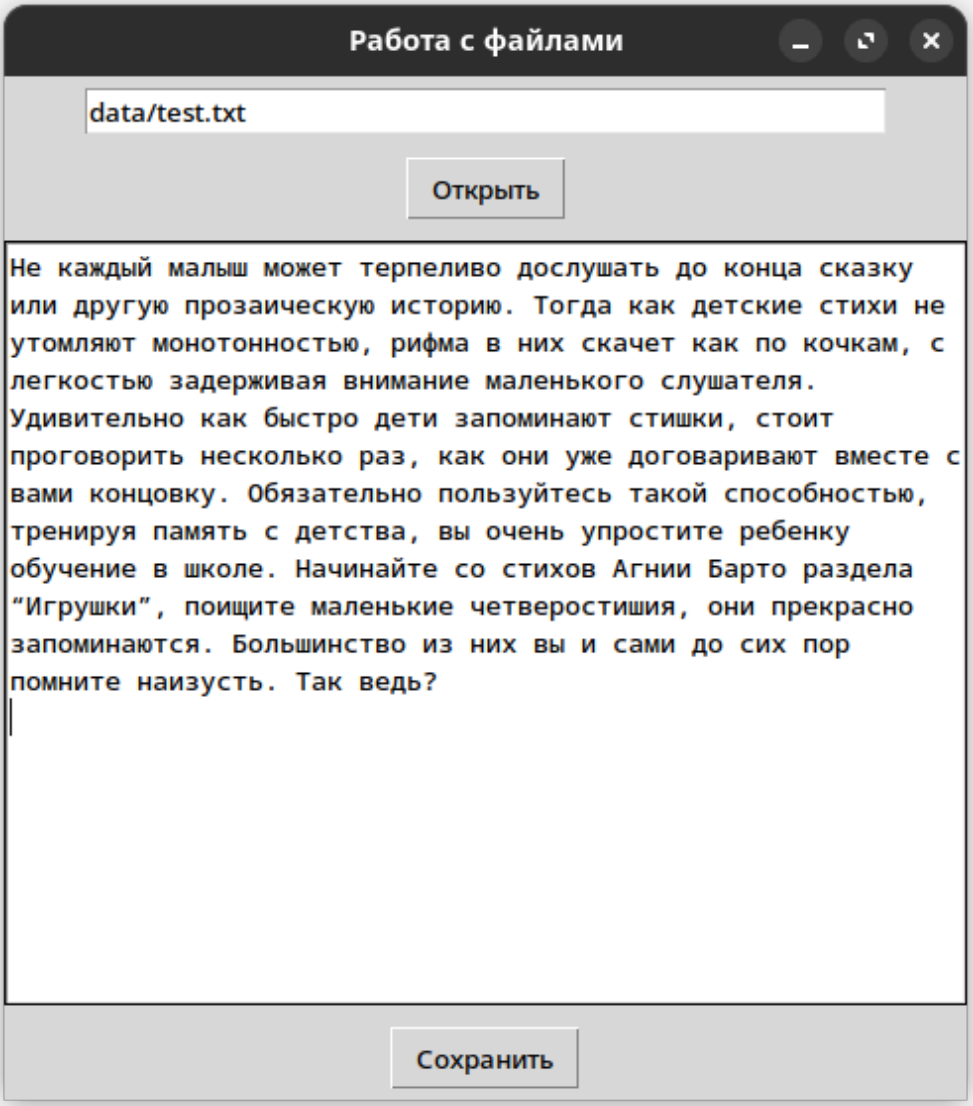
8. Решите задачу: напишите программу, состоящую из однострочного и многострочного текстовых полей и двух кнопок "Открыть" и "Сохранить". При клике на первую должен открываться на чтение файл, чье имя указано в поле класса Entry , а содержимое файла должно загружаться в поле типа Text . При клике на вторую кнопку текст, введенный пользователем в экземпляр Text , должен сохраняться в файле под именем, которое пользователь указал в однострочном текстовом поле. Файлы будут читаться и записываться в том же каталоге, что и файл скрипта, если указывать имена файлов без адреса. Для выполнения практической работы вам понадобится функция open языка Python и методы файловых объектов чтения и записи. Освежить знания о файлах можно из материала лабораторной работы 9.

