GUI - TkInter

GUI

Graficzny interfejs użytkownika (GUI) umożliwia nam korzystanie z urządzeń przy pomocy ikon, przycisków, pasków i innych graficznych elementów.

Samodzielne przygotowanie tych komponentów i mechanizmów wymagałoby dużego nakładu pracy. Stąd też powstała cała gama "gotowców" z których można skorzystać w różnych języka programowania.

Do jednych z najpopularniejszych wariantów dostępnych w Pythonie należą:

1. TkInter
   Ten zestaw narzędzi dostarczony jest razem z językiem Python. Umożliwia relatywnie łatwe rozpoczęcie tworzenia graficznych aplikacji.
   Do jego głównych wad zalicza się wygląd - domyślnie aplikacje stworzone z TkInter nie wyglądają zbyt nowocześnie. Spowodowało to powstanie licznych modyfikacji oryginalnej biblioteki mających na celu poprawienie tego aspektu.
2. Qt (Pyside, PyQt)
   Qt należy do najpopularniejszych frameworków GUI. Dostępny jest w większości języków programowania. W Pythonie dostępny jest z pomocą bibliotek Pyside i PyQt (ten pierwszy to oficjalny produkt firmy odpowiedzialnej za rozwój Qt).
   Do bardziej wymagających projektów dostępny jest QtDesigner umożliwiający wizualne opracowywanie graficznego szablonu dla aplikacji.
3. GTK (PyGObject)
   GTK wykorzystywany jest m.in. przez środowisko graficzne GNOME będący jednym z najpopularniejszych wyborów dla systemów opartych o Linux-a.
   Zapewnia przyzwoity wygląd aplikacji, ale nie zawsze dobrze integrujący się z środowiskami innymi niż GNOME.
   Istnieją nieoficjalne programy służące do projektowania okien dla GTK, takie jak Cambalanche.
4. Kivy
   Framework wykorzystywany m.in. przy bardziej nietypowych interfejsach np. na urządzenia mobilne.
5. DearPyGUI
   Zapewnia wsparcie wyświetlania z wykorzystaniem GPU, dzięki czemu w założeniu interakcja z aplikacją powinna być bardziej responsywna.
6. Flet
   To stosunkowy nowy wariant, umożliwia wykorzystanie frameworku Flutter w Pythonie.

TkInter

Do zaimportowanie TkInter-a można użyć:

```
import tkinter as tk
```

- podmiana nazwy tkinter na tk to dość popularne podejście

Pierwszą czynnością zazwyczaj jest stworzenie podstawowego okna:

```
import tkinter as tk


okno = tk.Tk()
okno.title("Jakiś tytuł")


okno.mainloop()
```

Metoda `mainloop` odpowiada za uruchomienie głównej pętli odpowiedzialnej za wyświetlanie okna.

Podstawowe komponenty

Pełną listę komponentów można znaleźć tutaj:

https://docs.python.org/3/library/tkinter.html

Elementy interfejsu przypisujemy pod zmienne i przekazujemy odpowiednie parametry. Zazwyczaj pierwszy parametr wskazuje na to do którego elementu dany komponent należy (w najprostszym wariancie będzie to okno aplikacji).

Do podstawowych aplikacji można skoncentrować się na:

Label

- czyli prosta etykieta z tekstem

```
etykieta = tk.Label(rodzic_komponentu, inne_parametry)
```

np.

```
text_label = tk.Label(okno, text="jakis napis")
```

```
pic_label = tk.Label(okno, image=obrazek)
```

Button

- przycisk

```
przycisk = tk.Button(text="jakis napis", command=nazwa_funkcji)
```

Input

- pole do wpisywania zawartości

```
wprowadz = tk.Entry(width=80)
```

Frame

Gdy chcemy pogrupować jakieś elementy (przyciski, etykiety etc.) z pomocą przychodzi nam `frame`. Taka ramka staje się rodzicem pozostałych komponentów.

To kontener na inne komponenty.

```
import tkinter as tk

okno = tk.Tk()
okno.title("Jakiś tytuł")

ramka = tk.Frame(okno, padding = 10)
text_label = tk.Label(ramka, text="jakis napis")
wprowadz = tk.Entry(ramka, width=80)

okno.mainloop()
```

Rozmieszczanie komponentów

Jeśli spróbujecie powyższego kodu zobaczycie, że nie działa to jeszcze poprawnie. Wynika to z tego, że nie zadeklarowaliśmy w jaki sposób mają być rozmieszczone komponenty.

Dwie podstawowe metody to:

Pack

Rozmieszcza komponenty półautomatycznie w ramach rodzica zgodnie z przekazanymi parametrami. Mogą one być umieszczone poziomo lub pionowo zgodnie z wartością parametru `side`.

Parametry:

- side - określa położenie elementu (`top`, `bottom`, `left`, `right`)
- fill - określa czy element ma być rozciągnięty w daną stronę (`x`, `y` , `both`)
- padx, pady - odstępy na zewnątrz komponentu
- ipadx, ipady - odstępy wewnątrz komponentu

np.

```
import tkinter as tk

okno = tk.Tk()
okno.title("Jakiś tytuł")

ramka = tk.Frame(okno)
ramka.pack()
text_label = tk.Label(ramka, text="jakis napis")
text_label.pack(side="left")
przycisk = tk.Button(ramka, text="jakis napis")
przycisk.pack(side="right")
wprowadz = tk.Entry(okno, width=80)
wprowadz.pack()

okno.mainloop()
```

Grid

Elementy rozmieszczane są na siatce wierszy i kolumn (można to sobie wyobrazić jako umieszczenie ich w komórkach arkusza kalkulacyjnego.)

https://www.pythontutorial.net/tkinter/tkinter-grid/

W przypadku "rozciągnięcia" elementu na kilka komórek trzeba:

- ustawić colspan / rowspan
- ustawić kierunek wypełnienia (n, s, w, e, we, ns)

CustomTkInter

Myślę, że dało się zauważyć, że domyślny TkInter jest dość brzydki. Z pomocą przychodzą nam różne warianty tego modułu, które mają na celu poprawę tego aspektu.

Jeśli chodzi o składnię w większości wypadków jest niemal identyczna z podstawowym TkInter - należy jedynie podmienić nazwę modułu.

```
import tkinter as tk
import customtkinter as ctk

okno = ctk.CTk()
okno.title("Jakiś tytuł")
okno.configure(padx=50, pady=50)

etykieta = ctk.CTkLabel(master=okno, text="Tekst etykiety")
etykieta.pack()

wprowadz = ctk.CTkEntry(master=okno, width=400)
wprowadz.pack()

# Do przycisku zazwyczaj dodaje się parametr
# command = nazwa_funkcji
# uruchamiajacy jakas funkcje po przyciśnięciu
przycisk = ctk.CTkButton(master=okno, text="Tekst przycisku")
przycisk.pack()

okno.mainloop()
```

Odno?niki

https://docs.python.org/3/library/tkinter.html

https://www.pythontutorial.net/tkinter/tkinter-grid/

John Elder, TKinter Widget Quick Reference Guide

Wersja #1
Utworzono 2024-10-11 15:24:36 UTC przez Przemek Jeske
Zaktualizowano 2024-10-11 15:25:40 UTC przez Przemek Jeske