Лабораторная работа № 4

I. Примеры

Прочитайте текст лабораторной работы, разберите и сделайте все примеры

Пример 6. Дополнительные окна (Toplevel) и стандартные диалоговые окна

В этом примере по нажатию кнопки будем создавать дополнительное окно (Toplevel). Также создадим несколько различных диалоговых окон. Дополнительную информацию см. [1, стр. 558 (Toplevel), 566 (диалоги)] и электронные ресурсы.

```
from tkinter import *
from tkinter.messagebox import * # подключаем диалоговые окна
from tkinter.colorchooser import askcolor #  диалоговое окно выбор цвета

root = Tk( )
root.title("Главное окно программы")
root.geometry("300x300+250+250")

# создаем фрейм для размещения на нем других компонент:
win1 = Frame(root, bg = '#555555')
# Помещаем фрейм на форму методом pack.
# Вместо параметра side (прижать к стороне) задаем параметр anchor (якорь):
# компонент будет прижат к северной (north) стороне и растянут по ширине.
win1.pack(anchor = "n", expand = YES, fill = X)

# Обработчик кнопки закрытия программы (запуск нескольких диалоговых окон):
def closeQuery( ):
    # при нажатии на кнопку уточняем у пользователя, закрывать ли программу:
    if askyesno('Выход из программы...', 'Закрыть программу? '):
        showwarning('Диалоговое окно', 'Внимание!!!')
        # Уничтожаем главное окно и поэтому закрываем программу:
        root.destroy( )
    else:
        showinfo('Диалоговое окно', 'Информация')

# Создаем кнопку закрытия программы:
Button(win1, text = 'Выход', command = closeQuery).pack(side = RIGHT,
           padx = 10, pady = 5)
```

```
# Создаем обработчик кнопки выбора цвета:
def my_set_color(w):
    # Запускаем диалоговое окно выбора цвета:
    res = askcolor( )
    # печатаем результат res, который вернул диалог:
    print("Диалоговое окно Цвет возвращает результат:", res)
    if res[1]:  # если цвет был выбран, то
        w.config(bg = res[1])       # меняем цвет у виджета w
        win1.config(bg = res[1])  # и меняем цвет у фрейма главного окна

# Обработчик нажатия на кнопку "Цвет" главного окна.
# В этом обработчике создаем новое окно (виджет Toplevel)
def newWindow( ):
    topL1 = Toplevel(root) # создание нового окна
    topL1.title("Дополнительное окно программы")
    topL1.transient(root) # – делаем его зависимым от главного окна
    # topL1.grab_set( )   # – устанавливаем фокус ввода
    # Создаем две кнопки в этом окне:
    b1 = Button(topL1, text= 'Color', command = (lambda: my_set_color(b1)), fg ='blue')
    b2 = Button(topL1, text = 'Цвет ', command = (lambda: my_set_color(b2)))
    b1.pack( )
    b2.pack( )

# Создаем кнопку "Цвет" главного окна:
Button(win1, text = 'Цвет', command = newWindow).pack(side = RIGHT,
           padx = 10, pady = 5)
root.mainloop( )
```

Когда размер программы увеличивается, становится сложно искать в тексте нужный код, исправлять и изменять программу. Поэтому код приходится структурировать. Например, выносить в отдельные функции повторяющиеся операции. Для каждого отдельного окна можно создать свой класс и все описывать в нем. Также удобно каждый такой класс выносить в отдельный модуль (файл).

