Бесплатная электронная книга

учусь tkinter

Free unaffiliated eBook created from
Stack Overflow contributors.

#tkinter

Около

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: tkinter

It is an unofficial and free tkinter ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official tkinter.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

глава 1: Начало работы с tkinter

замечания

Tkinter (« **Tk Inter** face») представляет собой стандартный кросс-платформенный пакет python для создания графических пользовательских интерфейсов (GUI). Он обеспечивает доступ к базовому интерпретатору Tcl с помощью инструментария Tk, который сам является межплатформенной многоязычной графической библиотекой пользовательского интерфейса.

Tkinter - не единственная библиотека графического интерфейса для python, но она входит в стандартную комплектацию. Дополнительные библиотеки GUI, которые можно использовать с python, включают wxPython , PyQt и kivy .

Величайшая сила Ткинтера - его вездесущность и простота. Он работает из коробки на большинстве платформ (linux, OSX, Windows) и поставляется с широким спектром виджетов, необходимых для большинства распространенных задач (кнопок, меток, холста для рисования, многострочного текста и т. Д.).

В качестве обучающего инструмента tkinter обладает некоторыми функциями, которые уникальны среди наборов графических интерфейсов, таких как именованные шрифты, теги привязки и трассировка переменных.

Различия между python 2 и 3

Tkinter практически не изменился между python 2 и python 3, причем основное отличие заключалось в том, что пакет tkinter и модули были переименованы.

Импорт в python 2.x

В python 2.x пакет tkinter называется `Tkinter` , а связанные пакеты имеют свои собственные имена. Например, следующий пример показывает типичный набор операторов импорта для python 2.x:

```
import Tkinter as tk
import tkFileDialog as filedialog
import ttk
```

Импорт в python 3.x

Хотя функциональность не сильно изменилась между python 2 и 3, имена всех модулей

tkinter изменились. Ниже приведен типичный набор операторов импорта для python 3.x:

```
import tkinter as tk
from tkinter import filedialog
from tkinter import ttk
```

Дальнейшее чтение

- Вопросы Tkinter по Stackoverflow
- Официальная документация Pton 3 tkinter
- Официальная документация Pton 2 tkinter
- Tkdocs.com - многоплатформенная документация по tk
- Введение Effbot в tkinter
- Справочное руководство Tkinter, New Mexico Tech

Версии

Tcl

Версия	Дата выхода
8,6	2016-07-27
8,5	2016-02-12
8,4	2013-06-01
8,3	2002-10-18
8,2	1999-12-16
8,1	1999-05-26
8,0	1999-03-09

питон

Версия	Дата выхода
3,6	2016-12-23
3,5	2015-09-13
3,4	2014-03-17

Версия	Дата выхода
3,3	2012-09-29
3,2	2011-02-20
3,1	2009-06-26
3.0	2008-12-03
2,7	2010-07-03
2,6	2008-10-02
2.5	2006-09-19
2,4	2004-11-30
2,3	2003-07-29
2,2	2001-12-21
2,1	2001-04-15
2,0	2000-10-16

Examples

Установка или настройка

Tkinter поставляется с предустановленными двоичными файлами установщика Python для Mac OS X и платформы Windows. Поэтому, если вы устанавливаете Python из официальных двоичных файлов для платформы Mac OS X или Windows, вы можете пойти с Tkinter.

Для версий Linux Debian вы должны установить его вручную, используя следующие команды.

Для Python 3

```
sudo apt-get install python3-tk
```

Для Python 2.7

```
sudo apt-get install python-tk
```

Linux-дистрибутивы с установкой yum могут установить модуль tkinter с помощью команды:

```
yum install tkinter
```

Проверка установки

Чтобы проверить, успешно ли вы установили Tkinter, откройте консоль Python и введите следующую команду:

```
import tkinter as tk # for Python 3 version
```

или же

```
import Tkinter as tk # for Python 2.x version
```

Вы успешно установили Tkinter, если приведенная выше команда выполняется без ошибок.

Чтобы проверить версию Tkinter, введите следующие команды в Python REPL:

Для python 3.X

```
import tkinter as tk
tk._test()
```

Для python 2.X

```
import Tkinter as tk
tk._test()
```

Примечание. Импорт `Tkinter as tk` не требуется, но является хорошей практикой, поскольку он помогает поддерживать совместимость между версией.

Привет, мир! (Минимальный)

Давайте проверим наши базовые знания о tkinter, создав классический «Hello, World!». программа.

Во-первых, мы должны импортировать tkinter, это будет зависеть от версии (см. Раздел замечаний «Различия между Python 2 и 3»)

В Python 3 модуль `tkinter` имеет строчную букву t:

```
import tkinter as tk
```

В Python 2 модуль `Tkinter` имеет прописную букву T:

```
import Tkinter as tk
```

Использование `as tk` не является строго необходимым, но мы будем использовать его, поэтому остальная часть этого примера будет работать одинаково для обеих версий.

теперь, когда мы импортируем модуль tkinter, мы можем создать корень нашего приложения, используя класс `Tk` :

```
root = tk.Tk()
```

Это будет действовать как окно для нашего приложения. (обратите внимание, что *дополнительные* окна должны быть экземплярами `Toplevel`)

Теперь, когда у нас есть окно, давайте добавим к нему текст с помощью `Label`

```
label = tk.Label(root, text="Hello World!") # Create a text label
label.pack(padx=20, pady=20) # Pack it into the window
```

После того, как приложение будет готово, мы можем запустить его (введите *основной цикл* событий) с `mainloop` метода `mainloop`

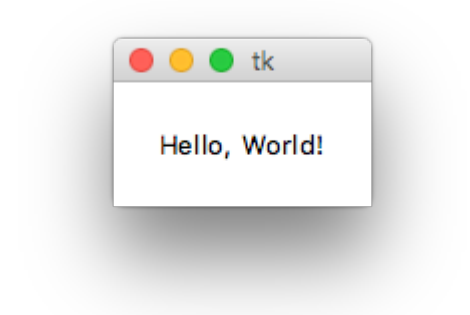
```
root.mainloop()
```

Это откроет и запустит приложение до тех пор, пока оно не будет остановлено закрывающимся окном или вызовом выхода из функций обратного вызова (см. `root.destroy()`), таких как `root.destroy()` .