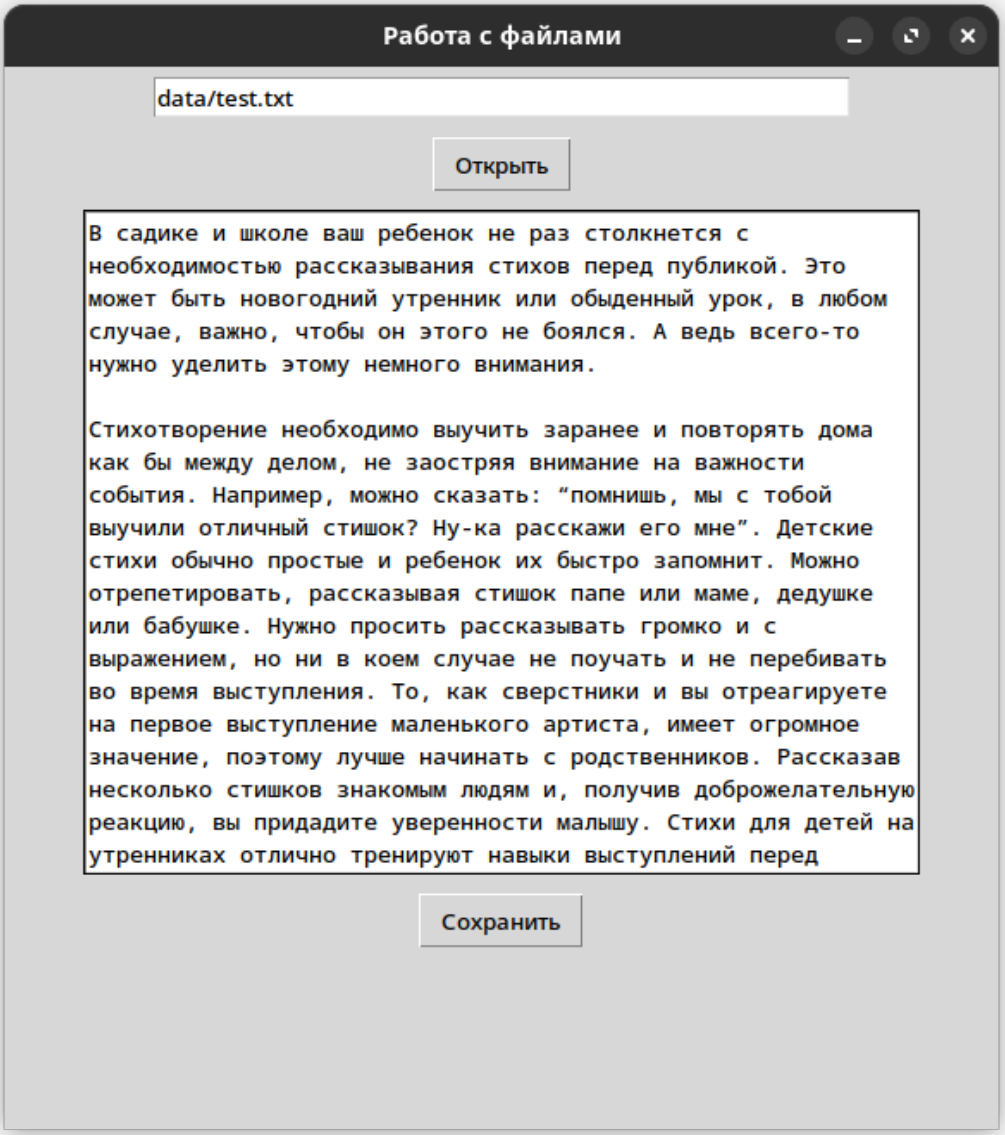


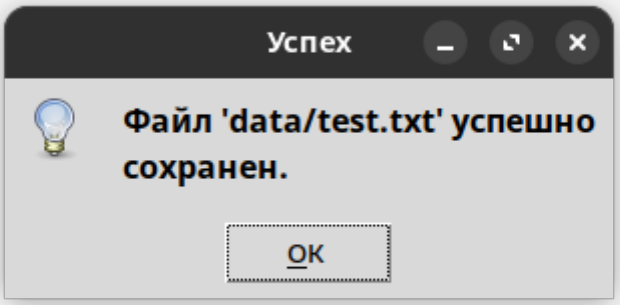
Решение:

```python
import tkinter as tk
from tkinter import messagebox


def open_file():
    file_name = entry_file_name.get()  # Получаем имя файла из поля Entry
    try:
        with open(file_name, "r", encoding="utf-8") as file:
            text_area.delete("1.0", tk.END)  # Очищаем поле Text
            text_area.insert(tk.END, file.read())  # Загружаем содержимое файла
    except FileNotFoundError:
        messagebox.showerror("Ошибка", f"Файл '{file_name}' не найден.")
    except Exception as e:
        messagebox.showerror("Ошибка", f"Не удалось открыть файл: {e}")


def save_file():
    file_name = entry_file_name.get()  # Получаем имя файла из поля Entry
    try:
        with open(file_name, "w", encoding="utf-8") as file:
            file.write(
                text_area.get("1.0", tk.END)
            )  # Сохраняем содержимое поля Text в файл
        messagebox.showinfo("Успех", f"Файл '{file_name}' успешно сохранен.")
    except Exception as e:
        messagebox.showerror("Ошибка", f"Не удалось сохранить файл: {e}")
```

