Мова програмування Python

Tkinter - створення графічного інтерфейсу на Python

Python має досить багато графічних інтерфейсів користувача, але тільки Tkinter вбудований в стандартну бібліотеку мови. Tkinter має ряд переваг. Він є крос платформним, тому той же код можна використовувати на Windows, macOS і Linux.

Візуальні ефекти відображаються через власні елементи поточної операційної системи, тому програми, створені за допомогою Tkinter, виглядають так, ніби вони належать до платформи, на якій вони працюють.

Хоча Tkinter є популярним графічним інтерфейсом Python, він має свої недоліки. Одна з них полягає в тому, що графічні інтерфейси, створені за допомогою Tkinter, виглядають застарілими. Якщо вам потрібен сучасний, помітний інтерфейс, то Tkinter може бути не зовсім тим, для цього є PyQt5, який розвивається сильніше в цьому відношенні.

Створення простого графічного інтерфейсу на Tkinter

Основним елементом графічного інтерфейсу Tkinter є вікно. Вікна - це контейнери, в яких розташовані всі елементи графічного інтерфейсу. Ці елементи графічного інтерфейсу, які включають текстові поля, ярлики та кнопки, називаються віджетами. Віджети розміщуються всередині вікон.

```
import tkinter as tk
```

Спробуйте створити нове вікно, призначивши його змінній вікна «window»:

```
window = tk.Tk()
```

Додавання нового віджета в додаток

Тепер, коли у вас є вікно, ви можете додати до нього віджет. Ми використовуємо клас tk.Label та додаємо текст до вікна.
Створіть **віджет мітки** (label) з текстом "Привіт, Tkinter!" і призначте його змінній під назвою greeting:

```
greeting = tk.Label(text="Привіт, Tkinter!")
```

Вікно, створене раніше, не змінюється. Створюється віджет ярлика, але він ще не доданий до вікна. Існує кілька способів додавання віджетів до вікна. Ви можете використовувати метод .pack() з віджета ярлика:

```
greeting.pack()
```

При використанні методу .pack() для розміщення віджета у вікні Tkinter встановлює розмір вікна якомога менше, поки віджет не впишеться в нього. Тепер давайте зробимо наступне:

```
window.mainloop()
```

Здається, що нічого не сталося, але це не зовсім так. window.mainloop() повідомляє Python, щоб запустити цикл подій Tkinter. Цей метод необхідний для таких подій, як натискання клавіш або кнопки, він також блокує виконання будь-якого коду, який слідує, поки вікно, на якому він був викликаний, не буде закрито.

Завдання 1 Створіть звичайне вікно в Tkinter

Напишіть повний скрипт для створення вікна в Tkinter з текстом «Python рулює!».

Робота з віджетами в Tkinter

Віджети є основою графічного інтерфейсу фреймворку Tkinter в Python. Це елементи, за допомогою яких користувачі взаємодіють з програмою. У Tkinter кожен віджет визначається класом. Нижче наведено список популярних віджетів від Tkinter:

Клас віджетів	Опис
Label	Використовується для відображення тексту або вставлення зображення у вікно програми.
Button	Кнопка, яка може мати текст, приймає певні дії, коли ви натискаєте на неї.
Entry	Віджет для введення одного рядка тексту. Еквівалент <input type="text"> в HTML.
Text	Віджет для введення великого тексту. Еквівалент <textarea>в HTML.
Frame	Прямокутна область, яка використовується для групування віджетів або додавання відстані між віджетами.

У подальших заняттях ми розглянемо на практиці, як працює кожен з перерахованих вище віджетів. Для отримання більш детальних списків віджетів Tkinter дивиться документацію.

Віджет Label — Відображення тексту та зображень

Віджети Label використовуються для відображення тексту або зображень. Текст на віджеті Label не може бути відредагований користувачем. Він тільки показується. Як показано в прикладах на початку цієї презентації, ви можете створити віджет Label за допомогою екземпляра класу Label і передавши рядок до текстового параметра:

