Тема 11. Виджеты (графические объекты) и их свойства

Рассмотрим часть графических объектов (виджет), содержащихся в библиотеке `Tkinter`: **кнопки, поля для ввода, метки, флажки, переключатели и списки.** Следует понимать, что графический интерфейс пользователя достаточно стандартен, и поэтому любые подобные библиотеки-модули (в том числе и `Tkinter`) содержат приблизительно одинаковые виджеты.

Каждый класс виджет имеет определенные свойства, значения которых можно задавать при их создании, а также программировать их изменение при действии пользователя и в результате выполнения программы.

Кнопки

Объект-кнопка создается вызовом класса Button модуля tkinter. При этом обязательным аргументом является лишь родительский виджет (например, окно верхнего уровня). Другие свойства могут указываться при создании кнопки или задаваться (изменяться) позже. Синтаксис:

переменная = Button (родит_виджет, [свойство=значение, … ….])

У кнопки много свойств, в примере ниже указаны лишь некоторые из них.

```
from tkinter import *

root = Tk()

but = Button(root,
           text="Это кнопка", #надпись на кнопке
           width=30,height=5, #ширина и высота
           bg="white",fg="blue") #цвет фона и надписи

but.pack()
root.mainloop()
```

bg и fg – это сокращения от background (фон) и foreground (передний план). Ширина и высота измеряются в знакоместах (количество символов).

Метки

Метки (или надписи) — это достаточно простые виджеты, содержащие строку (или несколько строк) текста и служащие в основном для информирования пользователя.

```
lab = Label(root, text="Это метка! \n Из двух строк.",
                                        font="Arial 18")
```

Однострочное текстовое поле

Такое поле создается вызовом класса Entry модуля tkinter. В него пользователь может ввести только одну строку текста.

```
ent = Entry(root,width=20,bd=3)
```

bd – это сокращение от borderwidth (ширина границы).

Элемент Entry представляет поле для ввода текста. Конструктор Entry принимает следующие параметры:

```
Entry(master, options)
```

Где master - ссылка на родительское окно, а options - набор следующих параметров:

- bg: фоновый цвет
- bd: толщина границы
- cursor: курсор указателя мыши при наведении на текстовое поле
- fg: цвет текста
- font: шрифт текста
- justify: устанавливает выравнивание текста. Значение LEFT выравнивает текст по левому краю, CENTER - по центру, RIGHT - по правому краю
- relief: определяет тип границы, по умолчанию значение FLAT
- selectbackground: фоновый цвет выделенного куска текста
- selectforeground: цвет выделенного текста
- show: задает маску для вводимых символов
- state: состояние элемента, может принимать значения NORMAL (по умолчанию) и DISABLED
- textvariable: устанавливает привязку к элементу StringVar
- width: ширина элемента

Определим элемент Entry и по нажатию на кнопку выведем его текст в отдельное окно с сообщением:

```
from tkinter import *
from tkinter import messagebox

def show_message():
    messagebox.showinfo("GUI Python", message.get())

root = Tk()
root.title("GUI на Python")
root.geometry("300x250")

message = StringVar()

message_entry = Entry(textvariable=message)
message_entry.place(relx=.5, rely=.1, anchor="c")
```

```
message_button = Button(text="Click Me", command=show_message)
message_button.place(relx=.5, rely=.5, anchor="c")

root.mainloop()
```

Для вывода сообщения здесь применяется дополнительный модуль messagebox, содержащий функцию showinfo(), которая собственно и выводит введенный в текстовое поле текст. Для получения введенного текста используется комп StringVar.

Теперь создадим более сложный пример с формой ввода:

```
from tkinter import *
from tkinter import messagebox



def display_full_name():
    messagebox.showinfo("GUI Python", name.get() + " " + surname.get())

root = Tk()
root.title("GUI на Python")

name = StringVar()
surname = StringVar()

name_label = Label(text="Введите имя:")
surname_label = Label(text="Введите фамилию:")

name_label.grid(row=0, column=0, sticky="w")
surname_label.grid(row=1, column=0, sticky="w")

name_entry = Entry(textvariable=name)
surname_entry = Entry(textvariable=surname)

name_entry.grid(row=0,column=1, padx=5, pady=5)
surname_entry.grid(row=1,column=1, padx=5, pady=5)



message_button = Button(text="Click Me", command=display_full_name)
message_button.grid(row=2,column=1, padx=5, pady=5, sticky="e")

root.mainloop()
```

Методы Entry

Элемент Entry имеет ряд методов. Основные из них:

- insert(index, str): вставляет в текстовое поле строку по определенному индексу

- get(): возвращает введенный в текстовое поле текст
- delete(first, last=None): удаляет символ по индексу first. Если указан параметр last, то удаление производится до индекса last. Чтобы удалить до конца, в качестве второго параметра можно использовать значение END.

