Урок 2. Виджеты Button, Label, Entry

В этом уроке рассмотрим подробнее три наиболее простых и популярных виджета GUI – кнопку, метку и однострочное текстовое поле. В tkinter объекты этих элементов интерфейса порождаются соответственно от классов Button, Label и Entry.

Свойства и методы виджетов бывают относительно общими, характерными для многих типов, а также частными, зачастую встречающимися только у какого-то одного класса. В любом случае список настраиваемых свойств велик. В этом курсе мы будем рассматривать только ключевые свойства и методы классов пакета tkinter.

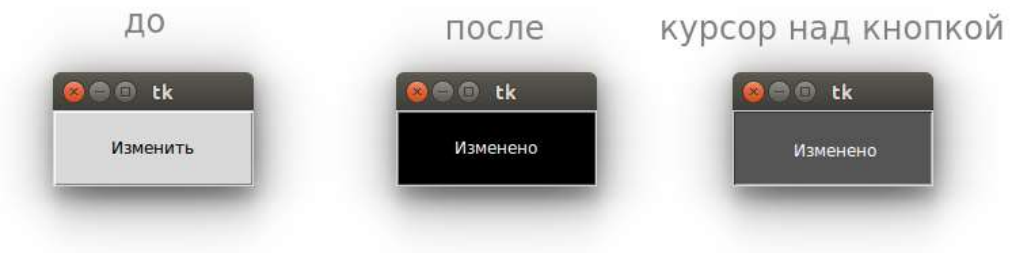
В Tkinter существует три способа конфигурирования свойств виджетов: в момент создания объекта, с помощью метода config(), он же configure(), путем обращения к свойству как к элементу словаря.

Button – кнопка

Самыми важными свойствами виджета класса Button являются text, с помощью которого устанавливается надпись на кнопке, и command для установки действия, то есть того, что будет происходить при нажатии на кнопку. По умолчанию размер кнопки соответствует ширине и высоте текста, однако с помощью свойств width и height эти параметры можно изменить. Единицами измерения в данном случае являются знакоместа. Такие свойства как bg, fg, activebackground и activeforeground определяют соответственно цвет фона и текста, цвет фона и текста во время нажатия (и установки курсора мыши над кнопкой).

```
from tkinter import *

root = Tk()
b1 = Button(text="Изменить", width=15, height=3)

def change():
    b1['text'] = "Изменено"
    b1['bg'] = '#000000'
    b1['activebackground'] = '#555555'
    b1['fg'] = '#ffffff'
    b1['activeforeground'] = '#ffffff'

b1.config(command=change)

b1.pack()
root.mainloop()
```

Здесь свойство command устанавливается с помощью метода config(). Однако можно было сделать и так: `b1['command'] = change`. Вот так будет выглядеть кнопка после запуска программы и после нажатия на нее:



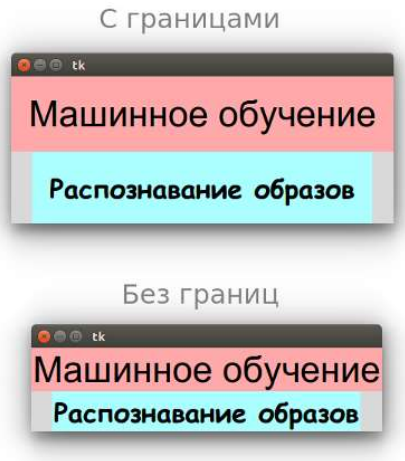
Label – метка

Виджет Label просто отображает текст в окне и служит в основном для информационных целей (вывод сообщений, подпись других элементов интерфейса). Свойства метки во многом схожи с таковыми у кнопки. Однако у меток нет опции command. Клик по метке не обрабатывается Tkinter. На примере объекта типа Label рассмотрим свойство font – шрифт.

```
from tkinter import *
root = Tk()
l1 = Label(text="Машинное обучение", font="Arial 32")
l2 = Label(text="Распознавание образов", font=("Comic Sans MS", 24, "bold"))
l1.config(bd=20, bg='#ffaaaa')
l2.config(bd=20, bg='#aaffff')
l1.pack()
l2.pack()
root.mainloop()
```

Значение шрифта можно передать как строку или как кортеж. Второй вариант удобен, если имя шрифта состоит из двух и более слов. После названия шрифта можно указать размер и стиль.

Также как font свойство bd есть не только у метки. С его помощью регулируется размер границ (единица измерения – пиксель):



Бывает, что метки и кнопки не присваивают переменным, если потом к ним в коде не приходится обращаться. Их создают от класса и сразу размещают:

```
from tkinter import *

def take():
    l['text'] = "Выдано"

root = Tk()
Label(text="Пункт выдачи").pack()
```

```
Button(text="Взять", command=take).pack()
l = Label(width=10, height=1)
l.pack()
root.mainloop()
```

В данном примере только у одной метки есть связь с переменной, так как одно из ее свойств может быть изменено в процессе выполнения программы.

Entry – однострочное текстовое поле

Текстовые поля предназначены для ввода информации пользователем. Однако нередко также для вывода, если предполагается, что текст из них будет скопирован. Текстовые поля как элементы графического интерфейса бывают однострочными и многострочными. В tkinter вторым соответствует класс Text, который будет рассмотрен позже.

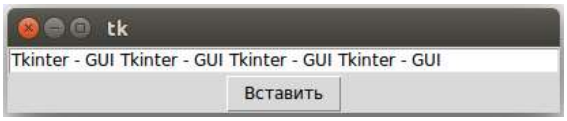
Свойства экземпляров Entry во многом схожи с двумя предыдущими виджетами. А вот методы – нет. Из текстового поля можно взять текст. За это действие отвечает метод get(). В текстовое поле можно вставить текст методом insert(). Также можно удалить текст методом delete().

Метод insert() принимает позицию, в которую надо вставлять текст, и сам текст.

Такой код

```
from tkinter import *
root = Tk()
e1 = Entry(width=50)
def insert():
    e1.insert(0,"Tkinter - GUI ")
b = Button(text="Вставить", command=insert)
e1.pack()
b.pack()
root.mainloop()
```

приведет к тому, что после каждого нажатия на кнопку будет вставляться новая фраза "Tkinter - GUI " перед уже существующей в поле строкой.



Если 0 в insert() заменить на константу END, то вставляться будет в конец. Можно указать любое число-индекс знакоместа, тогда вставка будет производиться куда-либо в середину строки.

Метод delete() принимает один или два аргумента. В первом случае удаляется один символ в указанной позиции. Во втором – срез между двумя указанными индексами, не включая последний. Если нужно полностью очистить поле, то первым аргументом должен быть 0, вторым – END.

Практическая работа

Напишите программу, состоящую из семи кнопок, цвета которых соответствуют цветам радуги. При нажатии на ту или иную кнопку в текстовое поле должен вставляться код цвета, а в метку – название цвета.

Коды цветов в шестнадцатеричной кодировке: #ff0000 – красный, #ff7d00 – оранжевый, #ffff00 – желтый, #00ff00 – зеленый, #007dff – голубой, #0000ff – синий, #7d00ff – фиолетовый.

Примерно должно получиться так:

Для выравнивания строки по центру в текстовом поле используется свойство justify со значением CENTER.