Графічний інтерфейс Python GUI (бібліотека Tkinter)

2024 рік

У посібнику наведені матеріали, що допоможуть набути основні навички розробки графічного інтерфейсу в Python з використанням бібліотеки Tkinter. Матеріали розміщені в порядку зростання складності їх використання. Поряд з розглянутими прикладами знаходяться QR-коди для швидкого завантаження фалу програми. Майже всі наведені приклади можна виконати на смартфоні (бажане середовище Pydroid 3).

Посібник адресовано вчителям інформатики закладів загальної середньої освіти, учням 7-8 класів, та всім бажаючим, що розпочинають вивчати використання графічного інтерфейсу.

Посилання для завантажень використаних середовищ:

- https://www.python.org/downloads/
- https://thonny.org/
- https://play.google.com/store/apps/details?id=ru.iiec.pydroid3&hl=uk&gl=US

Даний посібник створив для своїх учнів учитель інформатики Семенівського ліцею № 2 Семенівської селищної ради Кременчуцького району Кузуб Ю. А.

Зміст

Вступ

Tkinter - це стандартна бібліотека для створення графічного інтерфейсу користувача (GUI) в Python. Вона дозволяє розробникам створювати вікна, кнопки, поля введення та інші елементи інтерфейсу для взаємодії з користувачем.

Tkinter:

- має великий набір вбудованих віджетів, таких як кнопки, написи, поля введення, списки, прапорці, перемикачі та багато інших;
- має кілька методів розміщення віджетів відносно один одного, таких як pack(), grid() і place(), які дозволяють легко розташовувати елементи відповідно до потреб дизайну;
- дозволяє встановлювати обробники подій (Events) для відгуку на дії користувача, такі як натискання кнопок миші або клавіш, зміна значень в полі введення тощо.

Tkinter є потужним інструментом для створення простих та середньої складності GUI в Python. Це основний вибір для багатьох початківців та при цьому може бути корисним і для розробки більш складних додатків.

Вікно, напис, кнопка

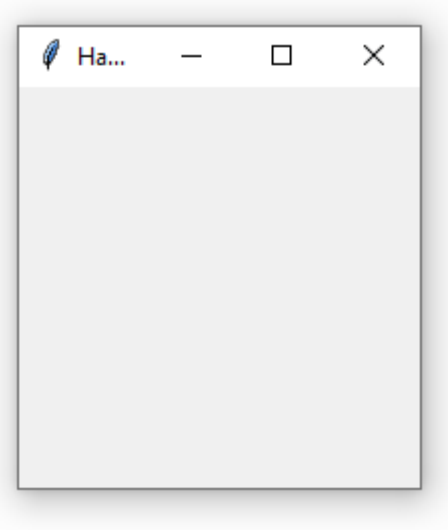
Ми розпочнемо зі створення вікна, в якому дізнаємося, як додавати об’єкти, такі, як написи, кнопки, поля, списки і т. д. Паралельно будемо експериментувати з їхніми властивостями.

Створення першого власного графічного інтерфейсу

Для початку, слід імпортувати Tkinter. Створюємо вікно, в якому вкажемо його назву:

```
from tkinter import *

window = Tk()  # створюється змінна window,  в якій зберігається об’єкт класу Tk()
window.title("Наш перший графічний інтерфейс") # заголовок вікна
window.mainloop()
```

Результат буде виглядати так:



Наш додаток працює. Останній рядок викликає функцію **mainloop()**. Ця функція запускає нескінченний цикл вікна, тому вікно буде чекати будь-якої взаємодії з користувачем, доки не буде закрите.

У разі, якщо ви не будете викликати функцію **mainloop()**, для користувача може нічого не відображатися.

Налаштування розмірів вікна програми

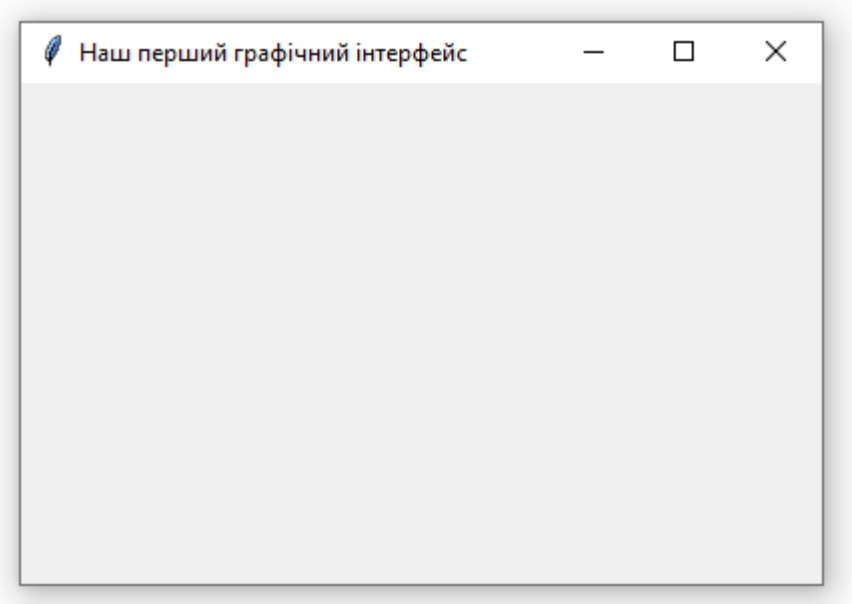
Ми можемо встановити розмір вікна за замовчуванням, використовуючи метод **geometry**:

```
window.geometry('400x250')
```

У наведеному вище рядку встановлюється вікно шириною до 400 пікселів і висотою до 250 пікселів.

```
from tkinter import *

window = Tk()
window.title("Наш перший графічний інтерфейс")
```


```
window.geometry('400x250')
window.mainloop()
```



Об’єкт Label (напис)

Щоб додати текст у наше вікно, ми створимо об’єкт **lbl**, за допомогою класу **Label**, наприклад:

```
lbl = Label(window, text="Привіт!")
```

Потім ми встановимо позицію напису у вікні за допомогою методу **grid**:

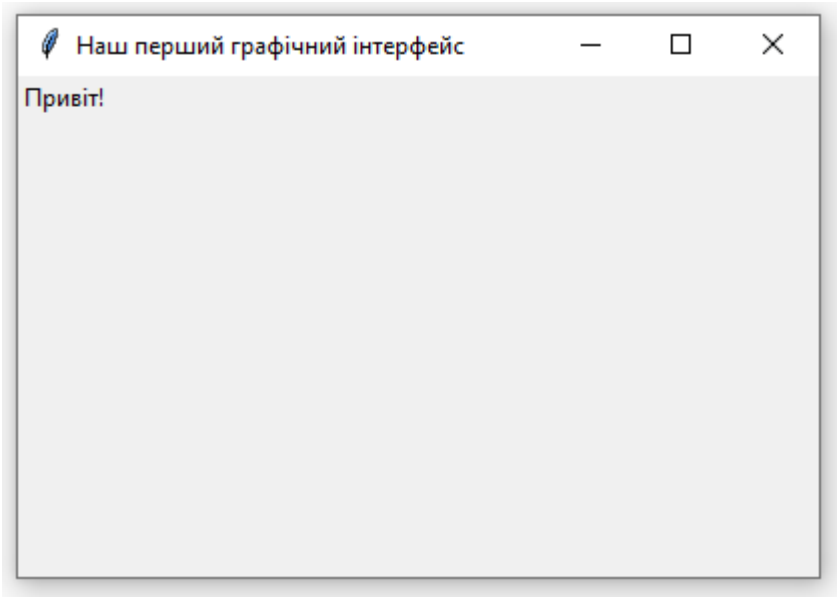
lbl.grid(column=0, row=0), де відповідно задається номер стовпця та рядка таблиці, згідно якої будуть розміщені елементи. Існують інші методи розміщення (**pack, place**), які розглянемо в подальших заняттях.

0,0	0,1	0,2
1,0	1,1	1,2
2,0	2,1	2,2
3,0	3,1	3,2

Повний код, буде виглядати так:

```
from tkinter import *

window = Tk()
window.title("Наш перший графічний інтерфейс")
window.geometry('400x250')
lbl = Label(window, text="Привіт!")
lbl.grid(column=0, row=0)
window.mainloop()
```

І ось як буде виглядати результат:

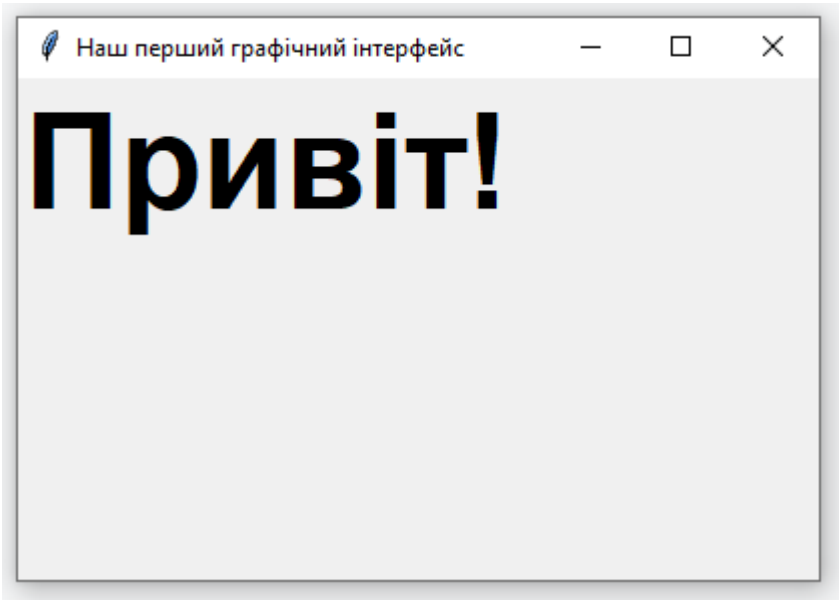


Якщо функція **grid** не буде викликана, текст не буде відображатися.

Налаштування розміру і шрифту тексту

Ми можемо задати шрифт тексту і розмір. Також можна змінити стиль шрифту. Для цього вкажемо параметр **font**:

lbl = Label(window, text="Привіт!", font=("Arial Bold", 50))



Зверніть увагу, що параметр **font** може бути використано і в інших об'єктах..

