

Тема практики: «Библиотека TkInter в Python»

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	5
1 Подготовка к работе	7
1.1 Теория	7
1.2 Установка библиотеки	7
1.2.1 Windows	7
1.2.2 Linux	8
1.3 Подключение библиотеки	8
2 Основные элементы библиотеки TkInter	9
2.1 Окно (Tk) и Метка (Label).....	9
2.2 Поле ввода (Entry) и кнопка (Button)	9
2.3 Фрейм (Frame)	11
2.4 Меню (Menu).....	11
2.5 Список (Listbox)	12
2.6 Флажок (Checkbox).....	14
2.7 Переключатель (Radiobutton).....	15
2.8 Прокрутка (Scrollbar)	16
2.9 Холст (Canvas)	17
3 Задания	19
3.1 Задание 4	19
3.2 Задание 5	19
3.3 Задание 6	19
3.4 Задание 7	19
3.5 Задание 8	19
3.6 Задание 9	19
3.7 Задание 10	19
ЗАКЛЮЧЕНИЕ	20
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	21
Приложение А Решение заданий, представленный в методическом руко- водстве.....	23
1.1 Задание 1	23
1.2 Задание 2	23
1.3 Задание 3	24
1.4 Задание 4	25

1.5	Задание 5	26
1.6	Задание 6	27
1.7	Задание 7	29
1.8	Задание 8	30
1.9	Задание 9	31
1.10	Задание 10	33
Приложение Б Тест		36
2.1	Вопросы	36
2.2	Ответы	37

ВВЕДЕНИЕ

Tkinter (сокращение от "Тк интерфейс") - это стандартная библиотека для создания графических пользовательских интерфейсов (GUI) в языке программирования Python. Она основана на библиотеке Tk, которая предоставляет инструменты для создания окон, кнопок, меню, полей ввода и других элементов интерфейса.

Библиотека Tkinter имеет свою историю, начиная с библиотеки Tk, которая была создана в 1988 году. Тогда Марком Л. Миллером была разработана Tk (Tool Command Language Kit). Tk представляет собой библиотеку для создания графических интерфейсов пользователя (GUI) на языке программирования Tcl (Tool Command Language). Tk обеспечивала набор виджетов (графических элементов интерфейса), таких как кнопки, окна, меню и другие, которые можно было использовать для создания приложений с графическим интерфейсом.

В 1991 году Гвидо ван Россум, создатель языка программирования Python, включил поддержку Tk в стандартную библиотеку Python, делая ее более доступной для разработчиков. Tkinter (Tk Interface) стала оберткой для Tk в Python, предоставляя Python-разработчикам простой способ создания графических интерфейсов. Tkinter стала стандартной библиотекой в Python, что обеспечило ее наличие в большинстве установок Python по умолчанию.

С течением времени и развитием Python и Tk, Tkinter была улучшена и расширена. Появились новые возможности, более удобные методы программирования GUI, и стало возможным создавать более сложные и красочные интерфейсы. Вместе с развитием экосистемы Python появились также альтернативные библиотеки для создания GUI, такие как PyQt, wxPython, Kivy и др. Однако Tkinter остается привлекательным выбором из-за своей простоты и интеграции со стандартной библиотекой Python. Некоторые проекты используют Tkinter в сочетании с другими библиотеками для создания более сложных и продвинутых приложений.

Tkinter остается популярным выбором для создания простых и средних по сложности графических интерфейсов в Python из-за своей легкости использования, интеграции со стандартной библиотекой Python и кросс-платформенной поддержки.

Благодаря этой методической разработке мы научимся создавать простейшие объекты с помощью библиотеки TkInter. Python – язык, удобный своей

быстротой написания. Поэтому быстрое создание GUI даже для небольшого приложения – важное умение для любого разработчика [1] [2].

1 Подготовка к работе

1.1 Теория

Программы с графическим интерфейсом пользователя событийно-ориентированные. Событийно-ориентированное ориентировано на события. То есть та или иная часть программного кода начинает выполняться лишь тогда, когда случается то или иное событие.