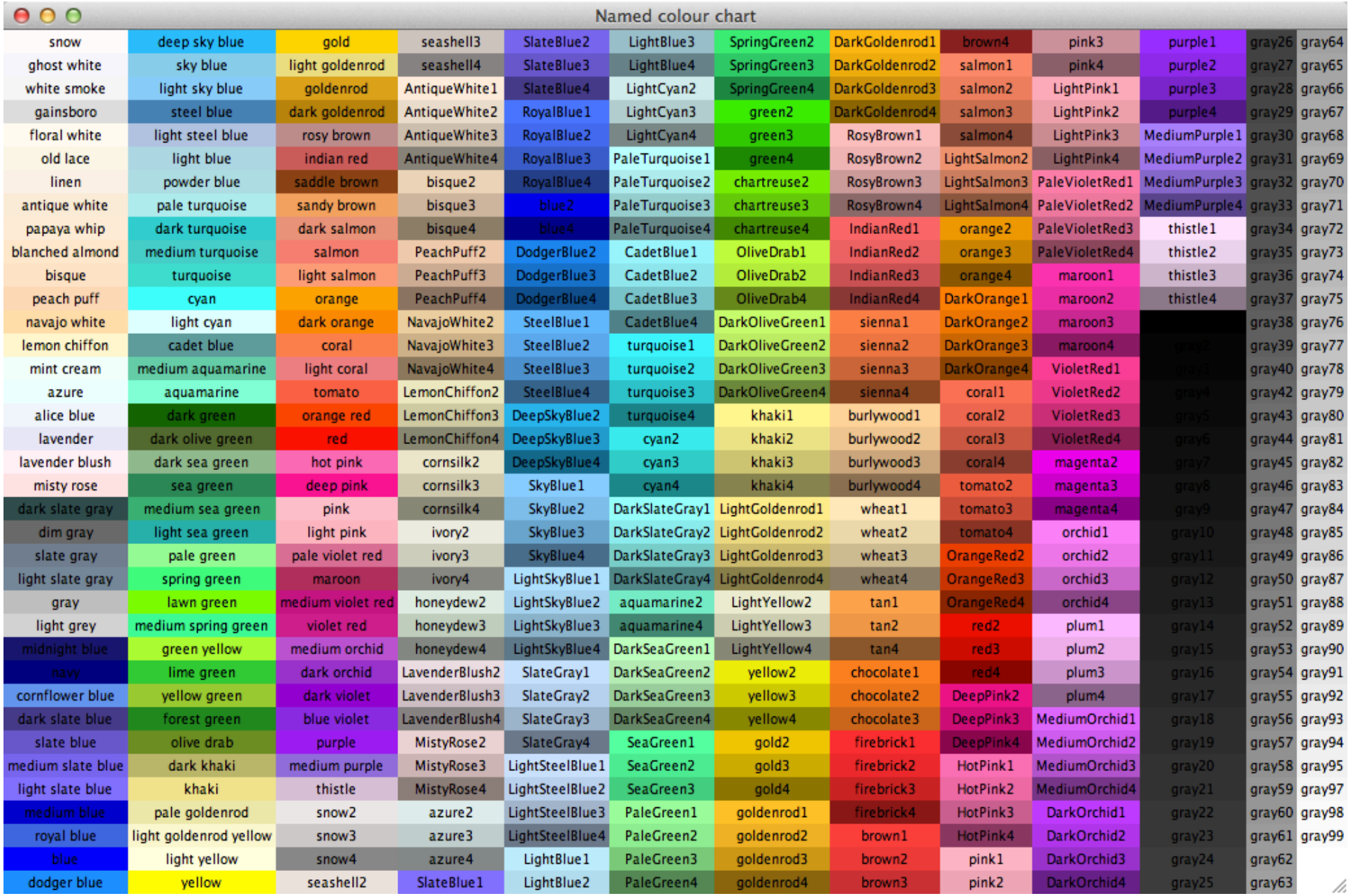
```
label = tk.Label(text="Hello, Tkinter")
```

Віджети Label відображають текст з встановленим за замовчуванням кольором системи та фоном. Зазвичай він чорно-білий. Тому, якщо у вашій операційній системі вказані інші кольори, ви їх побачите.
Ви можете змінити колір тексту та тла віджета Label через параметри foreground та background:

```
label = tk.Label(
    text="Привіт, Tkinter!",
    foreground="white",  # Встановлює білий текст
    background="black"  # Встановлює чорний фон
)
```

Деякі доступні кольори:
→ red — червоний;
→ orange — помаранчевий;
→ yellow — жовтий;
→ green — зелений;
→ blue — синій;
→ purple - бузковий.

Багато кольорів HTML мають однакову назву в Tkinter.