Спробуємо додати більше об'єктів GUI, наприклад, кнопки і подивитися, як опрацьовується натискання кнопок.

Об'єкт Button (кнопка)

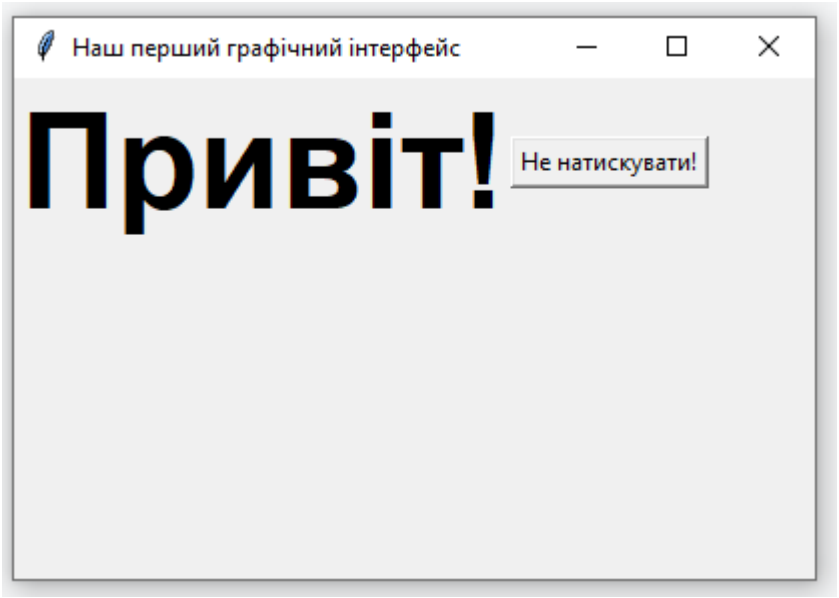
Почнемо з додавання кнопки в вікно. Кнопка створюється і додається у вікно так само, як і напис:

```
btn = Button(window, text="Не натискувати!")
btn.grid(column=1, row=0)
```

Наш код буде виглядати ось так:

```
from tkinter import *

window = Tk()
window.title("Наш перший графічний інтерфейс")
window.geometry('400x250')
lbl = Label(window, text="Привіт!", font=("Arial Bold", 50))
lbl.grid(column=0, row=0)
btn = Button(window, text="Не натискувати!")
btn.grid(column=1, row=0)
window.mainloop()
```

Результат буде таким:

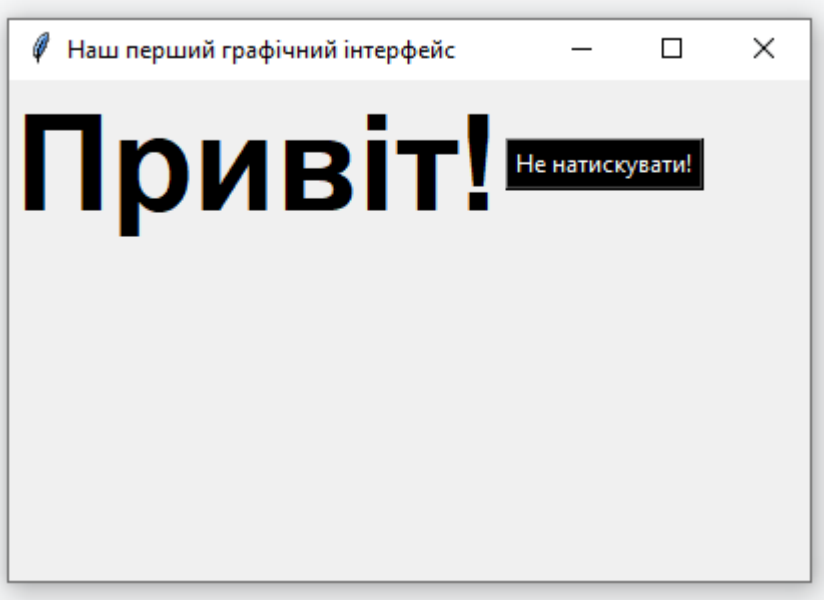


Зверніть увагу, що ми розміщуємо кнопку в другий стовпець вікна, що дорівнює 1. Якщо ви забудете і розмістите кнопку в тому ж стовпці, що дорівнює 0, то побачите кнопку зверху на написові.

Зміна кольору тексту і фону у Button

Ви можете поміняти колір тексту кнопки або будь-якого іншого елемента, використовуючи властивість **fg**. Крім того, ви можете поміняти колір фону будь-якого об’єкта, використовуючи властивість **bg**.

```
btn = Button(window, text="Не натискувати!", bg="black", fg="white")
```



Для зміни кольору вікна, можна скористатися командами:

window["bg"] = "yellow" або

window.config(bg="yellow")

Завдання.

Спробуйте створити власне вікно з написом і кнопкою та застосувати до них параметри кольору фону і шрифту.

Опрацювання події натискання кнопки. Введення даних з вікна (текстове поле).

На цьому занятті ми дізнаємося, як описати та використати функцію, яка опрацьовує подію натискання кнопки та розглянемо введення даних через текстове поле.

Опрацювання події натискання кнопки.

Під час виконання проєкту з об'єктами можуть відбуватися події, наприклад: клацання лівої кнопки миші, подвійне клацання лівої кнопки миші, рух вказівників по вікну, натискання клавіші на клавіатурі, створення вікна, закривання вікна.

Подія	Позначення в мові Python
Click – клацання лівої кнопки миші	**<Button-1>** або **<1>**
DblClick – подвійне клацання лівої кнопки миші	**<Double-Button-1>** або **<Double-1>**
MouseMove – рух вказівника по вікну	**<Motion>** (англ. motion – рух)
KeyPress – натискання будь-якої клавіші на клавіатурі	**<KeyPress>**
Create – створення вікна	**<Expose>** (англ. expose – розкривати)
Close – закривання вікна	**<FocusOut>** (англ. focus out – вихід фокуса)

Після настання події, викликається обробник події, який може бути реалізований у вигляді функції. У мові Python обробник події для вікна створюють як окрему підпрограму і її можна пов'язати з вікном, використовуючи метод **bind()**.

У загальному вигляді команда пов'язування обробника події з вікном, що має ім'я **window**, має такий вигляд:

window.bind('<подія>', <ім'я_обробника_події>)

Обробник події у загальному вигляді записують так:

def <ім'я_обробника_події> (event):

<команди, які будуть виконуватися під час настання події>

Якщо потрібно, щоб у результаті настання події змінилося значення певної властивості вікна, обробник події повинен містити команду змінення значення цієї властивості – команду присвоювання чи відповідний метод.

Для отримання поточного значення розмірів і положення вікна використовують

методи:

winfo_width() – ширина вікна;

winfo_height() – висота вікна;

winfo_x() – відстань лівої межі вікна від лівого краю екрана;

winfo_y() – відстань верхньої межі вікна від верхнього краю екрана.

Ми спробуємо використати подію натискання кнопки і скористаємося іншим способом виклику функції опрацювання події.

Для початку, ми запишемо функцію, яку потрібно виконати при натисканні кнопки:

```
def clicked():
    lbl.configure(text="Що ви робите! Я ж просив...")
```

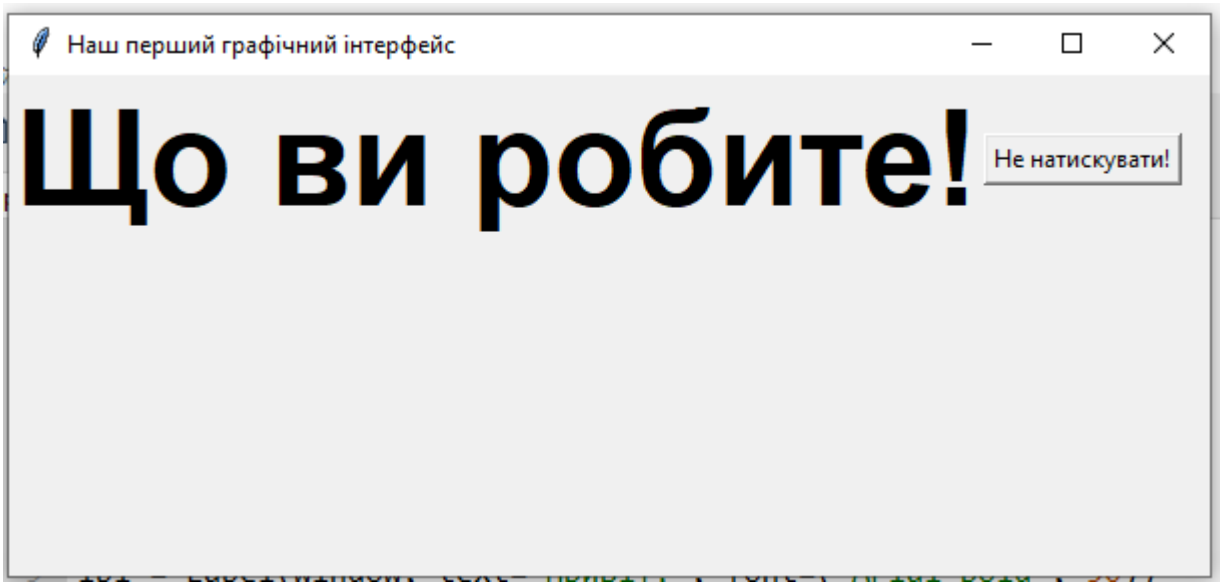
Потім ми підключаємо її за допомогою кнопки, вказавши в параметрі **command** функцію **clicked()**:

```
btn = Button(window, text="Не натискувати!", command=clicked)
```

Наш код буде виглядати ось так:

```
from tkinter import *

def clicked():
    lbl.configure(text="Що ви робите!")

window = Tk()
window.title("Наш перший графічний інтерфейс")
window.geometry('600x250')
lbl = Label(window, text="Привіт!", font=("Arial Bold", 50))
lbl.grid(column=0, row=0)

btn = Button(window, text="Не натискувати!", command=clicked)
btn.grid(column=1, row=0)
window.mainloop()
```

При натисканні на кнопку, результат, як і очікувалося, буде виглядати так:



Отримання введених даних з використанням класу Entry (текстове поле Tkinter)

У попередніх прикладах GUI Python ми ознайомилися зі способами додавання простих елементів, а тепер спробуємо отримати дані введення , використовуючи клас Tkinter Entry (текстове поле Tkinter).

