Лекція 31

Бібліотека tkinter

Пакувальники

Що таке пакувальник ?

Пакувальник або менеджер геометрії, менеджер розташування.

Пакувальник - це спеціальний механізм, який розміщає (упаковує) віджети на вікні.

В `tkinter` є три пакувальники:

`pack, place, grid.`

Зверніть увагу, що в одному віджеті можна використовувати тільки один тип упакування, при змішуванні різних типів упакування програма, швидше за все, не буде працювати.

Розглянемо кожний з них.

Пакувальник pack()

Менеджер геометрії **`pack()`** пакує віджети в рядки або стовпці.

Для керування цим менеджером можна використовувати його режим за замовчуванням, або задавати такі опції геометрії:

`fill`, **`expand`** і **`side`**.

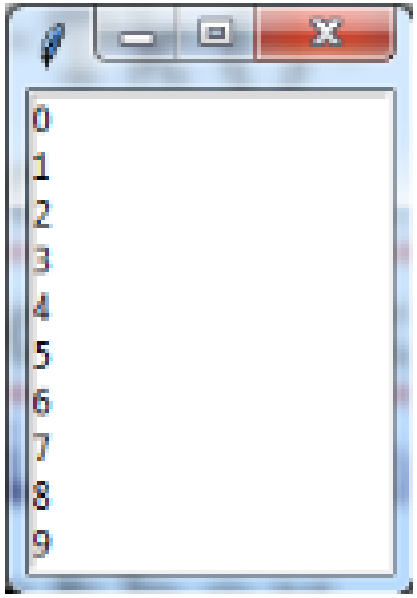
Застосування за замовчуванням

Розміщуємо віджет усередині контейнера-віджета, і він заповнює весь батьківський віджет.

Приклад: `Listbox` у кореневому вікні:

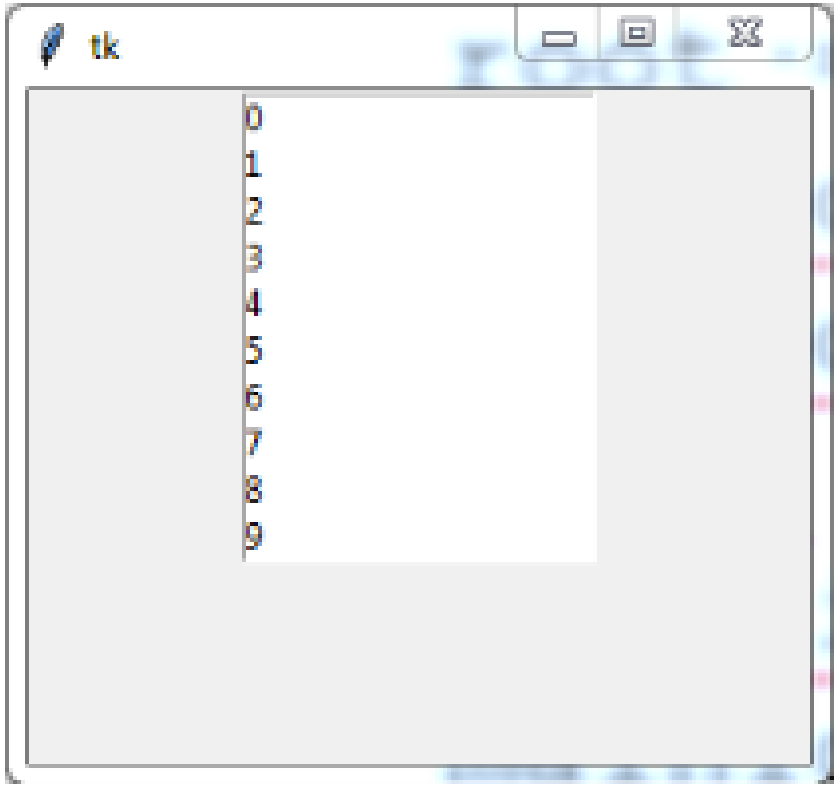
Приклад 1. Повне заповнення батьківського віджета

```
from tkinter import *
root = Tk()
listbox = Listbox(root)
listbox.pack()
for i in range(20):
    listbox.insert(END, str(i))
mainloop()
```



Пакування без розтягування

За замовчуванням `Listbox` має розмір 10 елементів. Однак наш `Listbox` містить 20 елементів. Але якщо спробувати показати їх усі за допомогою зміни розмірів вікна, то `tkinter` додасть відступи навколо `Listbox`:



Приклад 2. Пакування з розтягуванням

```
from tkinter import *
root = Tk()
listbox = Listbox(root)
for i in range(20):
    listbox.insert(END, str(i))
listbox.pack(fill=BOTH, expand=True)
mainloop()
```

Опція **fill** указує менеджерові, що віджет прагне заповнити весь простір, призначений для нього.