Объединяя все это:

```
import tkinter as tk # Python 3.x Version
#import Tkinter as tk # Python 2.x Version

root = tk.Tk()

label = tk.Label(root, text="Hello World!") # Create a text label
label.pack(padx=20, pady=20) # Pack it into the window

root.mainloop()
```

И что-то вроде этого должно появиться:



Привет, мир! (модульная, объектно-ориентированная)

```
import tkinter as tk

class HelloWorld(tk.Frame):
    def __init__(self, parent):
        super(HelloWorld, self).__init__(parent)

        self.label = tk.Label(self, text="Hello, World!")
        self.label.pack(padx=20, pady=20)

if __name__ == "__main__":
    root = tk.Tk()

    main = HelloWorld(root)
    main.pack(fill="both", expand=True)

    root.mainloop()
```

Примечание. Наследование можно получить только из любого виджета tkinter, включая корневое окно. Наследование от `tkinter.Frame` является, по крайней мере, самым гибким в том, что он поддерживает несколько интерфейсов документов (MDI), `tkinter.Frame` интерфейсы (SDI), одностраничные приложения и многостраничные приложения.

Прочитайте Начало работы с tkinter онлайн: https://riptutorial.com/ru/tkinter/topic/987/начало-работы-с-tkinter

глава 2: Tkinter Geometry Managers

Вступление

Для размещения виджетов есть три менеджера геометрии: `pack()` , `grid()` и `place()` .

Examples

упаковка ()

Менеджер `pack()` организует виджеты в блоках перед их размещением в родительском виджете. Он использует параметры `fill` , `expand` и `side` .

Синтаксис

```
widget.pack(option)
```

заполнить
Определяет, поддерживает ли виджет необходимое минимальное пространство или занимает дополнительное пространство, выделенное для него. Атрибуты: NONE (по умолчанию), X (заполнение по горизонтали), Y (заполнение по вертикали) или BOTH (заполнение как по горизонтали, так и по вертикали).

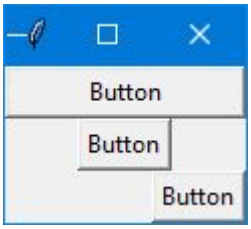
расширять
Когда установлено значение YES, виджет расширяется, чтобы заполнить любое пространство, которое не используется в родительском элементе виджета. Атрибуты: ДА, НЕТ.

Боковая сторона
Определяет, на какую сторону родителя виджета он упаковывает. Атрибуты: TOP (по умолчанию), BOTTOM, LEFT или RIGHT.

пример

```
from tkinter import *
root = Tk()
btn_fill = Button(root, text="Button")
btn_fill.pack(fill=X)

btn_expand = Button(root, text="Button")
btn_expand.pack(expand=YES)

btn_side = Button(root, text="Button")
btn_side.pack(side=RIGHT)

root.mainloop()
```

Результат



Сетка ()

Менеджер геометрии `grid()` организует виджетов в виде таблицы в родительском виджетах. Главный виджет разделяется на строки и столбцы, и каждая часть таблицы может содержать виджет. Он использует `column` , `columnspan` , `ipadx` , `ipady` , `padx` , `pady` , `row` , `rowspan` и `sticky` .

Синтаксис

```
widget.grid(options)
```

колонка
Столбец для размещения виджета. Столбец по умолчанию - 0, который является самым левым столбцом.

ColumnSpan
Сколько занимает виджет столбцов. По умолчанию используется значение 1.

Ipadx
Сколько пикселей пикселя для виджета горизонтально внутри границ виджета.

Ipady
Сколько пикселей пиксели для виджета вертикально внутри границ виджета.

Padx
Сколько пикселей пиксели для виджета горизонтально за пределами границ виджета.

Pady
Сколько пикселей пиксели для виджета вертикально вне границ виджета.

Строка
Строка для размещения виджета. Строка по умолчанию - 0, которая является самым верхним столбцом.

RowSpan
Сколько строк занимает виджет. По умолчанию используется значение 1.

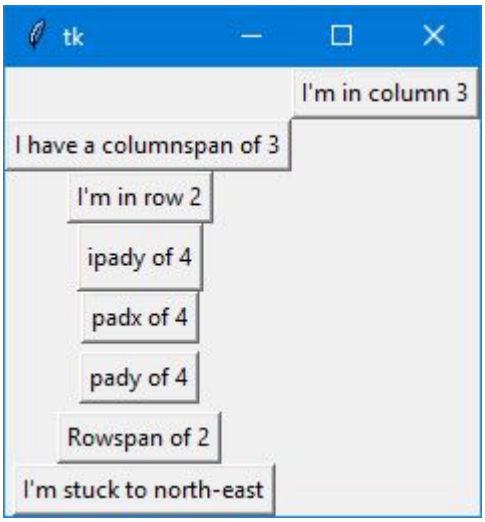
липкий
Когда виджет меньше ячейки, `sticky` используется для указания сторон и углов ячейки, к которой привязан виджет. Направление определяется направлениями компаса: N, E, S, W,

NE, NW, SE и SW и ноль. Это может быть конкатенация строк, например, NESW заставляет виджет занимать всю область ячейки.

пример

```
from tkinter import *
root = Tk()
btn_column = Button(root, text="I'm in column 3")
btn_column.grid(column=3)

btn_columnspan = Button(root, text="I have a columnspan of 3")
btn_columnspan.grid(columnspan=3)

btn_ipadx = Button(root, text="ipadx of 4")
btn_ipadx.grid(ipadx=4)

btn_ipady = Button(root, text="ipady of 4")
btn_ipady.grid(ipady=4)

btn_padx = Button(root, text="padx of 4")
btn_padx.grid(padx=4)

btn_pady = Button(root, text="pady of 4")
btn_pady.grid(pady=4)

btn_row = Button(root, text="I'm in row 2")
btn_row.grid(row=2)

btn_rowspan = Button(root, text="Rowspan of 2")
btn_rowspan.grid(rowspan=2)

btn_sticky = Button(root, text="I'm stuck to north-east")
btn_sticky.grid(sticky=NE)

root.mainloop()
```

Результат



место()

Менеджер `place()` организует виджетов, помещая их в определенную позицию в

родительском виджетах. Этот менеджер геометрии использует параметры `anchor` , `bordermode` , `height` , `width` , `relheight` , `relwidth` , `relx` , `rely` , `x` и `y` .