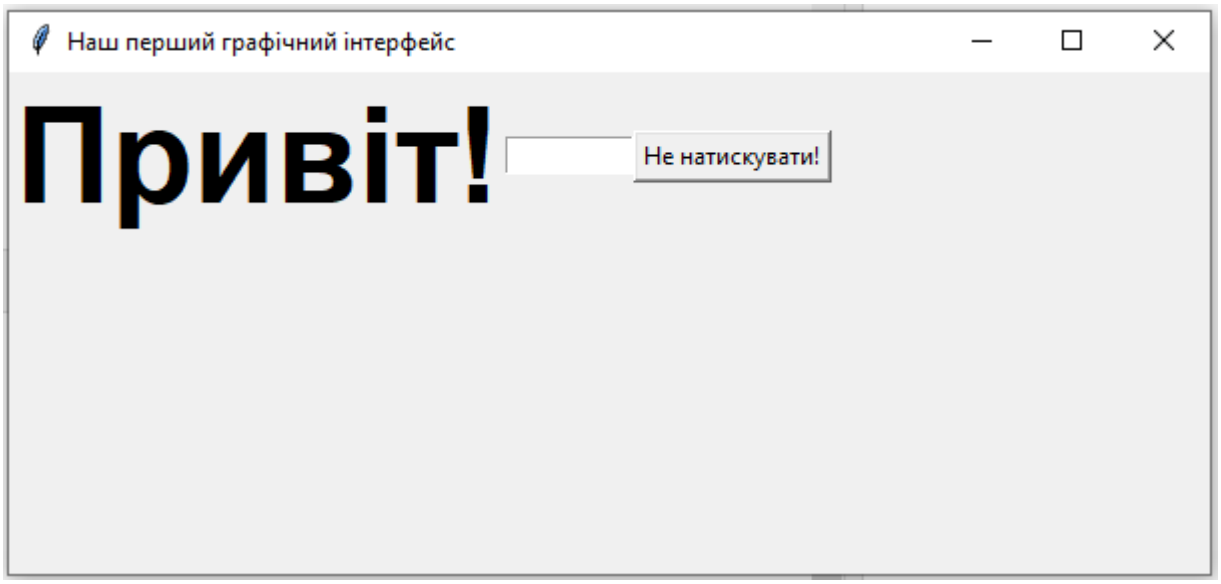
Ви можете створити текстове поле за допомогою класу **Tkinter Entry** таким чином:

txt = Entry(window, width=10) # **width** - ширина поля вводу в символах

Потім ви можете додати його у вікно, використовуючи функцію **grid**.

```
from tkinter import *

def clicked():
    lbl.configure(text="Що ви робите!")

window = Tk()
window.title("Наш перший графічний інтерфейс")
window.geometry('600x250')
lbl = Label(window, text="Привіт!")
lbl.grid(column=0, row=0)

txt = Entry(window,width=10)
txt.grid(column=1, row=0)

btn = Button(window, text="Не натискувати!", command=clicked)
btn.grid(column=2, row=0)
window.mainloop()
```

Отриманий результат буде виглядати так:



Тепер, якщо ви натиснете кнопку, вона покаже те ж саме старе повідомлення, але що ж буде з відображенням введеного тексту з об'єкта **txt** класу **Entry**?

Ви можете отримати введений текст, використовуючи функцію **get**. Ми можемо записати код для обраної функції таким чином:

```
def clicked():
    res = "Привіт {}".format(txt.get())
    lbl.configure(text=res)
```

Якщо ви натиснете на кнопку - з'явиться текст «Привіт» разом з введеним текстом в полі вводу.

Ось повний код:

```
from tkinter import *
def clicked():
    res = "Привіт {}".format(txt.get())
    lbl.configure(text=res)

window = Tk()
window.title("Наш перший графічний інтерфейс")
window.geometry('600x250')
lbl = Label(window, text="Привіт!")
lbl.grid(column=0, row=0)

txt = Entry(window,width=10)
txt.grid(column=1, row=0)
btn = Button(window, text="Натисніть!", command=clicked)
btn.grid(column=2, row=0)
window.mainloop()
```

Запустіть вищевказаний код і перевірте результат:

Кожен раз, коли ми запускаємо код, нам потрібно натиснути на поле для вводу, щоб налаштувати фокус на введення тексту (тобто зробити його вибраним для введення даних), а це не досить зручно. Як же налаштувати автоматичний фокус?

Установка фокусу на поле вводу

Тут все дуже просто, адже все, що нам потрібно зробити, - це викликати функцію **focus**:

```
txt.focus()
```

Коли ви запустите свій код, ви помітите, що текстове поле вже у фокусі (вибране), що дає можливість відразу написати текст.

Як зробити поле вводу неактивним

Щоб зробити поле для вводу неактивним, вимкніть властивість **state**:

```
txt = Entry(window,width=10, state='disabled')
```

Тепер ви не зможете ввести будь-якої текст.

Давайте розглянемо деякі способи редагування змісту поля вводу.

Щоб вивести в поле деякий текст, використовують команду виду:

<ім'я поля>.insert(<позиція>, <текст>),

де позиція – це позиція в полі, починаючи з якої буде вставлено текст. Якщо поле порожнє, то текст вставляється з позиції 0, навіть якщо в команді зазначено іншу позицію. Якщо ж у полі вже є деякий вміст, то текст можна вставити, починаючи з будь-якої позиції в ньому. Для вставлення тексту в кінець існуючого вмісту вказують позицію **END**.

Наприклад, щоб вивести в порожнє поле з іменем **txt** текст "Вивчаємо текстове поле", потрібно виконати команду:

```
txt.insert(0, 'Ми вивчаємо текстове поле')
```

Для заміни вмісту цього поля на **“Ми зараз вивчаємо текстове поле”** можна після попередньої команди додати команду:

txt.insert(2, ‘ зараз’)

А щоб отримати в полі текст **“Ми зараз вивчаємо текстове поле Entry”**, можна після попередніх команд додати команду:

entry.insert(END, ‘ Entry’)

Для очищення вмісту поля з іменем entry використовують команду:

entry.delete(0, END)

А зараз знову повернемося до способів розміщення об’єкті у вікні.

Метод place

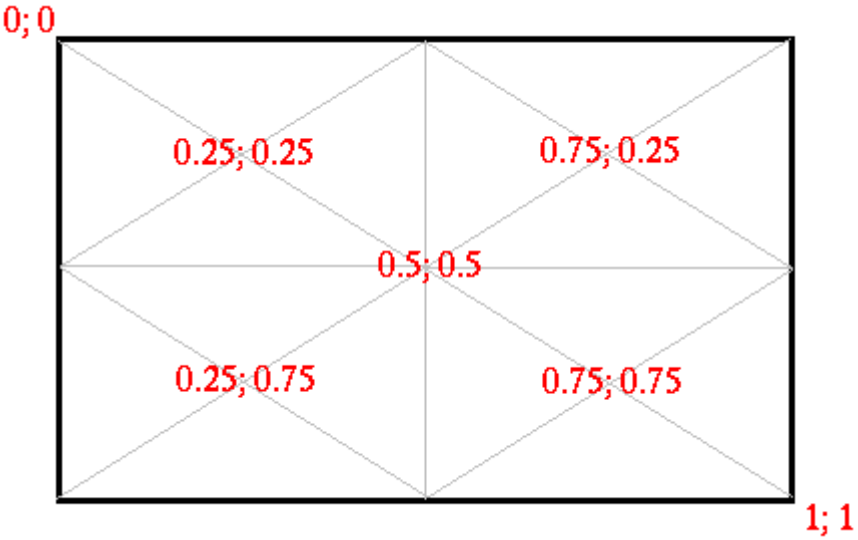
Методом **place** об’єкта вказується його положення, або в абсолютних значеннях (в пікселях), або в частках батьківського вікна, тобто відносно. Також абсолютно і відносно можна задавати розмір самого об’єкта.

Основними параметрами **place** є:

- **anchor** (Якір) - визначає положення елемента, для якого задаються координати. Приймає значення N, NE, E, SE, SW, W, NW або CENTER. За замовчуванням NW (верхній лівий кут).

- **relwidth**, **relheight** (відносні ширина і висота) - визначають розмір елемента в частках його батьківського елемента.

- **relx**, **rely**- визначають відносну позицію в батьківському вікні. Координата (0; 0) - у лівого верхнього кута, (1; 1) - у правого нижнього.



- **width**, **height**- абсолютний розмір елемента в пікселях. Значення за замовчуванням (коли дані опції опущені) прирівнюються до розміру, який визначається при його створенні і конфігурації.