Используем методы в программе:

```
from tkinter import *
from tkinter import messagebox


def clear():
    name_entry.delete(0, END)
    surname_entry.delete(0, END)


def display():
    messagebox.showinfo("GUI Python", name_entry.get() + " " +
surname_entry.get())

root = Tk()
root.title("GUI на Python")

name_label = Label(text="Введите имя:")
surname_label = Label(text="Введите фамилию:")

name_label.grid(row=0, column=0, sticky="w")
surname_label.grid(row=1, column=0, sticky="w")

name_entry = Entry()
surname_entry = Entry()

name_entry.grid(row=0,column=1, padx=5, pady=5)
surname_entry.grid(row=1,column=1, padx=5, pady=5)

# вставка начальных данных
name_entry.insert(0, "Tom")
surname_entry.insert(0, "Soyer")

display_button = Button(text="Display", command=display)
clear_button = Button(text="Clear", command=clear)

display_button.grid(row=2, column=0, padx=5, pady=5, sticky="e")
clear_button.grid(row=2, column=1, padx=5, pady=5, sticky="e")

root.mainloop()
```

При запуске программы сразу же в оба текстовых поля добавляется текст по умолчанию:

```
name_entry.insert(0, "Tom")
surname_entry.insert(0, "Soyer")
```

```
name_entry.insert("end", "Tom")
```

А так можно вставить текст «в конец поля»

Кнопка Clear очищает оба поля, вызывая метод delete:

```
def clear():
    name_entry.delete(0, END)
    surname_entry.delete(0, END)
```

Вторая кнопка, используя метод get, получает введенный текст:

```
def display():
    messagebox.showinfo("GUI Python", name_entry.get()
                        + " " + surname_entry.get())
```

Причем, как видно из примера, нам необязательно обращаться к тексту в Entry через переменные типа StringVar, мы можем это сделать напрямую через метод get

Многострочное текстовое поле

Text предназначен для предоставления пользователю возможности ввода не одной строки текста, а существенно больше.

```
tex = Text(root,width=40,
           font="Verdana 12",
           wrap=WORD)
```

Последнее свойство (wrap) в зависимости от своего значения позволяет переносить текст, вводимый пользователем либо по символам, либо по словам, либо вообще не переносить, пока пользователь не нажмет Enter.

Радиокнопки (переключатели)

Объект-радиокнопка никогда не используется по одному. Их используют группами, при этом в одной группе может быть «включена» лишь одна кнопка.

```
var=IntVar()
var.set(1)
rad0 = Radiobutton(root,text="Первая",
           variable=var,value=0)
rad1 = Radiobutton(root,text="Вторая",
           variable=var,value=1)
rad2 = Radiobutton(root,text="Третья",
           variable=var,value=2)
```

Одна группа определяет значение одной переменной, т. е. если в примере будет выбрана радиокнопка rad2, то значение переменной будет var будет 2. Изначально также требуется установить значение переменной (выражение var.set(1) задает значение переменной var равное 1).

Флажки

Объект checkbutton предназначен для выбора не взаимоисключающих пунктов в окне (в группе можно активировать один, два или более флажков или не один). В отличие от радиокнопок, значение каждого флажка привязывается к своей переменной, значение которой определяется опциями onvalue (включено) и offvalue (выключено) в описании флажка.

```
c1 = IntVar()
c2 = IntVar()
che1 = Checkbutton(root,text="Первый флажок",
          variable=c1,onvalue=1,offvalue=0)
che2 = Checkbutton(root,text="Второй флажок",
          variable=c2,onvalue=2,offvalue=0)
```

Списки

Вызов класса Listbox создает объект, в котором пользователь может выбрать один или несколько пунктов в зависимости от значения опции selectmode. В примере ниже значение SINGLE позволяет выбирать лишь один пункт из списка.

```
r = ['Linux','Python','Tk','Tkinter']
lis = Listbox(root,selectmode=SINGLE,height=4)
for i in r:
     lis.insert(END,i)
```

Изначально список (Listbox) пуст. С помощью цикла for в него добавляются пункты из списка (тип данных) r. Добавление происходит с помощью специального метода класса Listbox — insert. Данный метод принимает два параметра: куда добавить и что добавить.

