Тема 12. Програмування подій. Змінні Tkinter.

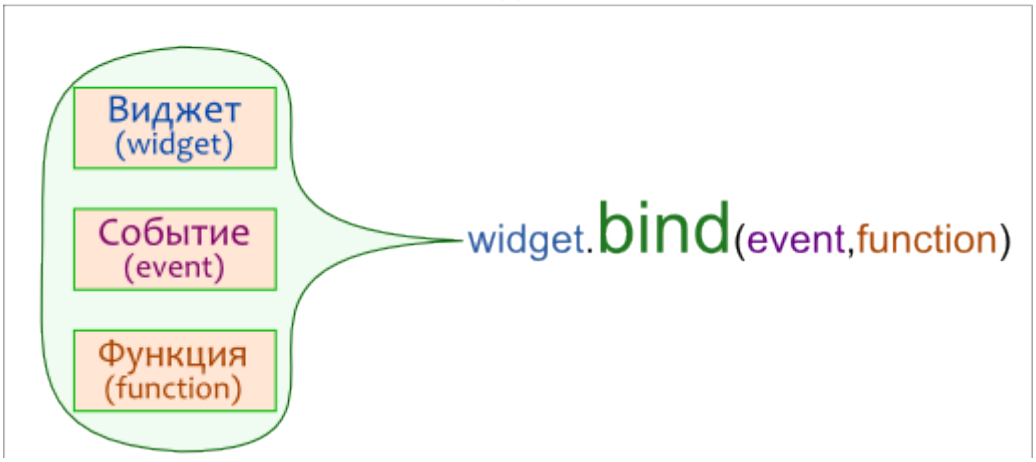
Метод bind модуля Tkinter.

Програми з графічним інтерфейсом користувача (GUI) повинні не просто красиво відображатися на екрані, але й виконувати будь-які дії, реалізуючи потреби користувача. На минулих уроках було розказано як створити GUI, на цьому уроці розглянемо, як додати йому функціональність, тобто можливість здійснювати за його допомогою ті чи інші дії.

На відміну від консольних додатків, які зазвичай виконуються при мінімальних зовнішніх впливах, графічний додаток зазвичай чекає на будь-які зовнішні впливи (клацань кнопкою миші, натискання клавіш на клавіатурі, зміни віджетів) і потім виконує закладену програмістом дію. З такого принципу роботи можна вивести наступну схему налаштування функціональності GUI: на віджет щось «впливає» з ? виконується якась функція (дія). Зовнішній вплив на графічний компонент називається подією. Події досить багато (основний їх перелік ми розглянемо на наступному занятті). На цьому занятті будемо використовувати лише два види подій: клацання лівою кнопкою миші () та натискання клавіші Enter ().

Одним із способів зв'язування віджету, події та функції (те, що має відбуватися після події) є використання методу bind. Синтаксис зв'язування представлений малюнку нижче.

Добавление функциональности графическому элементу с помощью метода bind



Розглянемо різні приклади додавання функціональності GUI.

приклад 1.

```
defoutput(event):
    s = ent.get()
    if s == "1":
```

```
          tex.delete(1.0,END)
          tex.insert(END,"Обслуговування клієнтів на
                                        другому
поверсі")
     elif s == "2":
          tex.delete(1.0,END)
          tex.insert(END,"Пластикові карти видають у
                                      сусідній будівлі")
     else:
          tex.delete(1.0,END)
          tex.insert(END,"Введіть 1 або 2 у поле
                                             зліва")

з tkinter import *
root = Tk()

ent = Entry(root,width=1) ## 16
but = Button(root,text="Вивести") ## 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()
```

Розглянемо код, починаючи з 16-го рядка.

У рядках 16-18 створюються три віджети: однорядкове текстове поле, кнопка та багаторядкове текстове поле. У першому полі користувач повинен щось ввести, потім натиснути кнопку і отримати відповідь у другому полі.

У рядках 20-22 використовується менеджер grid для розміщення віджетів. Властивості padx та pady визначають кількість пікселів від віджету до краю рамки (або комірки) по осях x та y відповідно.

У стоку 24 саме відбувається зв'язування кнопки з подією натискання лівої кнопки миші і функцією output. Всі ці три компоненти (віджет, подія та функція) зв'язуються за допомогою методу bind. В даному випадку при натисканні лівою кнопкою миші по кнопці but буде викликана функція output.

Отже, якщо раптом користувач клацне лівою кнопкою миші по кнопці, виконається функція output (ні в якому іншому випадку вона виконуватися не буде). Ця функція (рядки 1-11) виводить інформацію у друге текстове поле. Яку саме інформацію залежить від того, що користувач ввів у перше текстове поле. Як аргумент функції передається подія (у разі).