Named colour chart
snow ghost white white smoke gainsboro floral white old lace linen antique white papaya whip blanched almond bisque peach puff navajo white lemon chiffon mint cream azure alice blue lavender lavender blush misty rose dark slate gray dim gray slate gray light slate gray gray light grey midnight blue navy cornflower blue dark slate blue slate blue medium slate blue light slate blue medium blue royal blue blue dodger blue
deep sky blue sky blue light sky blue steel blue light steel blue light blue powder blue pale turquoise dark turquoise medium turquoise turquoise cyan light cyan cadet blue medium aquamarine aquamarine dark green dark olive green dark sea green sea green medium sea green light sea green pale green spring green lawn green medium spring green green yellow lime green yellow green forest green olive drab dark khaki khaki pale goldenrod light goldenrod yellow light yellow yellow
gold light goldenrod goldenrod dark goldenrod rosy brown indian red saddle brown sandy brown dark salmon salmon light salmon orange dark orange coral light coral tomato orange red red hot pink deep pink pink light pink pale violet red maroon medium violet red violet red medium orchid dark orchid dark violet blue violet purple medium purple thistle snow2 snow3 snow4 seashell2
seashell3 seashell4 AntiqueWhite1 AntiqueWhite2 AntiqueWhite3 AntiqueWhite4 bisque2 bisque3 bisque4 PeachPuff2 PeachPuff3 PeachPuff4 NavajoWhite2 NavajoWhite3 NavajoWhite4 LemonChiffon2 LemonChiffon3 LemonChiffon4 cornsilk2 cornsilk3 cornsilk4 ivory2 ivory3 ivory4 honeydew2 honeydew3 honeydew4 LavenderBlush2 LavenderBlush3 LavenderBlush4 MistyRose2 MistyRose3 MistyRose4 azure2 azure3 azure4 SlateBlue1
SlateBlue2 SlateBlue3 SlateBlue4 RoyalBlue1 RoyalBlue2 RoyalBlue3 RoyalBlue4 blue2 blue4 DodgerBlue2 DodgerBlue3 DodgerBlue4 SteelBlue1 SteelBlue2 SteelBlue3 SteelBlue4 DeepSkyBlue2 DeepSkyBlue3 DeepSkyBlue4 SkyBlue1 SkyBlue2 SkyBlue3 SkyBlue4 LightSkyBlue1 LightSkyBlue2 LightSkyBlue3 LightSkyBlue4 SlateGray1 SlateGray2 SlateGray3 SlateGray4 LightSteelBlue1 LightSteelBlue2 LightSteelBlue3 LightSteelBlue4 LightBlue1 LightBlue2
LightBlue3 LightBlue4 LightCyan2 LightCyan3 LightCyan4 PaleTurquoise1 PaleTurquoise2 PaleTurquoise3 PaleTurquoise4 CadetBlue1 CadetBlue2 CadetBlue3 CadetBlue4 turquoise1 turquoise2 turquoise3 turquoise4 DarkSlateGray1 DarkSlateGray2 DarkSlateGray3 DarkSlateGray4 aquamarine2 aquamarine4 DarkSeaGreen1 DarkSeaGreen2 DarkSeaGreen3 DarkSeaGreen4 SeaGreen1 SeaGreen2 SeaGreen3 PaleGreen1 PaleGreen2 PaleGreen3 PaleGreen4
SpringGreen2 SpringGreen3 SpringGreen4 green2 green3 green4 chartreuse2 chartreuse3 chartreuse4 OliveDrab1 OliveDrab2 OliveDrab4 DarkOliveGreen1 DarkOliveGreen2 DarkOliveGreen3 DarkOliveGreen4 khaki1 khaki2 khaki3 khaki4 LightGoldenrod1 LightGoldenrod2 LightGoldenrod3 LightGoldenrod4 LightYellow2 LightYellow3 LightYellow4 yellow2 yellow3 yellow4 gold2 gold3 gold4 goldenrod1 goldenrod2 goldenrod3 goldenrod4
DarkGoldenrod1 DarkGoldenrod2 DarkGoldenrod3 DarkGoldenrod4 RosyBrown1 RosyBrown2 RosyBrown3 RosyBrown4 IndianRed1 IndianRed2 IndianRed3 IndianRed4 sienna1 sienna2 sienna3 sienna4 burlywood1 burlywood2 burlywood3 burlywood4 wheat1 wheat2 wheat3 wheat4 tan1 tan2 tan4 chocolate1 chocolate2 chocolate3 firebrick1 firebrick2 firebrick3 firebrick4 brown1 brown2 brown3 brown4
salmon1 salmon2 salmon3 salmon4 LightSalmon2 LightSalmon3 LightSalmon4 orange2 orange3 orange4 DarkOrange1 DarkOrange2 DarkOrange3 DarkOrange4 coral1 coral2 coral3 coral4 tomato2 tomato3 tomato4 OrangeRed2 OrangeRed3 OrangeRed4 red2 red3 red4 DeepPink2 DeepPink3 DeepPink4 HotPink1 HotPink2 HotPink3 HotPink4 pink1 pink2
pink3 pink4 LightPink1 LightPink2 LightPink3 LightPink4 PaleVioletRed1 PaleVioletRed2 PaleVioletRed3 PaleVioletRed4 maroon1 maroon2 maroon3 maroon4 VioletRed1 VioletRed2 VioletRed3 VioletRed4 magenta2 magenta3 magenta4 orchid1 orchid2 orchid3 orchid4 plum1 plum2 plum3 plum4 MediumOrchid1 MediumOrchid2 MediumOrchid3 MediumOrchid4 DarkOrchid1 DarkOrchid2 DarkOrchid3 DarkOrchid4
purple1 purple2 purple3 purple4 MediumPurple1 MediumPurple2 MediumPurple3 MediumPurple4 thistle1 thistle2 thistle3 thistle4
gray26 gray27 gray28 gray29 gray30 gray31 gray32 gray33 gray34 gray35 gray36 gray37 gray38 gray39 gray40 gray42 gray43 gray44 gray45 gray46 gray47 gray48 gray49 gray50 gray51 gray52 gray53 gray54 gray55 gray56 gray57 gray58 gray59 gray60 gray61 gray62 gray63
gray64 gray65 gray66 gray67 gray68 gray69 gray70 gray71 gray72 gray73 gray74 gray75 gray76 gray77 gray78 gray79 gray80 gray81 gray82 gray83 gray84 gray85 gray86 gray87 gray88 gray89 gray90 gray91 gray92 gray93 gray94 gray95 gray97 gray98 gray99

