МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
«СЕВЕРОКАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Кафедра инфокоммуникаций

Институт цифрового развития

ОТЧЁТ

по лабораторной работе

Дисциплина: «Объектно – ориентированное программирование»

Выполнил: студент 3 курса
группы ИВТ-б-о-21-1
Мальцев Николай Артемович

Ставрополь 2023

Основы работы с tkinter

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

Ход работа:

Задание 1.

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".

Код программы:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
import tkinter as tk
from tkinter import messagebox

def sum():
    try:
        num1 = float(entry_num1.get())
        num2 = float(entry_num2.get())

        result = num1 + num2

        result_label.config(text=f"Результат: {result}")
    except:
        result_label.config(text="Ошибка: Введите числа")


def sub():
    try:
        num1 = float(entry_num1.get())
        num2 = float(entry_num2.get())

        result = num1 - num2
        result_label.config(text=f"Результат: {result}")
    except:
        result_label.config(text="Ошибка: Введите числа")

def div():
    try:
        num1 = float(entry_num1.get())
        num2 = float(entry_num2.get())

        if num2 != 0:
            result = num1 / num2
            result_label.config(text=f"Результат: {result}")
        else:
            raise ZeroDivisionError("Division by zero is not allowed.")
```

```
    except ZeroDivisionError:
        result_label.config(text="Ошибка: Деление на ноль")
    except:
        result_label.config(text="Ошибка: Введите числа")


def mult():
    try:
        num1 = float(entry_num1.get())
        num2 = float(entry_num2.get())

        result = num1 * num2

        result_label.config(text=f"Результат: {result}")
    except ValueError:
        result_label.config(text="Ошибка: Введите числа")
    except ZeroDivisionError:
        result_label.config(text="Ошибка: Деление на ноль")

# Создание главного окна
root = tk.Tk()
root.title("Простой калькулятор")

# Переменные для хранения чисел и операции
entry_num1 = tk.Entry(root)
entry_num2 = tk.Entry(root)
operation = tk.StringVar()
result_label = tk.Label(root, text="Результат:")

# Создание кнопок
button_add = tk.Button(root, text="+", command=sum)
button_subtract = tk.Button(root, text="-", command=sub)
button_multiply = tk.Button(root, text="*", command=mult)
button_divide = tk.Button(root, text="/", command=div)

# Размещение виджетов на экране
entry_num1.grid(row=0, column=0)
entry_num2.grid(row=0, column=1)
button_add.grid(row=1, column=0)
button_subtract.grid(row=1, column=1)
button_multiply.grid(row=2, column=0)
button_divide.grid(row=2, column=1)
result_label.grid(row=3, columnspan=2)

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

Результат работы программы:

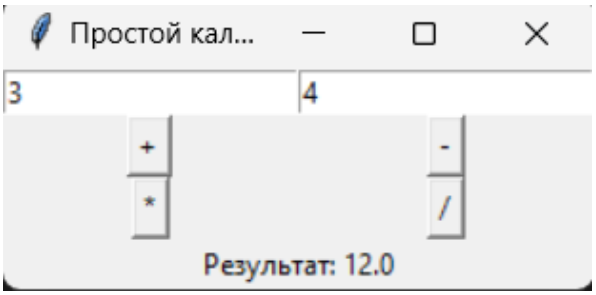

Рисунок 1. Результат работы программы к первому заданию

Задание 2.

Код программы:

```
import tkinter as tk

class RainbowColorApp:
    def __init__(self, root):
        self.root = root
        self.root.title("Цвета радуги")

        self.color_codes = {
            "Красный": "#ff0000",
            "Оранжевый": "#ff7d00",
            "Желтый": "#ffff00",
            "Зеленый": "#00ff00",
            "Голубой": "#007dff",
            "Синий": "#0000ff",
            "Фиолетовый": "#7d00ff"
        }

        self.current_color_name = tk.StringVar()
        self.current_color_code = tk.StringVar()

        self.label_color_name = tk.Label(root, textvariable=self.current_color_name,
font=("Arial", 14))
        self.label_color_name.pack(pady=10)

        self.entry_color_code = tk.Entry(root, textvariable=self.current_color_code,
state='readonly', font=("Arial", 12))
        self.entry_color_code.pack(pady=10)

        self.button_frame = tk.Frame(root)
        self.button_frame.pack()

        button_width = max(len(name) for name in self.color_codes.keys())  #
Выбираем максимальную длину имени цвета

        for color_name, color_code in self.color_codes.items():
            color_button = tk.Button(self.button_frame, text=color_name,
bg=color_code, width=button_width, command=lambda name=color_name, code=color_code:
self.on_button_click(name, code))
```

```
            color_button.pack(side=tk.TOP, pady=5)

    def on_button_click(self, color_name, color_code):
        self.current_color_name.set(color_name)
        self.current_color_code.set(color_code)

if __name__ == "__main__":
    root = tk.Tk()
    app = RainbowColorApp(root)
    root.mainloop()
```

Результат работы программы:

Рисунок 2. Результат работы программы ко второму заданию

Задание 3.