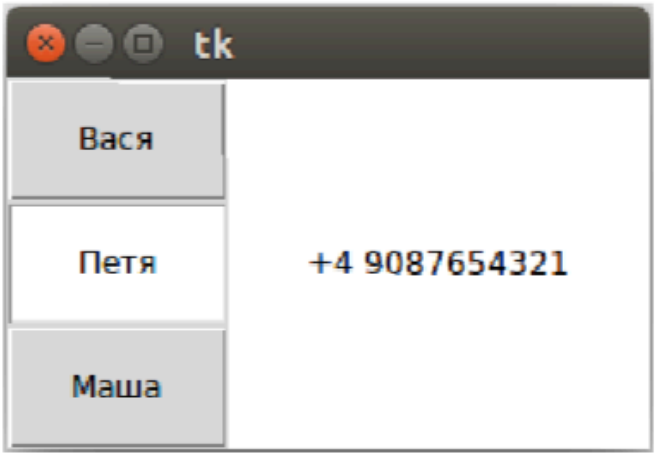
```python
# Создаем главное окно
window = tk.Tk()
window.title("Работа с файлами")

# Однострочное текстовое поле для ввода имени файла
entry_file_name = tk.Entry(window, width=50)
entry_file_name.pack(pady=5)

# Кнопка "Открыть"
button_open = tk.Button(window, text="Открыть", command=open_file)
button_open.pack(pady=5)

# Многострочное текстовое поле для содержимого файла
text_area = tk.Text(window, width=60, height=20, wrap=tk.WORD)
text_area.pack(pady=5)

# Кнопка "Сохранить"
button_save = tk.Button(window, text="Сохранить", command=save_file)
button_save.pack(pady=5)


if __name__ == "__main__":
    # Запуск приложения
    window.mainloop()
```


Работа с файлами
data/test.txt
Открыть
Не каждый малыш может терпеливо дослушать до конца сказку
или другую прозаическую историю. Тогда как детские стихи не
утомляют монотонностью, рифма в них скачет как по кочкам, с
легкостью задерживая внимание маленького слушателя.
Удивительно как быстро дети запоминают стишки, стоит
проговорить несколько раз, как они уже договаривают вместе с
вами концовку. Обязательно пользуйтесь такой способностью,
тренируя память с детства, вы очень упростите ребенку
обучение в школе. Начинайте со стихов Агнии Барто раздела
“Игрушки”, поищите маленькие четверостишия, они прекрасно
запоминаются. Большинство из них вы и сами до сих пор
помните наизусть. Так ведь?
Сохранить


Работа с файлами
data/test.txt
Открыть
В садике и школе ваш ребенок не раз столкнется с необходимостью рассказывания стихов перед публикой. Это может быть новогодний утренник или обыденный урок, в любом случае, важно, чтобы он этого не боялся. А ведь всего-то нужно уделить этому немного внимания.
Стихотворение необходимо выучить заранее и повторять дома как бы между делом, не заостряя внимание на важности события. Например, можно сказать: “помнишь, мы с тобой выучили отличный стишок? Ну-ка расскажи его мне”. Детские стихи обычно простые и ребенок их быстро запомнит. Можно отрепетировать, рассказывая стишок папе или маме, дедушке или бабушке. Нужно просить рассказывать громко и с выражением, но ни в коем случае не поучать и не перебивать во время выступления. То, как сверстники и вы отреагируете на первое выступление маленького артиста, имеет огромное значение, поэтому лучше начинать с родственников. Рассказав несколько стишков знакомым людям и, получив доброжелательную реакцию, вы придадите уверенности малышу. Стихи для детей на утренниках отлично тренируют навыки выступлений перед
Сохранить



9. Решите задачу: виджеты Radiobatton 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. Напишите программу, в которой имеется несколько объединенных в группу радиокнопок, индикатор которых выключен (indicatoron=0). Если какая-нибудь кнопка включается, то в метке должна отображаться соответствующая ей информация. Обычных кнопок в окне быть не должно.

