Python Tkinter (TkInterface) - Výukový materiál

https://www.geeksforgeeks.org/python/python-gui-tkinter/, https://tkinter.py.cz/python_tkinter.html

Úvod do Tkinter

Tkinter je vestavěná knihovna Pythonu pro vytváření grafických uživatelských rozhraní (GUI). Funguje jako lehký wrapper (nepřidává vlastní logiku, jen překlad Tk) kolem GUI toolkitu Tcl/Tk a nabízí vývojářům v Pythonu jednoduchý a intuitivní způsob, jak vytvářet desktopové aplikace. Podporuje správu layoutu, zpracování událostí a přizpůsobení, což z něj činí ideální nástroj pro rychlý vývoj GUI v Pythonu.

Proč potřebujeme Tkinter?

- Poskytuje vestavěný a snadno použitelný způsob vytváření GUI aplikací v Pythonu
- Nabízí různé widgety jako tlačítka, popisky, textová pole a menu pro vytváření interaktivních aplikací
- Eliminuje potřebu externích GUI frameworků - tkinter je součástí Pythonu
- Podporuje událostmi řízené programování, což ho činí vhodným pro responzivní uživatelská rozhraní

Vytvoření první Tkinter GUI aplikace

Pro vytvoření Tkinter aplikace v Pythonu postupujte podle těchto základních kroků:

1. **Importujte modul tkinter**: Importujte modul tkinter, který je nezbytný pro vytváření GUI komponent, `import tkinter`, nebo `from tkinter import *`
2. **Vytvořte hlavní okno (kontejner)**: Inicializujte hlavní okno (root) aplikace pomocí třídy `Tk()`
3. **Nastavte vlastnosti okna**: Můžete nastavit vlastnosti jako titulek a velikost okna
4. **Přidejte widgety do hlavního okna**: Můžete přidat libovolný počet widgetů jako tlačítka, popisky, vstupní pole atd. do hlavního okna pro návrh rozhraní. Widget má dvojí skupenství, jednak je to instance třídy ve skriptu a jednak je to zobrazený prvek na monitoru.
5. **Uspořádejte widgety**: Použijte geometry managery jako `pack()`, `grid()` nebo `place()` pro uspořádání widgetů v okně.
6. **Aplikujte triggery událostí na widgety**: Můžete připojit triggery událostí k widgetům, abyste definovali, jak reagují na interakce uživatelů, tzv. řízení událostí (event handling).

Ahoj světe

Existují dvě hlavní metody při vytváření Python aplikace s GUI:

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

1. Tk()

Pro vytvoření hlavního okna (root) v Tkinter používáme třídu `Tk()`.

2. mainloop()

Metoda `mainloop()` se používá ke spuštění aplikace, jakmile je připravena. Je to nekonečná smyčka, která udržuje aplikaci spuštěnou, čeká na výskyt událostí (jako jsou kliknutí na tlačítka) a zpracovává tyto události, dokud není okno zavřeno.

Příklad 1:

AhojSvete.py

```
from tkinter import *

okno = Tk() # Vytvoření hlavního okna
w = Label(okno, text="Nazdar světe!")
w.pack()

okno.mainloop()
```

Na prvním řádku importujeme modul tkinter abychom nemuseli u všech metod psát prefix `tkinter`.

Jméno *okno* je náš název pro instaci třídy *Tk*, která představuje nejvýše postavené okno se základními ovládacími prvky. Do tohoto okna se ukládají všechny další widgety.

Jméno *w* je náš název pro instanci třídy *Label*, která je podřízená objektu *okno* a nese námi vložený text.

`w.pack()` - nezbytný úkon, který zviditelní předtím deklarovaný objekt. Jde o akt správy geometrie, který si označíme jako expozici neboli výstavu. Metoda *pack()* může rozšířit widgety tak, aby vyplnily dostupný prostor nebo je umístit v pevné velikosti.

Poslední instrukce evokuje metodu `mainloop()` pro objekt *okno*, která generuje nekonečnou smyčku událostí.