Опція має допустимі значення:

1) `fill=BOTH` – віджет розширюється як горизонтально, так і вертикально
2) `fill=X` – віджет розширюється тільки горизонтально
3) `fill=Y` – віджет розширюється тільки вертикально.
4) `expand=True` – опція дозволяє розширювати віджет

Упакування віджетів один над одним

Щоб помістити кілька віджетів у стовпці, можна використовувати метод **`pack()`** без будь-яких опцій:

Приклад 3. Упакування віджетів один над одним.

```
from tkinter import *
root = Tk()
w = Label(root, text="Red", bg="red", fg="white")
w.pack()
w = Label(root, text="Green", bg="green", fg="black")
w.pack()
w = Label(root, text="Blue", bg="blue", fg="white")
w.pack()
mainloop()
```

Упакування віджетів з горизонтальним розширенням

Приклад 4. Розтягування віджетов по ширині батьківського віджета.

```
from tkinter import *
root = Tk()
w = Label(root, text="Red", bg="red", fg="white")
w.pack(fill=X)
w = Label(root, text="Green", bg="green", fg="black")
w.pack(fill=X)
w = Label(root, text="Blue", bg="blue", fg="white")
w.pack(fill=X)
mainloop()
```

Упакування віджетів одного поруч із іншим

Для упакування віджетів поруч, використовуємо опцію **side.**

Приклад 5.

```
from tkinter import *
root = Tk()
w = Label(root, text="Red", bg="red", fg="white")
w.pack(side=LEFT)
w = Label(root, text="Green", bg="green", fg="black")
w.pack(side=LEFT)
w = Label(root, text="Blue", bg="blue", fg="white")
w.pack(side=LEFT)
mainloop()
```

Упакування віджетів з вертикальним розширенням

Якщо необхідно зробити віджети по висоті рівними батьківському, використовуємо додатково опцію `fill=Y`.

Приклад 6.

```
from tkinter import *
root = Tk()
w = Label(root, text="Red", bg="red", fg="white")
w.pack(side=LEFT,fill=Y)

w = Label(root, text="Green", bg="green", fg="black")
w.pack(side=LEFT,fill=Y)

w = Label(root, text="Blue", bg="blue", fg="white")
w.pack(side=LEFT,fill=Y)
mainloop()
```

Пакування з різними опціями у одному вікні

Приклад 7.

```
from tkinter import *
root=Tk()
button1 = Button(text="1", font = ("Arial", 20))
button2 = Button(text="2", font = ("Arial", 20))
button3 = Button(text="3", font = ("Arial", 20))
button4 = Button(text="4", font = ("Arial", 20))
button1.pack(side='top')
button2.pack(side='bottom')
button3.pack(side='left')
button4.pack(side='right')
root.mainloop()
```

Огляд опцій пакувальника

anchor – указує точку відліку для урахування розміщення віджета .
За замовчуванням anchor=CENTER
Можливі значення опції anchor:
1) `anchor=W` – положення «West»
2) `anchor=E` – положення «East»
3) `anchor=N` – положення «North»
4) `anchor=S` – положення «South»

expand (розгортати)
True – віджет заповнює вільний простір

False – (за замовчуванням) віджет не розширюється

Застосування **anchor** показано на прикладі:

Приклад 8.

```
from tkinter import *
root=Tk()
button1 = Button(text="1", font = ("Arial", 20))
button2 = Button(text="2", font = ("Arial", 20))
button3 = Button(text="3", font = ("Arial", 20))
button4 = Button(text="4", font = ("Arial", 20))
button1.pack(anchor=W)
button2.pack(anchor=E)
button3.pack(anchor=N)
button4.pack(anchor=S)
root.mainloop()
```

fill (заповнювати)

