Тема 12. Программирование событий. Переменные Tkinter.

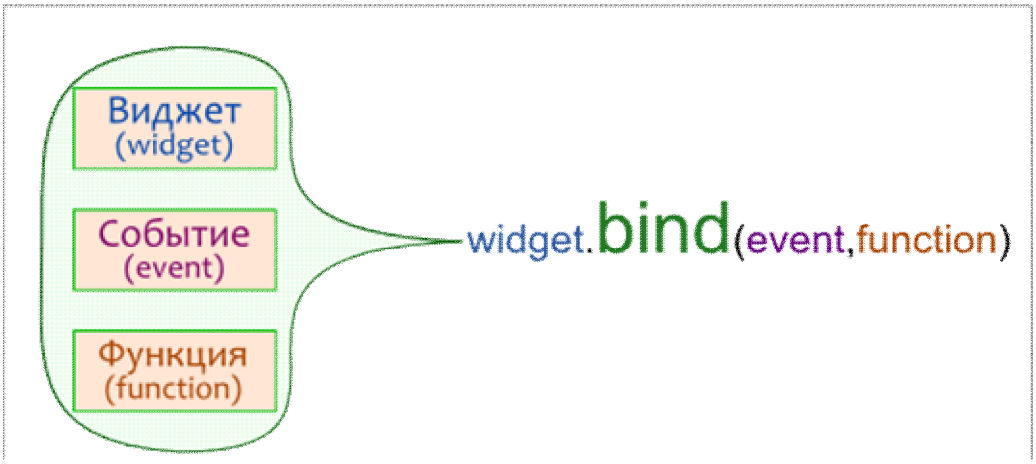
Метод bind модуля Tkinter.

Приложения с графическим интерфейсом пользователя (GUI) должны не просто красиво отображаться на экране, но и выполнять какие-либо действия, реализуя тем самым потребности пользователя. На прошлых уроках было рассказано как создать GUI, на этом уроке рассмотрим как добавить ему функциональность, т.е возможность совершать с его помощью те или иные действия.

В отличие от консольных приложений, которые обычно выполняются при минимальных внешних воздействиях, графическое приложение обычно ждет каких-либо внешних воздействий (щелчков кнопкой мыши, нажатий клавиш на клавиатуре, изменения виджетов) и затем выполняет заложенное программистом действие. Из такого принципа работы можно вывести следующую схему настройки функциональности GUI: на виджет что-то «влияет» из вне ? выполняется какая-то функция (действие). Внешнее воздействие на графический компонент называется событием. Событий достаточно много (основной их перечень мы рассмотрим на следующем занятии). На этом занятии будем использовать лишь два вида событий: щелчок левой кнопкой мыши () и нажатие клавиши `Enter` ().

Одним из способов связывания виджета, события и функции (того, что должно происходить после события) является использование метода `bind`. Синтаксис связывания представлен на рисунке ниже.

Добавление функциональности графическому элементу с помощью метода bind



Рассмотрим различные примеры добавления функциональности GUI.

Пример 1.

```
def output(event):
```

```
    s = ent.get()
    if s == "1":
        tex.delete(1.0,END)
        tex.insert(END,"Обслуживание клиентов на

                                        втором этаже")
    elif s == "2":
        tex.delete(1.0,END)
        tex.insert(END,"Пластиковые карты выдают в

                                    соседнем здании")
    else:
        tex.delete(1.0,END)
        tex.insert(END,"Введите 1 или 2 в поле

                                            слева")

from tkinter import *
root = Tk()

ent = Entry(root,width=1)  ## 16
but = Button(root,text="Вывести")  ## 17
tex =Text(root,width=20,height=3,font="12",wrap=WORD)

ent.grid(row=0,column=0,padx=20)    ## 22
but.grid(row=0,column=1)
tex.grid(row=0,column=2,padx=20,pady=10)

but.bind("<Button-1>",output)  ## 24

root.mainloop()
```

Рассмотрим код, начиная с 16-й строки.

В строках 16-18 создаются три виджета: однострочное текстовое поле, кнопка и многострочное текстовое поле. В первое поле пользователь должен что-то ввести, затем нажать кнопку и получить ответ во втором поле.

В строках 20-22 используется менеджер `grid` для размещения виджетов. Свойства `padx` и `pady` определяют количество пикселей от виджета до края рамки (или ячейки) по осям x и y соответственно.