Příklad 2:

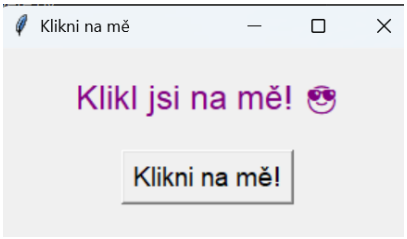
AhojSvete2.py

```
from tkinter import *

def zmen_text():
    label.config(text="Ahoj, Tkinter funguje!")

okno = Tk() # Vytvoření hlavního okna
okno.title("Můj první GUI program")
okno.geometry("300x150")

label = Label(okno, text="Klikni na tlačítko", font=("Arial", 12))
label.pack(pady=20)

tlacitko = Button(okno, text="Klikni", command=zmen_text)
tlacitko.pack()

okno.mainloop()
```

Úkol

Okno s tlačítkem `Button(master, text="Klikni", command=change_text)` které změní barvu textu v label.

```
def zmen_text():
    label.config(text="Klikl jsi na mě! 😎", fg=random.choice(colors))
```



KlikniNaMe.py

Tkinter Widgety

Existuje řada tkinter widgetů, které můžeme umístit do naší tkinter aplikace. Některé z hlavních widgetů jsou vysvětleny níže:

1. Label (Popisek)

Odkazuje na zobrazovací pole, kde zobrazujeme text nebo obrázek. Může mít různé možnosti jako font, pozadí, popředí atd, viz https://tkinter.py.cz/tkinter_labels.html

Syntaxe:

```
w = Label(master, option=value)
```

Parametr: master reprezentuje rodičovské okno

Příklad:

```
from tkinter import *

root = Tk()
w = Label(root, text='Ahoj', font=("Arial", 12), justify = RIGHT, padx = 2, fg=
"light green", bg="dark green")
w.pack()
root.mainloop()
```

Parametr "justify" lze použít pro zarovnání textu vlevo, vpravo, na střed (LEFT, RIGHT či CENTER). Parametr "padx" použijeme pro přidání dodatečné horizontální výstelky kolem textového popisku. Implicitní výstelka

(padding) je 1 pixel. Pro vertikální výstelku existuje parametr "pady". Pro barevnou změnu textu použijeme u popisku atribut "fg" a pro barevnou změnu pozadí použijeme atribut "bg".

2. Button (Tlačítko)

Klikatelné tlačítko, které může spustit akci, viz https://tkinter.py.cz/tkinter_buttons.html

Syntaxe:

```
w = Button(master, option=value)
```

Příklad 1:

```
import tkinter as tk

r = tk.Tk()
r.title('Počítání sekund')
button = tk.Button(r, text='Stop', width=25, command=r.destroy)
button.pack()
r.mainloop()
```

Příklad 2:

PocitaniSekund.py

```
import tkinter as tk

citac = 0
bezi = True

def pocitej():
    global citac
    if bezi:
        citac += 1
        label.config(text=str(citac)) # aktualni stav čítače
        root.after(1000, pocitej) # sleep 1000 ms, aby GUI nezamrzlo

def zastav():
    global bezi
    bezi = False
    label.config(text=f"Výsledek: {citac} s")

root = tk.Tk()
root.title("Počítání sekund")

label = tk.Label(root, text="0", font=("Arial", 16))
label.pack(pady=10)
```

```
tlacitko = tk.Button(root, text="Stop", width=20, command=zastav)
tlacitko.pack()

pocitej()
root.mainloop()
```

Proměnné *citac* a *bezi* používáme je jako globální, aby k nim měly přístup funkce. Jinak bychom museli použít třídy (class).

