Tkinter

Първа част

д-р Филип Андонов

23 юни 2022 г.

Съдържание

Tkinter

Tkinter е стандартен Python интерфейс към пакета за графични инструменти Tcl/Tk.
Tk и tkinter са налични на повечето Unix платформи, включително macOS, а дори и под Windows.
Можем да изпълним python[3] -m tkinter от командния ред за да се уверим, че той е инсталиран на системата и да видим версията му.
Tkinter поддържа различни версии на Tcl/Tk с или без поддръжка на нишки.

Tkinter

TKinter не е просто обвивка върху Tcl/Tk, а прави употребата им по по-питонски начин.

Разширения на Tk

През годините Tk е допълван с множество разширения, например с нови компоменти, които не са налични в ядрото. Някои от известните разширения са BLT, Tix, iWidgets и BWidgets.

Тъй като тези разширения са създадени отдавна, те може да не отразяват текущите конвенции в Tk, въпреки че най-вероятно ще работят.

Инсталиране

Повечето Linux дистрибуции включват версия на Python 3 по подразбиране. След като инсталирате Python трябва да се уверите, че Tkinter е инсталиран коректно. Имайте предвид, че в някои дистрибуции, интерпретаторът на Python 3 се казва python3, а не просто python. Ако получите грешка при внасяне на модула tkinter с import, трябва да проверите инсталацията и дали не се налага качването на допълнителни пакети.

Инсталиране - пакетен мениджър

На практика всички Linux дистрибуции имат Tcl/Tk в своите хранилища за пакети и могат да бъдат инсталирани. Например за Ubuntu базираните диструбиции можете да използвате apt с команда от типа на:
apt install tk8.6

Hello world

```
1 from tkinter import *
2 from tkinter import ttk
3 root = Tk()
4 ttk.Button(root, text="Hello World").grid()
5 root.mainloop()
```

Hello world

```
from tkinter import *
from tkinter import ttk
```

Тези два реда указват, от които се нуждае нашата Python програма. Първият модул, tkinter, е стандартното обвързване с Tk. Когато бъде внесен, той зарежда библиотеката Tk, инсталирана на системата. Модулът ttk е под-модул на tkinter. Той дава достъп до новите компоненти, налични от версия Tk 8.5.

Hello world

```
root = Tk()
root.title("Feet to Meters")
```

С тези два реда установяваме основния прозорец на рпиложението и му даваме заглавие "Feet to Meters."

Hello world

Първият компонент, който създаваме, е за въвеждане на стойността във футове, която искаме да конвертираме.

```
feet_entry = ttk.Entry(mainframe, width=7)
feet_entry.grid(column=2, row=1, sticky=(W, E))
```

Hello world

Трябва да направим две неща: да създадем визуалния компонент и да го покажем на екрана. Създаването на компонента изисква да укажем родителя му. Тук задаваме рамката за съдържание. Всички компоненти ще бъдат нейни наследници. Родителят се подава като първи аргумент на конструктора. При създаването можем по желание да направим някои настройки. Тук казваме, че искаме текстовото поле да е дълго 7 символа.
Позиционирането на визуалните компоненти става с помощта на grid. Ние указваме кой компопент в кой ред и колона на мрежата (grid) да отиде. Параметърът sticky указва как компонента да се подравни вътре в клетката, използвайки посоки на света. Стойност W означава закрепяне към лявата страна на клетката, (W, E) означава и към лявата и към дясната страна и т.н.

Hello world

Правим същото с останалите компоненти: етикет, показващ резултата, бутон "Calculate"и три статични етикета.

```
meters = StringVar()
ttk.Label(mainframe, textvariable=meters).grid(
    column=2, row=2, sticky=(W, E))

ttk.Button(mainframe, text="Calculate", command=
    calculate).grid(column=3, row=3, sticky=W)

ttk.Label(mainframe, text="feet").grid(column=3,
    row=1, sticky=W)
ttk.Label(mainframe, text="is equivalent to").
    grid(column=1, row=2, sticky=E)
ttk.Label(mainframe, text="meters").grid(column
    =3, row=2, sticky=W)
```

Довършителни работи по интерфейса:

```
for child in mainframe.winfo_children():
    child.grid_configure(padx=5, pady=5)
feet_entry.focus()
root.bind("<Return>", calculate)
```

Tkinter - следваща стъпка

Извършването на изчислението се намира в процедура, която се извиква когато се натисне бутона Calculate.

```
def calculate(*args):
    try:
        value = float(feet_entry.get())
        meters.set(int(0.3048 * value * 10000.0 +
            0.5)/10000.0)
    except ValueError:
        pass
```

Tkinter - следваща стъпка

Накрая казваме на Tk да стартира цикъла за проверка на събития, който притежава всяко GUI приложение.

```
root.mainloop()
```

Компоненти

Визуалните компоненти са всичко, което виждаме на екрана. В горния пример те бяха бутон, поле за въвеждане, етикети и рамка.