анкер
Указывает, к чему привязан виджет. Параметры - это направления компаса: N, E, S, W, NE, NW, SE или SW, которые относятся к сторонам и углам родительского виджета. По умолчанию используется NW (верхний левый угол виджета)

Bordermode
У Bordermode есть два варианта: `INSIDE` , что указывает на то, что другие опции относятся к родительскому внутри, (игнорируя границы родителя) и `OUTSIDE` , что является противоположным.

Рост
Укажите высоту виджета в пикселях.

ширина
Укажите ширину виджета в пикселях.

Relheight
Высота как float между 0.0 и 1.0, как часть высоты родительского виджета.

Relwidth
Ширина как float между 0.0 и 1.0, как часть ширины родительского виджета.

Relx
Горизонтальное смещение как float между 0.0 и 1.0, как часть ширины родительского виджета.

полагаться
Вертикальное смещение как float между 0.0 и 1.0, как часть высоты родительского виджета.

Икс
Горизонтальное смещение в пикселях.

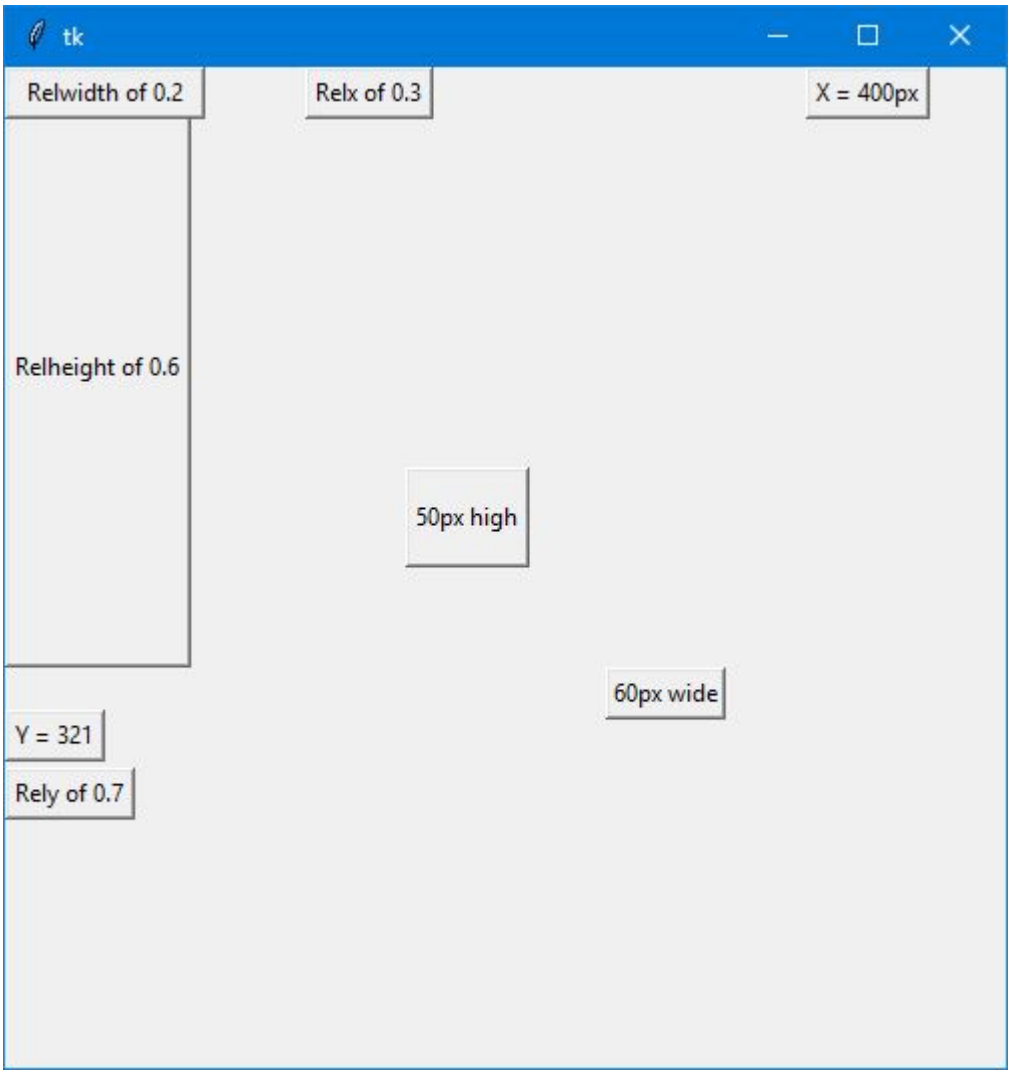
Y
Вертикальное смещение в пикселях.

пример

```
from tkinter import *
root = Tk()
root.geometry("500x500")

btn_height = Button(root, text="50px high")
btn_height.place(height=50, x=200, y=200)

btn_width = Button(root, text="60px wide")
```

```
btn_width.place(width=60, x=300, y=300)

btn_relheight = Button(root, text="Relheight of 0.6")
btn_relheight.place(relheight=0.6)

btn_relwidth= Button(root, text="Relwidth of 0.2")
btn_relwidth.place(relwidth=0.2)

btn_relx=Button(root, text="Relx of 0.3")
btn_relx.place(relx=0.3)

btn_rely=Button(root, text="Rely of 0.7")
btn_rely.place(rely=0.7)

btn_x=Button(root, text="X = 400px")
btn_x.place(x=400)

btn_y=Button(root, text="Y = 321")
btn_y.place(y=321)

root.mainloop()
```

Результат



Прочитайте Tkinter Geometry Managers онлайн:
https://riptutorial.com/ru/tkinter/topic/9620/tkinter-geometry-managers