В стоке 24 как раз и происходит связывание кнопки с событием нажатия левой кнопки мыши и функцией `output`. Все эти три компонента (виджет, событие и функция) связываются с помощью метода `bind`. В данном случае, при нажатии левой кнопкой мыши по кнопке `but` будет вызвана функция `output`.

Итак, если вдруг пользователь щелкнет левой кнопкой мыши по кнопке, то выполнится функция `output` (ни в каком другом случае она выполняться

не будет). Данная функция (строки 1-11) выводит информацию во второе текстовое поле. Какую именно информацию, зависит от того, что пользователь ввел в первое текстовое поле. В качестве аргумента функции передается событие (в данном случае).

Внутри веток `if-elif-else` используются методы `delete` и `insert`. Первый из них удаляет символы из текстового поля, второй — вставляет. 1.0 — обозначает первую строку, первый символ (нумерация символов начинается с нуля).

Пример 2.

```
li = ["red","green"]
def color(event):
     fra.configure(bg=li[0])
     li[0],li[1] = li[1],li[0]

def outgo(event):
     root.destroy()

from tkinter import *
root = Tk()
fra = Frame(root,width=100,height=100)    # 12 & 13
but = Button(root,text="Выход")
fra.pack()
but.pack()
root.bind("<Return>",color)              #  18
but.bind("<Button-1>",outgo)              #  19

root.mainloop()
```

Здесь создаются два виджета (строки 12, 13): фрейм и кнопка.

Приложение реагирует на два события: нажатие клавиши `Enter` в пределах главного окна (строка 18) и нажатие левой кнопкой мыши по кнопке `but` (строка 19). В первом случае вызывается функция `color`, во втором — outgo.

Функция color изменяет цвет фона (`bg`) фрейма (`fra`) с помощью метода configure, который предназначен для изменения значения свойств виджетов в процессе выполнения скрипта. В качестве значения опции bg подставляется первый элемент списка. Затем в списке два элемента меняются местами, чтобы при следующем нажатии Enter цвет фрейма снова изменился.

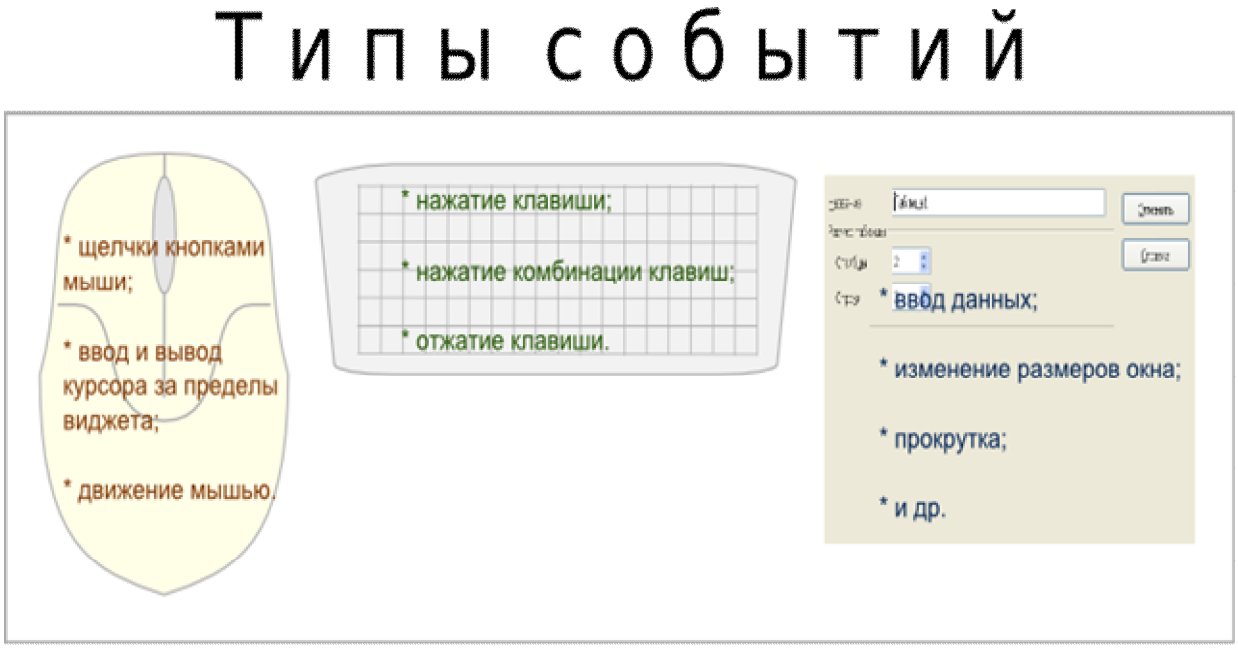
В функции outgo вызывается метод destroy по отношению к главному окну. Данный метод предназначен для «разрушения» виджета (окно закроется).