Компонент	Описание
Label	Етикет, представящ един ред описателен текст/изображение
Button	Обикновен бутон
Entry	поле за въвеждане на текст от потребителя
Menu	предоставя набор от команди на потребителя. Командите се съдържат в Menubutton
Checkbutton	Показва набор от възможни варианти като отметки. Потребителят може да избира множество варианти едновременно
Frame	Контейнер, който организира други компоненти

Компонент	Описание
Listbox	Предоставя списък от възможности на потребителя
Menubutton	Показва меню
Message	многоредов текст за въвеждане от потребителя
Radiobutton	Предоставя набор от взаимно изключващи се възможности на потребителя
Scale	компонент за плъзгач
Text	Многоредов текст за визуализиране

Таблица: Визуални компоненти

Компоненти

Всички компоненти имат достъп до специфични методи на геометричния мениджър, за да се организира разположението им в родителския компонент. Методите, които се предоставят са:

Метод	Описание
pack()	мениджърът организира компонентите в блокове преди да ги постави в родителския компонент
grid()	мениджърът организира компонентите в таблица
place()	Мениджърът показва компонентите на зададени координати

Таблица: Визуални компоненти

Работа с визуални компоненти

Първо трябва да установим кой компонент ни трябва. После да определим родителя. В Tk всички компоненти са част от йерархията на прозорците, с единичен корен най-отгоре.

Работа с визуални компоненти

Всеки компонент е отделен Python обект. Когато правим инстанция на компонент, казваме кой му е родителя. Единството изключение е корена, т.е. прозорецът от най-високо ниво.

```
root = Tk()
content = ttk.Frame(root)
button = ttk.Button(content)
```

Можем да запазваме референция към даден компонент в променлива или не, в зависимост от това дали ще ни трябва по-късно да се обръщаме към него. Тъй като е част от йерархията, автоматичното управление на паметта няма да го изтрие, дори да не пазим референция към него.

Основни компоненти

Нека разгледаме някои от основните компоненти, които ни позволяват да създаваме прост интерфейс.

Frame

Правоъгълник, който се визуализира и има за цел да организира интерфейса като групира компонентите. Често се използва като главен компонент за мениджъра на геометрия, за да разположи компонентите например в grid. Класът е ttk.Frame:

```
frame = ttk.Frame(parent)
```

Padding

Атрибутът padding се използва за задаване на допълнително разстояние около вътрешността на компонента. Могат да се определят разстоянията както за всички страни, така и за хоризонтални и вертикални, както и за всяка поотделно.

```
frame['padding'] = 5   # 5 pixels on all sides
frame['padding'] = (5,10)  # 5 on left and right,
    10 on top and bottom
frame['padding'] = (5,7,10,12) # left: 5, top: 7,
    right: 10, bottom: 12
```

Borders

Границите се използват за да разделим визуално компонента от това, което го заобикаля. Често се използва за да направи интерфейса да изглежда потънал или изпъкнал. За целта се определя ширината на границата (която по подразбиране е 0) и типа на релефа. Възможните стойности са flat(по подразбиране), raised, sunken, solid, ridge или groove.

```
frame['borderwidth'] = 2
frame['relief'] = 'sunken'
```

Основни компоненти - Frame

Често използвани настройки на Frame:

Настройка	Описание
bg	определя цвета на фона.
bd	размера на рамката около индикатора.
cursor	Определя вида на курсора когато е върху frame-а.
height	Височината
highlightcolor	Определя цвета, когато frame-ът е на фокус
relief	Типът на релефа.
width	Ширината на рамката.