Событийно-ориентированное программирование базируется на объектно-ориентированном. Даже если мы не будем разрабатывать собственные классы, мы обязательно будем пользоваться теми, что есть в TkInter. Все виджеты – это объекты-экземпляры, создаваемые от встроенных классов.

События могут быть разными: клик мыши, сработал временной фактор (истёк таймер), завершилась загрузка и прочее. Когда случается что-либо подобное, то, если был создан обработчик для соответствующего события, происходит выполнение определенной части программы, что приводит к какому-либо результату.

Чтобы написать GUI-программу, надо выполнить приблизительно следующее:

1. Создать главное окно.
2. Создать виджеты и выполнить конфигурацию их свойств (опций).
3. Определить события, то есть то, на что будет реагировать программа.
4. Описать обработчики событий, то есть то, как будет реагировать программа.
5. Расположить виджеты в главном окне.
6. Запустить цикл обработки событий.

Перейдём к практической части.

1.2 Установка библиотеки

1.2.1 Windows

На Windows Tkinter поставляется с установщиком Python. Нам просто нужно установить Python с www.python.org. Tkinter идет вместе с Python. Устанавливать отдельно не нужно. Установите флажок tcl / tk и IDE [3].

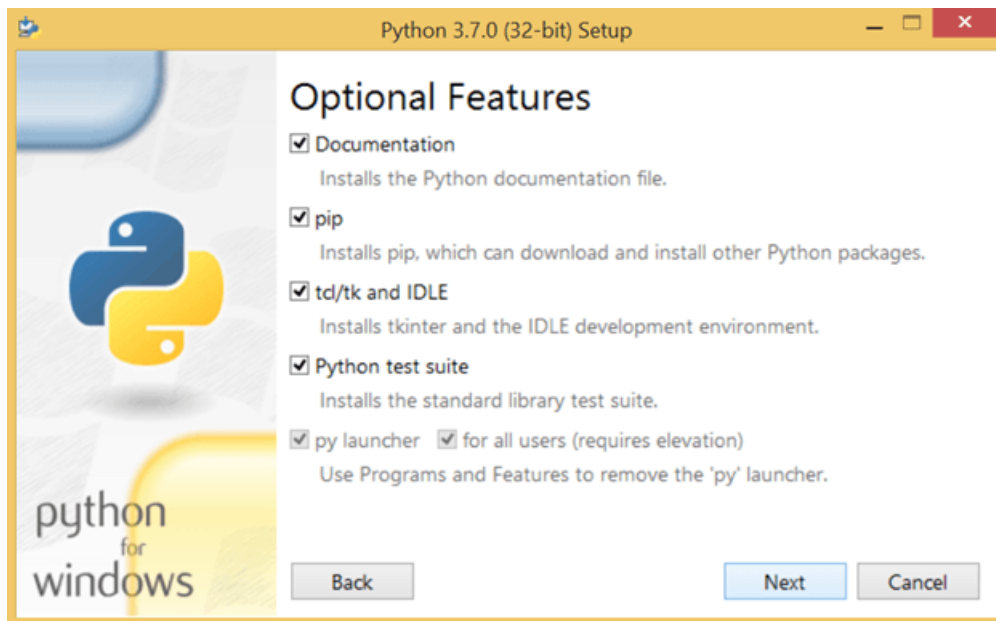


Рисунок 1 – Установка Python и TkInter на Windows

1.2.2 Linux

Введя эту команду на Ubuntu,

```
sudo apt-get install python3.11 python3-tk
```

мы установим интерпретатор Python, а также библиотеку TkInter [4].

Аналогично для Arch Linux:

```
sudo apt-get install python tk
```

1.3 Подключение библиотеки

Tkinter импортируется стандартно для модуля Python любым из способов:

```
1 import tkinter
2 from tkinter import *
3 import tkinter as tk
```

Теперь мы готовы к знакомству с основными элементами библиотеки.

