Лекція 30

Робота із зображеннями
Бібліотека tkinter (віджети)

Основне поняття про GUI

У різноманітті програм, які пишуть програмісти, виділяють додатки із графічним інтерфейсом користувача (GUI).

При створенні таких програм стають важливими не тільки алгоритми обробки даних, але й розробка для користувача програми зручного інтерфейсу, взаємодіючи з яким, він буде визначати поведінку програми.

Сучасний користувач в основному взаємодіє з програмою за допомогою різних кнопок, меню, значків, вводячи інформацію в спеціальні поля, вибираючи певні значення в списках і т. д. Ці "зображення" у певному змісті й формують GUI, в подальшому ми їх будемо називати ***віджетами*** (від англ. widget "штучка").

Послідовність кроків при створенні GUI

Послідовність кроків при створенні графічного додатка має свої особливості. Програма повинна виконувати своє основне призначення, бути зручною для користувача, реагувати на його дії.

Не вдаючись у подробиці розробки, розглянемо, які етапи приблизно потрібно пройти при програмуванні, щоб одержати програму з GUI:

1. Імпорт бібліотеки
2. Створення головного вікна
3. Створення віджетів
4. Установка їх властивостей
5. Визначення подій
6. Визначення обробників подій
7. Розташування віджетів на головному вікні
8. Відображення головного вікна

Імпорт модуля tkinter

Для Python віджети включені в спеціальну бібліотеку tkinter. Якщо її імпортувати в програму, то можна користуватися її компонентами, створюючи графічний інтерфейс.

Як і будь-який модуль, tkinter в Python можна імпортувати двома способами:

1. командою import tkinter
2. командою from tkinter import *.

У версії Python 3 ім'я модуля пишеться з малої букви (tkinter), хоча в більш ранніх версіях використовувалася прописна (Tkinter). Отже, перший рядок програми повинен мати такий вигляд:

```
from tkinter import *
```

Створення головного вікна

У сучасних операційних системах будь-який додаток користувача розташований у вікні, яке можна назвати головним, тому що в ньому розміщуються всі інші віджеты.

Об'єкт вікна верхнього рівня створюється при звертанні до класу **Tk** модуля **tkinter**.

Змінну, пов'язану з об'єктом-вікном, прийнято називати **root** (хоча зрозуміло, що можна назвати як завгодно, але так прийнято).

Другий рядок коду:

```
root = Tk()
```

Створення віджетів

Розглянемо встановлення кнопки.

Кнопка створюється при звертанні до класу Button модуля tkinter.

Об'єкт кнопка зв'язується з якою-небудь змінною.

У класу Button (як і всіх інших класів, за винятком Tk) є обов'язковий параметр — об'єкт, якому кнопка належить (кнопка не може "бути нічийною").

Поки в нас є єдине вікно (root), воно й буде аргументом, переданим у клас при створенні об'єкта-кнопки:

```
but = Button(root)
```

Установка властивостей віджетів

У кнопки багато властивостей:

розмір, колір тла й написи й ін. Ми розглянемо їх пізніше.

Установимо всього одну властивість — текст напису (text):

```
but["text"] = "Друк"
```

Визначення подій і їх обробників

При натисканні на кнопку повинні виконуватись певні дії. Дії (алгоритм), які відбуваються при тій або іншій події, можуть бути досить складним.

Тому часто їх оформляють у вигляді функції, а потім викликають, коли вона знадобиться.

Припустимо, що задача кнопки вивести яке-небудь повідомлення в потік виводу, використовуючи функцію print.

Тоді друк на екран оформимо у вигляді функції printer:

```
def printer(event):
    print ("Як завжди, черговий 'Hello World!'")
```

Параметр `event` – це яка-небудь подія.

Потрібно зв'язати цю подію з обробником (функцією printer). Для зв'язку призначений метод bind. Синтаксис зв'язування події з обробником має такий вигляд:

```
but.bind("<Button-1>",printer)
```

Розміщення віджетів

В будь-якому додатку віджети не розкидані по вікну аби як, а добре організовані.