Программирование событий в Tkinter

Обычно, чтобы графическое приложение что-то сделало, должно случиться какое-нибудь событие, т. е. воздействие на GUI из вне.

Типы событий

Можно выделить три основных типа событий: производимые мышью, нажатиями клавиш на клавиатуре, а также события, возникающие в результате изменения других графических объектов.



Способ записи

При вызове метода `bind` событие передается в качестве первого аргумента.

Название события заключается в кавычки, а также в знаки < и >. Событие описывается с помощью зарезервированных последовательностей ключевых слов.

События, производимые мышью

- **<Button-1>** - щелчок левой кнопкой мыши
- **<Button-2>** - щелчок средней кнопкой мыши
- **<Button-3>** - щелчок правой кнопкой мыши
- **<Double-Button-1>** - двойной клик левой кнопкой мыши
- **<Motion>** - движение мыши

и т. д.

Пример:

```
from tkinter import *
def b1(event):
     root.title("Левая кнопка мыши")
def b3(event):
     root.title("Правая кнопка мыши")
def move(event):
     root.title("Движение мышью")

root = Tk()
root.minsize(width = 500, height=400)

root.bind('<Button-1>',b1)
root.bind('<Button-3>',b3)
root.bind('<Motion>',move)

root.mainloop()
```

В этой программе меняется надпись в заголовке главного окна в зависимости от того двигается мышь, щелкают левой или правой кнопкой мыши.

Событие (`Event`) – это один из объектов `tkinter`. У событий есть атрибуты, как и у многих других объектов. В примере в функции `move` извлекаются значения атрибутов `x` и `y` объекта *event*, в которых хранятся координаты местоположения курсора мыши в пределах виджета, по отношению к которому было сгенерировано событие. В данном случае виджетом является главное окно, а событием – `<Motion>`, т. е. перемещение мыши.

В программе ниже выводится информация об экземпляре `Event` и некоторым его свойствам. Все атрибуты можно посмотреть с помощью команды `dir(event)`. У разных событий они одни и те же, меняются только значения. Для тех или иных событий часть атрибутов не имеет смысла, такие свойства имеют значения по умолчанию.

В примере хотя обрабатывается событие нажатия клавиши клавиатуры, в поля `x`, `y`, `x_root`, `y_root` сохраняются координаты положения на экране курсора мыши.

```
from tkinter import *


def event_info(event):
    print(type(event))
    print(event)
    print(event.time)
```

```
    print(event.x_root)
    print(event.y_root)


root = Tk()
root.bind('a', event_info)
root.mainloop()
```

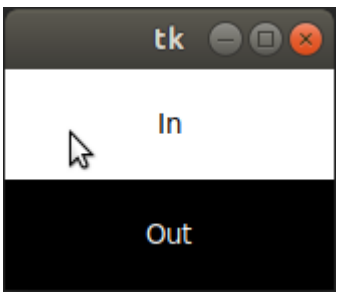
Пример выполнения программы:

```
<class 'tkinter.Event'>
<KeyPress event state=Mod2 keysym=a keycode=38
                              char='a' x=9 y=7>
8379853
37
92
```

Для событий с клавиатуры буквенные клавиши можно записывать без угловых скобок (например, `'a'`).

Для неалфавитных клавиш существуют специальные зарезервированные слова. Например, `<Return>` - нажатие клавиши Enter, <space> - пробел. (Заметим, что есть событие `<Enter>`, которое не имеет отношения к нажатию клавиши Enter, а происходит, когда курсор заходит в пределы виджета.)