`root.after(1000, pocitej)` → po 1000 ms (1 s) se funkce zavolá znovu (Tkinter alternativa k sleep, která nezamrzne GUI)

Úkol - odhad 10 sekund

PocitaniSekund10.py

Změňte kód tak, aby se čítač ukázal jen na první dvě sekundy a poté zmizel.

```
        if citac <= 3:
            label.config(text=str(citac))
        else:
            label.config(text="")  # potom zmizí
```

3. Entry (Vstupní pole)

Používá se k zadávání jednořádkového textového vstupu od uživatele. Pro víceřádkový textový vstup se používá widget Text, viz https://tkinter.py.cz/tkinter_entry_widgets.html

Syntaxe:

```
w = Entry(master, option=value)
```

Příklad:

Entry1.py

```
from tkinter import *

master = Tk()
Label(master, text='Křestní jméno').grid(row=0)
Label(master, text='Příjmení').grid(row=1)

e1 = Entry(master)
e2 = Entry(master)
e1.grid(row=0, column=1)
e2.grid(row=1, column=1)
```

```
mainloop()
```

Úkol - načtení vstupního pole

Přidejte do GUI tlačítko *Uložit*, které metodou `get` načte Entry *e1*, *e2* a zobrazí jako Label *vystup*.

Entry2.py

```
def uloz():
    jmeno = e1.get()
    prijmeni = e2.get()
    vystup.config(text=f"Jméno: {jmeno} {prijmeni}")

# Nové tlačítko - je třeba doplnit souřadnice v mřížce
Button(master, text="Uložit", command=uloz)
```

4. Checkbutton (Zaškrtávací políčko)

Zaškrtávací políčko může být zapnuto nebo vypnuto. Může být spojeno s proměnnou pro uložení svého stavu, viz https://tkinter.py.cz/tkinter_checkboxes.html

Syntaxe:

```
w = Checkbutton(master, option=value)
```

Příklad:

```
from tkinter import *

master = Tk()
var1 = IntVar()
Checkbutton(master, text='muž', variable=var1).grid(row=0, sticky=W)
var2 = IntVar()
Checkbutton(master, text='žena', variable=var2).grid(row=1, sticky=W)

mainloop()
```

Metoda grid() slouží k rozmisťování prvků do tabulky (mřížky), parametr sticky říká, ke které straně buňky se má widget „přilepit“. w - west,

> Checkbutton má parametr command, který se zavolá při každém kliknutí, podobně jako pro widget *Button*.

Úkol - zobrazit volbu

Přidejte do kódu tlačítko pro zobrazení volby a label pro výpis. Checkbutton.py

5. Radiobutton (Přepínač)

Umožňuje uživateli vybrat jednu možnost ze sady voleb. Jsou seskupeny sdílením stejné proměnné.

Syntaxe:

```
w = Radiobutton(master, option=value)
```

Příklad:

```
from tkinter import *

root = Tk()
v = IntVar()  # Radiobuttony musi sdilet jednu promennou, jinak o sobe nevi,
IntVar() jen jednou
Radiobutton(root, text='Motol', variable=v, value=1).pack(anchor=W)
Radiobutton(root, text='MIT', variable=v, value=2).pack(anchor=W)

mainloop()
```

*__Úkol:__ Přepište kód pro volbu pohlaví pro použití 'radiobutton'. Radiobutton.py

Message (Zpráva)

Je to widget pro zobrazení textových zpráv se zalamováním slov.

Syntaxe:

```
w = Message(master, option=value)
```

Příklad:

```
from tkinter import *

main = Tk()
ourMessage = 'Toto je naše zpráva'
messageVar = Message(main, text=ourMessage)
messageVar.config(bg='lightgreen')
messageVar.pack()

main.mainloop()
```

Text

Pro úpravu víceřádkového textu a formátování způsobu, jakým má být zobrazen.

Syntaxe:

```
w = Text(master, option=value)
```

Příklad:

```
from tkinter import *

root = Tk()
T = Text(root, height=2, width=30)
T.pack()
T.insert(END, 'GeeksforGeeks\nNEJLEPŠÍ WEBOVÁ STRÁNKA\n')

mainloop()
```

Canvas (Plátno)

Používá se ke kreslení obrázků a dalších složitých layoutů jako grafika, text a widgety, viz https://tkinter.py.cz/tkinter_canvas.html

Syntaxe:

```
w = Canvas(master, option=value)
```

Příklad: Canvas1.py

```
from tkinter import *

master = Tk()
w = Canvas(master, width=40, height=60)
w.pack()

canvas_height = 20
canvas_width = 200
y = int(canvas_height / 2)
w.create_line(0, y, canvas_width, y)

mainloop()
```

Barevné možnosti v Tkinter

Tento příklad demonstruje použití různých barevných možností ve widgetech Tkinter, včetně aktivního pozadí a barvy popředí, barev pozadí a popředí, barev zakázaného stavu a barev výběru.