Нехай потрібно просто кнопку будь-яким чином відобразити у вікні.

Найпростіший спосіб — це використання методу **pack**.

```
but.pack()
```

Без наявності даного коду, відображення віджета у полі вікна не відбудеться, оскільки не визначено місце його розташування

Відображення головного вікна

Ну й нарешті, головне вікно теж не з'явиться, поки не буде викликаний спеціальний метод mainloop:

```
root.mainloop()
```

Даний рядок коду повинен бути завжди наприкінці скрипта!

Код програми має такий вигляд:

Приклад 1.

```
from tkinter import *

def printer(event):
    print ("Як завжди, черговий 'Hello World!'")

root = Tk()

but = Button(root)

but["text"] = "Друк"

but.bind("<Button-1>",printer)

but.pack()

root.mainloop()
```

Об'єктно-орієнтований підхід

Приклад 2.

```
from tkinter import *
class But_print:
    def __init__(self):
        self.but = Button(root)
        self.but["text"] = "Друк"
        self.but.bind("<Button-1>",self.printer)
        self.but.pack()
    def printer(self,event):
        print ("Як завжди, черговий 'Hello World!'")

root = Tk()
obj = But_print()
root.mainloop()
```

Віджети бібліотеки tkinter

Кнопки

Об'єкт-Кнопка створюється викликом класу `Button` модуля `tkinter`.

Обов'язковим аргументом є лише батьківський віджет (наприклад, вікно верхнього рівня).

Інші властивості можуть вказуватися при створенні кнопки або задаватися (змінюватися) пізніше.

Синтаксис:

```
<змінна> = Button (<батьківський віджет>,
[властивість=значення, … …])
```

У кнопки багато властивостей, у прикладі зазначені лише деякі з них.

Приклад 3.

```
from tkinter import *

root = Tk()

but = Button(root,

text="Це кнопка", #напис на кнопці

width=30,height=5, #ширина и висота

bg="white",fg="blue") #колір тла і напису

but.pack()

root.mainloop()
```

Властивості й методи кнопки:

Властивість	Опис
activebackground	Колір тла, коли кнопка перебуває під курсором
activeforeground	Колір переднього плану, коли кнопка перебуває під курсором.
bd	Ширина рамки в пікселях. За замовчуванням дорівнює 2.
bg	Нормальний колір тла.
command	Функція або метод, який буде викликатися при натисканні кнопки.
fg	Нормальний передній план (текст) колір.
font	Шрифт тексту, який будуть використовуватися для друку лейбла кнопки.
height	Висота кнопки в текстових рядках

	(для текстових кнопок) або в пікселях (для зображень).
image	Зображення, яке буде відображатися на кнопці (замість тексту).
justify	Як показати кілька рядків тексту: LEFT- притиснути ліворуч кожний рядок; CENTER- центрувати; RIGHT- притиснути вправо кожний рядок.
padx	Додаткове заповнення ліворуч і праворуч від тексту.
pady	Додаткове заповнення зверху й знизу від тексту.
relief	Рельєф визначає тип границі. Деякі зі змінних SUNKEN, RAISED, GROOVE і RIDGE.
state	Установіть цю опцію в DISABLED, щоб кнопка стала сірою й

	недоступною. Якщо має значення ACTIVE, те стає активною, коли над нею покажчик миші. За замовчуванням NORMAL.
underline	За замовчуванням дорівнює -1, що означає, що символи тексту на кнопці не підкреслені. Якщо додатне, то відповідний текстовий символ буде підкреслений.
width	Ширина кнопки в символах (для відображення тексту) або пікселях (якщо відображення зображення).
Wraplength	Якщо це значення встановлене в додатне число, текстові рядки будуть завернуті, щоб відповідати довжині.

Використані методи для кнопки.