Припустимі значення:

X або - 'x' – заповнювати горизонтально,

Y або 'y' – заповнювати вертикально,

BOTH або 'both' – в обох напрямках,

NONE або 'none' – не заповнювати.

ipadx і ipady

Відстань, що вказує, який проміжок повинен залишатися усередині в кожної сторони дочірнього елемента керування.

padx і pady

Відстань, що вказує, який проміжок повинен залишатися ззовні в кожної сторони дочірнього елемента керування.

side (сторона)

Припустимі значення:

LEFT або "left" (ліворуч),

RIGHT або "right" (праворуч),

TOP або "top" (нагорі),

BOTTOM або "bottom" (унизу).

"Внутрішні" елементи керування (дочірні) упаковуються якнайближче до заданої сторони зовнішнього елемента (батька).

Більш докладну інформацію про пакувальника та його опції можна знайти у відповідній документації.

Пакувальник `grid()`

Цей пакувальник є таблицею з комірками, у яких містяться віджети.

Синтаксис: **`widget.grid(<аргументи>)`**

Аргументи

- **`row`** – номер рядка, у який поміщаємо віджет.
- **`rowspan`** – скільки рядків займає віджет
- **`column`** – номер стовпця, у який поміщаємо віджет. За замовчуванням `column = 0` (Самий лівий стовпець)
- **`columnspan`** – скільки стовпців займає віджет.

Приклад 9

```
from tkinter import *
root = Tk()
label1 = Label(root, text="example",font =
("Arial", 20))

button1 = Button(root, text = '1',font =
("Arial", 20))

button2 = Button(root, text = '2',font =
("Arial", 20))

label1.grid(row=0,column=1,columnspan=3)

button1.grid(row=1,column=0)
button2.grid(row=1,column=4)

root.mainloop()
```

- **padx / pady** – розмір зовнішньої границі (бордюру) по горизонталі й вертикалі.

- **ipadx / ipady** – розмір внутрішньої границі (бордюру) по горизонталі й вертикалі. Різниця між **pad** і **ipad** у тому, що при вказівці **pad** розширюється вільний простір, а при **ipad** розширюється віджет, який поміщаємо.

- **sticky** ("**n**", "**s**", "**e**", "**w**" або їх комбінація) – указує, до якої границі "приклеювати" віджет. Дозволяє розширювати віджет у зазначеному напрямку. Границі названі відповідно до сторін горизонту: "**n**" (північ) – верхня границя, "**s**" (південь) – нижня, "**w**" (захід) – ліва, "**e**" (схід) – права.

Приклад 10. Вплив зміни параметра sticky на розміщення кнопки Button

```
from tkinter import *
root = Tk()
label1 = Label(root, text="example",font = ("Arial", 20))
button1 = Button(root, text = '1',font = ("Arial", 20))
button2 = Button(root, text = '2',font = ("Arial", 20))
label1.grid(row=0,column=0,columnspan=3)
button1.grid(row=1,column=2, sticky = "e")
button2.grid(row=1,column=4)
root.mainloop()
```

Додаткові методи

`grid_configure` – синонім для **`grid`**.

`grid_slaves` - повертає список віджетів що належать даному віджетові. Віджети повертаються у вигляді посилань віджета tkinter.

`grid_size` – повертає розмір таблиці в рядках і стовпцях.

`grid_info` - повертає словник, що містить поточні параметри для осередку, використовуваної цим віджетом.

`grid_location` – приймає два аргументи: ***x*** і ***y*** (у пікселях). Повертає номер рядка й стовпця, в які потрапляють зазначені координати, або *-1,* якщо координати потрапили поза віджет.

Приклад 11. Використання додаткових методів

```
from tkinter import *
root = Tk()
label1 = Label(root, text="example",font = ("Arial", 20))
button1 = Button(root, text = '1',font = ("Arial", 20))
button2 = Button(root, text = '2',font = ("Arial", 20))
label1.grid(row=0,column=1,columnspan=3)
button1.grid(row=1,column=0, sticky = "w")
button2.grid(row=1,column=4)
print("Розмір сітки: ",root.grid_size())
print("Сітка містить: ",root.grid_slaves())
print("Параметри комірки: ", label1.grid_info())
print("Комірка: ", root.grid_location(150,100))
root.mainloop()
```

Методи налаштування комірок

Налаштування параметрів для стовпця комірки.