Таблица: Настройки на Frame

Label

Компонент, който показва текст или изображение, който потребителите виждат, но не могат да взаимодействат с него. Използва се за означаване на смисъла на други компоненти, за текстови описание и др. Представляват инстанции на класа ttk.Label.

```
label = ttk.Label(parent, text='Full name:')
```

Основни компоненти - Label

Показване на текст

Текстът е най-често употребявания атрибут на етикета. Той може да се промени по всяко време програмно, просто не директно от потребителя. Можем да направим така, че компонента да следа стойността на някаква променлива. Всеки път когато стойността се промени, етикетът ще покаже новата стойност. Това се постига с настройката textvariable. Tkinter позволява свързването на компоненти само към инстанции на класа StringVar, а не към произволни променливи в Python.

```
resultsContents = StringVar()
label['textvariable'] = resultsContents
resultsContents.set('New value to display')
```

Показване на изображения

Етикетите могат да показват и изображения вместо текст. За целта първо трябва да създадем обект от тип изображение, за зареждане на изображението в паметта. После да конфигурираме етикета да го показва.

```
image = PhotoImage(file='myimage.gif')
label['image'] = image
```

Основни компоненти - Label

Етикетите могат да показват едновременно и текст и изображение. За целта трябва да се използва настройката compound. Стойността по подразбиране е none, което означава да се показва само изображението ако е налично, в противен случай текста. Другите възможни стойности са text, image, center, top, left, bottom, and right.

Основни компоненти - Label

Шрифтове и цветове Не се счита за добър стил шрифтовете и цветовете да се сменят директно. Вместо това се препоръчва използването на стилове. Въпреки това, понякога се налага употребата на специфичен шрифт за даден етикет.
Установяването на шрифт става с настройката font.

Основни компоненти - Label

Шрифт	Описание
TkDefaultFont	Стандартният шрифт за всички компоненти, за които не е установен друг
TkTextFont	Използва се от компоненти за въвеждане
TkFixedFont	Стандартен шрифт с фиксирана ширина
TkMenuFont	Използва се за менюта
TkHeadingFont	Използва се за заглавия в списъци и таблици
TkCaptionFont	Използва се за заглавните ленти на диалогови прозорци
TkIconFont	Използва се за надписи на икони
TkTooltipFont	Използва се за подсказки

Таблица: Визуални компоненти

```
label['font'] = "TkDefaultFont"
```

Цветовете на шрифта и на фона също могат да се променят с помощта на настройките foreground и background. Могат да се използват както имена (например red) така и RGB стойности в шестнайсетичен вид (#ff340a).

Основни компоненти - Label

Разположението се определя от мениджъра на геометрии. Въпреки това могат да се зададат няколко настройки, указвайки как се разполага етикетът в рамките на пространството, зададено му от мениджъра. Ако пространството е по-голямо от необходимото на етикета, можем да използваме котва за да определим към кой край да се равни етикета. Възможните стойности са n, ne, e, se, s, sw, w, nw или center.

Основни компоненти - Label

Етикетите могат да показват повече от един ред текст. За целта текста трябва да съдържа символа за нов ред (
n). Етикетите могат автоматично да пренасят на нов ред с настройката wraplength, която определя максималната дължина на ред (в пиксели, сантиметри и т.н.).
Можем също да определяме подравняването на текста с настройката justify. Възможните стойности са left, center или right. Ако текстът е от един ред, по-добре да се използва котва.

Основни компоненти - Button

Button

Бутонът е компонент за взаимодействие с потребителя. Също като етикетите, той може да съдържа текст или изображение. Бутоните са инстанции на класа ttk.Button. Обикновено съдъранието на бутона и функцията, която се извиква при натискането им, се определят по време на създаването.

```
button = ttk.Button(parent, text='Okay', command=
    submitForm)
```

Основни компоненти - Button

Бутоните имат същите настройки за текст, изображение, и комбинация от двете както и етикетите.
Бутоните имат конфигурационна настройка default. Ако тя е активна, Tk приема бутонът за бутон по подразбиране в интерфейса, в противен случай е обикновен. Ако е по подразбиране може когато потребителят натисне Enter, той да се активира. На някои платформи той видимо ще се различава от другите бутони. Това, че бутонът е по подразбиране не го обвързва автоматично с Enter, това трябва да се направи ръчно.

```
action = ttk.Button(root, text="Action", default=
    "active", command=myaction)
root.bind('<Return>', lambda e: action.invoke())
```

Състояния

Бутоните започват в нормално състояние. Те реагират на движения с мишката, могат да бъдат натискани и да извикват асоцииранат с тях функция. Бутоните могат да бъдат и в изключено състояние, когато са посивени, не реагират на движения с мишката и не могат да бъдат натискани.
Всичко компоненти имат вътрешно състояние, което се състои от серия от двоични флагове. Те могат да се променят и проверяват с помощта на методите state и instate.

