Тема 11. Віджети (графічні об'єкти) та їх властивості

Розглянемо частину графічних об'єктів (віджет), які у бібліотеці Tkinter: кнопки, поля для введення, мітки, прапорці, перемикачі і списки. Слід розуміти, що графічний інтерфейс користувача досить стандартний, і тому будь-які подібні бібліотеки-модулі (в тому числі і Tkinter) містять приблизно однакові віджети.

Кожен клас віджет має певні властивості, значення яких можна задавати під час їх створення, а також програмувати їх зміну при дії користувача та в результаті виконання програми.

Кнопки

Об'єкт-кнопка створюється викликом класу Button tkinter. При цьому обов'язковим аргументом є лише батьківський віджет (наприклад вікно верхнього рівня). Інші властивості можуть вказуватися під час створення кнопки або задаватися (змінюватися) пізніше. Синтаксис:

змінна = Button (родить_віджет, [властивість = значення, … ….])

Кнопка має багато властивостей, у прикладі нижче вказані лише деякі з них.

```
з tkinter import *

root = Tk()

but = Button(root,
          text="Це кнопка", #напис на кнопці
          width=30,height=5, #ширина та висота
          bg="white",fg="blue") #колір фону та написи

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

bg та fg – це скорочення від background (фон) та foreground (передній план). Ширина та висота вимірюються в знайоместах (кількість символів).

Мітки

Мітки (або написи) — це досить прості віджети, що містять рядок (або кілька рядків) тексту та службовці, в основному, для інформування користувача.

```
lab = Label(root, text="Це мітка! \n З двох рядків.",
font="Arial 18")
```

Однорядкове текстове поле

Таке поле створюється викликом класу модуля Entry tkinter. Користувач може ввести лише один рядок тексту.

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

bd- Це скорочення від межіширини (ширина кордону).

Елемент Entry є полем для введення тексту. Конструктор Entry приймає такі параметри:

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

Де master – посилання на батьківське вікно, а options – набір наступних параметрів:

- bg: фоновий колір
- bd: товщина кордону
- cursor: курсор покажчика миші при наведенні на текстове поле
- fg: колір тексту
- font: шрифт тексту
- justify: встановлює вирівнювання тексту Значення LEFT вирівнює текст по лівому краю, CENTER - по центру, RIGHT - по правому краю
- relief: визначає тип кордону, за замовчуванням значення FLAT
- selectbackground: фоновий колір виділеного шматка тексту.
- selectforeground: колір виділеного тексту
- show: задає маску для символів, що вводяться.
- state: стан елемента може приймати значення NORMAL (за замовчуванням) і DISABLED
- textvariable: встановлює прив'язку до елемента StringVar
- width: ширина елемента

Визначимо елемент Entry та за натисканням на кнопку виведемо його текст в окреме вікно з повідомленням:

```
fromtkinter import *
fromtkinter import messagebox

defshow_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 виведення повідомлення тут застосовується додатковий модуль messagebox, що містить функцію showinfo(), яка і виводить введений у текстове поле текст. Для отримання введеного тексту використовується комп'ютер StringVar.

Тепер створимо складніший приклад із формою введення:

```
fromtkinter import *
fromtkinter import messagebox



defdisplay_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="Введіть ім'я:")
surname_label =Label(text="Введіть прізвище:")

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()
```

Методи Entry

Елемент Entry має низку методів. Основні з них:

- insert(index, str): вставляє в текстове поле рядок за певним індексом
- get(): повертає введений текстове поле текст
- delete(first, last=None): видаляє символ за індексом first. Якщо вказано параметр last, видалення проводиться до індексу last. Щоб видалити до кінця, в якості другого параметра можна використовувати значення END.

Використовуємо методи у програмі:

```
fromtkinter import *
fromtkinter import messagebox


defclear():
name_entry.delete(0, END)
surname_entry.delete(0, END)


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

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

name_label =Label(text="Введіть ім'я:")
surname_label =Label(text="Введіть прізвище:")

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 початкових даних
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 запуску програми відразу в обидва текстові поля додається текст за замовчуванням:

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

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

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

А так можна вставити текст «наприкінці поля»

Кнопка Clear очищає обидва поля, викликаючи метод delete:

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

Друга кнопка, використовуючи метод get, отримує введений текст:

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

Причому, як видно з прикладу, нам необов'язково звертатися до тексту в Entry через змінні типу StringVar, ми можемо зробити це безпосередньо через метод get

Багаторядкове текстове поле

Text призначений для надання користувачеві можливості введення не одного рядка тексту, а значно більше.

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

Остання властивість (wrap) в залежності від свого значення дозволяє переносити текст, що вводиться користувачем або за символами, або за словами, або взагалі не переносити, доки користувач не натисне Enter.

Радіокнопки (перемикачі)

Об'єкт-радіокнопка ніколи не використовується по одному. Їх використовують групами, при цьому в одній групі може бути включена лише одна кнопка.