глава 3: Виджет Tkinter Radiobutton

Синтаксис

- radobutton = tk.Radiobutton (родительский, ** kwargs)

параметры

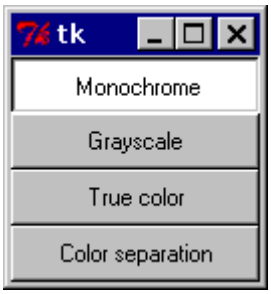
параметр	Описание
родитель	Виджеты tkinter существуют в иерархии. За исключением корневого окна, у всех виджетов есть родитель. Некоторые онлайн-уроки называют этот «мастер». Когда виджет будет добавлен на экран с пакетом, местом или сеткой, он появится внутри этого родительского виджета.
команда	функция, вызываемая каждый раз, когда пользователь меняет состояние радиообмена
indicatoron	1 или True для переключателей, 0 или False для кнопок
текст	Текст для отображения рядом с радиобарабаном.
значение	Когда выбран радиобъектив, соответствующая управляющая переменная устанавливается в значение.
переменная	Управляющая переменная, которую передает радиокарт, делится с другой радиообщей группы.

замечания

В этих примерах предполагается, что tkinter был импортирован либо `import tkinter as tk` (python 3), либо `import Tkinter as tk` (python 2).

Ссылка:

Чтобы превратить приведенный выше пример в «кнопочный ящик», а не в набор переключателей, установите для параметра индикатора значение 0. В этом случае нет отдельного индикатора переключателя, а выбранная кнопка отображается как SUNKEN вместо RAISED:



- effbot

Examples

Вот пример того, как переключаться на кнопки с кнопками:

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

rbvar = StringVar()
rbvar.set(" ")

rb1 = tk.Radiobutton(root, text="Option 1", variable=rbvar, value='a', indicatoron=0)
rb1.pack()

rb2 = tk.Radiobutton(root, text="Option 2", variable=rbvar, value='b', indicatoron=0)
rb2.pack()
```

Создать группу радиообъектов

Такая группа состоит из радиообъектов, которые имеют управляющую переменную, так что можно выбрать не более одного.

```
# control variable
var = tk.IntVar(parent, 0)

# group of radiobuttons
for i in range(1,4):
    tk.Radiobutton(parent, text='Choice %i' % i, value=i, variable=var).pack()

tk.Button(parent, text='Print choice', command=lambda: print(var.get())).pack()
```

Прочитайте Виджет Tkinter Radiobutton онлайн: https://riptutorial.com/ru/tkinter/topic/6338/виджет-tkinter-radiobutton

глава 4: Виджет входа Tkinter

Синтаксис

- entry = tk.Entry (*родительский* , ** *kwargs*)
- entry.get ()
- entry.insert (индекс, «значение»)
- entry.delete (start_index, end_index)
- entry.bind (событие, обратный вызов)

параметры

параметр	Описание
родитель	В hieararchy существуют виджеты tkinter. За исключением корневого окна, у всех виджетов есть родитель. Некоторые онлайн-уроки называют этот «мастер». Когда виджет будет добавлен на экран с `pack` , `place` или `grid` , он появится внутри этого родительского виджета
ширина	Ширина определяет *желаемую* ширину виджета на основе средней ширины символа. Для шрифтов с переменной шириной это зависит от ширины нулевого символа (`0`). Значение по умолчанию - 20. Обратите внимание, что фактическая ширина может быть больше или меньше в зависимости от того, как она добавляется к экрану.

замечания

В этих примерах предполагается, что tkinter был импортирован либо `import tkinter as tk` (python 3), либо `import Tkinter as tk` (python 2).

Examples

Создание виджета Entry и установка значения по умолчанию

```
entry = tk.Entry(parent, width=10)
entry.insert(0, "Hello, World!")
```

Получение значения виджета Entry

Значение виджета ввода можно получить с помощью метода `get` виджета:

```
name_entry = tk.Entry(parent)
...
name = name_entry.get()
```

Необязательно, вы можете связать экземпляр `StringVar` и получить значение из `StringVar` а не из виджета:

```
name_var = tk.StringVar()
name_entry = tk.Entry(parent, textvariable=name_var)
...
name = name_var.get()
```

Добавление проверки на виджет ввода

Чтобы ограничить символы, которые можно ввести в виджет ввода, к записи могут быть добавлены только цифры, например, команда проверки. Команда validate - это функция, возвращающая `True` если изменение принято, иначе `False` . Эта функция будет вызываться каждый раз, когда содержимое записи будет изменено. К этой функции могут быть переданы различные аргументы, такие как тип изменения (вставка, удаление), вставленный текст, ...

```
def only_numbers(char):
    return char.isdigit()

validation = parent.register(only_numbers)
entry = Entry(parent, validate="key", validatecommand=(validation, '%S'))
```

Параметр `validate` определяет тип события, который запускает проверку, здесь это любое нажатие клавиши в записи. `'%S'` в опции validatecommand означает, что вставленный или удаленный символ передается в аргументе функции `only_numbers` . Полный список возможностей можно найти здесь .

Получение int From Entry Widget

При использовании метода .get () все, что находится в виджетах записи, будет преобразовано в строку. Например, независимо от типа ввода (это может быть число или предложение), итоговый результат будет строкой. Если пользователь вводит 4, выход будет «4», как в строке. Чтобы получить int из виджета входа, сначала вызовите метод .get ().