Усередині гілок if-elif-else використовуються методи delete та insert. Перший видаляє символи з текстового поля, другий — вставляє. 1.0 -

позначає перший рядок, перший символ (нумерація символів починається з нуля).

приклад 2.

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

defoutgo(event):
     root.destroy()

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

root.mainloop()
```

Тут створюються два віджети (рядки 12, 13): кадр і кнопка.

Програма реагує на дві події: натискання клавіші Enter в межах головного вікна (рядок 18) та натискання лівою кнопкою миші по кнопці but (рядок 19). У першому випадку викликається функція color, у другому – outgo.

Функція color змінює колір фону (bg) кадру (fra) за допомогою методу configure, який призначений для зміни значення властивостей віджетів у процесі скрипту. Як значення опції bg підставляється перший елемент списку. Потім у списку два елементи змінюються місцями, щоб при наступному натисканні Enter колір кадру знову змінився.

У функції outgo викликається метод destroy по відношенню до головного вікна. Цей метод призначений для «руйнування» віджету (вікно закриється).

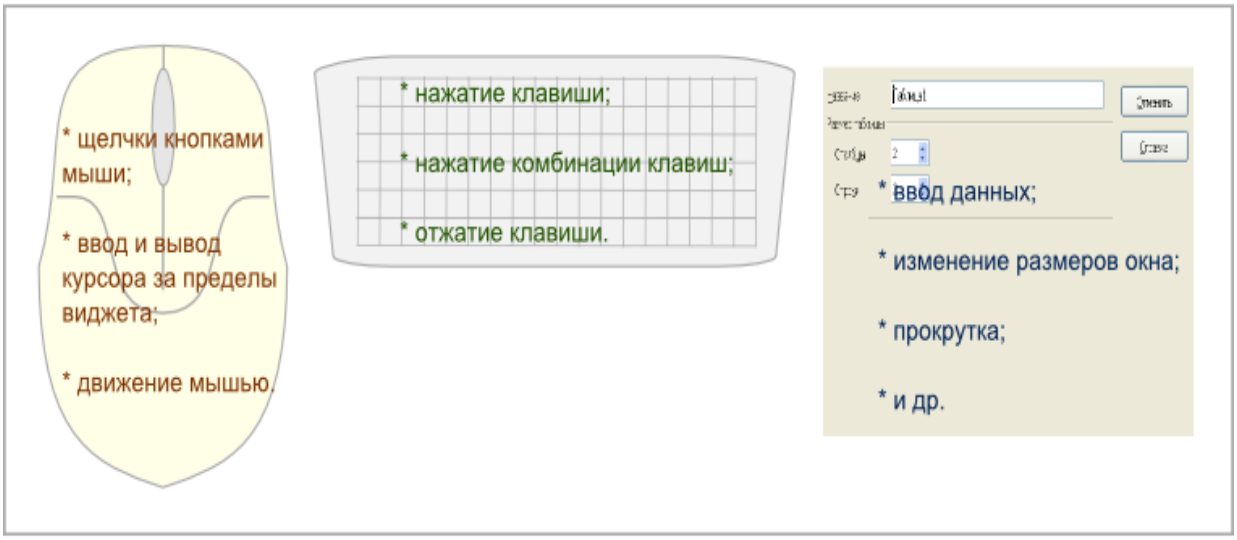
Програмування подій у Tkinter

Зазвичай, щоб графічний додаток щось зробило, має статися якесь подія, т. е. вплив на GUI з-за.

Типи подій

Можна виділити три основні типи подій: миші, натискання клавіш на клавіатурі, а також події, що виникають в результаті зміни інших графічних об'єктів.

Типы событий



Спосіб запису

При виклику методу bind подія передається як перший аргумент.

widget.bind(event,function)

Назва події полягає в лапки, а також знаки < і >. Подія описується за допомогою зарезервованих послідовностей ключових слів.

Події, що виробляються мишею

- **<Button-1>**- клацання лівою кнопкою миші
- **<Button-2>**- клацання середньою кнопкою миші
- **<Button-3>**- клацання правою кнопкою миші
- **<Double-Button-1>**- подвійний клік лівою кнопкою миші
- **<Motion>**- рух миші

і т.д.

Приклад:

```
з tkinter import *
defb1(event):
     root.title("Ліва кнопка миші")
defb3(event):
     root.title("Права кнопка миші")
defmove(event):
     root.title("Рух мишею")
```

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

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

root.mainloop()
```

У цій програмі змінюється напис у заголовку головного вікна залежно від того, що рухається миша, клацають лівою або правою кнопкою миші.