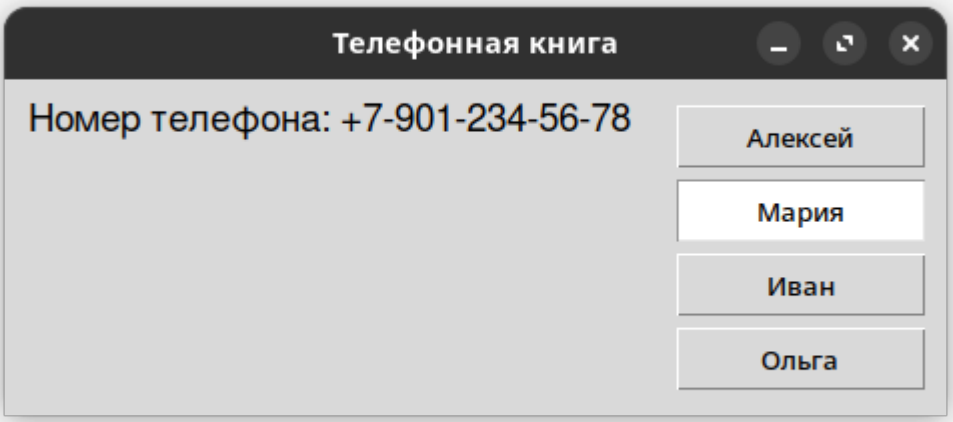


Помните, что свойство command есть не только у виджетов класса Button.

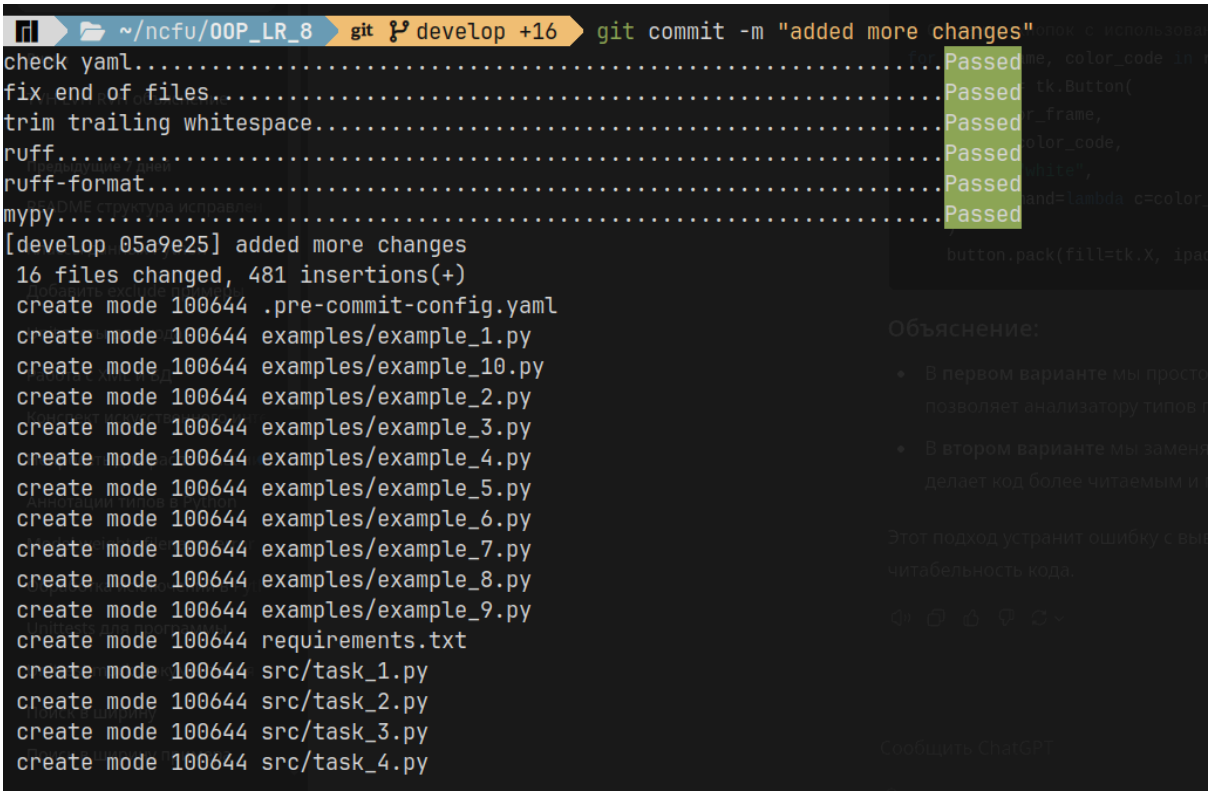
Решение:

```python
import tkinter as tk


def update_phone():
    label_phone.config(
        text=f"Номер телефона: {phonebook[selected_name.get()]}"
    )


# Создаем главное окно
window = tk.Tk()
window.title("Телефонная книга")

# Словарь с именами и номерами телефонов
phonebook = {
    "Алексей": "+7-900-123-45-67",
    "Мария": "+7-901-234-56-78",
    "Иван": "+7-902-345-67-89",
    "Ольга": "+7-903-456-78-90",
}