Рассмотрим программу:




```
from tkinter import *


def enter_leave(event):
    if event.type == '7':
        event.widget['text'] = 'In'
    elif event.type == '8':
        event.widget['text'] = 'Out'


root = Tk()

lab1 = Label(width=20, height=3, bg='white')
lab1.pack()
```

```
lab1.bind('<Enter>', enter_leave)
lab1.bind('<Leave>', enter_leave)

lab2 = Label(width=20, height=3, bg='black',
             fg='white')
lab2.pack()
lab2.bind('<Enter>', enter_leave)
lab2.bind('<Leave>', enter_leave)
root.mainloop()
```

В ней две метки используют одну и ту же функцию, и каждая метка использует эту функцию для обработки двух разных событий: ввода курсора в пределы виджета и вывода во границы.

Функция, в зависимости от того, по отношению к какому виджету было зафиксировано событие, изменяет свойства только этого виджета. Как изменяет, зависит от произошедшего события.

Свойство `event.widget` содержит ссылку на виджет, сгенерировавший событие.

Свойство **`event.type`** описывает, что это было за событие.

У каждого события есть имя и номер. С помощью выражения `print(repr(event.type))` можно посмотреть его полное описание.

При этом на одних платформах `str(event.type)` возвращает имя события (например, `'Enter'`), на других – строковое представление номера события (например, '7').

Вернемся к событиям клавиатуры. Сочетания клавиш пишутся через тире. В случае использования, так называемого модификатора, он указывается первым, детали на третьем месте. Например, `<Shift-Up>` - одновременное нажатие клавиш `Shift` и стрелки вверх, `<Control-B1-Motion>` – движение мышью с зажатой левой кнопкой и клавишей Ctrl.

```
from tkinter import *


def exit_win(event):
    root.destroy()


def to_label(event):
    t = ent.get()
    lbl.configure(text=t)


def select_all(event):

    def select_all2(widget):
        widget.selection_range(0, END)
        widget.icursor(END)  # курсор в конец
```

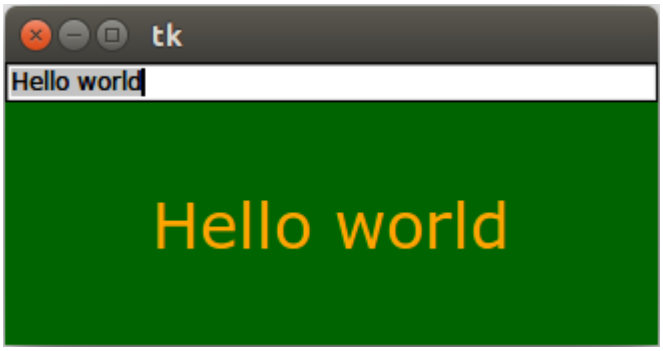
```
    root.after(10, select_all2, event.widget)


root = Tk()

ent = Entry(width=40)
ent.focus_set()
ent.pack()
lbl = Label(height=3, fg='orange',
            bg='darkgreen', font="Verdana 24")
lbl.pack(fill=X)

ent.bind('<Return>', to_label)
ent.bind('<Control-a>', select_all)
root.bind('<Control-q>', exit_win)

root.mainloop()
```



Здесь сочетание клавиш Ctrl+a выделяет текст в поле. Без `root.after()` выделение не работает. Метод `after` выполняет функцию, указанную во втором аргументе, через промежуток времени, указанный в первом аргументе. В третьем аргументе передается значение атрибута `widget` объекта *event*. В данном случае им будет поле *ent*. Именно оно будет передано как аргумент в функцию *select_all2* и присвоено параметру *widget*.

События, производимые с помощью клавиатуры

- Буквенные клавиши можно записывать без угловых скобок (например, 'L').
- Для неалфавитных клавиш существуют специальные зарезервированные слова
  * <Return> - нажатие клавиши Enter;
  * <space>- пробел;
  * и т. д.

- Сочетания клавиш пишутся через тире. Например:
  * <Control-Shift> - одновременное нажатие клавиш Ctrl и Shift.

Примерчик:

```
from tkinter import *

def exit_(event):
     root.destroy()
def caption(event):
     t = ent.get()
     lbl.configure(text = t)

root = Tk()

ent = Entry(root, width = 40)
lbl = Label(root, width = 80)

ent.pack()
lbl.pack()

ent.bind('<Return>',caption)
root.bind('<Control-z>',exit_)

root.mainloop()
```

При нажатии клавиши Enter в пределах текстовой строки (ent) вызывается функция caption, которая помещает символы из текстовой строки (ent) в метку (lbl). Нажатие комбинации клавиш Ctrl + z приводит к закрытию главного окна.