Большинство методов различных виджет мы рассмотрим по ходу изучения данного курса.

Виджеты (графические объекты) и их свойства

Продолжим рассматривать графические объекты (виджеты), содержащихся в библиотеке Tkinter. Это будут рамка (frame), шкала (scale), полоса прокрутки (scrollbar), окно верхнего уровня (toplevel).

Frame (рамка)

Как выяснится позже, рамки (фреймы) хороший инструмент для организации остальных виджет в группы внутри окна, а также оформления.

```
from tkinter import *

root = Tk()

fra1 = Frame(root,width=500,height=100,bg="darkred")
```

```
fra2 =

    Frame(root,width=300,height=200,bg="green",bd=20)
fra3 = Frame(root,width=500,height=150,bg="darkblue")

fra1.pack()
fra2.pack()
fra3.pack()

root.mainloop()
```

Данный скрипт создает три фрейма разного размера. Свойство bd (сокращение от `boderwidth`) определяет расстояния от края рамки до заключенных в нее виджетов (если они есть).

На фреймах также можно размещать виджеты как на основном окне (`root`). Здесь текстовое поле находится на рамке `fra2`.

```
ent1 = Entry(fra2,width=20)
ent1.pack()
```

Scale (шкала)

Назначение шкалы — это предоставление пользователю выбора какого-то значения из определенного диапазона. Внешне шкала представляет собой горизонтальную или вертикальную полосу с разметкой, по которой пользователь может передвигать движок, осуществляя тем самым выбор значения.

```
sca1 = Scale(fra3,orient=HORIZONTAL,length=300,
          from_=0,to=100,tickinterval=10,resolution=5)
sca2 = Scale(root,orient=VERTICAL,length=400,
          from_=1,to=2,tickinterval=0.1,resolution=0.1)
```

Свойства:

- `orient` определяет направление шкалы;
- `length` – длина шкалы в пикселях;
- `from_` и `to` – с какого значения шкала начинается и каким заканчивается (т. е. диапазон значений);
- `tickinterval` – интервал, через который отображаются метки для шкалы;
- `resolution` - минимальная длина отрезка, на которую пользователь может передвинуть движок.

Scrollbar (полоса прокрутки)

Данный виджет позволяет прокручивать содержимое другого виджета (например, текстового поля или списка). Прокрутка может быть как по горизонтали, так и по вертикали.

```
from tkinter import *

root = Tk()
```

```
tx = Text(root,width=40,height=3,font='14')
scr = Scrollbar(root,command=tx.yview)
tx.configure(yscrollcommand=scr.set)

tx.grid(row=0,column=0)
scr.grid(row=0,column=1)
root.mainloop()
```

В примере сначала создается текстовое поле (`tx`), затем полоса прокрутки (`scr`), которая привязывается с помощью опции `command` к полю `tx` по вертикальной оси (`yview`). Далее поле `tx` изменяется (конфигурируется) с помощью метода configure: устанавливается значение опции `yscrollcommand`.

Здесь используется незнакомый нам пока еще метод `grid`, представляющий собой другой способ расположения виджет на окне.

Toplevel (окно верхнего уровня)

С помощью класс `Toplevel` создаются дочерние окна, на которых также могут располагаться виджеты. Следует отметить, что при закрытии главного окна (или родительского), окно `Toplevel` также закрывается. С другой стороны, закрытие дочернего окна не приводит к закрытию главного.

```
win = Toplevel(root,relief=SUNKEN,bd=10,bg="lightblue")
win.title("Дочернее окно")
win.minsize(width=400,height=200)
```

Метод `title` определяет заголовок окна. Метод `minsize` конфигурирует минимальный размер окна (есть метод `maxsize`, определяющий максимальный размер окна). Если значение аргументов `minsize` будет таким же как у `maxsize`, то пользователь не сможет менять размеры окна.