- **x**, **y**- абсолютна позиція в пікселях. Значення за замовчуванням прирівнюються до нуля**.**

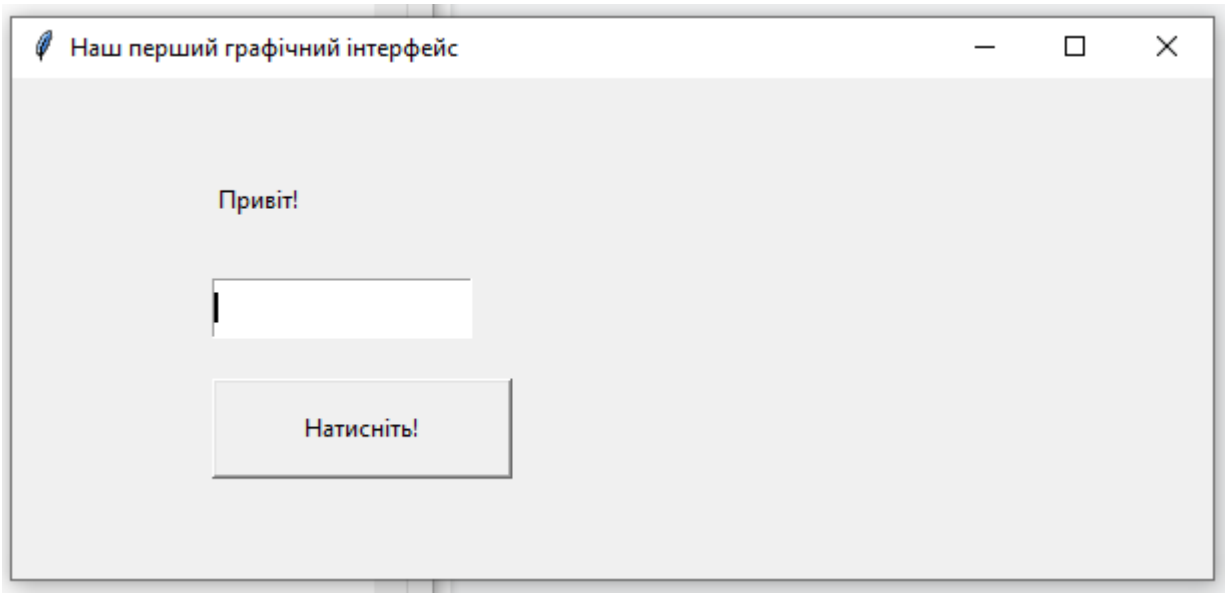
- **bordermode** - задає формат межі елемента. Може приймати значення INSIDE (за замовчуванням) і OUTSIDE.

Давайте розмістимо елементи нашого вікна в стовпчик один під одним на відстані 100 пікселів від лівого краю вікна.

Маємо такий код:

```
from tkinter import *

def clicked():
    res = "Привіт {}".format(txt.get())
    lbl.configure(text=res)

window = Tk()
window.title("Наш перший графічний інтерфейс")
window.geometry('600x250')
lbl = Label(window, text="Привіт!")
lbl.place(x=100, y=50)
txt = Entry(window,width=10)
txt.focus()
txt.place(x=100, y=100, height=30, width=130)
btn = Button(window, text="Натисніть!", command=clicked)
btn.place(x=100, y=150, height=50, width=150, bordermode=OUTSIDE)
window.mainloop()
```

Вигляд вікна:



Завдання.

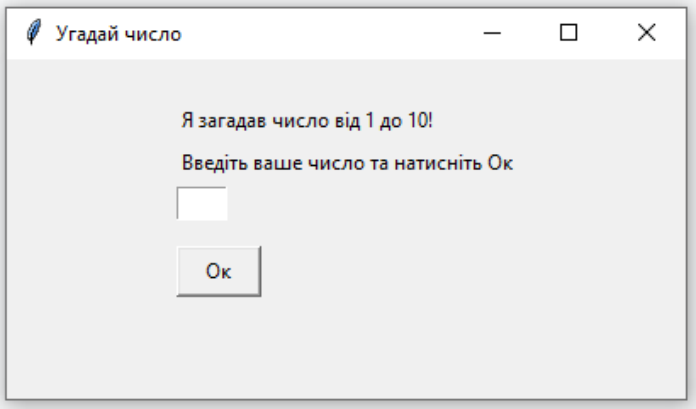
Спробуйте розташувати у вікні два поля для вводу, напис і кнопку. Створити програму яка виведе в напис суму двох цілих чисел введених в поля.

Зіскануйте для перевірки

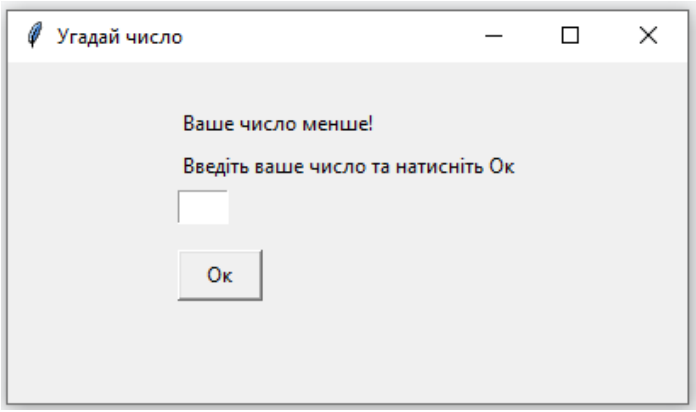
Практична робота. “Створення ігрової програми “Відгадай число”.

Завдання до практичної роботи:

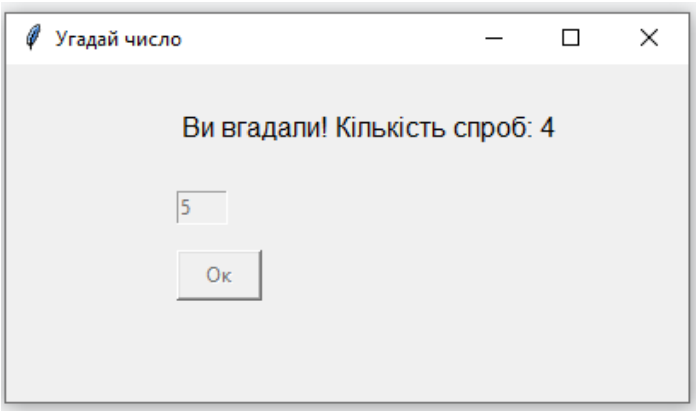
Використовуючи графічну бібліотеку **tkinter,** створити ігрову програму з віконним інтерфейсом, яка реалізує стратегію гри “Вгадай число”. При запуску програми випадковим чином генерується число від 1 до 10 та виводиться інтерфейс поданий на малюнку.



Фокус установлений на поле для вводу числа. Після вводу числа та натиснення клавіші Ок, програма повідомляє чи ваше число більше, чи менше за задумане та очищає поле вводу для наступного, пропонованого вами числа.



У разі правильної відповіді виводиться повідомлення шрифтом Arial 12 про те, що ви вгадали, та кількість спроб, які були використані. При цьому задумане число залишається в полі вводу, а саме поле вводу та кнопка деактивуються. Напис "Введіть ваше число та натисніть Ок" зникає з вікна.



Зіскануйте для перевірки

Додавання елементів Combobox (поле з випадаючим списком), Checkbutton (прапорець), Radiobutton (перемикач).

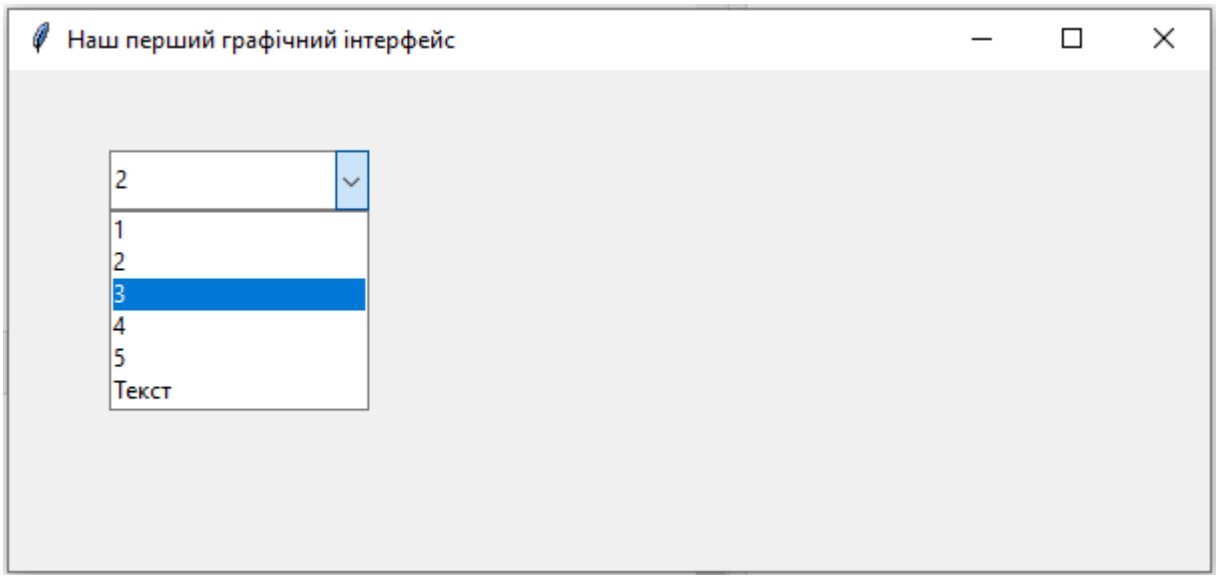
Додавання поля з випадаючим списком Combobox.

Щоб додати поле з випадаючим списком, використовуємо клас Combobox з модуля **ttk**, він містить більш сучасні та стилізовані об'єкти. Таким чином:

```
from tkinter.ttk import Combobox
combo = Combobox(window)
```

Потім додайте свої значення в поле зі списком.

```
from tkinter import *
from tkinter.ttk import Combobox

window = Tk()
window.title("Наш перший графічний інтерфейс")
window.geometry('600x250')
combo = Combobox(window)
combo['values'] = ("Січень", "Лютий", "Березень", "Квітень") #додаємо значення в список
combo.current(0)  #встановлення варіанту за замовчуванням
combo.place(x=50, y=40, height=30, width=130)
window.mainloop()
```



Щоб встановити елемент за замовчуванням, ви можете передати індекс потрібного елемента функції **current(...)**.