Medthod	Description
config()	Застосовуємо для зміни атрибутів кнопки після створення об’єкта.
flash()	Викликає блискання кнопки кілька разів між активним і неактивним положенням.

Приклад 3.1. Кнопка з флешом

```
from tkinter import *
root = Tk()
but_fl = Button(root,text="Це кнопка")
but_fl.config(bg = 'yellow')
but_fl.pack()
def flash(event):
    but_fl.config(bg = 'red')
    but_fl.flash()
    but_fl.after(500, lambda: but_fl.config(bg =
'lightgrey'))
but_fl.bind("<Button-1>", flash)
root.mainloop()
```

Використання messagebox

Приклад 4.

```
from tkinter import *
from tkinter import messagebox
top = Tk()
top.geometry("200x200")
def helloCallBack():
   msg=messagebox.showinfo( "Python", "Hello World")
B = Button(top, text ="Hello",
command = helloCallBack,
bd=4, bg="blue", width=10,fg="white",
activeforeground="black" )
B.place(x=75,y=75)
top.mainloop()
```

Лейбли

Лейблы (або написи) — це досить прості віджети, що містять рядок (або кілька рядків) тексту і використовуються в основному для інформування користувача.

Синтаксис:

<змінна> = Label ((<батьківський віджет>, <властивість>, ...)

<батьківський віджет> представляє об'єкт, який є предком даного віджета.

<властивість> – існує список найбільш часто використовуваних властивостей для цього віджета.

Ці параметри можуть бути використані як пари ключ-значення, розділені комами.

Властивість	Опис
anchor	Цей параметр визначає, де текст позиціонується, якщо віджет має більше простору, ніж потрібно для тексту. За замовчуванням anchor=CENTER, який центрує текст у доступному просторі.
bg	Нормальний колір тла відображається за лейблом.
bitmap	Установіть цей параметр рівним *bitmap* або *image* об'єкту, і лейбл відображатиме цю графіку.
bd	Розмір границі навколо індикатору. За замовчуванням 2 пікселя.
cursor	Якщо встановити в цей параметр вид курсору (*arrow, dot* і т.д.), курсор миші зміниться, коли буде

Властивість	Опис
	перебувати над лейблом.
font	Якщо ви відображаєте текст у цьому лейблі (використовуючи властивість text або textvariable) указуйте тут шрифт, який буде відображатися.
fg	Якщо ви відображаєте текст або bitmap у цьому лейблі, ця властивість визначає колір тексту.
height	Розмір по вертикалі
image	Для відображення статичного зображення необхідно встановити сюди об'єкт image.
justify	Як показати кілька рядків тексту: LEFT- притиснути ліворуч кожний рядок; CENTER- центрувати;

Властивість	Опис
	RIGHT- притиснути вправо кожний рядок
padx	Більше простору додається ліворуч і праворуч від тексту всередині віджета. Значення за замовчуванням 1.
pady	Додаткове заповнення ліворуч і праворуч від тексту.
relief	Визначає зовнішній вигляд декоративної границі навколо лейбла. Значення за замовчуванням FLAT. FLAT,RAISED,SUNKEN,GROOVE,RIDGE
text	Щоб відобразити один або кілька рядків тексту у віджеті лейбла, установіть цей параметр у рядок, що містить текст. Внутрішні

Властивість	Опис
	символи нового рядка ("\n"), розрив рядка.
width	Ширина етикетки(лейбла) в символах (не в пікселях!). Якщо цей параметр не заданий, мітка буде мати такий розмір, щоб відповідати його змісту.
wraplength	Ви можете обмежити кількість символів у кожному рядку, установивши цей параметр на потрібне число. Значення за замовчуванням 0 означає, що рядки будуть розбиті тільки на нових рядках.