Код программы:

```
import tkinter as tk

class RainbowColorApp:
    def __init__(self, root):
        self.root = root
        self.root.title("Цвета радуги")

        self.color_codes = {
            "Красный": "#ff0000",
            "Оранжевый": "#ff7d00",
            "Желтый": "#ffff00",
            "Зеленый": "#00ff00",
            "Голубой": "#007dff",
            "Синий": "#0000ff",
```

```
            "Фиолетовый": "#7d00ff"
        }

        self.current_color_name = tk.StringVar()
        self.current_color_code = tk.StringVar()

        self.label_color_name = tk.Label(root, textvariable=self.current_color_name,
font=("Arial", 14))
        self.label_color_name.pack(pady=10)

        self.entry_color_code = tk.Entry(root, textvariable=self.current_color_code,
state='readonly', font=("Arial", 12))
        self.entry_color_code.pack(pady=10)

        self.button_frame = tk.Frame(root)
        self.button_frame.pack()

        button_width = max(len(name) for name in self.color_codes.keys())  #
Выбираем максимальную длину имени цвета

        for color_name, color_code in self.color_codes.items():
            color_button = tk.Button(self.button_frame, width=button_width,
text=color_name, bg=color_code, command=lambda name=color_name, code=color_code:
self.on_button_click(name, code))
            color_button.pack(side=tk.LEFT, padx=5)

    def on_button_click(self, color_name, color_code):
        self.current_color_name.set(color_name)
        self.current_color_code.set(color_code)

if __name__ == "__main__":
    root = tk.Tk()
    app = RainbowColorApp(root)
    root.mainloop()
```

Результат работы программы:

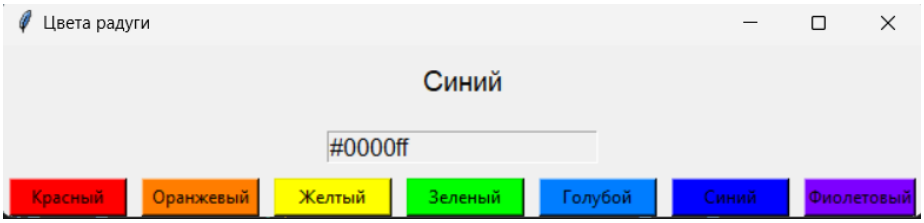

Рисунок 3. Результат работы программы к третей задачи

Задание 4.

Код программы:

```
import tkinter as tk
from tkinter import filedialog
```

```python
class FileEditorApp:
    def __init__(self, root):
        self.root = root
        self.root.title("Текстовый редактор")

        self.file_path_entry = tk.Entry(root, width=50)
        self.file_path_entry.pack(pady=10)

        self.text_content = tk.Text(root, wrap="word", width=50, height=10)
        self.text_content.pack(pady=10)

        self.button_open = tk.Button(root, text="Открыть", command=self.open_file)
        self.button_open.pack(side=tk.LEFT, padx=5)

        self.button_save = tk.Button(root, text="Сохранить", command=self.save_file)
        self.button_save.pack(side=tk.RIGHT, padx=5)

    def open_file(self):
        file_path = self.file_path_entry.get()
        try:
            with open(file_path, 'r') as file:
                content = file.read()
                self.text_content.delete(1.0, tk.END)  # Очищаем текущее содержимое
                self.text_content.insert(tk.END, content)
        except FileNotFoundError:
            tk.messagebox.showerror("Ошибка", "Файл не найден")

    def save_file(self):
        file_path = self.file_path_entry.get()
        content = self.text_content.get(1.0, tk.END)
        try:
            with open(file_path, 'w') as file:
                file.write(content)
            tk.messagebox.showinfo("Сохранено", "Файл успешно сохранен")
        except Exception as e:
            tk.messagebox.showerror("Ошибка", f"Не удалось сохранить файл:
{str(e)}")

if __name__ == "__main__":
    root = tk.Tk()
    app = FileEditorApp(root)
    root.mainloop()
```

Результат работы программы:

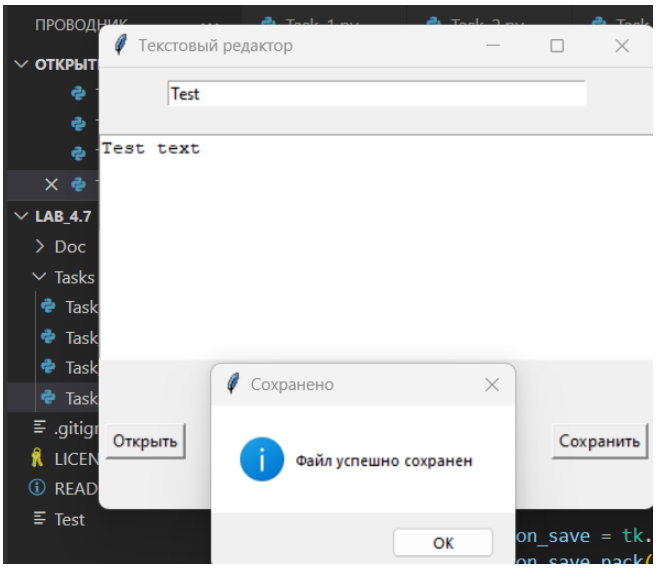

Рисунок 4. Результат работы программы к четвёртому заданию

Задание 5.