Подія (Event) – це один із об'єктів tkinter. Події мають атрибути, як і багато інших об'єктів. У прикладі функції move витягуються значення атрибутів x і y об'єкта event, в яких зберігаються координати розташування курсору миші в межах віджету, по відношенню до якого була згенерована подія. У разі віджетом є головне вікно, а подією – <Motion>, т. е. переміщення миші.

У програмі нижче виводиться інформація про примірник Event та деякі його властивості. Усі атрибути можна переглянути за допомогою команди dir(event). У різних подій вони ті самі, змінюються лише значення. Для тих чи інших подій частина атрибутів немає сенсу, такі властивості мають значення за умовчанням.

У прикладі хоча обробляється подія натискання клавіші клавіатури, поля x, y, x_root, y_root зберігаються координати положення на екрані курсора миші.

```
fromtkinter import *


defevent_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()
```

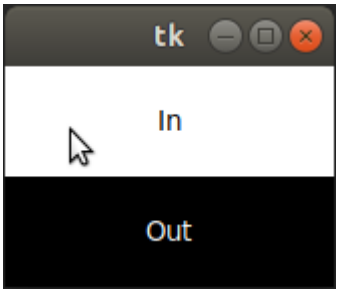
Приклад виконання програми:

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

Для подій з клавіатури можна записати буквені клавіші без кутових дужок (наприклад, 'a').

Для неалфавітних кнопок існують спеціальні зарезервовані слова. Наприклад, <Return> - натискання клавіші Enter, <space> - пробіл. (Зауважимо, що є подія <Enter>, яка не має відношення до натискання клавіші Enter, а відбувається, коли курсор заходить у межі віджету.)

Розглянемо програму:




```
fromtkinter import *


defenter_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()
```

У ній дві мітки використовують одну й ту саму функцію, і кожна мітка використовує цю функцію для обробки двох різних подій: введення курсору в межі віджету та виведення у межі.

Функція, залежно від того, стосовно якого віджету було зафіксовано подію, змінює властивості тільки цього віджету. Як змінює, залежить від події, що сталася.

Властивість event.widget містить посилання на віджет, що згенерував подію.

Властивість event.type описує, що це була подія.

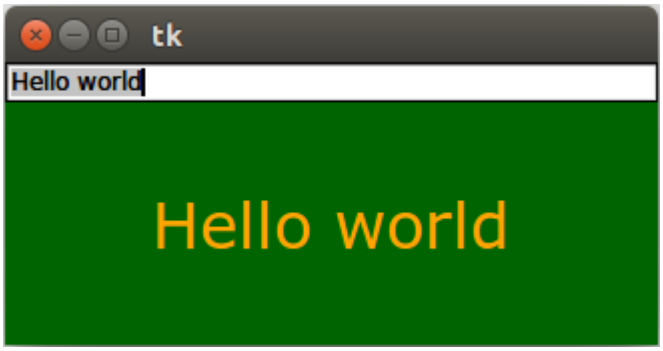
Кожна подія має ім'я та номер. За допомогою виразу print(repr(event.type)) можна переглянути його повний опис.

При цьому на одних платформах str(event.type) повертає ім'я події (наприклад, Enter), на інших – рядкове представлення номера події (наприклад, 7).

Повернімося до подій клавіатури. Поєднання клавіш пишуться через тире. У разі використання так званого модифікатора він вказується першим, деталі на третьому місці. Наприклад, <Shift-Up> - одночасне натискання клавіш Shift та стрілки вгору, <Control-B1-Motion> – рух мишею із затиснутою лівою кнопкою та клавішею Ctrl.

```
fromtkinter import *


defexit_win(event):
    root.destroy()


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


defselect_all(event):

    def select_all2(widget):
        widget.selection_range(0,END)
        widget.icursor(END)  курсор в кінець

    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()
```



Тут поєднання клавіш Ctrl+a виділяє текст у полі. Без root.after() виділення не працює. Метод after виконує функцію, вказану у другому аргументі, через проміжок часу, вказаний у першому аргументі. У третьому аргументі передається значення атрибута widget об'єкта event. У цьому випадку їм буде поле ent. Саме воно буде передано як аргумент у функцію select_all2 та присвоєно параметру widget.

Події, що виробляються за допомогою клавіатури

- Літерні клавіші можна записувати без кутових дужок (наприклад, 'L').
- Для неалфавітних клавіш існують спеціальні зарезервовані слова * <Return> - натискання клавіші Enter; * <space> - пробіл; * і т.д.