```
What_User_Wrote = Entry.get()
```

Теперь мы преобразуем эту строку в int так:

```
Convert_To_Int = int(What_User_Wrote)
```

Аналогичным образом, если вы хотите сэкономить время, вы можете просто сделать:

```
Convert_To_Int = int(Entry.get())
```

Вы можете использовать вышеуказанный метод, если вы не хотите преобразовывать str в int.

Прочитайте Виджет входа Tkinter онлайн: https://riptutorial.com/ru/tkinter/topic/4868/виджет-входа-tkinter

глава 5: Виджеты Ttk

Вступление

Примеры различных виджетов ttk. Ttk имеет в общей сложности 17 виджетов, одиннадцать из которых уже существуют в tkinter (tk).

Использование модуля ttk дает вашему приложению более современный и улучшенный внешний вид.

Синтаксис

- дерево = ttk.Treeview (мастер, ** kwargs)

параметры

параметр	Описание
мастер	В hieararchy существуют виджеты tkinter. За исключением корневого окна, у всех виджетов есть родительский элемент (также называемый « master»). Когда виджет будет добавлен на экран с пакетом, местом или сеткой, он появится внутри этого родительского виджета

замечания

В этих примерах предполагается, что tkinter был импортирован либо `import tkinter as tk` (python 3), либо `import Tkinter as tk` (python 2).

Также предполагается, что ttk был импортирован либо `from tkinter import ttk` (python 3), либо из `import ttk` (python 2).

Examples

Treeview: основной пример

Этот виджет используется для отображения элементов с иерархией. Например, Windows Explorer может быть воспроизведен таким образом. Некоторые красивые таблицы также можно сделать с помощью виджета `treeview` .

Создание виджета

```
tree=ttk.Treeview(master)
```

Определение столбцов

Вы можете определить, сколько столбцов, их ширина и минимальная ширина, когда пользователь попытается растянуть его. Определив `stretch=tk.NO` , пользователь не может изменить ширину столбца.

```
tree["columns"]=("one","two","three")
tree.column("#0", width=270, minwidth=270, stretch=tk.NO)
tree.column("one", width=150, minwidth=150, stretch=tk.NO)
tree.column("two", width=400, minwidth=200)
tree.column("three", width=80, minwidth=50, stretch=tk.NO)
```

Определение заголовков

```
tree.heading("#0",text="Name",anchor=tk.W)
tree.heading("one", text="Date modified",anchor=tk.W)
tree.heading("two", text="Type",anchor=tk.W)
tree.heading("three", text="Size",anchor=tk.W)
```

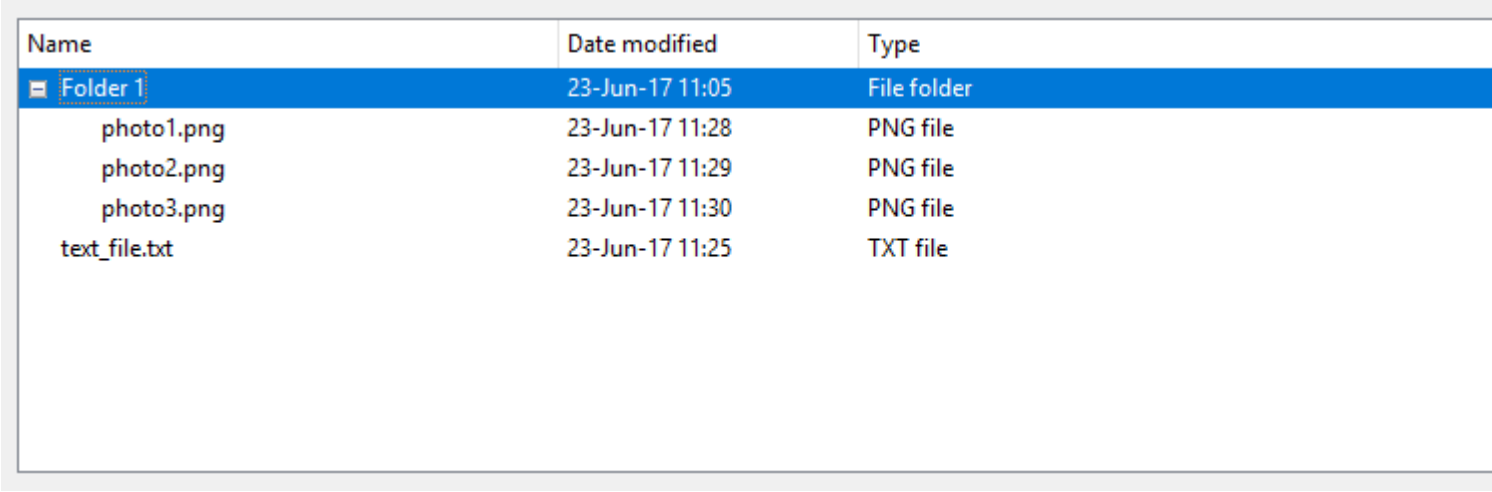
Вставить несколько строк

```
# Level 1
folder1=tree.insert("", 1, "", text="Folder 1", values=("23-Jun-17 11:05","File folder",""))
tree.insert("", 2, "", text="text_file.txt", values=("23-Jun-17 11:25","TXT file","1 KB"))
# Level 2
tree.insert(folder1, "end", "", text="photo1.png", values=("23-Jun-17 11:28","PNG file","2.6
KB"))
tree.insert(folder1, "end", "", text="photo2.png", values=("23-Jun-17 11:29","PNG file","3.2
KB"))
tree.insert(folder1, "end", "", text="photo3.png", values=("23-Jun-17 11:30","PNG file","3.1
KB"))
```

упаковка

```
tree.pack(side=tk.TOP,fill=tk.X)
```

В Windows из этого примера можно получить следующий снимок экрана.



Name	Date modified	Type
Folder 1	23-Jun-17 11:05	File folder
photo1.png	23-Jun-17 11:28	PNG file
photo2.png	23-Jun-17 11:29	PNG file
photo3.png	23-Jun-17 11:30	PNG file
text_file.txt	23-Jun-17 11:25	TXT file

Индикатор

Виджет `ttk.progress` полезен при работе с длинными вычислениями, чтобы пользователь знал, что программа запущена. Ниже приведен пример обновления контрольной панели каждые 0,5 секунды:

Функция обновления контрольной панели

```
def progress(currentValue):
    progressbar["value"]=currentValue
```

Установите максимальное значение

```
maxValue=100
```

Создать индикатор выполнения

```
progressbar=ttk.Progressbar(master,orient="horizontal",length=300,mode="determinate")
progressbar.pack(side=tk.TOP)
```

«Определенный» режим используется, когда индикатор прогресса находится под управлением программы.

Начальное и максимальное значения

```
currentValue=0
progressbar["value"]=currentValue
progressbar["maximum"]=maxValue
```

Эмулировать прогресс каждые 0,5 с