Код программы:

```
import tkinter as tk

class RadioButtonsApp:
    options = {"Рыбы": "19 февраля - 20 марта", "Козерог": "22 декабря - 19 января",
"Скорпион": "23 октября - 21 ноября"}

    def __init__(self, root):
        self.root = root
        self.root.title("Знаки зодиака")

        self.selected_option = tk.StringVar()

        self.label_result = tk.Label(root, text="Дата: ")
        self.label_result.pack(pady=10)

        self.radio_frame = tk.Frame(root)
        self.radio_frame.pack()

        for key in self.options:
            radio_button = tk.Radiobutton(self.radio_frame, text=key, width=10,
variable=self.selected_option, value=key, indicatoron=0, command=self.update_label)
            radio_button.pack(pady=5, anchor=tk.W)

    def update_label(self):
        selected_option_value = self.selected_option.get()
        date_range = self.options[selected_option_value]
        self.label_result.config(text=f"Дата: {date_range}")

if __name__ == "__main__":
    root = tk.Tk()
    app = RadioButtonsApp(root)
    root.mainloop()
```

Результат работы программы:

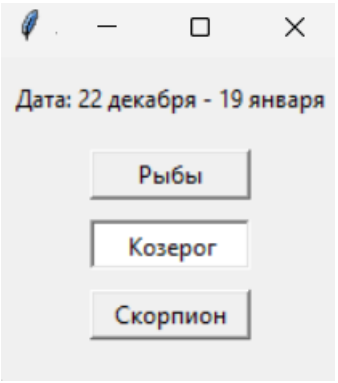

Рисунок 5. Результат работы программы к пятому заданию

Контрольные вопросы:

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

Основным средством для создания графического интерфейса в стандартной библиотеке Python является модуль tkinter.

2. Что такое Tkinter?

Tkinter — это стандартная библиотека Python для создания графического интерфейса пользователя. Она предоставляет набор инструментов и виджетов для построения окон, кнопок, текстовых полей, меток и других элементов управления.

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

Шаги для построения графического интерфейса с помощью Tkinter:

1. Создание главного окна (Tk).

2. Создание и настройка виджетов (кнопок, меток и т.д.).

3. Упаковка или размещение виджетов в окне с использованием метода pack(), grid() или place().

4. Запуск цикла обработки событий (mainloop()).

4. Что такое цикл обработки событий?

Цикл обработки событий — это бесконечный цикл, который ожидает и обрабатывает события, такие как нажатия кнопок, перемещения мыши и другие взаимодействия с пользователем. В Tkinter это обеспечивает метод mainloop().

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

Экземпляр класса Tk представляет главное окно приложения. Его цель — создать основное окружение для построения графического интерфейса с использованием Tkinter.

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

Button: Кнопка, предназначенная для выполнения действия п нажатии.

Label: Метка для отображения текста или изображения.

Entry: Однострочное текстовое поле для ввода данных.

Text: Многострочное текстовое поле для ввода и отображения текста.

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

Метод pack() используется для размещения виджетов в родительском контейнере. Он автоматически управляет размерами виджетов и их расположением в окне.

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

side: Задает сторону (верх, низ, лево, право), на которую будет упакован виджет.

fill: Определяет, как виджет заполняет доступное пространство ("x", "y", "both", "none").

expand: Указывает, следует ли расширять виджет для заполнения доступного пространства.

9. Как осуществляется управление полосами прокрутки в виджете Text?

Для управления полосами прокрутки в виджете Text используются виджеты Scrollbar и их методы привязки (yview и xview).

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

Тэги в виджете Text используются для применения форматирования, стилей и связывания событий к определенным частям текста.

11. Как осуществляется вставка виджетов в текстовое поле?

Для вставки виджетов, таких как кнопки и изображения, в текстовое поле используются методы window_create() и insert().

12. Для чего предназначены виджеты Radiobutton и Checkbutton?

Radiobutton: Позволяет пользователю выбирать один из нескольких взаимоисключающих вариантов.

Checkbutton: Предоставляет пользователю возможность включать или выключать определенные опции, независимо друг от друга.

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

Переменные Tkinter, такие как StringVar, IntVar и DoubleVar, предоставляют связь между значениями переменных и виджетами, что позволяет автоматически обновлять виджеты при изменении переменных.

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

Для связи с Radiobutton используется параметр variable с объектом переменной (StringVar, IntVar и т.д.).

Для связи с Checkbutton используется параметр variable с объектом IntVar, который принимает 0 или 1 в зависимости от состояния флажка.