Якщо ви хочете зрозуміти тему більш детально, особливо в кольорах системи macOS і Windows, керованих поточною темою ОС, перегляньте сторінку значень кольорів.
Ви також можете вказати колір, використовуючи шістнадцяткові значення RGB, які часто використовуються в CSS для стилю сайтів:

```
label = tk.Label(text="Привет, Tkinter!", background="#34A2FE")
```

Тепер фон став приємним синім кольором. Шістнадцяткові значення RGB, на відміну від назв кольорів, кодуються, але це робить їх більш керованими. На щастя, доступні інструменти для легкого і швидкого отримання шістнадцяткових колірних кодів.

Якщо ви не хочете регулярно вводити foreground і background, можна використовувати скорочені версії параметрів - fg і bg. Вони також відповідають за встановлення **кольору тексту** та **кольору тла**.

```
label = tk.Label(text="Привіт, Tkinter!", fg="white", bg="black")
```

Також можна керувати шириною та висотою ярлика за допомогою параметрів width і height :

```
import tkinter as tk

window = tk.Tk()

label = tk.Label(
    text="Привіт, Tkinter!",
    fg="white",
    bg="black",
    width=20,
    height=20
)

label.pack()
window.mainloop()
```

Може здатися дивним, що ярлик у вікні не квадратний, хоча параметр як ширини, так і висоти становить 20. Все тому, що ширина і висота вимірюються в текстових одиницях.
Горизонтальна текстова одиниця визначається шириною символу "0", або числом нуль, у системному шрифті за замовчуванням. Аналогічно, одна вертикальна текстова одиниця визначається висотою символу "0".

Ярлики чудово підходять для відображення тексту, але користувач не може вводити дані через них. Наступні три віджети дозволяють користувачеві вводити інформацію.

Створення кнопки за допомогою віджета Button

Віджети Button потрібні для створення кнопок що можна натиснути. Їх можна налаштувати таким чином, щоб при натисканні була викликана певна функція.

Існує багато подібностей між віджетами Button і Label. В основному, кнопка - це просто ярлик, на який можна натиснути. Ті ж аргументи також використовуються для створення стилів віджетів ярликів і кнопок.