```
divisions=10
for i in range(divisions):
    currentValue=currentValue+10
    progressbar.after(500, progress(currentValue))
    progressbar.update() # Force an update of the GUI
```

Прочитайте Виджеты Ttk онлайн: https://riptutorial.com/ru/tkinter/topic/10622/виджеты-ttk

глава 6: Добавление изображений в ярлык / кнопку

Вступление

Это показывает правильное использование изображений и правильное отображение изображений.

Examples

Форматы файлов, поддерживаемые Tkinter

Tkinter поддерживает .ppm файлы из PIL (Python Imaging Library), .JPG, .PNG и .GIF.

Для импорта и изображения сначала необходимо создать ссылку, например:

```
Image = PhotoImage(filename = [Your Image here])
```

Теперь мы можем добавить это изображение в Button и Labels так, используя обратный вызов «img»:

```
 Lbl = Label (width=490, img=image)
```

Использование форматов .GIF.

Чтобы отобразить gif, вам нужно показать его по кадру, как анимацию.

Анимированный gif состоит из нескольких кадров в одном файле. Tk загружает первый кадр, но вы можете указать разные кадры, передав индексный параметр при создании изображения. Например:

```
frame2 = PhotoImage(file=imagefilename, format="gif -index 2")
```

Если вы загрузите все кадры в отдельные PhotoImages, а затем используйте события таймера, чтобы переключить отображаемый кадр (label.configure (image = nextframe)). Задержка на таймере позволяет контролировать скорость анимации. Нет ничего, что обеспечило бы вам количество кадров на изображении, кроме того, что он не смог создать кадр, если вы превысите количество кадров.

Прочитайте Добавление изображений в ярлык / кнопку онлайн:
https://riptutorial.com/ru/tkinter/topic/9746/добавление-изображений-в-ярлык---кнопку

глава 7: Настроить стили ttk

Вступление

Стиль новых виджетов ttk является одним из самых мощных аспектов ttk. Помимо того факта, что это совершенно другой способ работы, чем традиционный пакет tk, он позволяет выполнять огромную настройку ваших виджетов.

Examples

Настройка древовидной структуры

Выбрав Treeview: Основной пример , можно показать, как настроить базовое древовидное представление.

В этом случае мы создаем стиль «mystyle.Treeview» со следующим кодом (см. Комментарии, чтобы понять, что делает каждая строка):

```
style = ttk.Style()
style.configure("mystyle.Treeview", highlightthickness=0, bd=0, font=('Calibri', 11)) # Modify
the font of the body
style.configure("mystyle.Treeview.Heading", font=('Calibri', 13,'bold')) # Modify the font of
the headings
style.layout("mystyle.Treeview", [('mystyle.Treeview.treearea', {'sticky': 'nswe'})]) # Remove
the borders
```

Затем создается виджет, создающий этот стиль:

```
tree=ttk.Treeview(master,style="mystyle.Treeview")
```

Если вы хотите иметь другой формат в зависимости от строк, вы можете использовать `tags` :

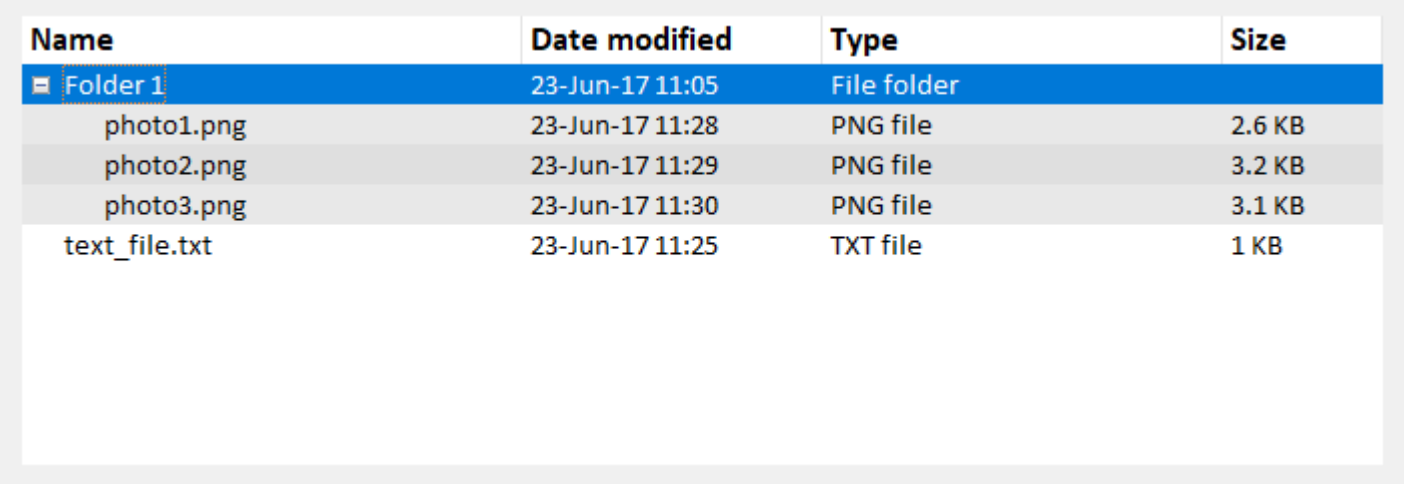
```
tree.insert(folder1, "end", "", text="photo1.png", values=("23-Jun-17 11:28","PNG file","2.6
KB"),tags = ('odd',))
tree.insert(folder1, "end", "", text="photo2.png", values=("23-Jun-17 11:29","PNG file","3.2
KB"),tags = ('even',))
tree.insert(folder1, "end", "", text="photo3.png", values=("23-Jun-17 11:30","PNG file","3.1
KB"),tags = ('odd',))
```

Затем, например, цвет фона может быть связан с тегами:

```
tree.tag_configure('odd', background='#E8E8E8')
tree.tag_configure('even', background='#DFDFDF')
```

Результатом является древовидная структура с измененными шрифтами как для тела, так

и для заголовков, без границ и разных цветов для строк:



Name	Date modified	Type	Size
Folder 1	23-Jun-17 11:05	File folder	
photo1.png	23-Jun-17 11:28	PNG file	2.6 KB
photo2.png	23-Jun-17 11:29	PNG file	3.2 KB
photo3.png	23-Jun-17 11:30	PNG file	3.1 KB
text_file.txt	23-Jun-17 11:25	TXT file	1 KB

Примечание. Чтобы создать приведенное выше изображение, вы должны добавить / изменить вышеупомянутые строки кода в примере примера Treeview: Basic .

Прочитайте Настроить стили ttk онлайн: https://riptutorial.com/ru/tkinter/topic/10624/настроить-стили-ttk

глава 8: Несколько окон (виджеты TopLevel)

Examples

Разница между Tk и Toplevel

`Tk` является абсолютным корнем приложения, это первый виджет, который необходимо создать, и графический интерфейс будет закрыт, когда он будет уничтожен.

`Toplevel` - это окно в приложении, закрытие окна уничтожит все дочерние виджеты, размещенные в этом окне {1}, но не выключит программу.

```
try:
    import tkinter as tk #python3
except ImportError:
    import Tkinter as tk #python2

#root application, can only have one of these.
root = tk.Tk()

#put a label in the root to identify the window.
label1 = tk.Label(root, text="""this is root
closing this window will shut down app""")
label1.pack()

#you can make as many Toplevels as you like
extra_window = tk.Toplevel(root)
label2 = tk.Label(extra_window, text="""this is extra_window
closing this will not affect root""")
label2.pack()

root.mainloop()
```

Если ваша программа python представляет только одно приложение (которое почти всегда будет), то у вас должен быть только один экземпляр `Tk` , но вы можете создать столько окон `Toplevel` сколько хотите.

```
try:
    import tkinter as tk #python3
except ImportError:
    import Tkinter as tk #python2

def generate_new_window():
    window = tk.Toplevel()
    label = tk.Label(window, text="a generic Toplevel window")
    label.pack()

root = tk.Tk()
```

```
spawn_window_button = tk.Button(root,
                                text="make a new window!",
                                command=generate_new_window)
spawn_window_button.pack()

root.mainloop()
```

{1}: если Toplevel (`A = Toplevel(root)`) является родителем другого верхнего уровня (`B = Toplevel(A)`), то закрывающее окно A также закрывает окно B.

расположение стека окон (метод .lift)

Самый простой случай, чтобы поднять конкретное окно над другими, просто вызовите метод `.lift()` в этом окне (либо `Toplevel` либо `Tk`)

```
import tkinter as tk #import Tkinter as tk #change to commented for python2

root = tk.Tk()

for i in range(4):
    #make a window with a label
    window = tk.Toplevel(root)
    label = tk.Label(window,text="window {}".format(i))
    label.pack()
    #add a button to root to lift that window
    button = tk.Button(root, text = "lift window {}".format(i), command=window.lift)
    button.grid(row=i)

root.mainloop()
```

Однако, если это окно уничтожено, пытаясь его поднять, это приведет к ошибке:

```
Exception in Tkinter callback
Traceback (most recent call last):
  File "/.../tkinter/__init__.py", line 1549, in __call__
    return self.func(*args)
  File "/.../tkinter/__init__.py", line 785, in tkraise
    self.tk.call('raise', self._w, aboveThis)
_tkinter.TclError: bad window path name ".4385637096"
```

Часто, когда мы пытаемся поставить конкретное окно перед пользователем, но оно было закрыто, хорошей альтернативой является воссоздание этого окна:

```
import tkinter as tk #import Tkinter as tk #change to commented for python2

dialog_window = None

def create_dialog():
    """creates the dialog window
  ** do not call if dialog_window is already open, this will
     create a duplicate without handling the other
if you are unsure if it already exists or not use show_dialog()"""
    global dialog_window
```

```
    dialog_window = tk.Toplevel(root)
    label1 = tk.Label(dialog_window,text="this is the dialog window")
    label1.pack()
    #put other widgets
    dialog_window.lift() #ensure it appears above all others, probably will do this anyway

def show_dialog():
    """lifts the dialog_window if it exists or creates a new one otherwise"""
    #this can be refactored to only have one call to create_dialog()
    #but sometimes extra code will be wanted the first time it is created
    if dialog_window is None:
        create_dialog()
        return
    try:
        dialog_window.lift()
    except tk.TclError:
        #window was closed, create a new one.
        create_dialog()


root = tk.Tk()

dialog_button = tk.Button(root,
                          text="show dialog_window",
                          command=show_dialog)
dialog_button.pack()
root.mainloop()
```

Таким образом, функция `show_dialog` покажет диалоговое окно независимо от того, существует оно или нет, также обратите внимание, что вы можете вызвать `.winfo_exists()` чтобы проверить, существует ли это, прежде чем пытаться поднять окно, а не обертывать его в `try:except` .

Существует также метод `.lower()` который работает так же, как метод `.lift()` , за исключением опускания окна в стеке:

```
import tkinter as tk #import Tkinter as tk #change to commented for python2

root = tk.Tk()
root.title("ROOT")
extra = tk.Toplevel()
label = tk.Label(extra, text="extra window")
label.pack()

lower_button = tk.Button(root,
                         text="lower this window",
                         command=root.lower)
lower_button.pack()

root.mainloop()
```

Вы заметите, что он опускается ниже других приложений, и только ниже ниже определенного окна вы можете передать его `.lower()` , аналогичным образом это также можно сделать с помощью `.lift()` чтобы только поднять окно над другим один.

Прочитайте Несколько окон (виджеты TopLevel) онлайн:

https://riptutorial.com/ru/tkinter/topic/6439/несколько-окон--виджеты-toplevel-

глава 9: Отсрочка функции

Синтаксис

- widget.after (delay_ms, callback, * args)

параметры

параметр	Описание
delay_ms	Время (миллисекунды), которое задерживает вызов функции `callback` функции
Перезвоните	Функция, вызываемая после данного `delay_ms` . Если этот параметр не задан, `.after` действует аналогично `time.sleep` (в миллисекундах)

замечания

Синтаксис принимает `widget` принятый методом. `.after` он был создан (т.е. `widget=tk.Label(parent)`)

Examples

.после()

`.after(delay, callback=None)` - это метод, определенный для всех виджетов tkinter. Этот метод просто вызывает функцию `callback` функции после заданной `delay` в ms. Если функция не указана, она действует подобно `time.sleep` (но в миллисекундах вместо секунд)

Вот пример того , как создать простой таймер с помощью `after` того, как :