Приклад 5. Приклад використання Label

```
from tkinter import *
root = Tk()
root.title("Приклад лейблів")
root.geometry("500x500")
lab1 = Label(root, text="Це мітка! \n з двох
рядків.", font="Arial 18")
lab1.place(x=150,y=100)
lab2 = Label(root, text="Це мітка з околом 10
пікс",
bd=10, fg = "Yellow", bg = "Green", font="Arial
18", cursor = "dot")
lab2.place(x=100,y=200)

photo=PhotoImage(file="python.gif")
lab3 = Label(root, image = photo,
width="150",height="100")
lab3.place(x=160,y=300)
root.mainloop()
```

Однорядкове текстове поле Entry

Таке поле створюється викликом класу Entry модуля tkinter.

В нього користувач може ввести тільки один рядок тексту.

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

s=ent.get()

bd – це скорочення від borderwidth (ширина границі).

width - задає довжину елемента в знакомісцях.

ent.get() – зчитує введений текст.

Приклад 6. Приклад використання Entry

```
from tkinter import *
top = Tk()
top.geometry("500x500")

lb=Label(text= "Введіть дані",font = ("Arial",20))
lb.place(x=140,y=150)


en =Entry(top,width=20,bd=3, font = ("Arial",20))
en.place(x=90,y=200)
def callback():
    print(en.get())

b = Button(top, text="get", width=10,
command=callback, font = ("Arial",20))
b.place(x=150,y=250)

top.mainloop()
```

Багаторядкове текстове поле

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

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

Остання властивість (wrap) залежно від свого значення дозволяє переносити текст, що вводиться користувачем, або по символах, або по словах, або взагалі не переносити, поки користувач не натисне `Enter`.

Для зчитування текстової інформації з Text використовуємо метод get(), для вставки- insert(), а для видалення - del()

Метод insert

Формат: `insert(index, text)`

Вставка тексту в задану позицію.

Для першого параметра зазвичай вибирають

INSERT або END

Якщо у першому параметрі задати INSERT, то текст, який задано у другому параметрі буде вставлений у поточну позицію курсора.

При задаванні **END** текст вставляють у кінці тексту.

Приклад 7. Застосування методу insert

```
from tkinter import *
root=Tk()

lb=Label(text="Введіть дані",font=("Arial",20))
en =Entry(root,width=20,bd=3,font=("Arial",20))
text1=Text(root,height=3,width=40,font=("Arial",
14), wrap=WORD)
def callback():
    text1.insert(END, en.get())
b = Button(root, text="Додати", width=10,
command=callback, font = ("Arial",20))

lb.pack()
en.pack()
b.pack()
text1.pack()

root.mainloop()
```

Метод delete.

Формат: `delete(start, end=None)`

Видаляє символ (або вбудований об'єкт) з заданої позиції.

Перший аргумент: start

1. **`delete(INSERT)` – видаляємо один символ після курсора**
2. **`delete(INSERT, END)`- видаляємо всі символи від курсора до кінця**
3. **`delete(1.0, 1.3)` - видаляємо три перші символи**
4. **`delete(1.0, 2.0)` – видаляємо перший рядок**

Формат позиції: `x.y`, де **`x`** – це рядок що змінюється від 1, а **`y`** – стовпець (від 0).

Приклад 8. Застосування методу delete

```
from tkinter import *
root=Tk()
lb=Label(text= "Введіть дані",font = ("Arial",20))
text1=Text(root,height=3,width=40,
font=("Arial",14),wrap=WORD)
def callback():
    text1.delete(1.0,END)
b = Button(root, text="Очистити", width=10,
command=callback, font = ("Arial",20))

lb.pack()
text1.pack()
b.pack()

root.mainloop()
```

Метод get.

Формат: get(start, stop)

Зчитує текст, починаючи з позиції `start` і до позиції `stop`.