Състояния

```
b.state(['disabled'])    # set the disabled flag
b.state(['!disabled'])   # clear the disabled flag
b.instate(['disabled']) # true if disabled, else
    false
b.instate(['!disabled'])# true if not disabled,
    else false
b.instate(['!disabled'], cmd)# execute 'cmd' if
    not disabled
```

Основни компоненти - Button

Пълният набор от флагове за компонентите, които се адаптират към темата, са:

- active
- disabled
- focus
- pressed
- selected
- background
- readonly
- alternate
- invalid

Checkbutton

Бутон с две стабилни състояния. Когато бъде натиснат, той променя състоянието си и извиква функция. Често се използват за включване и изключване на дадена опция.

```
Checkbutton1 = IntVar()
Button1 = Checkbutton(root, text = "Tutorial",
                      variable = Checkbutton1,
                      onvalue = 1,
                      offvalue = 0,
                      height = 2,
                      width = 10)
```

Основни компоненти - Checkbutton

Отметките (Checkbuttons) са инстанции на класа ttk.Checkbutton.

```
measureSystem = StringVar()
check = ttk.Checkbutton(parent, text='Metric',
   command=metricChanged, variable=measureSystem,
   onvalue='metric', offvalue='imperial')
```

Основни компоненти - Checkbutton

Отметките използват много от същите настройки като обикновените бутони. Настройките text, textvariable, image, compound и command са налични и тук.
За разлика от обикновените бутони имат стойност. Настройката variable е подобна textvariable. Тя се променя всеки път когато се превключи състоянието на отметката. По подразбиране стойностите са 1 и 0, но може да бъдат променени с настройките onvalue и offvalue. Отметката не установява автоматично свързаната променлива. Тя трябва да бъде инициализирана отделно.

Основни компоненти - Checkbutton

Когато свързаната променлива не съдържа нито една от двете стойности, компонентът преминава в неустановено състояние. В такъв случай флагът alternate на състоянието бива вдигнат. Може да се провери с

```
check.instate(['alternate'])
```

Свързани променливи

Освен класа StringVar, Tkinter предоставя други класове за обвързване на различни типове променливи - булеви, целочислени и с плаваща запетая. Можем винаги да използваме StringVar и да конвертираме, но в някои случаи е по-удобно да ползвам друг тип.

```
s = StringVar(value="abc") # default value is ''
b = BooleanVar(value=True) # default is False
i = IntVar(value=10)       # default is 0
d = DoubleVar(value=10.5)  # default is 0.0
```

Radiobutton

Радиобутонът предоставя възможност за избор измежду взаимно изключващи се варианти. За разлика от отметката, вариантите не са ограничени до два.
Радиобутоните винаги се ползват в група, като група от тях е обвързана с единичен избор. Радиобутоните са инстанции на класа ttk.Radiobutton.

```
radioStation = StringVar()
home = ttk.Radiobutton(parent, text='FM+',
    variable=radioStation, value='FM+')
office = ttk.Radiobutton(parent, text='Z-Rock',
    variable=radioStation, value='Z-Rock')
cell = ttk.Radiobutton(parent, text='N-Joy',
    variable=radioStation, value='N-Joy')
```

Основни компоненти - Radiobutton

Радиобутоните имат почти същите настройки като отметките. Една от разликите е, че вместо onvalue и offvalue, те имат само една настройка - value. Всеки радиобутон в множество е свързан с една и съща променлива, но с различна стойност. Когато свързаната променлива има съответната стойност, бутонът е включен. Ако свързаната променлива не съществува, радиобутонът също е в неопределено състояние. За да имаме повече от една група радиобутони, трябва различните групи да са обвързани с различни променливи.

Основни компоненти - Entry

Entry

Този компонент се използва за въвеждане на един ред от потребителя. Компонентите от този тип са инстанции на класа ttk.Entry class.

```
username = StringVar()
name = ttk.Entry(parent, textvariable=username)
```

Основни компоненти - Entry

Конфигурационна настройка, налична за този компонент е ширината на полето, което определя броя на символите. Компонентите за въвеждане на текст също имат вътрешна стойност, достъпна през свързана променлива. Използва се textvariable.