2 Основные элементы библиотеки TkInter

Рассмотрим несколько основных элементов библиотеки.

2.1 Окно (Tk) и Метка (Label)

Окно представляет собой основной контейнер для размещения всех виджетов. Внутри него будут помещаться все следующие элементы. Метка же используется для отображения текста или изображения. В коде ниже представлено создание этих двух элементов: окна и метки внутри него [5] [6].

```
1 import tkinter as tk
2
3 root = tk.Tk()
4
5 root.title("Label Widget Demo")
6 label = tk.Label(root, text="This is a label")
7 label.pack()
8
9 root.mainloop()
```

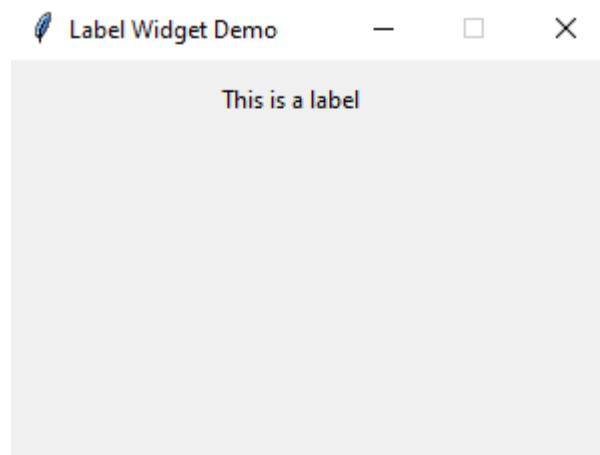


Рисунок 2 – Окно с меткой

2.2 Поле ввода (Entry) и кнопка (Button)

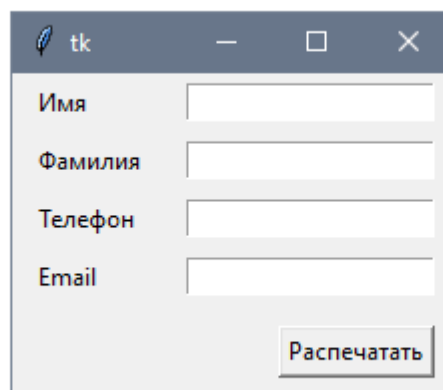
Поле ввода предназначено для ввода текста. Кнопка выполняет определенное действие при нажатии. В следующем коде представлено создание этих элементов:


```

1 import tkinter as tk
2
3 def on_button_click():
4     text = entry.get()
5     label.config(text="Вы ввели: " + text)
6
7 root = tk.Tk()
8
9 entry = tk.Entry(root)
10 entry.pack()
11
12 button = tk.Button(root, text="Подтвердить",
13     ↪ command=on_button_click)
14
15 label = tk.Label(root, text="")
16 label.pack()
17
18 root.mainloop()

```

Здесь можно наблюдать возможную форму ввода, использующую только метки, поля ввода и кнопку .



The image shows a screenshot of a Tkinter window titled "tk". Inside the window, there is a form with four input fields, each preceded by a label: "Имя", "Фамилия", "Телефон", and "Email". Below these fields is a button labeled "Распечатать". The window has a standard title bar with minimize, maximize, and close buttons.

Рисунок 3 – Кнопка и поле ввода

Задание 1. Реализуйте форму, представленную на рисунке 3 [7].

2.3 Фрейм (Frame)

Фрейм представляет собой контейнер для организации других виджетов.

Пример реализации фрейма представлен в коде ниже:

```
1 import tkinter as tk
2
3 root = tk.Tk()
4
5 frame = tk.Frame(root)
6 frame.pack()
7
8 label = tk.Label(frame, text="Это фрейм!")
9 label.pack()
10
11 root.mainloop()
```