Щоб отримати елемент **select**, ви можете використовувати функцію **get** таким чином:
combo.get()

Додавання елемента "Прапорець" Checkbutton

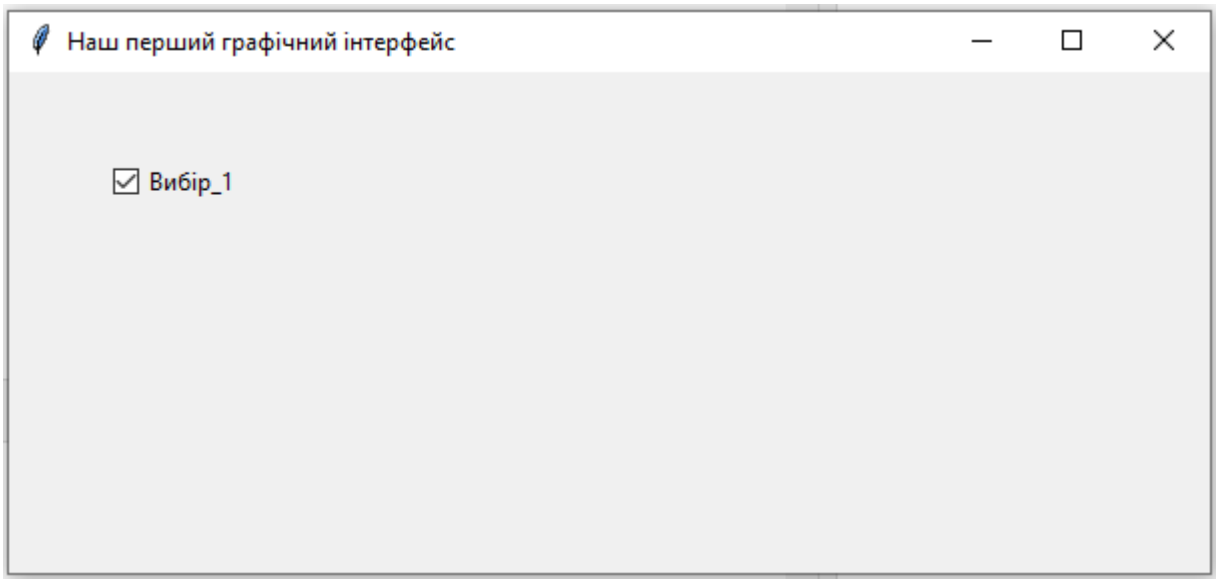
Щоб додати елемент **checkbutton**, використовуємо клас **Checkbutton**:

```
from tkinter.ttk import Checkbutton
chk = Checkbutton(window, text='Вибір_1')
```

Крім того, ви можете задати значення за замовчуванням, передавши його в параметр **var** в **Checkbutton**:

```
from tkinter import *
from tkinter.ttk import Checkbutton

window = Tk()
window.title("Наш перший графічний інтерфейс")
window.geometry('600x250')
chk_state = BooleanVar()
chk_state.set(True)  # задаємо стан прапорця (вибраний, не вибраний)
chk = Checkbutton(window, text='Вибір_1', var=chk_state)
chk.place(x=50, y=40, height=30, width=130)
window.mainloop()
```

Подивимося на результат:



Встановлення стану Checkbutton

Для цього ми створюємо змінну типу **BooleanVar**, яка не є стандартною змінною Python, це змінна Tkinter, потім передаємо її класу **Checkbutton**, щоб встановити стан True у наведеному вище прикладі.

Ви можете встановити для BooleanVar значення false, щоб прапорець був не вибраний.

Так само можна використовувати **IntVar** замість **BooleanVar**, але тоді потрібно встановлювати значення 0 і 1.

```
chk_state = IntVar()
chk_state.set(0) # False
chk_state.set(1) # True
```

Ці приклади працюють так само як і **BooleanVar**.

Додавання елемента RadioButton

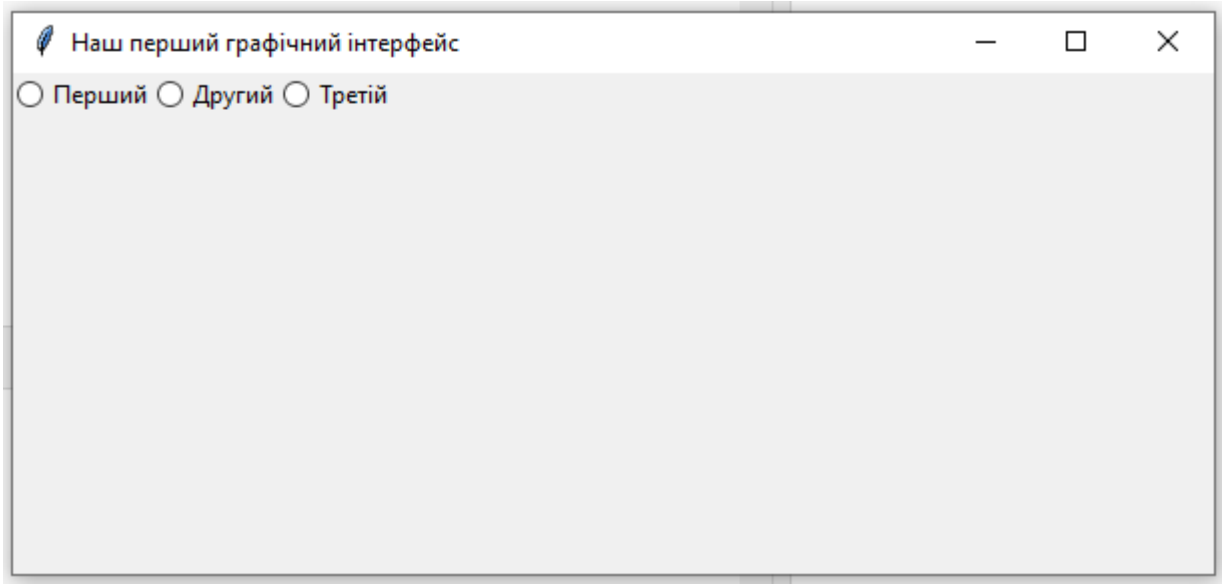
Щоб додати radio кнопки, використовуємо клас **RadioButton**:

rad1 = Radiobutton(window,text='Перша', value=1)

Зверніть увагу, що ви повинні встановити **value** для кожної radio кнопки з унікальним значенням, інакше вони не будуть працювати.

```
from tkinter import *
from tkinter.ttk import Radiobutton

window = Tk()
window.title("Наш перший графічний інтерфейс")
window.geometry('600x250')
rad1 = Radiobutton(window, text='Перший', value=1)
rad2 = Radiobutton(window, text='Другий', value=2)
rad3 = Radiobutton(window, text='Третій', value=3)
rad1.grid(column=0, row=0)
rad2.grid(column=1, row=0)
rad3.grid(column=2, row=0)
window.mainloop()
```

Результат цього коду буде наступний:



Крім того, ви можете задати **command** будь-якій з цих кнопок для виклику певної функції. Якщо користувач натискає на таку кнопку, вона запустить код функції. По суті ми опрацьовуємо подію вибору кнопки.

Наприклад:

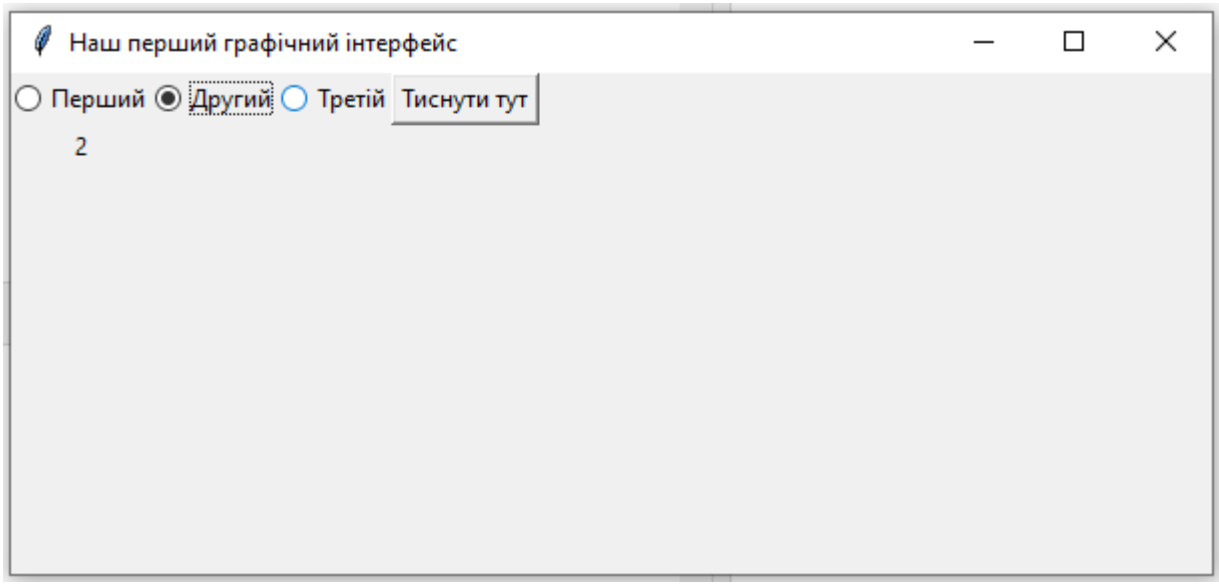
rad1 = Radiobutton(window,text='Перша', value=1, command=clicked)

```
def clicked():
    # Команди функції
```

Отримання значення RadioButton (обрано RadioButton)

Щоб отримати поточну обрану radio кнопку або її значення, ви можете передати параметр змінної і отримати його значення.

```
from tkinter import *
from tkinter.ttk import Radiobutton

def clicked():
    lbl.configure(text=selected.get())

window = Tk()
window.title("Наш перший графічний інтерфейс")
window.geometry('600x250')
selected = IntVar()
rad1 = Radiobutton(window, text='Перший', value=1, variable=selected)
rad2 = Radiobutton(window, text='Другий', value=2, variable=selected)
rad3 = Radiobutton(window, text='Третій', value=3, variable=selected)
btn = Button(window, text="Тиснути тут", command=clicked)
lbl = Label(window)
rad1.grid(column=0, row=0)
rad2.grid(column=1, row=0)
rad3.grid(column=2, row=0)
btn.grid(column=3, row=0)
lbl.grid(column=0, row=1)
window.mainloop()
```



Кожен раз, коли ви вибираєте radiobutton, значення змінної буде змінено на значення обраної кнопки.

Завдання.

Спробуйте розташувати елементи цього вікна по центру, один під одним з допомогою методу **place**.