Основни компоненти - Entry

Съдържанието на текстовата стойност може да да се променя и без да се минава през свързаната променлива. Методът get връща текущата стойност, а методите delete и insert променят съдържанието.

```
print('current value is %s' % name.get())
name.delete(0,'end')            # delete between
    two indices, 0-based
name.insert(0, 'your name')     # insert new text
    at a given index
```

Следене за промени

Компонентите от тип Entry нямат настройка command за обратно извикване на функция. За да следим за промени, трябва да следим за промени в свързаната променлива.

```
def it_has_been_written(*args):
    ...
username.trace_add("write", it_has_been_written)
```

Следене за промени

Използвания в предходния пример метод trace_add е част от сложна система за наблюдаване на променливи и функции за обратно повикване при четене, писане или изтриване. Могат да се извикват множество функции за обратно повикване, да се добавят или премахват с trace_remove.
Tkinter позволява наблюдаването за промени на StringVar и подобните му.

Пароли

Полетата за въвеждане могат да се използват за пароли, където същинското съдържание не се показва. За целта трябва да се установи стойност на настройката show.

```
passwd = ttk.Entry(parent, textvariable=password,
    show="*")
```

Валидиране на входа

Когато текстовото поле приема стойности с допълнителни ограничения - телефонен номер, пощенски код, адрес на електронна поща и т.н. можем да валидираме въведеното. Критериите за валидиране се поставят в настройката validatecommand. Това е булева функция, казваща дали въведеното преминава валидацията или не. В примера по-долу валидирането се извършва на всяко натискане на клавиш, тъй като е подадена стойност 'key' на конфигурационната настройка validate.

```
import re
def check_num(newval):
    return re.match('^[0-9]*$', newval) is not
        None and len(newval) <= 5
check_num_wrapper = (root.register(check_num), '%
    P')

num = StringVar()
e = ttk.Entry(root, textvariable=num, validate='
    key', validatecommand=check_num_wrapper)
e.grid(column=0, row=0, sticky='we')
```

Основни компоненти - Entry

Тук се използва процентна субституция(%P), която съдържа новата стойност ако промяната е валидна. Методът register (който може да се извиква върху всеки компонент, не само root), създава Tcl процедура, която ще извика нашата Python функция. Процентната субституция ще бъде подадена на нейните параметри.

Combobox

Компонент, който комбинира поле за въвеждане и падащ списък. Това позволява на потребителя да избира от предварително зададен набор от стойности, но също така да въвежда и своя стойност. Комбинираните полета са инстанции на класа ttk.Combobox.

```
countryvar = StringVar()
country = ttk.Combobox(parent, textvariable=
    countryvar)
```

Основни компоненти - Combobox

Списъкът с предварително зададени стойности се подава на конфигурационната настройка values.

```
country['values'] = ('USA', 'Canada', 'France')
```

Основни компоненти - Combobox

Подобно на текстовите полета за въвеждане, настройката textvariable свързва променлива с текущата стойност на комбинираното поле. Отново свързаната променлива трябва да бъде инициализирана в кода.
Комбинираното поле ще генерира виртуално събитие «ComboboxSelected» към което може да се свържем за когато стойността се промени. Това събитие не се поражда при писане в текстовото поле, само при избор от списъка.

```
country.bind('<<ComboboxSelected>>', function)
```

Основни компоненти - Combobox

За да се ограничи избора само до предварително зададения набор от стойности се използва състоянието readlonly.

```
country.state(["readonly"])
```

Основни компоненти - Combobox

За да получим текущата стойност на комбинираното поле, независимо дали е въведено от потребителя или избрано от предварително дефинирания списък, можем да използваме метода get. За да променим стойността, използваме метода set. За да установим индекса на избрания елемент, можем да използваме метода current без аргументи.

Задачи

Да се направи графично приложение на Tkinter за изчисляване на BMI. Потребителят въвежда височината си в метри и теглото си в килограми и индексът на телесна маса се изчислява като теглото се раздели на повдигнатата на квадрат височина в метри. Тълкуването на стойностите може да се намери в Интернет.
Да се направи графични приложение за игра на бесеница.

Благодаря за вниманието!

Въпроси?

References

Backup Slide

This is a backup slide, useful to include additional materials to answer questions from the audience.

The package appendixnumberbeamer is used to refrain from numbering appendix slides.