**Виджеты**

**Виджет** (англ. *Widget*) — примитив графического интерфейса пользователя, имеющий стандартный внешний вид и выполняющий стандартные действия. Другие названия: элемент управления и элемент интерфейса.

**ПРОИСХОЖДЕНИЕ ТЕРМИНА «ВИДЖЕТ»**

Слово «widget» впервые фиксируется в 1926 году в том же значении, что и gadget — «приспособление, штуковина». Предполагают, что на его форму повлияло gadget, а происходит он из which it — «этот, как его».

Слово «Виджет» также используется и как название класса вспомогательных мини-программ — графических модулей, которые размещаются в рабочем пространстве соответствующей родительской программы и служат для украшения рабочего пространства, развлечения, решения отдельных рабочих задач или быстрого получения информации из интернета без помощи веб-браузера.

**ТИПОВЫЕ ЭЛЕМЕНТЫ ИНТЕРФЕЙСА**

В большинстве существует стандартный набор элементов интерфейса, включающий следующие элементы управления:

* кнопка (button)
* радиокнопка (radio button)
* флажок/переключатель (check box)
* значок (иконка, icon)
* список (list box)
* дерево — иерархический список (tree view)
* раскрывающийся список (combo box, drop-down list)
* поле редактирования (textbox, edit field)
* элемент для отображения табличных данных (grid view)
* меню (menu)
* главное меню окна (main menu)
* контекстное меню (popup menu)
* ниспадающее меню (pull down menu)
* окно (window)
* панель (panel)
* диалоговое окно (dialog box)
* модальное окно (modal window)
* вкладка (tab)
* панель инструментов (toolbar)
* полоса прокрутки (scrollbar)
* ползунок (slider)
* строка состояния (status bar)
* всплывающая подсказка (tooltip, hint)

## **Toplevel**

Toplevel - окно верхнего уровня. Обычно используется для создания многоконных программ, а также для диалоговых окон.

### МЕТОДЫ ВИДЖЕТА

* title - заголовок окна
* overrideredirect - указание оконному менеджеру игнорировать это окно. Аргументом является True или False. В случае, если аргумент не указан – получаем текущее значение. Если аргумент равен True, то такое окно будет показано оконным менеджером без обрамления (без заголовка и бордюра). Может быть использовано, например, для создания splashscreen при старте программы.
* iconify / deiconify - свернуть / развернуть окно
* withdraw - "спрятать" (сделать невидимым) окно. Для того, чтобы снова показать его надо использовать метод deiconify.
* minsize и maxsize - минимальный / максимальный размер окна. Методы принимают два аргумента – ширина и высота окна. Если аргументы не указаны - возвращают текущее значение.
* state - получить текущее значение состояния окна. Может возвращать следующие значения: normal (нормальное состояние), icon (показано в виде иконки), iconic (свёрнуто), withdrawn (не показано), zoomed (развёрнуто на полный экран, только для Windows и Mac OS X)
* resizable - может ли пользователь изменять размер окна. Принимает два аргумента - возможность изменения размера по горизонтали и по вертикали. Без аргументов возвращает текущее значение.
* geometry - устанавливает геометрию окна в формате ширина x высота + x + y (пример: geometry('600x400+40+80') - поместить окно в точку с координатам 40,80 и установить размер в 600x400). Размер или координаты могут быть опущены (geometry('600x400') - только изменить размер, geometry('+40+80') - только переместить окно).
* transient - сделать окно зависимым от другого окна, указанного в аргументе. Будет сворачиваться вместе с указанным окном. Без аргументов возвращает текущее значение.
* protocol - получает два аргумента: название события и функцию, которая будет вызываться при наступлении указанного события. События могут называться WM\_TAKE\_FOCUS (получение фокуса), WM\_SAVE\_YOURSELF (необходимо сохраниться, в настоящий момент является устаревшим), WM\_DELETE\_WINDOW (удаление окна).
* tkraise (синоним lift) и lower - поднимает (размещает поверх всех других окон) или опускает окно. Методы могут принимать один необязательный аргумент: над/под каким окном разместить текущее.
* grab\_set - устанавливает фокус на окно, даже при наличии открытых других окон
* grab\_release - снимает монопольное владение фокусом ввода с окна

Эти же методы могут быть использованы для корневого (root) окна.

