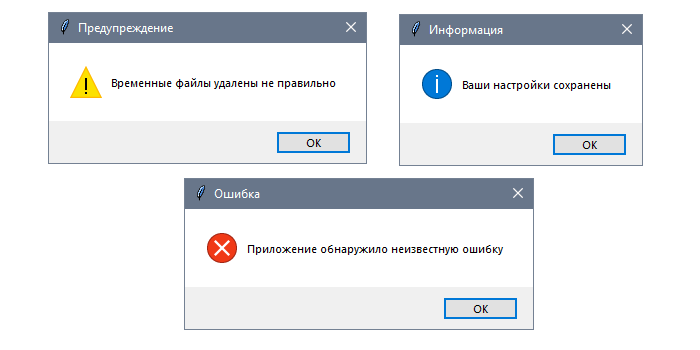
**Окна с уведомлениями**

**from** tkinter **import** messagebox

messagebox.showerror(**"Ошибка"**, **"Приложение обнаружило неизвестную ошибку"**)  
messagebox.showinfo(**"Информация"**, **"Ваши настройки сохранены"**)  
messagebox.showwarning(**"Предупреждение"**,**"Временные файлы удалены не правильно"**)  


**Таймер в tkinter**

… #создайте окно

sec=0 #начальное значение времени

**def** sec():  
 **global** sec #объявить переменную времени глобальной  
 sec=sec+1 #добавить единицу времени  
 **if** sec<60:  
 window.after(1000,sec) #Запускает функцию sec() через 1000 мс

1000 мс = 1секунда

lbl.configure(text=sec)

... #добавьте виджет Label

**Виджет Radiobutton**

При запуске программы включенной окажется первая радиокнопка, так как значение ее опции value совпадает с текущим значением переменной r\_var. Если кликнуть по второй радиокнопке, то она включится, а первая выключится. При этом значение r\_var станет равным 1.

В программном коде обычно требуется "снять" данные о том, какая из двух кнопок включена. Делается это с помощью метода get экземпляров переменных Tkinter.

**from** tkinter **import** \*

**def** change():

**if** var.get() == 0:

label['bg'] = 'red'

**elif** var.get() == 1:

label['bg'] = 'green'

**elif** var.get() == 2:

label['bg'] = 'blue'

root = Tk()

var = IntVar()

var.set(0)

red = Radiobutton(text="Red", variable=var, value=0)

green = Radiobutton(text="Green",variable=var, value=1)

blue = Radiobutton(text="Blue",variable=var, value=2)

button = Button(text="Изменить", command=change)

label = Label(width=20, height=10)

red.pack()

green.pack()

blue.pack()

button.pack()

label.pack()

root.mainloop()

**Виджет Checkbutton**

У каждого флажка должна быть своя переменная. Иначе при включении одного флажка, другой будет выключаться, так как значение общей tkinter-переменной изменится и не будет равно значению опции onvalue первого флажка.

**from** tkinter **import** \*

 /////

var1 = BooleanVar()

var1.set(0)

c1 = Checkbutton(text="First",variable=var1,

onvalue=1, offvalue=0,command=show)

c1.pack(padx=10)

var2 = IntVar()

var2.set(-1)

c2 = Checkbutton(text="Second",variable=var2,

onvalue=1, offvalue=0,command=show)

c2.pack(padx=10)

//////

С помощью методов **select и deselect** флажков можно их программно включать и выключать. То же самое относится к радиокнопкам

**Виджет Text**

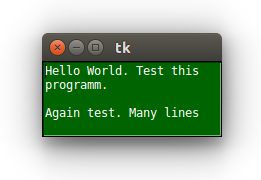
**from** tkinter **import** \*

root = Tk()

text = Text(width=25, height=5, bg="darkgreen", fg='white', wrap=WORD)

text.pack()

root.mainloop()



Значение WORD опции wrap позволяет переносить слова на новую строку целиком, а не по буквам.

Основные методы у Text такие же как у Entry – get, insert, delete. Однако, если в случае однострочного текстового поля было достаточно указать один индекс элемента при вставке или удалении, то в случае многострочного надо указывать два – номер строки и номер символа в этой строке (другими словами, номер столбца). При этом нумерация строк начинается с единицы, а столбцов – с нуля.

**from** tkinter **import** \*  
root = Tk()  
  
text = Text(width=20, height=7)  
text.pack(side=LEFT)  
  
scroll = Scrollbar(command=text.yview) # для добавления скроллбара на

текстовое поле  
scroll.pack(side=LEFT, fill=Y)  
  
text.config(yscrollcommand=scroll.set)  
  
root.mainloop()

**Frame - рамка**

Для сложных интерфейсов используют отдельные области позиционирования элементов. Фрейм - вспомогательный виджет, создание которого происходит при помощи класса Frame().

Фреймы размещают на главном окне, а уже в фреймах – виджеты:

from tkinter import \*

root = Tk()

**frame\_top = Frame()**

frame\_top.pack()

label\_1 = Label(**frame\_top**, width=7, height=4, bg='yellow', text="1")

label\_1.pack(side=LEFT)

root.mainloop()

Кроме Frame существует похожий класс LabelFrame – фрейм с подписью. В отличие от простого фрейма у него есть свойство text:

from tkinter import \*

root = Tk()

**frame\_top = LabelFrame(text="Верх")**

frame\_top.pack()

label\_1 = Label(frame\_top, width=7, height=4, bg='yellow', text="1")

label\_1.pack(side=LEFT)

label\_2 = Label(frame\_top, width=7, height=4, bg='orange', text="2")

label\_2.pack(side=LEFT)

frame\_bottom = LabelFrame(text="Низ")

frame\_bottom.pack()

label\_3 = Label(frame\_bottom, width=7, height=4, bg='lightgreen', text="3")

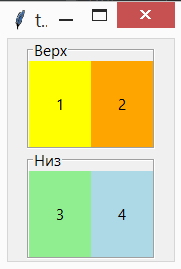
label\_3.pack(side=LEFT)

label\_4 = Label(frame\_bottom, width=7, height=4, bg='lightblue', text="4")

label\_4.pack(side=LEFT)

root.mainloop()

Результат выполнения:



#### PhotoImage

PhotoImage позволяет использовать полноцветное изображение. Кроме того у этого класса есть несколько (достаточно примитивных) методов для работы с изображениями. PhotoImage гарантированно понимает формат GIF. Аргументы конструктора:

* **file** – путь к файлу с изображением;
* **data** – вместо пути к файлу можно указать уже загруженные в память данные изображения. Изображения в формате GIF могут быть закодированы с использованием base64. Данная возможность удобна для встраивания изображения в программу;
* **format** – явное указание формата изображения;
* **width**, **height** – ширина и высота изображения;
* **gamma** – коррекция гаммы;
* **palette** – палитра изображения.

**from** tkinter **import** \*  
root = Tk()  
root.title(**"Изображения"**)  
img = PhotoImage(file=**"lab1.gif"**)  
label = Label(root, image=img)  
label.pack()  
root.mainloop()