Додавання елемента ScrolledText (текстова область Tkinter)

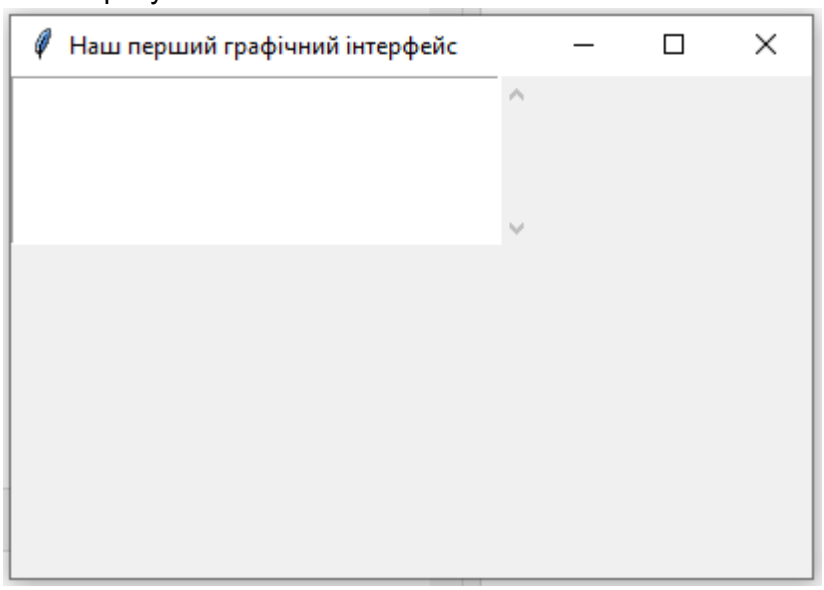
Створення ScrolledText

Щоб додати елемент **ScrolledText**, використовуємо клас **scrolledText**:
from tkinter import scrolledtext
txt = scrolledtext.ScrolledText(window,width=40,height=10)

Потрібно вказувати ширину і висоту **ScrolledText**, інакше він займає все вікно.

```
from tkinter import *
from tkinter import scrolledtext
window = Tk()
window.title("Наш перший графічний інтерфейс")
window.geometry('400x250')
txt = scrolledtext.ScrolledText(window, width=40, height=10)
txt.grid(column=0, row=0)
window.mainloop()
```

Маємо результат:



Налаштування вмісту Scrolledtext

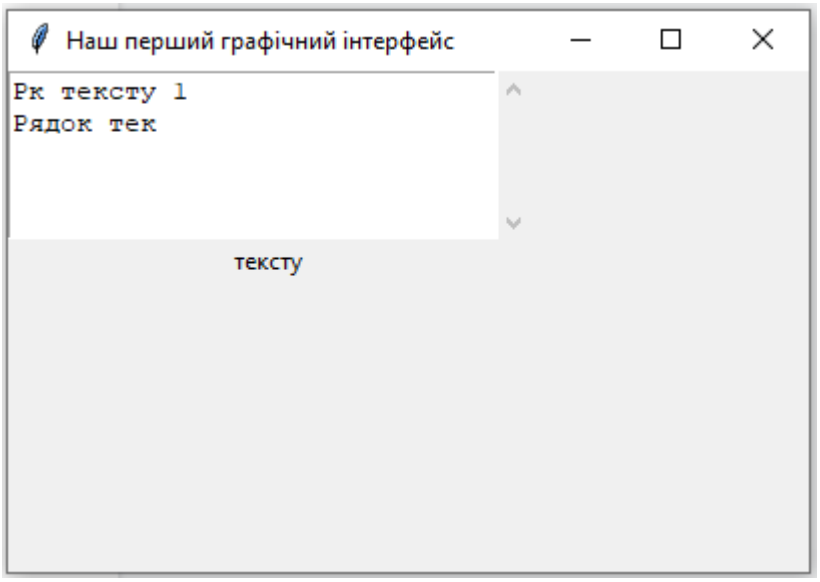
Використовуйте метод **insert**, щоб налаштувати вміст **Scrolledtext**:
txt.insert(INSERT, 'Рядок тексту')

Видалення / Очищення вмісту Scrolledtext

Щоб очистити вміст даного віджету, використовуйте метод **delete**:
txt.delete(1.0, END) # передаємо координати очищення
Наприклад:

```
from tkinter import *
from tkinter import scrolledtext
window = Tk()
```

```
window.title("Наш перший графічний інтерфейс")
window.geometry('400x250')
txt = scrolledtext.ScrolledText(window, width=30, height=5)
txt.grid(column=0, row=0)
txt.insert(INSERT, "Рядок тексту 1\n")   #\n - перехід на новий рядок
txt.insert(INSERT, "Рядок тексту 2\n")
txt.delete(1.1,1.4)   #вилучити текст між 1 рядок 1 символ до 1 рядок 4 символ
txt.delete(2.9, END)   #вилучити текст між 2 рядок 9 символ до кінця тексту
s=txt.get(1.3, 1.9)   #вибір тексту між 3 та 9 символом 1 рядка
lbl = Label(window, text=s)
lbl.grid(column=0, row=1)
window.mainloop()
```



Завдання.

Спробуйте розташувати елементи цього вікна по центру, один під одним з допомогою методу **place**.

Спливаюче вікно з повідомленням

Створення спливаючого вікна

Для створення і виведення спливаючого вікна повідомлень з повідомленням використаємо об’єкт **messagebox** графічної бібліотеки **Tkinter**.

```
from tkinter import messagebox
messagebox.showinfo('Заголовок вікна', 'Текст у вікні')
```

Давайте виведемо вікно повідомлень, при натисканні на кнопку користувачем.

```
from tkinter import *
from tkinter import messagebox
def clicked():
    messagebox.showinfo('Заголовок вікна', 'Текст у вікні')
window = Tk()
window.title("Наш перший графічний інтерфейс")
window.geometry('400x250')
btn = Button(window, text='Натисніть!', command=clicked)
btn.grid(column=0, row=0)
window.mainloop()
```

Коли натиснути на кнопку, з'явиться інформаційне вікно. Пам’ятайте, закрити вікно основної програми можна тільки після закриття спливаючого вікна.

Показ повідомлень про попередження і помилки

В спливаючому вікні можна показати попередження або повідомлення про помилку. Для цього потрібно використати відповідну функцію повідомлення: **showwarning** або **showerror**.

```
messagebox.showwarning('Заголовок', 'Текст')  # виводить вікно попередження
messagebox.showerror('Заголовок', 'Текст')  # виводить вікно з інформацією про помилку
```

Використання діалогових вікон з вибором варіанта

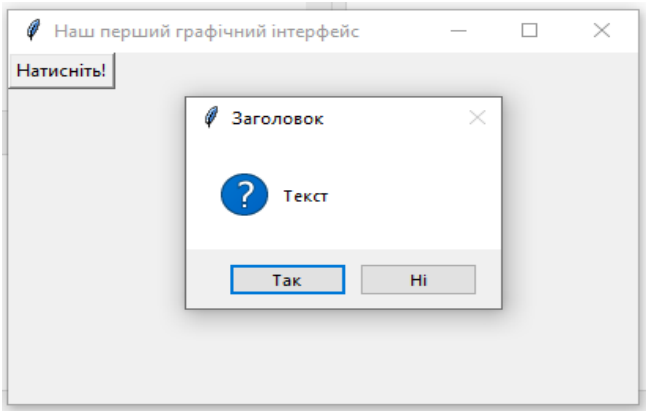
Щоб показати користувачеві повідомлення "Так / Ні", можна використовувати одну з таких функцій **messagebox**:

```
from tkinter import *
from tkinter import messagebox
def clicked():
    res = messagebox.askquestion('Заголовок', 'Текст') # Так/ні
    res = messagebox.askyesno('Заголовок', 'Текст') # Так/ні
    res = messagebox.askyesnocancel('Заголовок', 'Текст')  # Так/ні/скасувати
    res = messagebox.askokcancel('Заголовок', 'Текст')  # Ok/Скасувати
    res = messagebox.askretrycancel('Заголовок', 'Текст') # Повторити/скасувати
window = Tk()
window.title("Наш перший графічний інтерфейс")
window.geometry('400x250')
btn = Button(window, text='Натисніть!', command=clicked)
btn.grid(column=0, row=0)
window.mainloop()
```

Ви можете вибрати відповідний стиль повідомлення відповідно до ваших потреб. Просто залиште у наведеному вище прикладі відповідний рядок і виконайте скрипт. Крім того, можна перевірити, яка кнопка натиснута, використовуючи змінну результату **res**. Якщо ви клацнете OK,

Так(Yes), або Повторити(Rretry), значення стане True, а якщо, буде обрано Ні(No) або Cancel , значення буде False. Винятком є **askquestion**, вона повертає yes/no.
Єдиною функцією, яка повертає одне з трьох значень, є функція **askyesnocancel**; вона повертає True / False / None .
Перевіримо це, модифікувавши попередній код.

```
from tkinter import *
from tkinter import messagebox
def clicked():
    res = messagebox.askquestion('Заголовок', 'Текст') # Так/ні
    lbl.configure(text=res)
    res = messagebox.askyesno('Заголовок', 'Текст') # Так/ні
    lbl.configure(text=str(res))
    res = messagebox.askyesnocancel('Заголовок', 'Текст')  # Так/ні/скасувати
    lbl.configure(text=str(res))
    res = messagebox.askokcancel('Заголовок', 'Текст')  # Ok/Скасувати
    lbl.configure(text=str(res))
    res = messagebox.askretrycancel('Заголовок', 'Текст') # Повторити/скасувати
    lbl.configure(text=str(res))
window = Tk()
window.title("Наш перший графічний інтерфейс")
window.geometry('400x250')
btn = Button(window, text='Натисніть!', command=clicked)
lbl = Label(window)
btn.grid(column=0, row=0)
lbl.grid(column=0, row=1)
window.mainloop()
```



Завдання.