Наприклад, наступний код створює кнопку з синім фоном і жовтим текстом. Ширина і висота також встановлюються з індикаторами 25 і 5 текстових одиниць:

```
import tkinter as tk

window = tk.Tk()

button = tk.Button(
    text="Натисни на мене!",
    width=25,
    height=5,
    bg="blue",
    fg="yellow",
)

button.pack()
window.mainloop()
```

Віджет Entry - Однорядне текстове поле

У випадках, коли ви хочете отримати текстову інформацію від користувача, наприклад адресу електронної пошти, використовується віджет Entry. Він відображає невелике текстове поле, де користувач може вводити текст.
Створення віджета Entry практично нічим не відрізняється від створення ярлика і кнопки. Наприклад, наступний код створює віджет з синім фоном і жовтим текстом довжиною 50 текстових одиниць:

```
entry = tk.Entry(fg="yellow", bg="blue", width=50)
```

У випадку з віджетом однорядного текстового поля (Entry), цікавим є не процес створення стилю, а як отримати цей ввід від користувача. Є три основні операції, які можна виконати за допомогою віджета однорядного текстового поля (Entry):

→ Отримання всього тексту через .get()
→ Видалення тексту за допомогою .delete()
→ Вставлення нового тексту за допомогою .insert()

Щоб краще зрозуміти, як працюють віджети Entry, створіть один екземпляр елемента і спробуйте ввести в нього текст. Відкрийте оболонку Python і виконайте такі дії:
Спочатку імпортуйте tkinter і створіть нове вікно:

```
import tkinter as tk
window = tk.Tk()
```

Тепер створіть віджети Label та Entry :

```
label = tk.Label(text="Им'я")
entry = tk.Entry()
```

Тут ярлик визначає, що користувач повинен написати у віджеті Entry. Обмежень на вхідні дані немає, це просто натяк на те, що потрібно ввести "Ім'я". Віджети стануть видимими у вікні після виконання методу .pack() для кожного з них:

```
label.pack()
entry.pack()
```

Зауважте, що у вікні віджет ярлика автоматично центрується на віджеті текстового поля. Це особливість упаковки pack, яка буде описана далі.

Тепер у віджеті текстового поля є текст, тільки він ще не відправлений в програму. Щоб отримати текст і призначити його значення змінній name, скористайтеся методом .get():

```
name = entry.get()
```

Ви також можете видалити текст за допомогою методу .delete(). Цей метод приймає аргумент, який є цілим числом, і повідомляє Python, який символ видалити. Наприклад, у наведеному нижче фрагменті коду показано, як можна видалити перший символ із текстового поля за допомогою методу .delete(0):

```
entry.delete(0)
```

Зауважте, що, як і рядки в Python, текст в віджеті однорядного текстового поля індексується, а індексування починається з 0.

Якщо потрібно видалити кілька символів із текстового поля, потрібно передати другий цілочисельний аргумент .delete(), вказавши індекс символу, на якому має завершитися процес видалення. Наприклад, такий код видаляє перші чотири літери з текстового поля:

```
entry.delete(0, 4)
```

Метод .delete() працює аналогічно методу рядка slice(), щоб видалити деякі символи з рядка. Перший аргумент визначає початковий індекс видалення, останній індекс визначає, де саме має припинитися процес видалення.

Щоб видалити весь текст з текстового поля, другий аргумент методу .delete() використовує спеціальну константу tk.END:

```
entry.delete(0, tk.END)
```

Ви також можете вставити текст в віджет однорядкового текстового поля за допомогою методу .insert():

```
entry.insert(0, "Петров")
```

Перший аргумент повідомляє методу .insert(), куди вставити текст. Якщо в текстовому полі немає тексту, новий текст завжди буде вставлений на початку віджета, незалежно від того, яке значення передається як перший аргумент.

Наприклад, якщо ви визиваєте .insert() з першим аргументом 100, а не 0, як згадувалося вище, вивід буде таким самим.

У тому випадку, якщо текстове поле вже містить текст, то .insert() вставе новий текст у вказане положення та перемістіть існуючий текст вправо:

```
entry.insert(0, "Іван ")
```

Віджети однорядкових текстових полів відмінно підходять для отримання невеликої кількості тексту від користувача, однак вони відображають лише один рядок тексту, тому не підходять для великого обсягу інформації. Для таких випадків краще використовувати віджети Text.