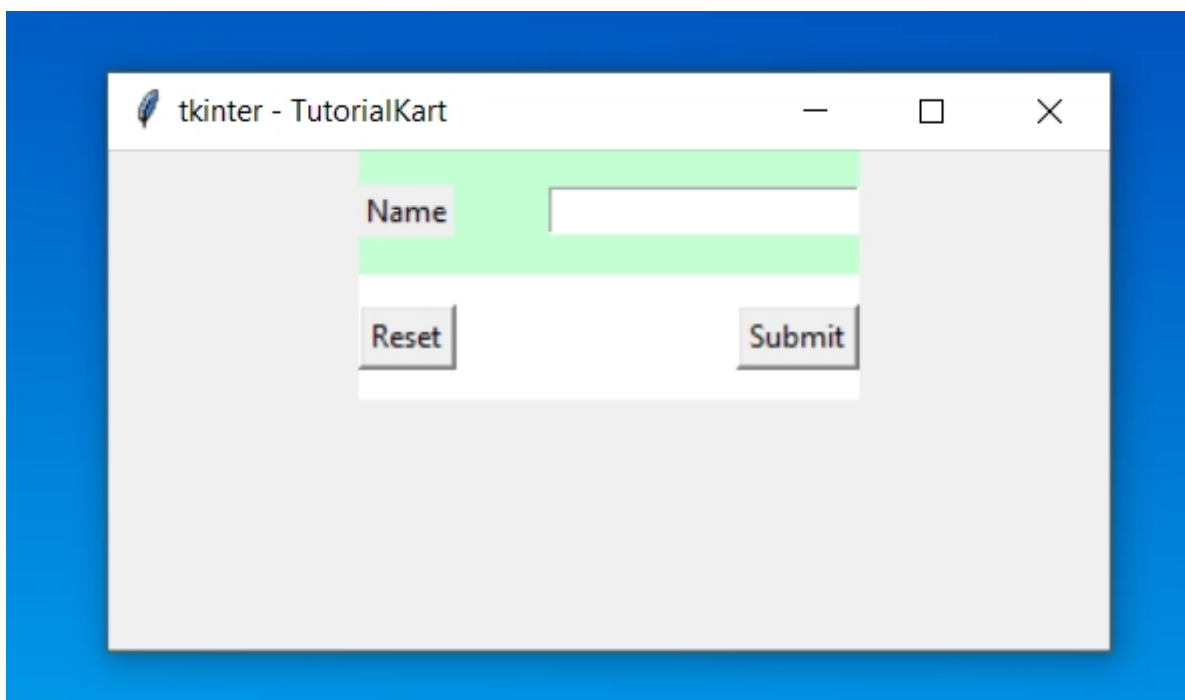


Рисунок 4 – Пример Frame

Задание 2. Реализуйте форму, представленную на рисунке 4 [8].

2.4 Меню (Menu)

Меню используется для создания панели меню [9]. Ниже представлен код реализации примера меню:

```

1 import tkinter as tk
2
3 def on_menu_click():
4     label.config(text="Выбран пункт меню")
5
6 root = tk.Tk()
7
8 menu_bar = tk.Menu(root)
9 root.config(menu=menu_bar)
10
11 file_menu = tk.Menu(menu_bar, tearoff=0)
12 menu_bar.add_cascade(label="Файл", menu=file_menu)
13 file_menu.add_command(label="Открыть", command=on_menu_click)
14 file_menu.add_command(label="Сохранить", command=on_menu_click)
15 file_menu.add_separator()
16 file_menu.add_command(label="Выход", command=root.destroy)
17
18 label = tk.Label(root, text="")
19 label.pack()
20
21 root.mainloop()

```

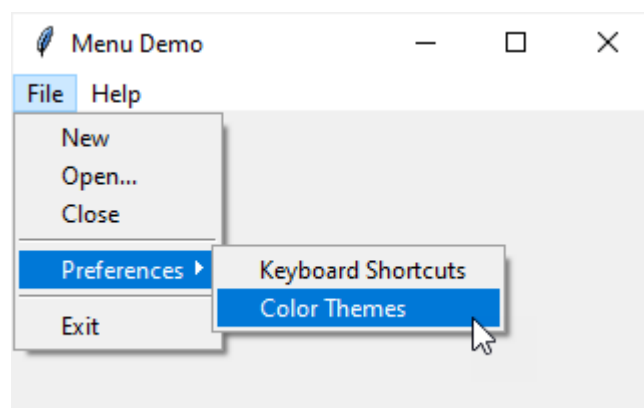


Рисунок 5 – Пример Menu

2.5 Список (Listbox)