```
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)
```

Одна група визначає значення однієї змінної, тобто якщо в прикладі буде обрана радіокнопка rad2, то змінної буде var буде 2. Спочатку також потрібно встановити значення змінної (вираз var.set(1) задає значення змінної var рівне 1).

Прапорці

Об'єкт checkbutton призначений для вибору не взаємовиключних пунктів у вікні (у групі можна активувати один, два або більше прапорців або один). На відміну від радіокнопок, значення кожного прапорця прив'язується до своєї змінної, значення якої визначається опціями onvalue (включено) та offvalue (вимкнено) в описі прапорця.

```
c1 = IntVar()
c2 = IntVar()
che1 = Checkbutton(root,text="Перший прапорець",
          variable=c1,onvalue=1,offvalue=0)
che2 = Checkbutton(root,text="Другий прапорець",
          variable=c2,onvalue=2,offvalue=0)
```

Списки

Виклик класу Listbox створює об'єкт, в якому користувач може вибрати один або кілька пунктів залежно від значення опції selectmode. У прикладі нижче значення SINGLE дозволяє вибирати лише один пункт зі списку.

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

Спочатку список (Listbox) порожній. За допомогою циклу for до нього додаються пункти зі списку (тип даних) r. Додавання відбувається за допомогою спеціального методу класу Listbox – insert. Даний метод приймає два параметри: куди додати та що додати.

Більшість методів різних віджет ми розглянемо під час вивчення даного курсу.

Віджети (графічні об'єкти) та їх властивості

Продовжимо розглядати графічні об'єкти (віджети), які у бібліотеці Tkinter. Це будуть рамка (frame), шкала (scale), смуга прокручування (scrollbar), вікно верхнього рівня (toplevel).

Frame (рамка)

Як з'ясується пізніше, рамки (фрейми) є гарним інструментом для організації інших віджет у групи всередині вікна, а також оформлення.

```
з 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()
```

Цей скрипт створює три кадри різного розміру. Властивість bd (скорочення від boderwidth) визначає відстані від краю рамки до віджетів, що у неї віджетів (якщо вони є).

На фреймах можна розміщувати віджети як на основному вікні (root). Тут текстове поле знаходиться на рамці fra2.

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

Scale (шкала)

Призначення шкали – це надання користувачеві вибору певного значення з певного діапазону. Зовні шкала є горизонтальною або вертикальною смугою з розміткою, по якій користувач може пересувати движок, здійснюючи тим самим вибір значення.

```
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)
```

Властивості:

- `orient`визначає напрямок шкали;
- `length`- Довжина шкали в пікселях;
- `from_` і to - з якого значення шкала починається і яким закінчується (тобто діапазон значень);
- `tickinterval`– інтервал, через який відображаються позначки для шкали;
- `resolution`- Мінімальна довжина відрізка, на яку користувач може пересунути двигун.

Scrollbar (смуга прокручування)

Цей віджет дозволяє прокручувати вміст іншого віджету (наприклад, текстового поля або списку). Прокручування може бути як по горизонталі, так і по вертикалі.

```
з 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()
```

У прикладі спочатку створюється текстове поле (tx), потім смуга прокручування (scr), яка прив'язується за допомогою опції command до поля tx вертикальної осі (yview). Далі поле tx змінюється (конфігурується) за допомогою методу configure: встановлюється значення опції yscrollcommand.

Тут використовується незнайомий нам поки ще метод grid, що є іншим способом розташування віджет на вікні.

Toplevel (вікно верхнього рівня)

За допомогою класу Toplevel створюються дочірні вікна, на яких також можуть розташовуватися віджети. Слід зазначити, що при закритті головного вікна (або батьківського) вікно Toplevel також закривається. З іншого боку, закриття дочірнього вікна не призводить до закриття головного вікна.

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

Метод title визначає заголовок вікна. Метод minsize конфігурує мінімальний розмір вікна (є метод maxsize, що визначає максимальний розмір вікна). Якщо значення аргументів minsize буде таким самим, як у maxsize, то користувач не зможе змінювати розміри вікна.