Віджет Text - введення великого тексту у Tkinter

Віджети Text використовуються для введення тексту, як і віджети Entry. Різниця полягає в тому, що Text може містити кілька рядків тексту. За допомогою віджета Text користувач може вводити цілі абзаци або сторінки тексту. Як і у випадку з віджетами Entry, ви можете виконати три основні операції над віджетами Text:

→ Отримання тексту за допомогою методу .get()
→ Видалення тексту за допомогою методу .delete()
→ Вставлення тексту за допомогою методу .insert(

Незважаючи на те, що назви методів такі ж, як ті, що використовуються в Entry, процес реалізації дещо відрізняється.

У оболонці Python потрібно створити нове порожнє вікно, а всередині нього за допомогою методу .pack() розмістити текстовий віджет:

```
import tkinter as tk
window = tk.Tk()
text_box = tk.Text()
text_box.pack()
```

За замовчуванням текстові поля набагато більше, ніж однорядні текстові віджети Entry.

Щоб активувати текстове поле, натисніть на будь-яку точку всередині вікна. Введіть слово " Hello ". Натисніть кнопку Enter на клавіатурі, а потім введіть слово " World " у другому рядку.

Як і у випадку з віджетами Entry, ви можете отримати текст свого текстового віджета за допомогою методу .get(). Однак виклик .get() без аргументів не повертає весь текст, як це відбувається з однорядними текстовими віджетами Entry. З'явиться виняток TypeError:

Метод .get() для текстового віджета запитує принаймні один аргумент. Виклик .get() з одним індексом повертає один символ. Щоб отримати кілька символів, необхідно передати початкові та кінцеві індекси.
Індекси в віджетах Text діють інакше, ніж у віджетах Entry. Оскільки віджети Text можуть отримувати кілька рядків тексту, індекс повинен містити такі два пункти:

→ Номер рядка, де розташований символ;
→ Розташування символу в рядку.

Номери рядків починаються з 1, а позиція символів – з 0. Для отримання індексу створюється рядок формату "<line>.<char>", де <line> замінено номером рядка, а <char> номером символів. Наприклад, "1.0" означає перший символ у першому рядку, а "2.3" - четвертий символ у другому рядку.

Використовуйте індекс "1.0", щоб отримати першу букву раніше створеного текстового поля:

```
text_box.get("1.0")
```

У слові “Hello" 5 букв, буква о знаходиться під індексом 4, так як відлік символів починається з 0, а слово “Hello" починається з першого рядка в текстовому полі.
Як і у випадку зі зрізами рядків в Python, щоб отримати все слово “Hello" з текстового поля, остаточний індекс повинен бути на один більше, ніж останній символ для читання.

Таким чином, щоб отримати слово “Hello" з текстового поля, "1.0" використовується для першого індексу, а "1,5" для другого індексу:

```
text_box.get("1.0", "1.5")
```

Щоб отримати весь текст з текстового віджета, встановіть індекс на "1.0" і використовуйте спеціальну постійну tk.END для другого індексу:

```
text_box.get("1.0", tk.END)
```

Зауважте, що текст, повернутий за допомогою методу .get(), містить символи переходу до нового рядка \n. У цьому прикладі також можна побачити, що кожен рядок у віджеті Text містить символ нового рядка в кінці, включаючи останній рядок тексту в текстовому полі.

Використовуйте метод .delete() для видалення символів з текстового віджета. Він працює точно так само, як і метод .delete() для віджетів Entry.
Існує два способи використання .delete():

→ С одним аргументом;
→ С двумя аргументами.

За допомогою параметра з одним аргументом можемо передати індекс символа для видалення до .delete(). Наприклад, такий код видаляє перший символ H з текстового поля:

```
text_box.delete("1.0")
```

У версії з двома аргументами передаються два індекси, щоб видалити групу символів, починаючи з першого символу і закінчуючи другим, але не включаючи його. Наприклад, щоб видалити залишки частини “ello" з першого рядка текстового поля, використовуються індекси "1.0" і "1.4":

```
text_box.delete("1.0", "1.4")
```

Зауважте, що текст з першого рядка видалено, але ліворуч є порожній рядок, за яким слідує слово Word у другому рядку:

Хоча ви не можете його побачити, на першому рядку залишився лише один символ - це символ нового рядка \n. Ви можете побачити його самі, викликавши метод .get():

```
text_box.get("1.0")
```

Коли ви видалите цей символ, решта вмісту текстового віджета переміститься вгору на рядок:

```
text_box.delete("1.0")
```

Спробуємо очистити решту текстового віджета. Встановіть "1.0" як початковий індекс і tk.END як другий індекс:

```
text_box.delete("1.0", tk.END)
```

Ви також можете вставити текст у текстовий віджет за допомогою методу .insert():

```
text_box.insert("1.0", "Hello")
```

Метод вставляє слово “Hello" на початку текстового віджета. Використовується знайомий формат «<line>.<column>», а також метод .get() для уточнення положення вставки тексту.