Формат:

```
grid_columnconfigure (індекс,<опції>)
```

індекс - індекс стовпця решітки .

<опції> -

minsize=<число>

Визначає мінімальний розмір стовпця. Якщо стовпець повністю порожній, він не буде відображатися, навіть якщо ця опція встановлена.

weight= =<число>

Відносна вага використовується для розподілу додаткового простору між стовпцями. Стовпець з вагою 2 зростатиме вдвічі швидше, ніж стовпець з вагою 1. Значення за замовчуванням дорівнює 0, що означає, що стовпець не буде рости взагалі.

Налаштування параметрів для рядка комірки.

Формат:

```
grid_rowconfigure (індекс,<опції>)
```

індекс - індекс рядка решітки .

<опції> -

minsize=<число>

Визначає мінімальний розмір рядка. Якщо рядок повністю порожній, він не буде відображатися, навіть якщо ця опція встановлена.

weight= =<число>

Відносна вага використовується для розподілу додаткового простору між рядками. Рядок з вагою 2 зростатиме вдвічі швидше, ніж рядок з вагою 1. Значення за замовчуванням дорівнює 0, що означає, що рядок не буде рости взагалі.

Приклад 12. Текстовий віджет з двома скролбарами

```
from tkinter import *
root=Tk()
text = Text(wrap=NONE)
vscrollbar = Scrollbar(orient='vert', command=text.yview)
text['yscrollcommand'] = vscrollbar.set
hscrollbar = Scrollbar(orient='hor', command=text.xview)
text['xscrollcommand'] = hscrollbar.set
text.grid(row=0, column=0, sticky='nsew')
vscrollbar.grid(row=0, column=1, sticky='ns')
hscrollbar.grid(row=1, column=0, sticky='ew')

root.rowconfigure(0, weight=1)
root.columnconfigure(0, weight=1)
root.mainloop()
```

Приклад 13. Розміщає 12 лейблів у сітці розміром 3x4.

```
import tkinter

root = tkinter.Tk()

for r in range(3):

    for c in range(4):

        lb = tkinter.Label(root, text='R%s/C%s'%(r,c),
borderwidth=10, font = ("Arial", 20))

        lb.grid( row=r, column=c)

root.mainloop(  )
```

Приклад 14. Вікно введення даних про студента

```
from tkinter import *
root = Tk( )
l1=Label(root, text="Група", font = ("Arial", 20))
l1.grid(row=0, sticky=W) # за замовчуванням column=0
l2=Label(root, text="П.І.Б.", font = ("Arial", 20))
l2.grid(row=1,sticky=W) # за замовчуванням column=0
l3=Label(root, text="№", font = ("Arial", 20))
l3.grid(row=2, sticky=W) # за замовчуванням column=0
e1 = Entry(root, font = ("Arial", 20))
e2 = Entry(root, font = ("Arial", 20))
e3 = Entry(root,font = ("Arial", 20))
e1.grid(row=0, column=1)
e2.grid(row=1, column=1)
e3.grid(row=2, column=1)
root.mainloop()
```

Використання опцій злиття стовпців або рядків

Опції columnspan та rowspan

Ці опції дозволяють виділяти під віджети більше, ніж одну комірку.

Опція **columnspan=<число>** опція використовується для виділення під один віджет більше одного стовпця.

Опція **rowspan=<число>** опція використовується для виділення під один віджет більше одного рядка.

Приклад 15.

```
from tkinter import *
root = Tk( )
l1=Label(root, text="Group",font='arial 20')
l1.grid(sticky=E)
l2=Label(root, text="Name",font='arial 20')
l2.grid(sticky=E)
l3=Label(root, text="№",font='arial 20')
l3.grid(sticky=E)
button1=Button(root,text='ok',width=3,bg='red',fg='black',
font='arial 20')
button2=Button(root,text='no',width=3,bg='black',fg='red',
font='arial 20')
p = PhotoImage(file="Tux.gif")
iml = Label(root, image=p)
```

```
e1 = Entry(root,font='arial 20')
e2 = Entry(root,font='arial 20')
e3 = Entry(root,font='arial 20')
e1.grid(row=0, column=1)
e2.grid(row=1, column=1)
e3.grid(row=2, column=1)
cb = Checkbutton(root,text="Life's good",font='arial 20')
cb.grid(columnspan=2, sticky=W)
iml.grid(row=0, column=2, columnspan=2, rowspan=2,
sticky=W+E+N+S, padx=5, pady=5)

button1.grid(row=2, column=2)
button2.grid(row=2, column=3)

root.mainloop(  )
```

Пакувальник `place`

Пакувальник **`place`** є найпростішим із трьох загальних менеджерів геометрії, передбачених в **`tkinter`**.