Příklad:

```
import tkinter as tk

root = tk.Tk()
root.title("Barevné možnosti v Tkinter")

# Vytvoření tlačítka s aktivním pozadím a barvami popředí
button = tk.Button(root, text="Klikni na mě", activebackground="blue",
activeforeground="white")
button.pack()

# Vytvoření popisku s barvami pozadí a popředí
label = tk.Label(root, text="Ahoj, Tkinter!", bg="lightgray", fg="black")
label.pack()

# Vytvoření widgetu Entry s barvami výběru
entry = tk.Entry(root, selectbackground="lightblue", selectforeground="black")
entry.pack()

root.mainloop()
```

Správa geometrie

Tkinter také nabízí přístup ke geometrické konfiguraci widgetů, která může organizovat widgety v rodičovských oknech. Existují hlavně tři třídy geometry managerů. Nejjednodušší je metoda *pack*, doporučovaná a nejvšestrannější je metoda *grid*.

Podrobné ukázky: https://tkinter.py.cz/tkinter_layout_management.html

pack() metoda

Organizuje widgety v blocích před umístěním do rodičovského widgetu. Widgety mohou být zabaleny shora, zdola, zleva nebo zprava. Může rozšířit widgety tak, aby vyplnily dostupný prostor nebo je umístit v pevné velikosti.

Příklad:

```
import tkinter as tk

root = tk.Tk()
root.title("Příklad Pack")

# Vytvoření tří tlačítek
button1 = tk.Button(root, text="Tlačítko 1")
button2 = tk.Button(root, text="Tlačítko 2")
button3 = tk.Button(root, text="Tlačítko 3")
```

```
# Zabalení tlačítek vertikálně
button1.pack()
button2.pack()
button3.pack()

root.mainloop()
```

grid() metoda

Organizuje widgety v mřížce (struktura podobná tabulce) před umístěním do rodičovského widgetu. Každému widgetu je přiřazen řádek a sloupec. Widgety mohou pokrývat více řádků nebo sloupců pomocí rowspan a columnspan.

Příklad: Grid1.py

```
import tkinter as tk

root = tk.Tk()
root.title("Příklad Grid")

# Vytvoření tří popisků
label1 = tk.Label(root, text="Popisek 1")
label2 = tk.Label(root, text="Popisek 2")
label3 = tk.Label(root, text="Popisek 3")

# Uspořádání popisků v mřížce 2x2
label1.grid(row=0, column=0)
label2.grid(row=0, column=1)
label3.grid(row=1, column=0, columnspan=2)

root.mainloop()
```

Úkol - vysvětlete kód

Jak bude vypadat GUI? Grid2.py

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

for i in range(1, 10):
    lbl = tk.Label(root, text=i, width=5)
    lbl.grid(row=(i - 1) // 3, column=(i - 1) % 3)

root.mainloop()
```

Grid3.py

```
from tkinter import *
from random import sample

colours = ['red','green','orange','white','yellow','blue']
coloursshuffled = sample(colours,len(colours))

for r in range(len(colours)):
    Label(text=colours[r], relief=RIDGE, width=15).grid(row=r,column=0)
    Entry(bg=coloursshuffled[r], relief=SUNKEN,width=10).grid(row=r,column=1)

mainloop()
```

Testováví znalosti barev Grid3b.py

place() metoda

Organizuje widgety jejich umístěním na konkrétní pozice určené programátorem. Widgety jsou umístěny na konkrétní souřadnice x a y. Velikosti a pozice mohou být specifikovány v absolutních nebo relativních termínech.

Příklad:

```
import tkinter as tk

root = tk.Tk()
root.title("Příklad Place")