Варіанти виконання зчитування

1. get(1.0, 1.5) – зчитуємо 5 перших симоволів
2. get(1.0, 2.0) – зчитуємо перший рядок
3. get(1.0, END) – зчитуємо всі симоволи
4. get (INSERT,END) зчитуємо від курсора до кінця тексту

Приклад 9. Застосування методу get

```
from tkinter import *
root=Tk()
start=1.0
stop=2.0
lb=Label(text= "Зчитані дані",font =
("Arial",20))
text1=Text(root,height=3,width=40,
font=("Arial",14),wrap=WORD)
def callback():
    lb.config(text = text1.get(start,stop))
b = Button(root, text="Зчитати", width=10,
command=callback, font = ("Arial",20))

lb.pack()
text1.pack()
b.pack()
root.mainloop()
```

Віджет Listbox

Listbox – це віджет, який являє собою список, з елементів якого користувач може вибирати один або кілька пунктів. Має додаткову властивість selectmode,

selectmode = SINGLE, дозволяє користувачеві вибрати тільки один елемент списку

selectmode = EXTENDED, дозволяє користувачеві вибрати будь-яку кількість елементів списку

`insert(index, *elements)`- вставка елементів

`delete(start,stop)` – видалення елементів

`get(first, last=None)`- зчитування елементів

Приклад 10.

```
from tkinter import *
root=Tk()
root.title("listbox")
en =Entry(root,width=20,bd=3, font = ("Arial",20))
lb1=Label(text= "EXTENDED",font = ("Arial",14))
listbox1=Listbox(root,height=5,width=15,
font=("Arial",16),selectmode=EXTENDED)
lb2=Label(text= "SINGLE",font = ("Arial",14))
listbox2=Listbox(root,height=5,width=15,
font=("Arial",16),selectmode=SINGLE)
list1=["Київ","Хмельницький","Львів","Одеса"]
list2=["Канберра","Сідней","Мельбурн","Аделаїда"]
for i in list1:
    listbox1.insert(END,i)
for i in list2:
    listbox2.insert(END,i)
def callback():
    listbox1.insert(ACTIVE,en.get())
```

```
b = Button(root, text="Insert", width=10, command=callback, font =
("Arial",20))
def callback1():
    listbox1.delete(ACTIVE,END)
b1 = Button(root, text="Delete", width=10, command=callback1, font =
("Arial",20))
def callback2():
    lb1.config(text=listbox1.get(ACTIVE))
b2 = Button(root, text="get", width=10, command=callback2, font =
("Arial",20))
en.pack()
b.pack()
lb1.pack()
listbox1.pack()
lb2.pack()
listbox2.pack()
b1.pack()
b2.pack()
root.mainloop()
```

Віджет **Frame**

Віджет Frame (рамка) призначений для організації віджетів усередині вікна.

Приклад 11.

```
# encoding: utf-8
from tkinter import *
root=Tk()
frame1=Frame(root,bg='green',bd=5)
frame2=Frame(root,bg='red',bd=5)
button1=Button(frame1,text='Перша кнопка')
button2=Button(frame2,text='Друга кнопка')
frame1.pack()
frame2.pack()
button1.pack()
button2.pack()
root.mainloop()
```

Властивість bd відповідає за товщину краю рамки.

Віджет **Checkbutton**

Checkbutton – дозволяє відзначити „галочкою“ пункт у вікні. При використанні декількох пунктів потрібно кожному присвоїти свою змінну.

IntVar() – спеціальний клас бібліотеки для роботи із цілими числами.

StringVar() – спеціальний клас бібліотеки для роботи із рядкамми.

variable - властивість, відповідальна за прикріплення до віджету змінної.
onvalue, offvalue – властивості, які присвоюють прикріпленій до віджету змінній значення, яке залежить від стану (onvalue – при вибраному пункті, offvalue – при невибраному пункті).