Він дозволяє точно встановити положення й розмір вікна або в абсолютному значенні, або відносно іншого вікна.

Це, як правило, не дуже гарна ідея – використовувати `Place` для звичайних і діалогових вікон.

Просто прийдеться виконати багато роботи, щоб усе запрацювало як годиться. Рекомендується використовувати `pack` або `grid` для таких цілей.

Синтаксис:

```
widget.place(<place_options>)
```

Основні опції пакувальника `place`

- **`anchor:`** Точка відліку для розміщення віджета встановлюється буквами по сторонах горизонту:
- **`N`** – північ,
- **`E`** – схід,
- **`S`** – південь,
- **`W`** – захід,
- **`N`**e – північний схід,
- **`Nw`** – північний захід,
- **`Se`** – південний захід, або
- **`Sw`** –південний захід.
- **`Nw`** - Значення за замовчуванням: (верхній лівий кут батьківського віджета).

- **bordermode:**

INSIDE (за замовчуванням) для індикації того, що враховують розташування усередині (ігноруючи розташування ззовні);

OUTSIDE – навпаки.

- **height, width:** висота й ширина в пікселях.
- **relheight, relwidth:** висота й ширина виражена дробом від 0.0 до 1.0, як частка від висоти й ширини батьківського віджета.
- **relx, rely:** Горизонтальний і вертикальний зсув як дріб між 0.0 і 1.0, у частках від висоти й ширини батьківського віджета.
- **x, y:** Горизонтальний і вертикальний зсув у пікселах.

Приклади роботи пакувальника `place`

Наступна команда центрує віджет усередині батька.

```
w.place(relx=0.5, rely=0.5, anchor=CENTER)
```

Скористаємося цією властивістю для розміщення кнопки в центрі батьківського об'єкта `root`.

Приклад 16.

```
from tkinter import *
root = Tk()
def helloCallBack():
   print( "Hello Python")
B = Button(root, height=2, width=10, text
="Hello", command = helloCallBack)
B.place( relx=0.5, rely=0.5,anchor=CENTER)
root.mainloop()
```

Поясненя роботи прикладу 16

Віджет міститься на середині довжини й висоти батьківського об'єкта **root**. Точка відліку для віджета **Button** узята на його середині, оскільки anchor=CENTER. При натисканні на кнопку опція command викликає функцію hellocallback(),яка видає у вихідний потік stdout повідомлення **"Hello Python"**

В **root** додали ще один об'єкт **Label**. Тепер потрібно самостійно стежити, щоб зображення цих об'єктів не перетиналися.

Приклад 17.

```
from tkinter import *
root = Tk()
def helloCallBack():
   print( "Hello Python")
B = Button(root, height=2, width=10, text
="Hello", command = helloCallBack)
L = Label(root, text= "Welcome button",
fg="red",)
B.place( relx=0.5, rely=0.5,anchor=CENTER)
L.place( relx=0.5, rely=0.25,  anchor=S)
root.mainloop()
```

Застосуваня пакувльника place при використанні віджета Frame

Розглянемо використання віджета `Frame`, який є контейнером для інших віджетів.

У цьому випадку батьком віджета `Frame` є `root`.

А батьком віджетів `Label` і `Button` є віджет `Frame`.

Усі об'єкти впаковані за допомогою пакувальника (менеджера геометрії) `place`.

Хоча для цих цілей можна використовувати як `pack`, так і `grid`

Приклад 18.

```
from tkinter import *
root = Tk()
pane=Frame(root, height=150, width=150, bg =
"green", relief = RAISED, bd = 10.)
pane.place( relx=0.5, rely=0.5,anchor=CENTER
)
def helloCallBack():
   print( "Hello Python")
B = Button(pane, height=2, width=10, text
="Hello", command = helloCallBack)
L = Label(pane, text= "Welcome button",
fg="red",bg = "yellow")
B.place( relx=0.5, rely=0.5,anchor=CENTER)
L.place( relx=0.5, rely=0.25,  anchor=S)
root.mainloop()
```

message box

Модуль **messagebox** використовується для відображення вікон повідомлень у додатках.