Перевірте, що відбувається, коли ви намагаєтеся вставити слово “World" у другий рядок:

```
text_box.insert("2.0", "World")
```

Якщо ви хочете помістити текст у новий рядок, то вам потрібно буде вставити новий символ рядка \n вручну:

```
text_box.insert("2.0", "\nWorld")
```

Метод .insert() виконує такі дві дії:

→ Вставляє текст у вказану позицію, якщо текст уже є в цій позиції або після нього;
→ Додає текст до вказаного рядка, якщо номер символу перевищує індекс останнього символу в текстовому полі.

Зазвичай недоцільно намагатися відстежувати індекс останнього символу. Найкращий спосіб вставити текст в кінці віджета Text - це передати tk.END як перший параметр у методі .insert():

```
text_box.insert(tk.END, "\nВстав мене в новий рядок!")
```

Віджети «Label», «Button», «Entry» і «Text» — це лише мала частина віджетів, доступних у Tkinter. Інші віджети включають прапорці, перемикачі, смуги прокрутки та індикатори

Використання віджета Frame у Tkinter

Віджети Frame важливі для впорядкування макета віджетів у вашому додатку.

Перш ніж заглибитися в деталі макетів віджетів, давайте докладніше розглянемо, як працює віджет кадру і як ви можете вставити в них інші віджети. Наступний скрипт створює порожній віджет кадру і додає його до головного вікна програми:

```
import tkinter as tk

window = tk.Tk()
frame = tk.Frame()
frame.pack()

window.mainloop()
```

Метод frame.pack() розміщує кадр у вікні, тому розмір вікна стає таким же маленьким, наскільки це можливо, для того, щоб відповідати кадру.

Кадри найкраще розглядати як контейнери для інших віджетів.

Ви можете вставити будь-який віджет у кадр, встановивши головний атрибут цього віджета:

```
frame = tk.Frame()
label = tk.Label(master=frame)
```

Щоб побачити, як це працює, давайте напишемо скрипт для створення двох віджетів Frame під назвою frame_a і frame_b.
У цьому скрипті frame_a містить ярлик з текстом "I'm in Frame A", а frame_b містить ярлик з текстом "I'm in Frame B".

```
import tkinter as tk

window = tk.Tk()

frame_a = tk.Frame()
frame_b = tk.Frame()

label_a = tk.Label(master=frame_a, text="I'm in Frame A")
label_a.pack()

label_b = tk.Label(master=frame_b, text="I'm in Frame B")
label_b.pack()

frame_a.pack()
frame_b.pack()

window.mainloop()
```

Зауважте, що frame_a кадру розміщується у вікні перед кадром frame_b. У вікні, що відкриється, показано ярлик у frame_a кадрі над ярликом у frame_b кадрі

Тепер давайте подивимося, що станеться, коли ви змінюєте порядок вставки frame_a.pack() і frame_b.pack():

```
import tkinter as tk

window = tk.Tk()

frame_a = tk.Frame()
label_a = tk.Label(master=frame_a, text="I'm in Frame A")
label_a.pack()

frame_b = tk.Frame()
label_b = tk.Label(master=frame_b, text="I'm in Frame B")
label_b.pack()

# Вставка рамок у вікні змінилися місцями.
frame_b.pack()
frame_a.pack()

window.mainloop()
```

Тепер етикетка label_b зверху. Оскільки мітка label_b призначена frame_b кадру, вона переміщується там, де розташований кадр frame_b.

Усі чотири типи віджетів, які обговорюються — Label, Button, Entry і Text — мають атрибут master, який встановлюється під час їх створення. Таким чином, ви можете керувати тим, якому кадру призначено віджет.

Віджети Frame чудово підходять для логічної організації інших віджетів у вікні додатку. Пов'язаним віджетам можна призначити один і той же кадр, так що якщо кадр коли-небудь рухається у вікні, пов'язані віджети рухаються разом з ним.
На додаток до логічного групування віджетів, кадри можуть трохи поліпшити зовнішній вигляд вашого додатка.