Список предоставляет возможность отображения списка элементов, из которого пользователь может выбирать [10]. В коде ниже представлено кон-

струирование списка языков программирования:

```
1 import tkinter as tk
2
3 def on_select(event):
4     selected_item = listbox.get(listbox.curselection())
5     label.config(text="Выбрано: " + selected_item)
6
7 root = tk.Tk()
8
9 listbox = tk.Listbox(root)
10 listbox.pack()
11
12 listbox.insert(1, "Пункт 1")
13 listbox.insert(2, "Пункт 2")
14 listbox.insert(3, "Пункт 3")
15
16 listbox.bind("<<ListboxSelect>>", on_select)
17
18 label = tk.Label(root, text="")
19 label.pack()
20
21 root.mainloop()
```

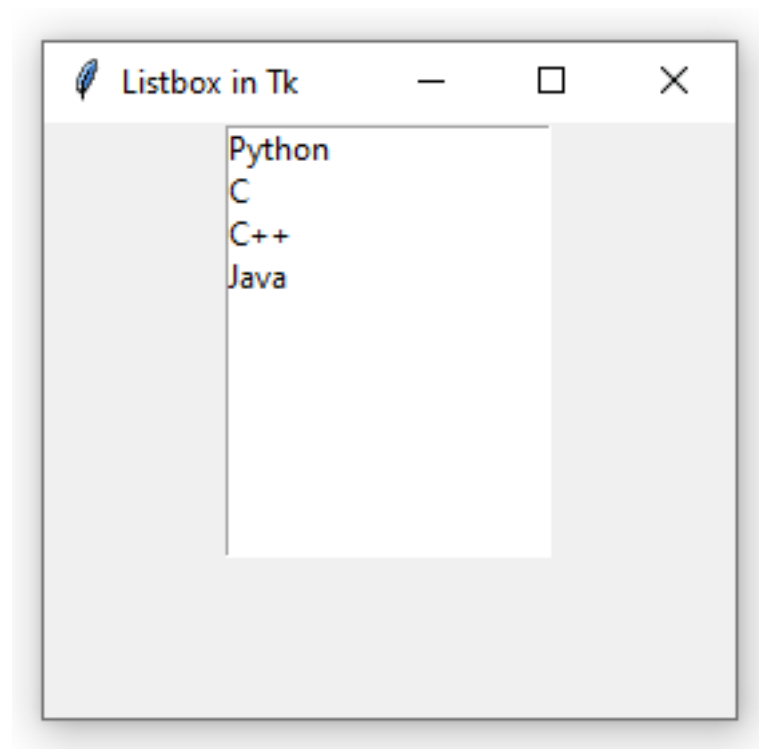


Рисунок 6 – Пример Listbox

2.6 Флажок (Checkbox)

Флажок позволяет пользователю включать или выключать какую-то опцию [11]. В представленном коде описано самое простое описание элемента Checkbox:

```
1 import tkinter as tk
2
3 def on_checkbox_click():
4     label.config(text="Флажок включен" if var.get() else "Флажок
    ↪   выключен")
5
6 root = tk.Tk()
7
8 var = tk.BooleanVar()
9 checkbox = tk.Checkbutton(root, text="Включить опцию",
    ↪   variable=var, command=on_checkbox_click)
10 checkbox.pack()
11
12 label = tk.Label(root, text="")
```

```

13 label.pack()
14
15 root.mainloop()