Модуль **messagebox** забезпечує ряд функцій, які можна використовувати, щоб відобразити відповідне повідомлення.

Синтаксис:

```
<змінна> =
messagebox.Functionname(<"заголовок">,
<"повідомлення">, [, опції])
```

Functionname – ім'я відповідної функції `messagebox`.

`"заголовок"` – текст, який відображається в смузі заголовка вікна повідомлення

"повідомлення" – текст, який відображається як повідомлення.

опції – необов'язковий параметр для налаштування конфігурації вікна й виду кнопки.

Функції для діалогового вікна `(Functionname)`:

- `showinfo()`
- `showwarning()`
- `showerror ()`
- `askquestion()`
- `askokcancel()`
- `askyesno ()`
- `askretrycancel`

Для виклику цього модуля не достатньо інструкції

```
from tkinter import *
```

Необхідно додатково записати окремий виклик модуля

```
from tkinter import messagebox
```

Виклик діалогового вікна з пакувальником pack()

Приклад 19.

```
from tkinter import *
from tkinter import messagebox

root = Tk()
def hello():
    messagebox.showinfo("Say Hello", "Hello
World")

B1 = Button(root, text = "Say Hello",
command = hello)
B1.pack()

mainloop()
```

Використання двох кнопок, розміщених на `Frame` і пакувальника `grid`

Приклад 20

```
from tkinter import *
from tkinter import messagebox
root = Tk()
app = Frame(root, bg = "green" , bd=10)
app.grid(row=0, column = 0)
def dialog():
    var = messagebox.showinfo("test" , "This is my first message")
button2 = Button(app, text = "Info" , width=5, command=dialog)
button2.grid(padx=110, pady=80)
button1 = Button(app, text = " exit " , width=3, command=exit)
button1.grid(row=1, column = 2)
mainloop()
```

Меню Menu

Основні функціональні можливості віджета:

- спливаюче вікно (`pop-up`),
- меню верхнього рівня (`toplevel`),
- меню, що випадає (`pull-down`).

Синтаксис.

```
<змінна> = Menu(<батьківське вікно>,
                <опція>, ... )
```

<батьківське вікно> – представляє батьківське вікно. Найбільше часто – це `root`.

<опція> – приведемо список найбільш часто використовуваних опцій для цього віджета. Ці параметри можуть бути використані як пари ключ-значення, розділені комами.

Опції	Опис
`activebackground`	Колір тла, яким буде відображатися вибір, коли він перебуває під мишкою.
`activeborderwidth`	Визначає ширину границі, проведеної навколо вибору, коли він перебуває під мишкою. За замовчуванням 1 піксель.
`activeforeground`	Колір переднього плану, яким буде відображатися вибір, коли він перебуває під мишкою.
`bg`	Колір тла для вибору не під мишкою.
`bd`	Ширина границі навколо всіх варіантів. Значення за замовчуванням 1.
`cursor`	Курсор, який з'являється при

Опції	Опис
	наведенні курсору миші на вибір, але тільки тоді, коли меню виключене.
`disabledforeground`	Колір тексту для елементів у стані `DISABLED`.
`font`	Шрифт за замовчуванням для текстових варіантів.
`fg`	Колір переднього плану використовується для варіантів вибору, що не перебувають під мишею.
`postcommand`	Ви можете встановити цю опцію в процедуру, і ця процедура буде викликатися щоразу, коли хтось відвідує це меню.
`relief`	3 D-Ефект за замовчуванням для

Опції	Опис
	меню при `relief=RAISED`.
`image`	Для виводу зображення на цій кнопці меню.
`selectcolor`	Визначає колір, відображуваний в `checkbutton`, коли вони обрані.
`tearoff`	Меню може бути виключене, перша позиція (позиція 0) у списку вибору займає головний елемент (`tearoff`), а додаткові варіанти будуть додані, починаючи з позиції 1. Якщо ви встановили `tearoff=0`, меню не буде мати функцію `tearoff`, і вибір буде доданий, починаючи з позиції 0.