Пример 7. Привязка событий с помощью метода bind

```
from tkinter import *

root = Tk( )
root.geometry("400x300+150+100")
```

```
root.title("Click me")
# Создаем функции-обработчики событий:
def showPosEvent(event):
    print("Widget = %s X = %s Y = %s" % (event.widget, event.x, event.y))

def onLeftClick(event):
    print('Левая клавиша мыши')
    showPosEvent(event)

def onMiddleClick(event):
    print('Средняя клавиша мыши')
    showPosEvent(event)

def onRightClick(event):
    print('Правая клавиша мыши')
    showPosEvent(event)

def onDoubleLeftClick(event):
    print('Левая клавиша мыши - двойной щелчок')
    showPosEvent(event)

def onLeftDrag(event):
    print('Движение')
    showPosEvent(event)

def onKeyPress(event):
    print('клавиша ', event.char)
    print(event)
    # Печать всех параметров из event:
    #for a in dir(event):
    #      if not a.startswith('__'):
    #           print(a,'=>',getattr(event, a))

def onArrowKey(event):
    print('клавиша стрелка вверх')



w = Label(root, text= 'Hello bind world')
w.config(height = 5, font = 20)
```

```
w.config(bg = 'red')
w.pack(expand=YES, fill = BOTH)
w.bind('<Button-1>', onLeftClick) # щелчок левой кнопки мыши
w.bind('<Button-2>', onMiddleClick) # щелчок средней
w.bind('<Button-3>', onRightClick) # щелчок правой
w.bind('<Double-1>', onDoubleLeftClick) # двойной щелчок левой
w.bind('<B1-Motion>', onLeftDrag) # движение и нажатие левой кнопки мыши
w.bind('<KeyPress>', onKeyPress) # нажатие клавиши на клавиатуре
w.bind('<Up>', onArrowKey) # клавиша стрелка вверх

w.focus( )
mainloop( )
```

Дополнительную информацию см. [1, стр. 585] и электронные ресурсы, например, https://ru.wikiversity.org/wiki/Курс_по_библиотеке_Tkinter_языка_Python#Привязка_событий .

Пример 8. Менеджеры размещения виджетов pack, grid и place

Для размещения компонент мы использовали метод pack (упаковщик). Он прижимает компоненты к какому-нибудь краю. Есть еще метод grid (сетка), позволяет разбивать контейнер сеткой и размещать компоненты в ее ячейки. Также есть метод place (место), при размещении указываются координаты размещения. В одном контейнере эти методы смешивать нельзя!

В главном окне разместим три фрейма, на каждом из них разместим компоненты тремя различными способами.

```
from tkinter import *
import random # Подключаем модуль random
root = Tk( )
root.title("Менеджеры размещения компонентов")
root.geometry("400x400+300+250")

# На первом фрейме используем метод pack:
win1 = Frame(root, bg = '#555555')
win1.pack(expand = YES, fill = BOTH)
# Разметим метку и кнопку методом pack:
Label(win1, text = "Метод pack").pack(side = LEFT,padx = 20, pady = 10)
# При нажатии на кнопку будем слева добавлять еще кнопки:
def add1():
    Label(win1, text = "Метка").pack(side=LEFT, padx=20, pady=10)
```

```
Button(win1, text = "Добавить", command=add1).pack(side = LEFT, padx = 20,
           pady = 10)

# На втором фрейме используем метод grid:
win2 = Frame(root, bg='#888888')
win2.pack(expand = YES, fill = BOTH)
# Поместим метку и кнопку в первой строке сетки.
Label(win2, text = "Метод grid").grid(row = 0, column = 0)
# При нажатии на кнопку будем добавлять компоненты в следующие строки:
s = 0
def add2( ):
    global s
    s += 1
    Label(win2, text = "x" + str(s) + " = ").grid(row=s,column=0, padx=20, pady=10)
    Entry(win2).grid(row=s,column=1, padx=20, pady=10)

Button(win2, text = "Добавить", command=add2).grid(row=0,column=0, padx=20,
pady=10)
add2()  # Добавили компоненты методом add2

# На третьем фрейме используем метод place:
win3 = Frame(root, bg = '#BBBBBB')
win3.pack(expand = YES, fill = BOTH)
# Поместим кнопку на фрейм.
# и при ее нажатии будем добавлять метку в случайное место контейнера.
random.seed( ) # Инициализация генератора случайных чисел
def add3():
    Label(win3, text = "Метод place").place(x = random.randint(0, 400),
            y = random.randint(0, 100))
Button(win3, text = "Добавить", command = add3).place(x = 10, y = 10)

root.mainloop()
```