I. Переменные Tkinter.

Библиотека Tkinter содержит специальные классы, объекты которых выполняют роль переменных для хранения значений о состоянии различных виджет. Изменение значения такой переменной ведет к изменению и свойства виджета, и наоборот: изменение свойства виджета изменяет значение ассоциированной переменной.

Существует несколько таких классов Tkinter, предназначенных для обработки данных разных типов.

1. **StringVar**() - для строк;
2. **IntVar**() - целых чисел;
3. **DoubleVar**() - дробных чисел;
4. **BooleanVar**() - для обработки булевых значений (true и false).

Пример 1.

Раньше мы уже использовали переменную-объект типа **IntVar**() при создании группы радиокнопок:

```
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)
```

Здесь создается объект класса `IntVar` и связывается с переменной `var`. С помощью метода `set` устанавливается начальное значение, равное 1. Три радиокнопки относятся к одной группе: об этом свидетельствует одинаковое значение опции (свойства) `variable`. `Variable` предназначена для связывания переменной Tkinter с радиокнопкой. Опция value определяет значение, которое будет передано переменной, если данная кнопка будет в состоянии "включено". Если в процессе выполнения скрипта значение переменной `var` будет изменено, то это отразится на группе кнопок. Например, это делается во второй строчке кода: включена кнопка `rad1`.

Если метод `set` позволяет устанавливать значения переменных, то метод `get`, наоборот, позволяет получать (узнавать) значения для последующего их использования.

```
def display(event):
     v = var.get()
     if v == 0:
          print ("Включена первая кнопка")
     elif v == 1:
          print ("Включена вторая кнопка")
     elif v == 2:
          print ("Включена третья кнопка")

but = Button(root,text="Получить значение")
but.bind('<Button-1>',display)
```

При вызове функции `display` в переменную v “записывается“ значение, связанное в текущий момент с переменной `var`. Чтобы получить значение переменной `var`, используется метод `get` (вторая строчка кода).

Пример 2.

Несколько сложнее обстоит дело с флажками. Поскольку состояния флажков независимы друг друга, то для каждого должна быть введена собственная ассоциированная переменная-объект.

```
from tkinter import *

root = Tk()
```

```
var0=StringVar() # значение каждого флажка ...
var1=StringVar() # ... хранится в собственной
переменной
var2=StringVar()
# если флажок установлен, то в ассоциированную
переменную ...
# ...(var0,var1 или var2) заносится значение onvalue,
...
# ...если флажок снят, то - offvalue.
ch0 = Checkbutton(root,text="Окружность",variable=var0,
          onvalue="circle",offvalue="-")
ch1 = Checkbutton(root,text="Квадрат",variable=var1,
          onvalue="square",offvalue="-")
ch2 =
Checkbutton(root,text="Треугольник",variable=var2,
          onvalue="triangle",offvalue="-")

lis = Listbox(root,height=3)
def result(event):
     v0 = var0.get()
     v1 = var1.get()
     v2 = var2.get()
     l = [v0,v1,v2] # значения переменных заносятся в
                                                 список
     lis.delete(0,2) # предыдущее содержимое удаляется
                                               из Listbox
     for v in l: # содержимое списка l последовательно
          lis.insert(END,v) # ...вставляется в Listbox

but = Button(root,text="Получить значения")
but.bind('<Button-1>',result)

ch0.deselect()# "по умолчанию" флажки сняты
ch1.deselect()
ch2.deselect()

ch0.pack()
ch1.pack()
ch2.pack()
but.pack()
lis.pack()

root.mainloop()
```

Пример 3.

Помимо свойства (опции) variable, связывающей виджет с переменной-объектом Tkinter (IntVar, StringVar и др.), у многих виджет существует опция textvariable, которая определяет текст-содержимое или текст-надпись виджета. Несмотря на то, что «текстовое свойство» может быть установлено для виджета и изменено в процессе выполнения кода без использования ассоциированных переменных, иногда такой способ изменения оказывается более удобным.

```
from tkinter import *
root = Tk()
v = StringVar()
ent1 = Entry (root, textvariable =
                              v,bg="black",fg="white")
ent2 = Entry(root, textvariable = v)
ent1.pack()
ent2.pack()
root.mainloop()
```

Здесь содержимое одного текстового поля немедленно, отображается в другом, т.к. оба поля привязаны к одной и той же переменной v.