```

```python
# Переменная для хранения выбранного имени
selected_name = tk.StringVar()
selected_name.set("")  # По умолчанию ничего не выбрано

# Левая колонка для отображения номера телефона
frame_left = tk.Frame(window, padx=10, pady=10)
frame_left.pack(side=tk.LEFT, fill=tk.BOTH)

label_phone = tk.Label(frame_left, text="Номер телефона:", font=("Arial", 14))
label_phone.pack()

# Правая колонка для радиокнопок
frame_right = tk.Frame(window, padx=10, pady=10)
frame_right.pack(side=tk.RIGHT, fill=tk.BOTH)

# Создание радиокнопок
for name in phonebook:
    rb = tk.Radiobutton(
        frame_right,
        text=name,
        value=name,
        variable=selected_name,
        command=update_phone,
        indicatoron=0,  # Отключаем индикатор
        width=15,
        pady=5,
    )
    rb.pack(anchor="w", pady=2)


if __name__ == "__main__":
    # Запуск приложения
    window.mainloop()
```



10. Зафиксируйте сделанные изменения в репозитории.



11. Выполните слияние ветки для разработки с веткой main (master).

```
~/ncfu/OOP_LR_8  main  git merge develop
Обновление 1efc7be..05a9e25
Fast-forward
 .pre-commit-config.yaml | 24 ++++++++++++++++++++++++
 examples/example_1.py   | 30 ++++++++++++++++++++++++++++++
 examples/example_10.py  | 28 ++++++++++++++++++++++++++++
 examples/example_2.py   | 33 +++++++++++++++++++++++++++++++++
 examples/example_3.py   | 16 ++++++++++++++++
 examples/example_4.py   | 10 ++++++++++
 examples/example_5.py   | 13 +++++++++++++
 examples/example_6.py   | 14 ++++++++++++++
 examples/example_7.py   | 16 ++++++++++++++++
 examples/example_8.py   | 16 ++++++++++++++++
 examples/example_9.py   |  9 +++++++++
 requirements.txt        |  1 +
 src/task_1.py           | 75 +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 src/task_2.py           | 60 ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 src/task_3.py           | 69 +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 src/task_4.py           | 67 +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 16 files changed, 481 insertions(+)
 create mode 100644 .pre-commit-config.yaml
 create mode 100644 examples/example_1.py
 create mode 100644 examples/example_10.py
 create mode 100644 examples/example_2.py
 create mode 100644 examples/example_3.py
 create mode 100644 examples/example_4.py
 create mode 100644 examples/example_5.py
 create mode 100644 examples/example_6.py
 create mode 100644 examples/example_7.py
 create mode 100644 examples/example_8.py
 create mode 100644 examples/example_9.py
 create mode 100644 requirements.txt
 create mode 100644 src/task_1.py
 create mode 100644 src/task_2.py
 create mode 100644 src/task_3.py
 create mode 100644 src/task_4.py
```

12. Отправьте сделанные изменения на сервер GitHub.

```
~/ncfu/OOP_LR_8  main ↑1  git push
Перечисление объектов: 21, готово.
Подсчет объектов: 100% (21/21), готово.
При сжатии изменений используется до 12 потоков
Сжатие объектов: 100% (19/19), готово.
Запись объектов: 100% (20/20), 8.10 КиБ | 4.05 МиБ/с, готово.
Total 20 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/m4g0med0v/OOP_LR_8.git
   1efc7be..05a9e25  main -> main
```

Контрольные вопросы

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

В стандартной библиотеке Python для построения графического интерфейса пользователя (GUI) имеются следующие основные средства:

- **Tkinter** — самый популярный и встроенный в Python набор для создания GUI.
- **PyQt** и **PySide** — обертки для библиотеки Qt.
- **wxPython** — обертка для библиотеки wxWidgets.
- **Kivy** — библиотека для создания многосенсорных приложений, также поддерживающая мобильные платформы.