```

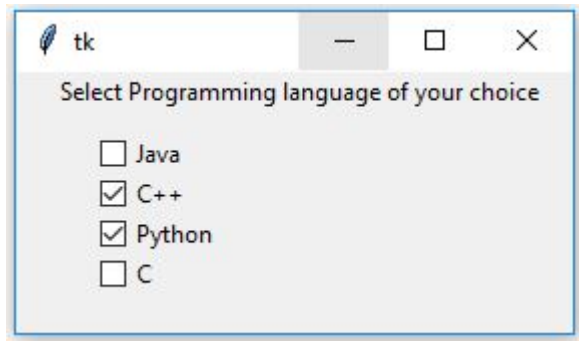


Рисунок 7 – Пример Checkbox

2.7 Переключатель (Radiobutton)

Переключатель позволяет выбрать один вариант из группы [12]. В коде ниже представлено создание простейшей группы элементов, переключающихся с помощью данного элемента.

```

1 import tkinter as tk
2
3 def on_radio_change():
4     label.config(text="Выбрано: " + var.get())
5
6 root = tk.Tk()
7
8 var = tk.StringVar(value="Опция 1")
9
10 radio1 = tk.Radiobutton(root, text="Опция 1", variable=var,
    ↪ value="Опция 1", command=on_radio_change)
11 radio2 = tk.Radiobutton(root, text="Опция 2", variable=var,
    ↪ value="Опция 2", command=on_radio_change)
12
13 radio1.pack()
14 radio2.pack()
15

```

```

16 label = tk.Label(root, text="")
17 label.pack()
18
19 root.mainloop()

```

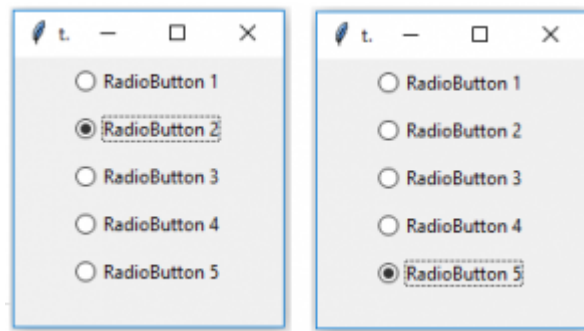


Рисунок 8 – Пример Radiobutton

Задание 3. Реализуйте форму, представленную на рисунке 8.

2.8 Прокрутка (Scrollbar)

Прокрутка используется для обеспечения возможности прокрутки содержимого виджетов, таких как текстовые поля или списки. Есть возможность создавать Scrollbar как справа (достаточно обычное место), так и во всех 4 сторонах [13].

```

1 import tkinter as tk
2
3 root = tk.Tk()
4
5 scrollbar = tk.Scrollbar(root)
6 scrollbar.pack(side=tk.RIGHT, fill=tk.Y)
7
8 text = tk.Text(root, yscrollcommand=scrollbar.set)
9 text.pack()
10
11 scrollbar.config(command=text.yview)
12
13 root.mainloop()

```

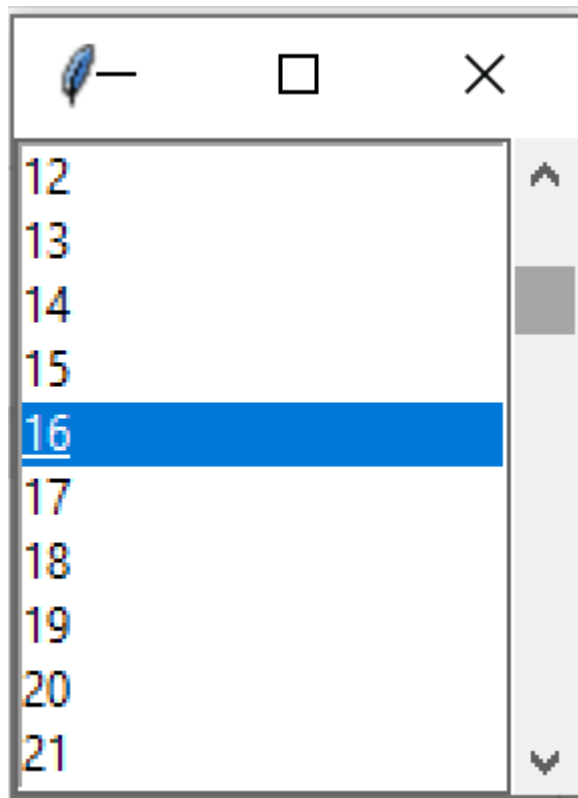


Рисунок 9 – Пример Scrollbar

2.9 Холст (Canvas)