Дополнительную информацию см. [1, стр. 726] и электронные ресурсы, например, ru.wikiversity.org/wiki/Курс_по_библиотеке_Tkinter_языка_Python#Упаковщики

Пример 9. Виджет Canvas, рисование

```
from tkinter import *

root = Tk( )
root.title("Виджет Canvas, рисование ")
root.geometry("400x400+250+250")

# создаем фрейм для размещения на нем кнопки:
win = Frame(root, bd = 1, relief = RAISED)
win.pack(anchor = "n", expand = YES, fill = X)
# создаем область рисования Canvas:
can = Canvas(root)
can.pack(expand = YES, fill = BOTH)

# Создаем обработчик кнопки 'Рисовать':
def ris( ):
    can.create_line(20, 50, 200, 40, fill = 'red', width = 5)
    can.create_arc(30,150,300,310, fill = 'blue')
    can.create_rectangle(300,10, 350,40, width = 2, outline = 'green')
    can.create_polygon(50,200,10,300,70,255,fill = 'magenta')

# Создаем кнопку:
Button(win, text = 'Рисовать', command = ris).pack(padx = 10, pady = 5)
root.mainloop( )
```

Дополнительную информацию см. [1, стр. 709] и электронные ресурсы.

Пример 10. Динамическое выполнение кода на Python. Методы eval и exec.

Методы eval и exec не относятся к методам tkinter и вообще к графическим интерфейсам. Но они могут расширить возможности программы. Методы позволяют выполнить код, записанный в строке. Метод eval вычисляет выражение, записанное в строке, и возвращает результат, а exec способен выполнять многострочные блоки кода.

В этом примере пользователь в двух редакторах может задать две целые величины x, y. А в третьем редакторе написать любую функцию от x, y по правилам Python и по нажатию кнопки, эта функция будет вычислена.

Эти методы опасны, особенно если программа используется в Internet. Пользователь может записать не математическую функцию, а, например, код удаления файлов. Поэтому нужно как-то ограничить его.

```
from tkinter import *
from math import *   # подключаем математические функции
root = Tk()
root.title("Методы eval  и exec")
# создаем фрейм для размещения компонент задающих  x, y:
win1 = Frame(root)
win1.pack(anchor = "n", expand = YES, fill = X)

lx = Label(win1, text = "x = ")
lx.pack(side=LEFT, padx=10, pady=10)

entX = Entry(win1)
entX.insert(0, 0)
entX.pack(side = LEFT, padx=10, pady=10)
entX.focus()

ly = Label(win1, text = "y = ")
ly.pack(side=LEFT, padx=10, pady=10)

entY = Entry(win1)
entY.insert(0, 0)
entY.pack(side = LEFT, padx=10, pady=10)

# создаем второй фрейм для задания функции и вычисления:
win2 = Frame(root)
win2.pack(anchor = "n", expand = YES, fill = X)
Label(win2, text = "Функция: ").pack(side=LEFT, padx=10, pady=10)
entF = Entry(win2)
entF.pack(side = LEFT, padx=10, pady=10, expand=YES, fill=X)
entF.insert(0, "x + y")

# Создаем обработчик кнопки:
def  res():
    try:
        x = int(entX.get())
    except ValueError:
        showerror("Ошибка заполнения", "Переменная x не является целым числом")
        return
    try:
```

```
        y = int(entY.get())
    except ValueError:
        showerror("Ошибка заполнения", "Переменная y не является целым числом")
        return
    F = entF.get()
    #print(x, y, eval(F))
    labF['text'] = eval(F)

# Создаем кнопку и метку:
Button(win2, text='Вычислить', command = res).pack(side = LEFT, padx=10, pady=5)
labF = Label(win2, text = " ")
labF.pack(side=LEFT, padx=10, pady=10)

root.mainloop()
```

II. Задание

На основе созданных программ выполните следующие задания. Информацию о tkinter см. [1, 4–12].