Пример:

from tkinter import \*

def window\_deleted():

print ('Окно закрыто')

root.quit() # явное указание на выход из программы

root=Tk()

root.title('Пример приложения')

root.geometry('500x400+300+200') # ширина=500, высота=400, x=300, y=200

root.protocol('WM\_DELETE\_WINDOW', window\_deleted) # обработчик закрытия окна

root.resizable(True, False) # размер окна может быть изменен только по горизонтали

root.mainloop()

Таким способом можно предотвратить закрытие окна (например, если закрытие окна приведёт к потере введённых пользователем данных).

## **Button**

Виджет Button - самая обыкновенная кнопка, которая используется в тысячах программ. Пример кода:

from tkinter import \*

root = Tk()

button1 = Button(root, text = 'ok', width = 25, height = 5, bg = 'black', fg = 'red', font = 'arial 14')

button1.pack()

root.mainloop()

Разберем этот небольшой код. За создание окна отвечает класс Tk(), и первым делом нужно создать экземпляр этого класса. Этот экземпляр принято называть root, хотя вы можете назвать его как угодно. Далее создаётся кнопка, при этом мы указываем её свойства (начинать нужно с указания окна, в примере - root). Здесь перечислены некоторые из них:

* text - какой текст будет отображён на кнопке (в примере – ок)
* width, height - соответственно, ширина и высота кнопки.
* bg - цвет кнопки (сокращенно от background, в примере цвет - чёрный)
* fg - цвет текста на кнопке (сокращённо от foreground, в примере цвет - красный)
* font - шрифт и его размер (в примере - arial, размер - 14)

## **Label**

Label - это виджет, предназначенный для отображения какой-либо надписи без возможности редактирования пользователем. Имеет те же свойства, что и перечисленные свойства кнопки.

## **Entry**

Entry - это виджет, позволяющий пользователю ввести одну строку текста. Имеет дополнительное свойство bd (сокращённо от borderwidth), позволяющее регулировать ширину границы.

* borderwidth - ширина бордюра элемента
* width - задаёт длину элемента в знакоместах.

## **Text**

Text - это виджет, который позволяет пользователю ввести любое количество текста. Имеет дополнительное свойство wrap, отвечающее за перенос (чтобы, например, переносить по словам, нужно использовать значение WORD). Например:

from tkinter import \*

root = Tk()

text1 = Text(root, height = 7, width = 7, font = 'Arial 14', wrap = WORD)

text1.pack()

root.mainloop()

## **Listbox**

Listbox - это виджет, который представляет из себя список, из элементов которого пользователь может выбирать один или несколько пунктов. Имеет дополнительное свойство selectmode, которое, при значении SINGLE, позволяет пользователю выбрать только один элемент списка, а при значении EXTENDED – любое количество. Пример:

from tkinter import \*

root = Tk()

listbox1 = Listbox(root, height = 5, width = 15, selectmode = EXTENDED)

listbox2 = Listbox(root, height = 5, width = 15, selectmode = SINGLE)

list1 = ["Москва", "Санкт-Петербург", "Саратов", "Омск"]

list2 = ["Канберра", "Сидней", "Мельбурн", "Аделаида"]

for i in list1:

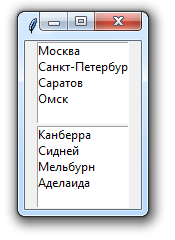
listbox1.insert(END, i)

for i in list2:

listbox2.insert(END, i)

listbox1.pack()

listbox2.pack()

root.mainloop()

## **Frame**

Виджет Frame (рамка) предназначен для организации виджетов внутри окна. Рассмотрим пример:

from tkinter import \*

root = Tk()

frame1=Frame(root, width = 100, heigh = 100, bg = 'green', bd = 5)

frame2 = Frame(root, width = 150, heigh = 75, bg = 'red', bd = 5)

button1 = Button(frame1, text = 'Первая кнопка')

button2 = Button(frame2, text='Вторая кнопка')

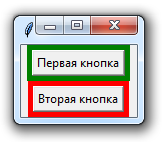
frame1.pack()

frame2.pack()

button1.pack()

button2.pack()

root.mainloop()

Свойство bd отвечает за расстояние от края рамки до виджетов.