Холст предоставляет область для рисования графики и различных объектов [14].

```
1 import tkinter as tk
2
3 def draw_circle():
4     canvas.create_oval(50, 50, 150, 150, fill="blue")
5
6 root = tk.Tk()
7
8 canvas = tk.Canvas(root, width=200, height=200)
9 canvas.pack()
10
11 button = tk.Button(root, text="Нарисовать круг",
12     ↪ command=draw_circle)
13 button.pack()
14 root.mainloop()
```


Примером такого холста может являться следующее окно:

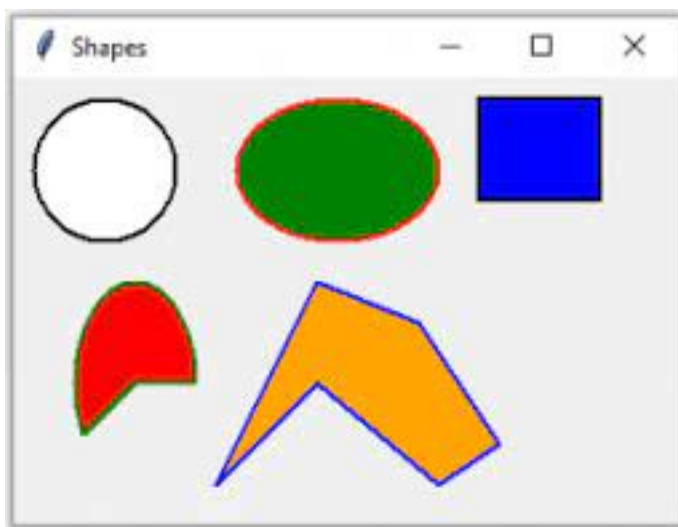


Рисунок 10 – Пример Canvas

3 Задания

Помимо заданий выше предлагается выполнить ещё несколько:

3.1 Задание 4

Создайте приложение с главным окном. Добавьте метку с текстом "Привет, Tkinter!" и кнопку "Нажми меня". При нажатии кнопки измените текст метки.

3.2 Задание 5

Создайте простой калькулятор с использованием виджетов Entry, Button и Label. Позвольте пользователю вводить числа, выбирать оператор (+, -, *, /) и выводить результат.

3.3 Задание 6

Разработайте приложение для управления списком задач. Используйте Listbox для отображения задач. Добавьте Entry для ввода новой задачи и кнопку для добавления ее в список.

3.4 Задание 7

Создайте простой менеджер паролей. Используйте Entry для ввода логина и пароля, кнопку для входа, и метку для отображения результатов.

3.5 Задание 8

Создайте игру, в которой компьютер загадывает число от 1 до 100, а игрок пытается угадать его. При каждой попытке компьютер должен сообщить, было ли угаданное число больше, меньше или равно загаданному.

3.6 Задание 9

Разработайте простой таймер обратного отсчета. Пользователь может ввести время в секундах, нажать кнопку "Старт" и по завершении отсчета появится уведомление.

3.7 Задание 10

Создайте игру "Камень, ножницы, бумага". Игрок выбирает один из трех вариантов, а затем компьютер также делает свой выбор. После этого выводится результат раунда.

ЗАКЛЮЧЕНИЕ

В методической разработке были рассмотрены основы библиотеки Tkinter в Python. Мы изучили историю создания, теоретические аспекты, основные элементы, а также провели практические упражнения, создав простые приложения. Предложенные практические задания должны помочь лучше понять основы Tkinter и обеспечить ресурсами для дальнейшего изучения и разработки графических интерфейсов с использованием этой библиотеки в Python.