Опції	Опис
`Title`	Як правило, заголовок вікна меню `tearoff` буде таким же, як текст кнопки меню каскаду, який приводить до цього меню. Якщо ви прагнете змінити назву цього вікна, установіть параметр заголовка на цей рядок.

Методи, що доступні на об'єктах меню:

Методи	Опис
`add_command (options)`	Додає пункт меню в меню.
`add_radiobutton(options)`	Створює елемент меню `radio button`.
`add_checkbutton(options)`	Створює пункт меню `checkbutton`.
`add_cascade(options)`	Створює нове ієрархічне меню, зв'язуючи дане меню з батьківським меню.
`add_separator()`	Додає роздільник у меню.
`add(type, options)`	Додає певний тип пункту меню в меню.
`delete(startindex [,`	Видаляє пункти меню,

Методи	Опис
`endindex])`	починаючи від `startindex` до `endindex`.
`entryconfig(index, options)`	Дозволяє змінити пункт меню, який ідентифікується за допомогою індексу, і змінити його параметри.
`index(item)`	Повертає порядковий номер даного лейбла пункту меню.
`insert_separator (index)`	Включити новий роздільник у позиції, зазначеній індексом.
`invoke (index)`	Викликає команду зворотного виклику,

Методи	Опис
	пов'язаного з вибором позиції індексу.
`type (index)`	Повертає тип вибору зазначеного індексу: `"cascade"`, `"checkbutton"`, `"command"`, `"radiobutton"`, `"separator"` або `"tearoff"`.

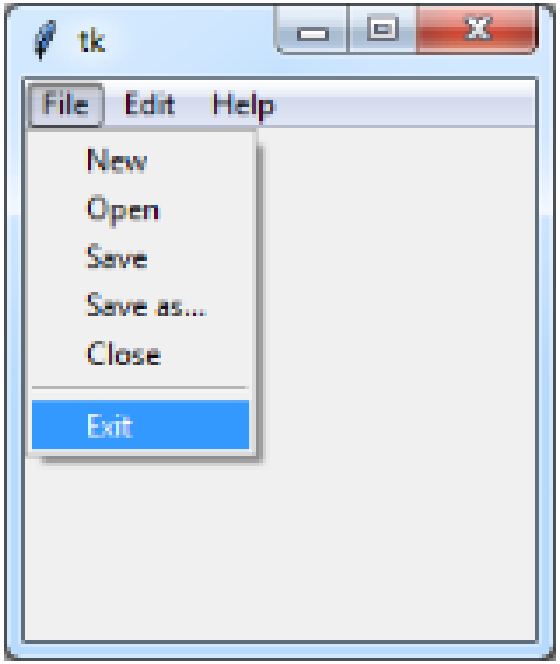
Розглянемо роботу меню на прикладі

Приклад 21

```
from tkinter import *
def donothing():
   filewin = Toplevel(root)
   button = Button(filewin, text="Do nothing button")
   button.pack()

root = Tk()
menubar = Menu(root)
filemenu = Menu(menubar, tearoff=0)
filemenu.add_command(label="New", command=donothing)
filemenu.add_command(label="Open", command=donothing)
filemenu.add_command(label="Save", command=donothing)
filemenu.add_command(label="Save as...", command=donothing)
filemenu.add_command(label="Close", command=donothing)
```

```
filemenu.add_separator()

filemenu.add_command(label="Exit", command=root.quit)
menubar.add_cascade(label="File", menu=filemenu)
editmenu = Menu(menubar, tearoff=0)
editmenu.add_command(label="Undo", command=donothing)

editmenu.add_separator()

editmenu.add_command(label="Cut", command=donothing)
editmenu.add_command(label="Copy", command=donothing)
editmenu.add_command(label="Paste", command=donothing)
editmenu.add_command(label="Delete", command=donothing)
editmenu.add_command(label="Select All", command=donothing)

menubar.add_cascade(label="Edit", menu=editmenu)
```

```
helpmenu = Menu(menubar, tearoff=0)
helpmenu.add_command(label="Help Index",
command=donothing)
helpmenu.add_command(label="About...", command=donothing)
menubar.add_cascade(label="Help", menu=helpmenu)
root.config(menu=menubar)
root.mainloop()
```



Menubutton

Синтаксис

```
<змінна> = Menubutton (<батьківське вікно>, <опція>, ... )
```

<батьківське вікно> – представляє батьківське вікно. Найбільше часто – це `root`.