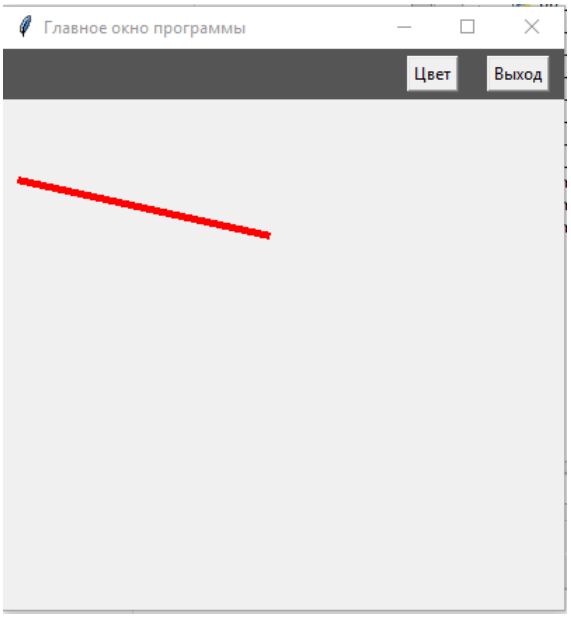
1. Создайте программу с кнопкой. При нажатии на кнопку запускается дополнительное окно Toplevel.
2. Создайте программу с кнопкой. При нажатии на кнопку запускается стандартное диалоговое окно showerror.
3. Создайте программу с кнопкой. При нажатии на кнопку запускается стандартное диалоговое окно askyesno.
4. Создайте программу с кнопкой. При нажатии на кнопку запускается стандартное диалоговое окно showwarning.
5. Создайте программу с кнопкой. При нажатии на кнопку запускается стандартное диалоговое окно askokcancel.

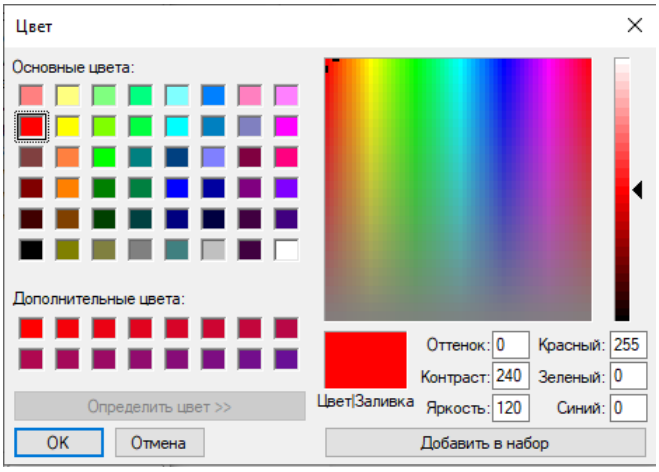
---

6. Создайте программу, выводящую координаты мыши, при ее движении над виджетом.
7. С помощью метода размещения grid создайте таблицу редакторов Entry. Заполните их при создании случайными числами. Создайте кнопку, подсчитывающую сумму чисел в редакторах.
8. С помощью метода размещения place создайте таблицу меток Label. Задайте им разные цвета фона. Задайте им обработчики щелчка мыши: при щелчке мыши на метке цвет фона в окне меняется на цвет метки.