## **Checkbutton**

Checkbutton - это виджет, который позволяет отметить "галочкой" определенный пункт в окне. При использовании нескольких пунктов нужно каждому присвоить свою переменную. Разберем пример:

from tkinter import \*

root=Tk()

var1=IntVar()

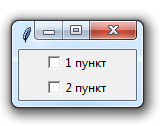
var2=IntVar()

check1=Checkbutton(root, text = '1 пункт', variable = var1, onvalue = 1, offvalue = 0)

check2 = Checkbutton(root, text = '2 пункт', variable = var2, onvalue = 1, offvalue=0)

check1.pack()

check2.pack()

root.mainloop()

IntVar() - специальный класс библиотеки для работы с целыми числами. variable - свойство, отвечающее за прикрепление к виджету переменной. onvalue, offvalue - свойства, которые присваивают прикреплённой к виджету переменной значение, которое зависит от состояния (onvalue - при выбранном пункте, offvalue - при невыбранном пункте).

## **Radiobutton**

Виджет Radiobutton выполняет функцию, схожую с функцией виджета Checkbutton. Разница в том, что в виджете Radiobutton пользователь может выбрать лишь один из пунктов. Реализация этого виджета несколько иная, чем виджета Checkbutton:

from tkinter import \*

root = Tk()

var = IntVar()

rbutton1 = Radiobutton(root, text = '1', variable = var, value = 1)

rbutton2 = Radiobutton(root, text = '2', variable = var, value = 2)

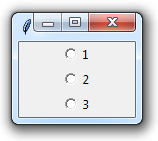
rbutton3 = Radiobutton(root, text = '3', variable = var, value = 3)

rbutton1.pack()

rbutton2.pack()

rbutton3.pack()

root.mainloop()



В этом виджете используется уже одна переменная. В зависимости от того, какой пункт выбран, она меняет своё значение. Самое интересное, что, если присвоить этой переменной какое-либо значение, поменяется и выбранный виджет.

## **Scale**

Scale (шкала) - это виджет, позволяющий выбрать какое-либо значение из заданного диапазона. Свойства:

* orient - как расположена шкала на окне. Возможные значения: HORIZONTAL, VERTICAL (горизонтально, вертикально).
* length - длина шкалы.
* from\_ - с какого значения начинается шкала.
* to - каким значением заканчивается шкала.
* tickinterval - интервал, через который отображаются метки шкалы.
* resolution - шаг передвижения (минимальная длина, на которую можно передвинуть движок)

Пример кода:

from tkinter import \*

root = Tk()

def getV(root):

a = scale1.get()

print("Значение", a)

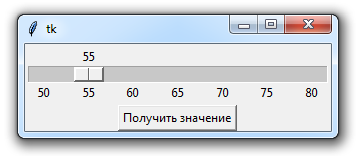
scale1 = Scale(root, orient = HORIZONTAL, length = 300, from\_ = 50, to = 80, tickinterval = 5, resolution = 5)

button1 = Button(root, text = "Получить значение")

scale1.pack()

button1.pack()

button1.bind("<Button-1>",getV)

root.mainloop()

Здесь используется специальный метод get(), который позволяет снять с виджета определенное значение, и используется не только в Scale.

## **Scrollbar**

Этот виджет даёт возможность пользователю "прокрутить" другой виджет (например, текстовое поле) и часто бывает полезен. Использование этого виджета достаточно нетривиально. Необходимо сделать две привязки: command полосы прокрутки привязываем к методу xview/yview виджета, а xscrollcommand/yscrollcommand виджета привязываем к методу set полосы прокрутки.

Рассмотрим на примере:

from tkinter import \*

root = Tk()

text = Text(root, height = 3, width = 60)

text.pack(side = 'left')

scrollbar = Scrollbar(root)

scrollbar.pack()

# первая привязка

scrollbar['command'] = text.yview

# вторая привязка

text['yscrollcommand'] = scrollbar.set

root.mainloop()