<опція> – приведемо список найбільше часто використовуваних опцій для цього віджета.

Ці параметри можуть бути використані в якості пар ключ-значення, розділених комами.

Опції	Опис
activebackground	Колір тла при наведенні курсору миші на menubutton.
activeforeground	Колір переднього плану, коли миша перебуває над menubutton.
anchor	Цей параметр визначає, де текст позиціонується, якщо віджет має більше простору, ніж потрібно для тексту. За замовчуванням anchor=CENTER, який центрує текст.
bg	Нормальний колір тла, відображуваного за лейблом і індикатором.
bitmap	Для відображення растрового зображення на кнопку MENU

Опції	Опис
	установіть цей параметр **на ім'я растрового**.
`bd`	Розмір границі навколо індикатору. За замовчуванням 2 пікселя.
`cursor`	Курсор, який з'являється при наведенні курсору миші на `menubutton`.
`direction`	Установите `direction=LEFT`, щоб відобразити меню ліворуч від кнопки; використовуйте `direction=RIGHT`, щоб відобразити меню праворуч від кнопки; або напрямок `direction='above'`, щоб помістити меню над кнопкою.

Опції	Опис
disabledforeground	Колір переднього плану в ситуації, коли menubutton відключений.
fg	Колір переднього плану, коли миша не перебуває над menubutton.
height	Висота menubutton у рядках тексту (не пикселах). За замовчуванням відповідає розміру тексту в menubutton.
highlightcolor	Колір у фокусі підсвічування, коли віджет має фокус.
image	Для виводу зображення на menubutton
justify	Цей параметр визначає, де текст розташований, коли текст не

Опції	Опис
	заповнює `menubutton`: використовувати `ustify=LEFT` для розміщення зліва (за замовчуванням), використовувати `ustify=CENTER` для центрування, або `ustify=RIGHT` на праву виправдати.
`menu`	Щоб зв'язати `menubutton` з набором варіантів, установіть цей параметр в об'єкт `Menu`, що містить ці вибори. Цей об'єкт `Menu` повинен бути створений шляхом передачі зв'язаної `menubutton` конструктору як перший аргумент.

Опції	Опис
padx	Скільки місця залишити ліворуч і праворуч від тексту menubutton. Значення за замовчуванням 1.
pady	Скільки місця залишити вище й нижче тексту menubutton. Значення за замовчуванням 1.
relief	Вибирає тривимірні пограничні ефекти затінення. За замовчуванням піднята.
state	Зазвичай menubuttons реагує на мишу. Установіть state=DISABLED, щоб зробити menubutton сірою. У цьому стані вона не буде відповідати на запити.

Опції	Опис
`text`	Для відображення тексту на `menubutton` установіть в цей параметр рядок, що містить потрібний текст. Знак нового рядка ("\n") усередині рядка буде викликати розриви рядків.
`textvariable`	Можна зв'язати змінну управління класу `Stringvar` із цим `menubutton`. Встановлення цієї змінної керування змінить відображуваний текст.
`underline`	Зазвичай в `menubutton` текст відображається без підкреслення. Щоб підкреслити один із символів, установіть для цього параметра індекс цього

Опції	Опис
	символу.
`width`	Ширина віджета в символах. Значення за замовчуванням 20.
`wraplength`	Як правило, рядки не переносяться. Ви можете встановити цю опцію на ряд (низку) символів, і всі рядки будуть розбиті на куски не довші, ніж це число.

Приклад 4

```
from tkinter import *
top = Tk()
mb=  Menubutton ( top, text="python", relief=RAISED )
mb.grid()
mb.menu  =  Menu ( mb, tearoff = 0 )
```

```
mb["menu"]  =  mb.menu
datVar = IntVar()
oopVar = IntVar()
guiVar = IntVar()
mb.menu.add_checkbutton ( label="data structures",
variable=datVar )
mb.menu.add_checkbutton ( label="OOP",variable=oopVar )
mb.menu.add_checkbutton ( label="GUI", variable=guiVar )
mb.pack()
top.mainloop()
```