Приклад 12.

```
from tkinter import *
root=Tk()
var1=IntVar()
var2=StringVar()

def fch1():
    print(var1.get())
def fch2():
    print(var2.get())
check1=Checkbutton(root,text='1 пункт',
variable=var1, onvalue=1, offvalue=0, command =
fch1)
check2=Checkbutton(root,text='2 пункт',
variable=var2, onvalue="Yes", offvalue="No",
command = fch2)
check1.pack()
check2.pack()
root.mainloop()
```

Віджет Radiobutton

Віджет Radiobutton виконує функцію, схожу з функцією віджета Checkbutton. Різниця в тому, що у віджеті Radiobutton користувач може вибрати лише один з пунктів. Реалізація цього віджета дещо інша, ніж віджета Checkbutton:

variable - властивість, яка закріплює змінну до Radiobutton.

value – властивість, яка надає значення прикріпленій до віджету змінній.

anchor=W – встановлює положення якоря «захід - West»

Дані встановлюємо та зчитуємо через методи змінної `get()` та `set()`.

Приклад 13.

```
from tkinter import *
root=Tk()
var=IntVar()
vars=StringVar()
vars.set("one")
var.set(1)
rbutton1=Radiobutton(root,text='1',variable=var,value=1)
rbutton2=Radiobutton(root,text='2',variable=var,value=2)
rbutton3=Radiobutton(root,text='3',variable=var,value=3)
rbutton1s=Radiobutton(root,text='one',variable=vars,value="one")
rbutton2s=Radiobutton(root,text='two',variable=vars,value="two")
rbutton3s=Radiobutton(root,text='three',variable=vars,value="three")
rbutton1.pack(anchor=W)
rbutton2.pack(anchor=W)
rbutton3.pack(anchor=W)
rbutton1s.pack(anchor=W)
rbutton2s.pack(anchor=W)
rbutton3s.pack(anchor=W)
```

Приклад 13_1.

```
from tkinter import *
root = Tk()
v = IntVar()
v.set(1)
languages = [("Python",1), ("Perl",2),
("Java",3), ("C++",4), ("C",5)]
lb = Label(root, text="""Choose your favourite
programming language:""",
justify = LEFT, padx = 20)
lb.pack()
def ShowChoice():
    print(v.get())
for txt, val in languages:
    rbl=Radiobutton(root, text=txt, padx = 20,
variable=v, command=ShowChoice, value=val)
    rbl.pack(anchor=W)
mainloop()
```

Віджет Scale

Scale (шкала) – це віджет, що дозволяє вибрати будь-яке значення із заданого діапазону.

Властивості:

- **`orient`** – як розташована шкала у вікні. Можливі значення: HORIZONTAL, VERTICAL (горизонтально, вертикально).
- **`length`** – довжина шкали.
- **`from_`** – з якого значення починається шкала.
- **`to`** – яким значенням закінчується шкала.
- **`tickinterval`** – інтервал, через який відображаються мітки шкали.
- **`resolution`** – крок пересування (мінімальна довжина, на яку можна пересунути движок)

Приклад 14.

```
from tkinter import *
root = Tk()
def getV(root):
    a = scale1.get()
    print("Значення", a)
scale1 = Scale(root,orient=HORIZONTAL,length=300, from_=50,
to=80,tickinterval=5,resolution=5)
button1 = Button(root,text="Отримати значення")
scale1.pack()
button1.pack()
button1.bind("<Button-1>",getV)
root.mainloop()
```

Тут використовується спеціальний метод get(), який дозволяє зняти з виджета певне значення, і використовується не тільки в Scale.

Scrollbar

Цей виджет дає можливість користувачеві "прокрутити" інший віджет (наприклад, текстове поле) і часто буває корисним. Використання цього віджета досить нетривіально. Необхідно зробити дві прив'язки: command смуги прокручування прив'язуємо до методу xview/yview віджета, а xscrollcommand/yscrollcommand віджета прив'язуємо до методу set смуги прокручування.

Приклад 15.

```
from tkinter import *
root = Tk()
text = Text(root, height=3, width=60)
text.pack(side='left')
scrollbar = Scrollbar(root)
scrollbar.pack(side='left')
# перша прив'язка
scrollbar['command'] = text.yview
# друга прив'язка
text['yscrollcommand'] = scrollbar.set
root.mainloop()
```