# Vytvoření popisku
label = tk.Label(root, text="Popisek")

# Umístění popisku na konkrétní souřadnice
label.place(x=50, y=50)

root.mainloop()
```

Zpracování událostí v Tkinter

V Tkinter jsou události akce, které nastávají, když uživatel interaguje s GUI, jako je stisknutí klávesy, kliknutí myší nebo změna velikosti okna. Zpracování událostí nám umožňuje definovat, jak by naše aplikace měla reagovat na tyto interakce, seznam všech událostí s podrobným popisem pro pokročilejší uživatele na https://tkinter.py.cz/tkinter_events_binds.html

Události a vazby

Události v Tkinter jsou zachyceny a spravovány pomocí mechanismu nazývaného vazby (bindings). Vazba propojuje událost s callback funkcí (také známou jako handler události), která je volána, když nastane událost.

U widgetů Button, Checkbutton, Radiobutton, Scale a Spinbox lze tuto funkci připojit jako hodnotu parametru command (command="callback"). Callback může být buď nativní nebo uživatelsky definovaná funkce (která nepřijímá žádné argumenty), metoda, výraz lambda či prostý argument.

Syntaxe:

```
widget.bind(event, handler)

# například
tlacitko.bind('<Button-1>', levy_klik) # při kliknutí levou myší na widget
tlacitko se zavolá funkce levý klik

tlacitko.bind('<Double-1>', dvojklik) # při dvojkliku levou myší na widget
tlacitko se zavolá funkce dvojklik
```

- `widget`: Widget Tkinter, ke kterému chcete událost navázat
- `event`: Řetězec, který specifikuje typ události (např. `<Button-1>` pro levé kliknutí myší)
- `handler`: Callback funkce, která bude provedena, když nastane událost, funkce musí být definovaná se vstupním parametrem event()

Události klávesnice a myši

Události klávesnice jsou spouštěny, když uživatel stiskne klávesu na klávesnici. Události myši jsou spouštěny akcemi myši, jako je kliknutí nebo pohyb myši.

Příklad: UdalostiBind.py

```
import tkinter as tk

def on_key_press(event):
    print(f"Stisknuta klávesa: {event.keysym}")

def on_left_click(event):
    print(f"Levé kliknutí na ({event.x}, {event.y})")

def on_right_click(event):
    print(f"Pravé kliknutí na ({event.x}, {event.y})")

def on_mouse_motion(event):
    print(f"Myš přesunuta na ({event.x}, {event.y})")

root = tk.Tk()
root.title("Pokročilý příklad zpracování událostí")

root.bind("<KeyPress>", on_key_press)
root.bind("<Button-1>", on_left_click)
root.bind("<Button-3>", on_right_click)
root.bind("<Motion>", on_mouse_motion)

root.mainloop()
```

V tomto pokročilém příkladu je zpracováno více typů událostí současně. Funkce `on_mouse_motion` je volána vždy, když je myš přesunuta v okně, což demonstruje, jak můžeme sledovat a reagovat na kontinuální události.

Objekt události

Objekt události je předán callback funkci, když nastane událost. Obsahuje užitečné informace o události:

- `event.keysym`: Symbol klávesy (např. 'a', 'Enter')
- `event.x` a `event.y`: Souřadnice x a y události myši
- `event.widget`: Widget, který spustil událost

Například callback funkce, která vypiše, jakou klávesu uživatel stisknul:

```
def klavesnice(event):
    label.config(text=f"Stiskl jsi klávesu: {event.keysym}")
root.bind("<KeyPress>", klavesnice)
```

📋 Zadání pro žáky

UdalostiReseni.py

Vytvořte program, který Zobrazí velké tlačítko s nápisem 👉 NEMAČKEJ MĚ Po každém kliknutí myší se text změní (např. „Říkal jsem NEMAČKEJ MĚ!“), zvýší se počitadlo kliknutí Po 5 kliknutích se text změní na: 👉 „Tak dost!!! 😠“

Prozkoumejte více

- https://howto.py.cz/tkinter.html
- programujte.com
- python.cz
- An Introduction to Tkinter, F. Lundh - podrobná kniha v angličtině, 156 stránek