Атрибут relief в Tkinter — зміна стилю кадру

Кадри можуть змінювати свій стиль за допомогою атрибута relief, який створює межу навколо кадру. Можна призначити relief будь-яке з таких значень:

→ tk. FLAT: Без ефекту кадру (за замовчуванням);
→ tk. SUNKEN : Створюється ефект поглиблення елемента;
→ tk. RAISED: Створюється ефект опуклості елемента;
→ tk. GROOVE: Створює ефект рамки, що врізається в текстуру, свого роду виїмка;
→ tk. RIDGE : Створюється ефект опуклої виїмки.

Щоб застосувати ефект кадру, потрібно встановити значення атрибута borderwidth більше 1. Цей атрибут налаштовує ширину рамки кадру в пікселях. Кращий спосіб зрозуміти, як виглядає кожен ефект , - перевірити на практиці.

Ось скрипт, який ставить п'ять кадрів у вікно, де кожен має своє власне значення аргументу relief:

```
import tkinter as tk

border_effects = {
    "flat": tk.FLAT,
    "sunken": tk.SUNKEN,
    "raised": tk.RAISED,
    "groove": tk.GROOVE,
    "ridge": tk.RIDGE,
}

window = tk.Tk()

for relief_name, relief in border_effects.items():
    frame = tk.Frame(master=window, relief=relief, borderwidth=5)
    frame.pack(side=tk.LEFT)
    label = tk.Label(master=frame, text=relief_name)
    label.pack()

window.mainloop()
```

Розбір сценарію за рядками коду:
Рядки з 3 по 9 створюють словник, ключі якого є назвами різних ефектів рельєфу, доступних в Tkinter. Значення є відповідними об'єктами Tkinter. Цей словник призначено змінній border_effects
Рядок 13 запускає цикл for для кожного елемента словника border_effects
Рядок 14 створює новий віджет Frame і присвоює його об'єкту window. Атрибут relief встановлено на відповідний елемент рельєфу в словнику border_effects, а атрибут border встановлено на 5 пікселів таким чином, щоб ефект був видимим
Рядок 15 розміщує віджет кадру у вікні за допомогою методу .pack(). Ключове слово " side " повідомляє Tkinter, де розмістити межу. Докладніше про те, як це працює, ви дізнаєтеся далі.
Рядки 16 і 17 створюють віджет текстового ярлика, щоб відобразити назву рельєфу і помістити його в новостворений кадр.

Правила іменування віджетів у Tkinter

При створенні віджета ви можете дати йому будь-яке ім'я, за умови, що це ім'я ще не зареєстроване самим Python, наприклад, with, for, range і т.д.
Зазвичай рекомендується включити ім'я класу віджета в ім'я змінної, призначеної екземпляру віджета. Наприклад, якщо для відображення імені користувача використовується віджет Label, ви можете назвати віджет label_user_name. Віджет Entry, який використовується для визначення віку користувача, можна назвати entry_age.

Коли ви включаєте ім'я класу віджета в ім'я змінної, ви допомагаєте собі (і кожному, кому потрібно прочитати ваш код), зрозуміти, до якого типу віджета належить ім'я змінної.

Однак використання повної назви класу віджетів може призвести до довгих імен змінних, тому ви можете використовувати абревіатуру для позначення кожного типу віджета. Надалі використовуються наступні скорочені префікси для назви віджетів:

Клас віджетів	Префікс імені змінної	Приклад
Label	lbl	lbl_name
Button	btn	btn_submit
Entry	ent	ent_age
Text	txt	txt_notes
Frame	frm	frm_address

Сьогодні ми дізналися, як створити вікно, використовувати віджети та працювати з кадрами. На даний момент ми змогли створити кілька простих вікон, які відображають повідомлення, але це ще не повноцінний додаток.

Далі ми будемо розглядати управління макетом додатка за допомогою потужних менеджерів геометрії Tkinter.

Завдання 2 - створіть текстове поле та вставте в нього текст

Завдання: Створіть однорядне текстове поле та вставте на нього текст.
Напишіть скрипт для відображення віджета однорядного текстового поля Entry шириною 40 текстових блоків з кольоровим фоном і білим текстом;
Використовуйте метод .insert(), щоб вставити текст: "Як ваше ім'я?".