---

9. Создайте программу (см. рисунок ниже) с Frame вверху и Canvas, занимающим всю оставшуюся область. На фрейме находятся две кнопки: “Цвет” и “Выход”. При нажатии на кнопку “Цвет” запускается диалоговое окно выбора цвета и рисуется со случайными координатами выбранным цветом указанный в задании объект. При нажатии на кнопку “Выход” запускается диалоговое окно, требующее у пользователя подтверждения выхода и выход из программы.
   Объект и выбор толщины кисти для рисования:
   a) выводить линию; ширина задается с помощью Scale;
   b) выводить овал; ширина задается с помощью Radiobutton;
   c) выводить прямоугольник, ширина задается с помощью диалогового окна с вводом значения.





10. **Рисование мышкой.** Создайте программу (см. рисунок выше) с Frame вверху и Canvas, занимающим всю оставшуюся область. На фрейме находятся две кнопки: “Цвет” и “Выход”. При нажатии на кнопку “Цвет” запускается диалоговое окно выбора цвета и происходит выбор текущего цвета. Добавьте возможность рисования с помощью мыши: при нажатии на кнопку мыши начинаем рисовать, при отпускании – прекращаем. При нажатии на кнопку “Выход” запускается диалоговое окно, требующее у пользователя подтверждения выхода и выход из программы.
    Добавьте выбор толщины кисти для рисования:
    a) задается с помощью Scale;
    b) задается с помощью Radiobutton;
    c) задается с помощью диалогового окна с вводом значения.
11. **(Виджет изображение PhotoImage)** Создайте программу, которая загружает рисунок из файла на компонент PhotoImage. Имя файла выбирайте с помощью стандартного диалога askopenfilename (см. пример из [1, стр. 571]).
    Замечание. Изображение PhotoImage см. [1, стр. 633] и электронные ресурсы.

---

12. На основе примера 10 создайте программу, вычисляющую логические выражения. В редакторы для x, y пользователь записывает через пробел значения. Например, значения вектора x = ”1 0”, значения вектора y = ”0 1”. А в редактор формул логическое выражение, например, F = “not (x1 and y1) or (x2 and y2) = 1” и по нажатию кнопки вычисляется результат.
13. На основе примера 10 создайте следующую программу. Есть 2 редактора для формул, и в них записано, например, “x + 1” и “x * 2”. В программе есть список K размерности 11 (n=0..10), состоящий из нулей, только K[1] = 1. В цикле по x от 1 до 10 запустите последовательное вычисление K[x + 1] += K[x] и K[2*x] += K[x].

Литература

[1] Лутц М. Программирование на Python, том I, 4-е издание. Пер.с англ. СПб.: Символ-Плюс, 2011. 992 с.

[2] Лутц М. Изучаем Python, 4-е издание. Пер.с англ. СПб.: Символ-Плюс, 2011. 1280 с.

[3] Лутц М. Программирование на Python, том II, 4-е издание. Пер.с англ. СПб.: Символ-Плюс, 2011. 992 с.

Интернет-ресурсы

[4] https://www.python.org, https://docs.python.org/3/ (Documentation Python)

[5] https://docs.python.org/3/library/tkinter.html#module-tkinter (tkinter doc)

[6] http://www.pythonware.com/library/, http://effbot.org/tkinterbook/

[7] https://www.tcl.tk/, https://www.tcl.tk/man/tcl8.6/TkCmd/contents.htm (Tk Documentation)

[8] https://tkdocs.com/tutorial/index.html (TkDocs)

Интернет-ресурсы (русскоязычные электронные учебники)

[9] https://younglinux.info/tkinter.php (Tkinter. Программирование GUI на Python. Курс)

[10] https://metanit.com/python/tutorial/9.1.php (Глава 9 электронного учебника по Python. «Создание графического интерфейса»)

[11] https://ru.wikiversity.org/wiki/Курс_по_библиотеке_Tkinter_языка_Python#Entry (Курс по библиотеке Tkinter языка Python)

[12] https://pythonru.com/uroki/obuchenie-python-gui-uroki-po-tkinter#toc (Обучение Python GUI (уроки по Tkinter))