Чому у деяких функціях, які повертають значення обраної кнопки, при присвоєнні напису значення, ми використали **str(res)?**

Використання елементів Spinbox та Progressbar

Використання Spinbox (лічильник)

Лічильник Spinbox призначений для вибору із запропонованого переліку значень. Він має дві кнопки для перегляду списку вгору та вниз, за допомогою яких обирається необхідне значення. У більшості випадків компонент-лічильник використовують для отримання числових даних, але він також може працювати з нечислові значення - рядками

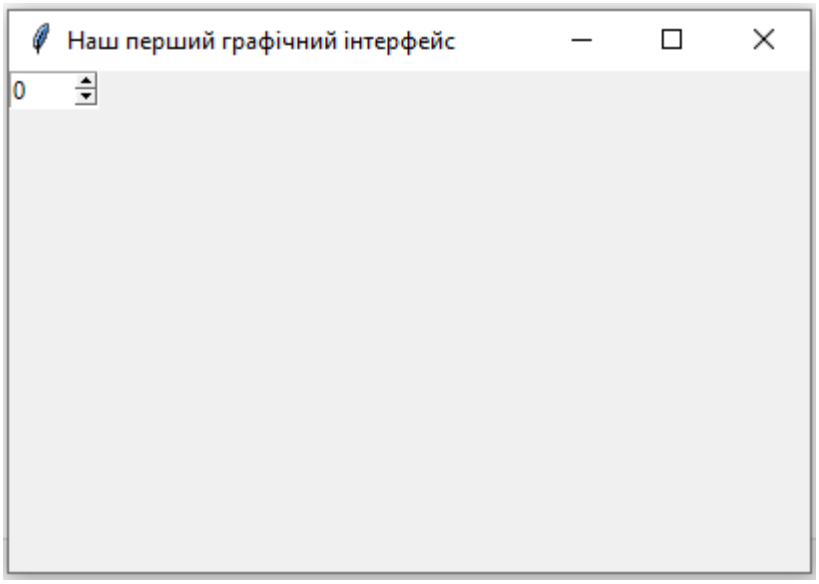
Для створення спінбокса, використовуємо екземпляр класу **Spinbox**:
spin = Spinbox(window, from_=0, to=100)

Таким чином, ми створюємо об'єкт **Spinbox**, і передаємо параметри **from**_ і **to**, щоб вказати діапазон.
Крім того, ми можемо задати ширину стрічки з допомогою параметра **width**:

spin = Spinbox(window, from_=0, to=100, width=5)

Наприклад:

```
from tkinter import *

window = Tk()
window.title("Наш перший графічний інтерфейс")
window.geometry('400x250')
spin = Spinbox(window, from_=0, to=100, width=5)
spin.grid(column=0, row=0)
window.mainloop()
```



Ви можете вказати числа для Spinbox, замість використання всього діапазону таким чином:

spin = Spinbox(window, values=(3, 8, 11), width=5)

В результаті при переборі значень будуть показані тільки ці 3 числа: 3, 8 і 11.

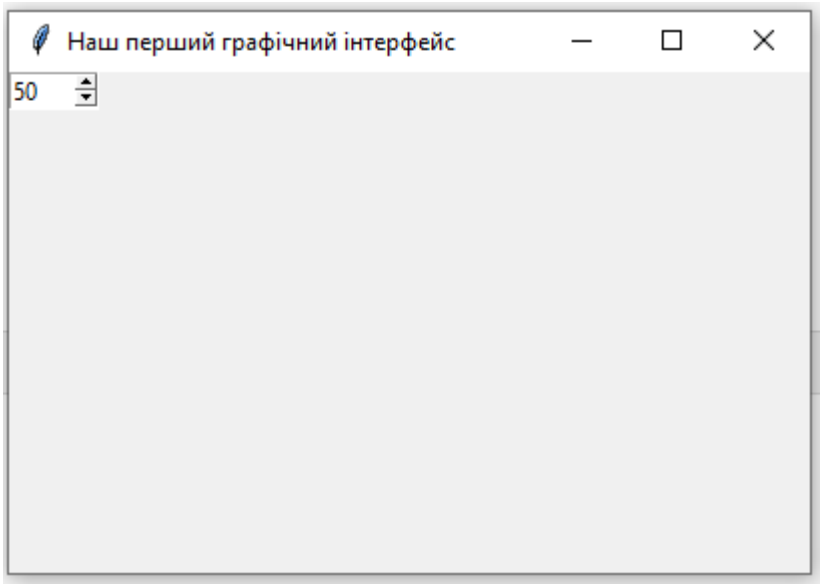
Задати значення за замовчуванням для Spinbox

У разі, якщо вам потрібно задати значення за замовчуванням для Spinbox, ви можете передати значення параметру **textvariable** таким чином:

```
var = IntVar()
var.set(50)
spin = Spinbox(window, from_=0, to=100, width=5, textvariable=var)
```

Тепер, якщо ви запустите програму, вона покаже 50 як значення за замовчуванням для Spinbox**.**

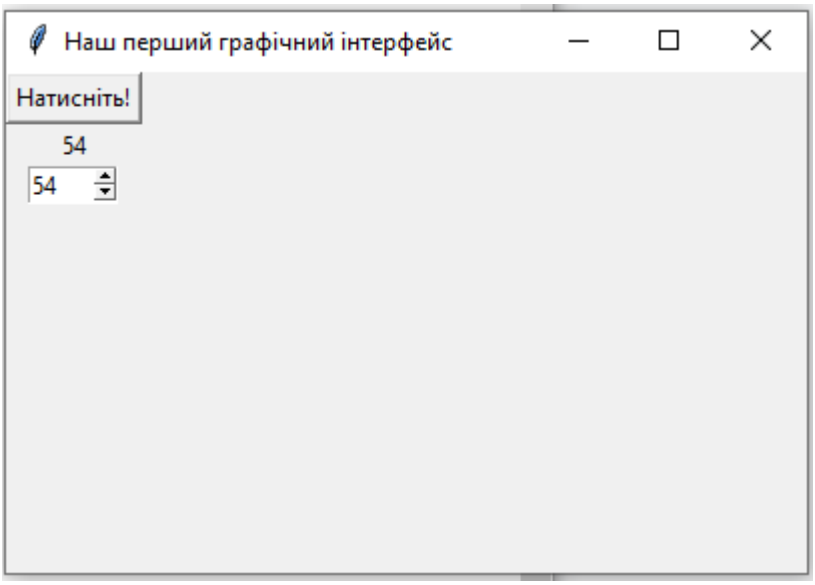
```
from tkinter import *

window = Tk()
window.title("Наш перший графічний інтерфейс")
window.geometry('400x250')
var = IntVar()
var.set(50)
spin = Spinbox(window, from_=0, to=100, width=5, textvariable=var)
spin.grid(column=0, row=0)
window.mainloop()
```



Для отримання значення Spinbox використаємо метод **get().**

```
from tkinter import *
from tkinter import messagebox

def clicked():
    res = spin.get() # отримуємо значення Spinbox
    lbl.configure(text=res)

window = Tk()
window.title("Наш перший графічний інтерфейс")
window.geometry('400x250')
btn = Button(window, text='Натисніть!', command=clicked)
```

```
lbl = Label(window)
btn.grid(column=0, row=0)
lbl.grid(column=0, row=1)
var = IntVar()
var.set(50)
spin = Spinbox(window, from_=0, to=100, width=5, textvariable=var)
spin.grid(column=0, row=3)
window.mainloop()
```



Використання Progressbar (індикатор прогресу)

Індикатор прогресу корисний для відображення стану операції або завдання. Progressbar може вказувати на розвиток певного процесу (наприклад, завантаження файлу з Інтернету) або просто представляти, що виконується операція, у тих випадках, коли час, що залишився, невідомий.

Щоб створити даний об’єкт, використовуйте екземпляр класу **progressbar:**

```
from tkinter.ttk import Progressbar
bar = Progressbar(window, length=200)
```

Методи Progressbar:

start — запускає нескінчений цикл завантаження. Крок довжиною 1 буде виконано у зазначений час (у мілісекундах);

stop — зупиняє цикл завантаження;

step — просуває процес завантаження на задану кількість кроків.

Значення progressbar можна встановити таким чином:

```
bar['value'] = 30
```

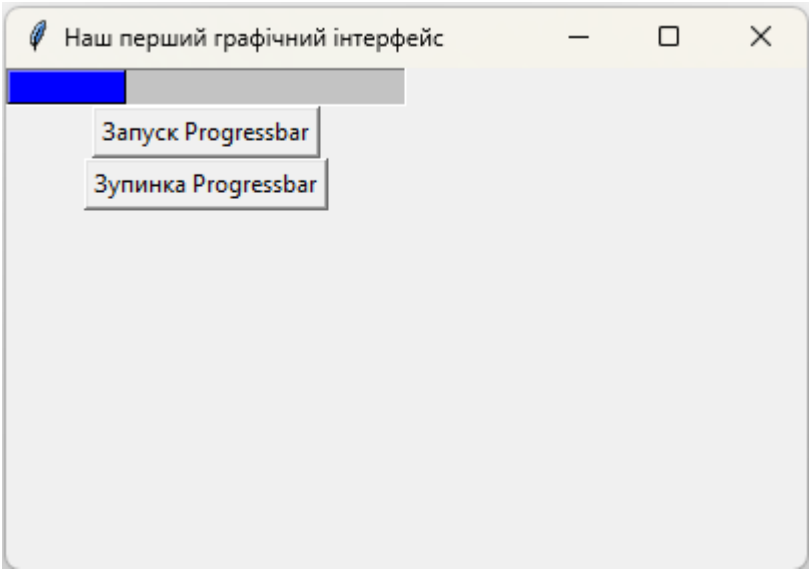
Це значення можна встановити на основі будь-якого процесу, або при виконанні завдання.

Як змінити колір Progressbar

Для зміни кольору спочатку потрібно створити стиль і задати колір фону, а потім налаштувати створений стиль на Progressbar. Розглянемо приклад в якому налаштуємо відображення індикатора прогресу та створиму емуляцію його роботи:

```
from tkinter import *
from tkinter import ttk
from tkinter.ttk import Progressbar

def clicked1():
    bar.start(100) # запускаємо нескінченний цикл завантаження

def clicked2():
    bar.stop()  # зупиняємо цикл завантаження

window = Tk()
window.title("Наш перший графічний інтерфейс")
window.geometry('400x250')
style = ttk.Style()  #створюємо стиль та задаємо його параметри
style.theme_use('default')
style.configure("black.Horizontal.TProgressbar", background='blue')
bar = Progressbar(window, length=200, style='black.Horizontal.TProgressbar')  #створюємо
bar['value'] = 1  # задаємо початкове значення
bar.grid(column=0, row=0)
btn1 = Button(window, text="Запуск Progressbar", command=clicked1)
btn1.grid(column=0, row=1)
btn2 = Button(window, text="Зупинка Progressbar", command=clicked2)
btn2.grid(column=0, row=2)

window.mainloop()
```

В результаті отримуємо наступне:



Завдання.