- Поєднання клавіш пишуться через тире. Наприклад: * <Control-Shift> - одночасне натискання клавіш Ctrl та Shift.

Прикладник:

```
з tkinter import *

defexit_(event):
     root.destroy()
defcaption(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 натисканні клавіші Enter в межах текстового рядка (ent) викликається функція caption, яка поміщає символи з текстового рядка (ent) до мітки (lbl). Натискання комбінації клавіш Ctrl+z призводить до закриття головного вікна.

I. Змінні Tkinter.

Бібліотека Tkinter містить спеціальні класи, об'єкти яких виконують роль змінних для зберігання значень про стан різних віджетів. Зміна значення такої змінної веде до зміни властивості віджету, і навпаки: зміна властивості віджета змінює значення асоційованої змінної.

Існує кілька класів Tkinter, призначених для обробки даних різних типів.

1. **StringVar**() - для рядків;
2. **IntVar**() - цілих чисел;
3. **DoubleVar**() - дробових чисел;
4. **BooleanVar**() - для обробки булевих значень (true та false).

приклад 1.

Раніше ми вже використовували змінну-об'єкт типу IntVar() при створенні групи радіокнопок:

```
var=IntVar()
var.set(1)
rad0 =
Radiobutton(root,text="Перша",variable=var,value=0)

rad1 =
Radiobutton(root,text="Друга",variable=var,value=1)

rad2 =
Radiobutton(root,text="Третя",variable=var,value=2)
```

Тут створюється об'єкт класу IntVar та зв'язується зі змінною var. За допомогою методу set встановлюється початкове значення 1. Три радіокнопки відносяться до однієї групи: про це свідчить однакове значення опції (властивості) variable. Variable призначена для зв'язування змінної Tkinter із радіокнопкою. Опція value визначає значення, яке буде передано змінною, якщо ця кнопка буде у стані "увімкнено". Якщо у процесі виконання скрипта значення змінної var буде змінено, це позначиться групі кнопок. Наприклад, це робиться в другому рядку коду: увімкнена кнопка rad1.

Якщо метод set дозволяє встановлювати значення змінних, то метод get, навпаки, дозволяє отримувати (пізнавати) значення подальшого їх використання.

```
defdisplay(event):
     v = var.get()
     if v == 0:
          print ("Увімкнена перша кнопка")
     elif v == 1:
          print ("Включена друга кнопка")
     elif v == 2:
          print ("Увімкнена третя кнопка")

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

При виклику функції display в змінну v “записується“ значення, пов'язане з змінною var. Щоб отримати значення змінної var, використовується метод get (другий рядок коду).

приклад 2.

Дещо складніше справа з прапорцями. Оскільки стани прапорців незалежні один одного, то для кожного має бути введена власна асоційована змінна-об'єкт.

```
з tkinter import *

root = Tk()

var0=StringVar() # значення кожного прапорця ...
var1=StringVar() # ... зберігається у власній змінній
var2=StringVar()
# якщо прапорець встановлений, то асоційовану
змінну ...
# ... (var0, var1 або var2) заноситься значення
onvalue, ...
# ...якщо прапорець знято, то - offvalue.
ch0 = Checkbutton(root,text="Коло",variable=var0,
          onvalue="circle",offvalue="-")
ch1 = Checkbutton(root,text="Квадрат",variable=var1,
          onvalue="square",offvalue="-")
ch2 = Checkbutton(root,text="Трикутник",variable=var2,
          onvalue="triangle",offvalue="-")

lis = Listbox(root,height=3)
defresult(event):
     v0 = var0.get()
     v1 = var1.get()
     v2 = var2.get()
```

```
     l = [v0,v1,v2] # значення змінних заносяться до
списку
     lis.delete(0,2) # попередній вміст видаляється з
Listbox
     for v in l: # вміст списку l послідовно
          lis.insert(END,v) # ...вставляється в Listbox

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

ch0.deselect()# "за замовчуванням" прапорці зняті
ch1.deselect()
ch2.deselect()

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

root.mainloop()
```

приклад 3.

Крім властивості (опції) variable, що зв'язує віджет зі змінною-об'єктом Tkinter (IntVar, StringVar та ін), у багатьох віджет існує опція textvariable, яка визначає текст-вміст або текст-напис віджету. Незважаючи на те, що «текстова властивість» може бути встановлена для віджету і змінена в процесі виконання коду без використання асоційованих змінних, іноді такий спосіб зміни виявляється зручнішим.

```
з 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()
```

Тут вміст одного текстового поля негайно відображається в іншому, т.к. обидва поля прив'язані до однієї і тієї ж змінної v.