2. Что такое Tkinter?

Tkinter — это стандартная библиотека Python для создания графических интерфейсов. Она является оберткой над библиотекой **Tk**, которая предоставляет инструменты для создания оконных приложений. Tkinter используется для быстрого и удобного создания простых графических интерфейсов.

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

Для построения графического интерфейса с помощью Tkinter обычно выполняются следующие шаги:

- **Импорт библиотеки Tkinter**: import tkinter as tk
- **Создание окна приложения**: создается экземпляр класса Tk().
- **Добавление виджетов**: создаются виджеты (например, кнопки, метки, поля ввода) и размещаются в окне.
- **Управление размещением виджетов**: размещение виджетов в окне с помощью методов, таких как pack(), grid() или place().

- **Запуск цикла обработки событий**: запуск метода mainloop(), который начинает обработку событий и отображает интерфейс.

4. Что такое цикл обработки событий?

Цикл обработки событий — это непрерывный процесс, в котором программа ожидает и обрабатывает пользовательские действия (например, клики мыши, нажатия клавиш). В Tkinter цикл обработки событий инициируется с помощью метода mainloop(), который сохраняет окно в рабочем состоянии и продолжает отвечать на действия пользователя.

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

Экземпляр класса Tk является основным окном приложения и служит корнем для всех виджетов и элементов управления. Это окно будет контейнером для всех других виджетов, и его методы позволяют управлять поведением и внешним видом приложения.

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

Button: представляет кнопку, на которую можно нажать для выполнения действия.

Label: представляет текстовую метку, которая используется для отображения информации или текста.

Entry: текстовое поле для ввода одной строки текста.

Text: текстовое поле для ввода и отображения нескольких строк текста (многострочное).

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

Метод pack() используется для размещения виджетов в окне. Он автоматически выстраивает виджеты по определенному принципу (вертикально или горизонтально). Этот метод удобен для быстрого размещения элементов, но ограничен в плане гибкости по сравнению с другими методами размещения, такими как grid() и place().

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

Метод pack() размещает виджеты на основе их порядка и ориентации. Управление размещением осуществляется с помощью параметров:

- side (например, top, bottom, left, right): определяет, с какой стороны родительского контейнера будет размещен виджет.
- fill: определяет, как виджет будет заполнять доступное пространство (x, y, both).
- expand: если установлено в True, виджет будет пытаться занять все доступное пространство в родительском контейнере.

9. Как осуществляется управление полосами прокрутки в виджете Text?

Для добавления полос прокрутки в виджет Text необходимо использовать виджет Scrollbar:

- Создается экземпляр Scrollbar и привязывается к текстовому полю с помощью метода config().

- Привязывается действие на прокрутку, например, с использованием вертикальной или горизонтальной прокрутки.

```
scrollbar = tk.Scrollbar(window)
text = tk.Text(window, yscrollcommand=scrollbar.set)
scrollbar.config(command=text.yview)
```

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

Тэги в виджете Text позволяют выделять части текста для дальнейшего стилизования или применения к ним особых операций. Тэги могут использоваться для изменения цвета текста, шрифта или других атрибутов. Они также могут служить для навигации по определенным участкам текста.

11. Как осуществляется вставка виджетов в текстовое поле?

Вставка виджетов в текстовое поле (Text) выполняется через использование метода window.create_window(). Этот метод позволяет вставить другие виджеты (например, кнопки или метки) в определенные части текста.

12. Для чего предназначены виджеты Radiobutton и Checkbutton?

Radiobutton: предназначен для создания группы взаимно исключающих кнопок. Пользователь может выбрать только одну кнопку из группы.

Checkbutton: используется для создания флажков (checkbox). Пользователь может выбрать несколько флажков одновременно.

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

Переменные Tkinter (например, StringVar, IntVar, DoubleVar, BooleanVar) используются для связывания значений виджетов с данными в программе. Эти переменные облегчают работу с состоянием виджетов и позволяют легко получать или обновлять их значения.

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

Для связывания переменной с виджетами Radiobutton и Checkbutton используется параметр variable, который указывает на переменную Tkinter. Для Radiobutton устанавливается одно значение, а для Checkbutton — несколько значений.