Зв’яжіть Progressbar із Spinbox, щоб при виборі певного значення у лічильнику, після натиснення кнопки, це значення відображалося індикатором прогресу.

Зіскануйте для перевірки

Створення власного віконного меню. Поле для завантаження файлів.

Додавання панелі меню

Для додавання панелі меню, використовуємо екземпляр класу класу **Menu**:

```
from tkinter import Menu

menu = Menu(window)
menu.add_command(label='Файл')
window.config(menu=menu)
```

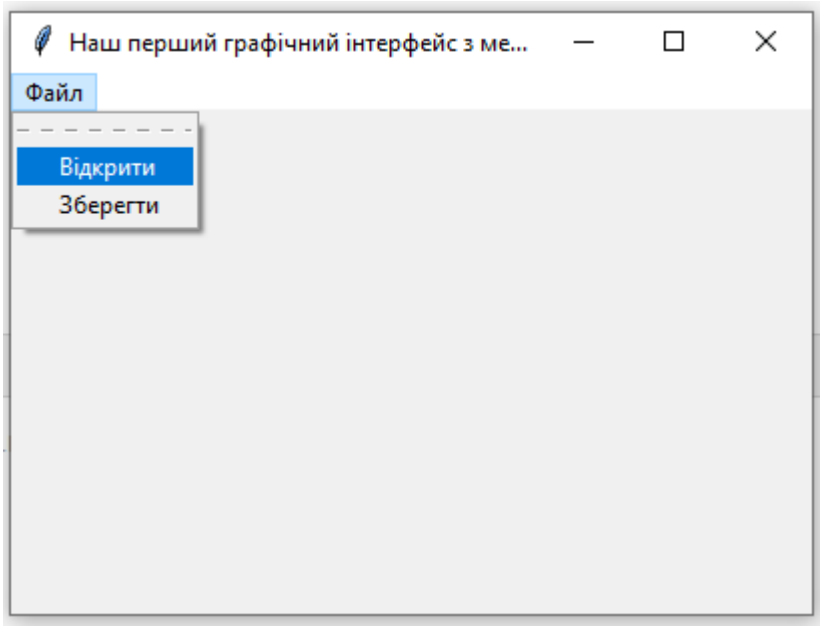
Спочатку ми створюємо меню, потім додаємо наш перший пункт підменю. Ви можете додавати пункти меню в будь-яке меню, скориставшись пунктом **add_cascade()** таким чином:

```
menu.add_cascade(label='Новий пункт', menu=new_item)
```

Наш код може виглядати так:

```
from tkinter import *
from tkinter import Menu

window = Tk()
window.title("Наш перший графічний інтерфейс з меню")
window.geometry('400x250')
menu = Menu(window)  # створюємо новий об'єкт класу меню
new_item = Menu(menu)  # створюємо пункт меню
new_item.add_command(label='Відкрити') #додаємо пункт підменю Відкрити
new_item.add_command(label='Зберегти') #додаємо пункт підменю Зберегти
menu.add_cascade(label='Файл', menu=new_item)  #додаємо меню до вікна
window.config(menu=menu)
window.mainloop()
```

Результат роботи:



Ви можете помітити пунктирну лінію на початку, якщо ви натиснете на цей рядок, вона відобразить пункти меню в невеликому окремому вікні. Цю функцію можна вимкнути, за допомогою параметру **tearoff** :
new_item = Menu(menu, tearoff=0)

Тобто досить відредагувати **new_item**, як у наведеному вище прикладі і він більше не буде відображати пунктирну лінію.

Ви так само можете ввести будь-який код, який працює, при натисканні користувачем на будь-який елемент меню, для цього задаємо властивість команди:
new_item.add_command(label='Відкрити', command=clicked), але не забуваємо описати відповідну функцію.

Додавання поля завантаження файлу

Для додавання поля з файлом, використовуємо екземпляр класу **filedialog**:
from tkinter import filedialog
file = filedialog.askopenfilename()

Після того, як ви виберете файл і натиснете "Відкрити"; файлова змінна буде містити вибраний шлях до файлу. Крім того, ви можете вибрати кілька файлів:
files = filedialog.askopenfilenames()

Вибір файлів за типом (фільтр файлів)

Можливість вказати тип файлів доступна при використанні параметра **filetypes**, однак при цьому важливо вказати розширення файлів.
file = filedialog.askopenfilename(filetypes = (("Text files","*.txt"),("all files","*.*")))

Є можливість вказати папку, використовуючи метод **askdirectory**:
dir = filedialog.askdirectory()

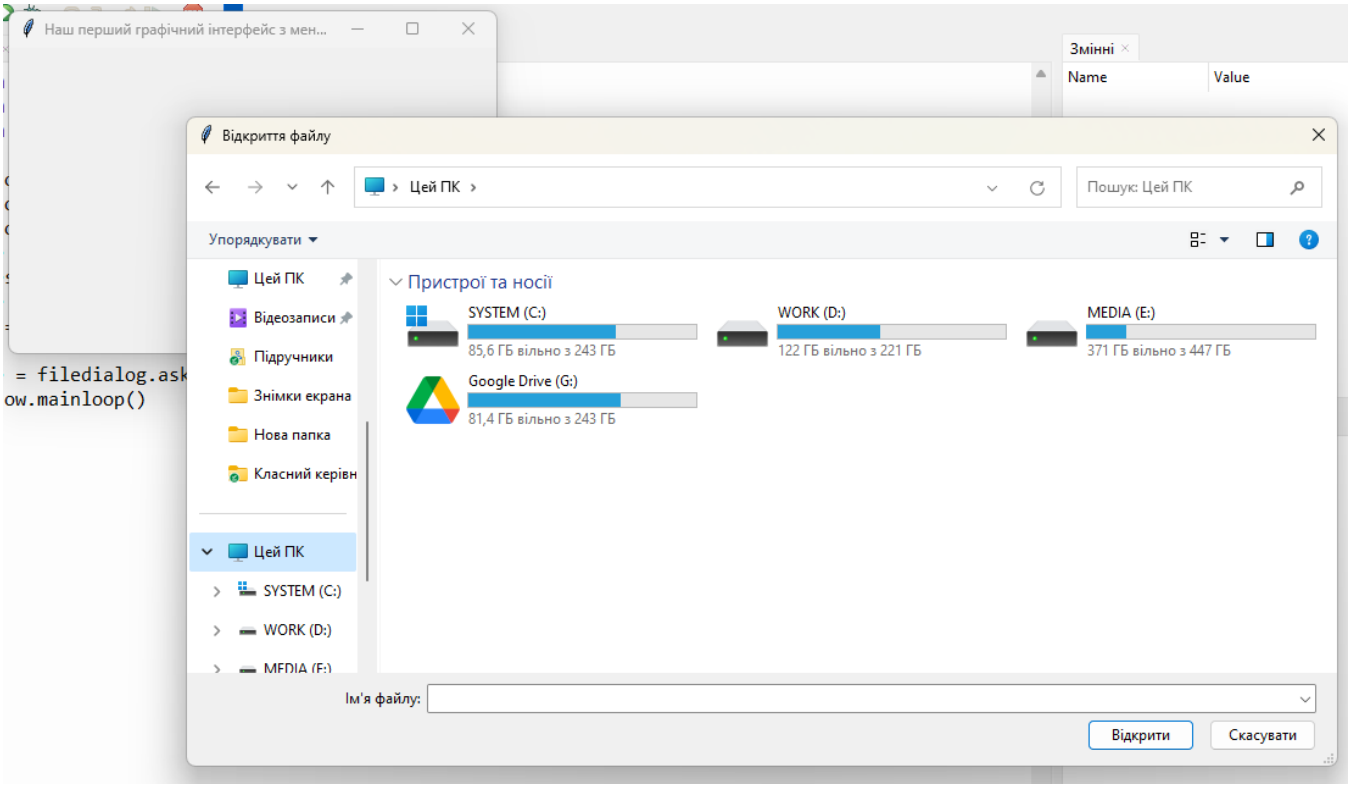
Щоб вказати початкову папку для діалогового вікна файлу, вкажіть **initialdir** в такий спосіб:

from os import path
file = filedialog.askopenfilename(initialdir= path.dirname(__file__))

Виконуємо наведений код і спостерігаємо як працюють вікна відкриття файлів.

```
from tkinter import *
from tkinter import filedialog
from os import path

window = Tk()
window.title("Вибір файлів")
window.geometry('400x250')
file = filedialog.askopenfilename()  # вибрати один файл у вікні діалогу
files = filedialog.askopenfilenames()  # вибрати декілька файлів у вікні діалогу
file = filedialog.askopenfilename(filetypes = (("Text files","*.txt"),("all files","*.*")))
# фільтруємо файли по вказаних типах
dir = filedialog.askdirectory()
file = filedialog.askopenfilename(initialdir= path.dirname("D:\IDE\\")) # вказуємо шлях до папки
```

window.mainloop()



Завдання.

Додайте в попереднє завдання функцію опрацювання події вибору меню і за допомогою неї виведіть вікно вибору файлу, встановіть фільтри вибору файлів з розширенням *.py та будь яких. Вивести ім’я вибраного файла у вікні.

Зіскануйте для перевірки

Використані джерела

1. Ривкінд Й. Я., Лисенко Т. І., Чернікова Л. А., Шакотько В. В. Інформатика : підручник для 8-го кл. закл. заг. серед. освіти. Київ : Генеза, 2021. 256 с.
2. ДистОсвіта - дистанційне навчання інформатики. Алгоритми та програми. URL: https://dystosvita.org.ua/course/view.php?id=8#section-2 (дата звернення: 30.04.2024)
3. Офіційна документація для Python 3.9.19. URL: https://docs.python.org/uk/3.9/index.html (дата звернення: 30.04.2024)