```
# import tkinter
try:
    import tkinter as tk
except ImportError:
    import Tkinter as tk

class Timer:
    def __init__(self, parent):
        # variable storing time
        self.seconds = 0
        # label displaying time
        self.label = tk.Label(parent, text="0 s", font="Arial 30", width=10)
        self.label.pack()
        # start the timer
```

```
        self.label.after(1000, self.refresh_label)

    def refresh_label(self):
        """ refresh the content of the label every second """
        # increment the time
        self.seconds += 1
        # display the new time
        self.label.configure(text="%i s" % self.seconds)
        # request tkinter to call self.refresh after 1s (the delay is given in ms)
        self.label.after(1000, self.refresh_label)

if __name__ == "__main__":
    root = tk.Tk()
    timer = Timer(root)
    root.mainloop()
```

Прочитайте Отсрочка функции онлайн: https://riptutorial.com/ru/tkinter/topic/6724/отсрочка-функции

глава 10: Прокрутка виджета

Вступление

Полоски прокрутки могут быть добавлены в виджеты Listbox, Canvas и Text. Кроме того, виджеты ввода можно прокручивать по горизонтали. Чтобы иметь возможность прокручивать другие типы виджетов, вы должны поместить их в виджет Canvas или Text.

Синтаксис

- scrollbar = tk.Scrollbar (родительский, ** kwargs)

параметры

параметр	Описание
родитель	Виджеты tkinter существуют в иерархии. За исключением корневого окна, у всех виджетов есть родитель. Некоторые онлайн-уроки называют этот «мастер». Когда виджет будет добавлен на экран с пакетом, местом или сеткой, он появится внутри этого родительского виджета
восток	Ориентация полосы прокрутки, `"vertical"` (значение по умолчанию) или `"horizontal"`

замечания

В этих примерах предполагается, что tkinter был импортирован либо `import tkinter as tk` (python 3), либо `import Tkinter as tk` (python 2).

Examples

Подключение вертикальной полосы прокрутки к текстовому виджету

Соединение между виджетами и полосой прокрутки происходит в обоих направлениях. Полоса прокрутки должна быть развернута вертикально, чтобы она имела ту же высоту, что и виджет.

```
text = tk.Text(parent)
text.pack(side="left")

scroll_y = tk.Scrollbar(parent, orient="vertical", command=text.yview)
scroll_y.pack(side="left", expand=True, fill="y")
```

```
text.configure(yscrollcommand=scroll_y.set)
```

Прокрутка виджета Canvas горизонтально и вертикально

Принцип по существу тот же, что и для виджета «Текст», но макет `Grid` используется для размещения полос прокрутки вокруг виджета.

```
canvas = tk.Canvas(parent, width=150, height=150)
canvas.create_oval(10, 10, 20, 20, fill="red")
canvas.create_oval(200, 200, 220, 220, fill="blue")
canvas.grid(row=0, column=0)

scroll_x = tk.Scrollbar(parent, orient="horizontal", command=canvas.xview)
scroll_x.grid(row=1, column=0, sticky="ew")

scroll_y = tk.Scrollbar(parent, orient="vertical", command=canvas.yview)
scroll_y.grid(row=0, column=1, sticky="ns")

canvas.configure(yscrollcommand=scroll_y.set, xscrollcommand=scroll_x.set)
```

В отличие от виджета «Текст» прокручиваемая область Canvas не обновляется автоматически при изменении ее содержимого, поэтому нам нужно определить ее и обновить вручную с `scrollregion` аргумента `scrollregion` :

```
canvas.configure(scrollregion=canvas.bbox("all"))
```

`canvas.bbox("all")` возвращает координаты прямоугольника, соответствующие всему содержимому холста.

Прокрутка группы виджетов

Когда окно содержит много виджетов, они могут быть не все видимыми. Однако ни окно (экземпляр Tk, ни Toplevel), ни кадр не прокручиваются. Одним из решений, позволяющих прокручивать содержимое окна, является размещение всех виджетов в кадре, а затем встраивание этого кадра в холст с `create_window` метода `create_window` .

```
canvas = tk.Canvas(parent)
scroll_y = tk.Scrollbar(parent, orient="vertical", command=canvas.yview)

frame = tk.Frame(canvas)
# group of widgets
for i in range(20):
    tk.Label(frame, text='label %i' % i).pack()
# put the frame in the canvas
canvas.create_window(0, 0, anchor='nw', window=frame)
# make sure everything is displayed before configuring the scrollregion
canvas.update_idletasks()

canvas.configure(scrollregion=canvas.bbox('all'),
                 yscrollcommand=scroll_y.set)
```

```
canvas.pack(fill='both', expand=True, side='left')
scroll_y.pack(fill='y', side='right')
```

Прочитайте Прокрутка виджета онлайн: https://riptutorial.com/ru/tkinter/topic/8931/прокрутка-виджета

кредиты

S. No	Главы	Contributors
1	Начало работы с tkinter	Billal BEGUERADJ, Bryan Oakley, Community, J.J. Hakala, j_4321, JGreenwell, Mike - SMT, Neil A., Nico Brubaker, Razik, ryneke, Tadhg McDonald-Jensen, tao, Yamboy1
2	Tkinter Geometry Managers	Henry
3	Виджет Tkinter Radiobutton	j_4321, nbro, Parviz Karimli
4	Виджет входа Tkinter	Angrywasabi, Bryan Oakley, double_j, j_4321
5	Виджеты Ttk	David Duran
6	Добавление изображений в ярлык / кнопку	Angrywasabi
7	Настроить стили ttk	David Duran
8	Несколько окон (виджеты TopLevel)	Tadhg McDonald-Jensen
9	Отсрочка функции	David Duran, Neil A., Tadhg McDonald-Jensen
10	Прокрутка виджета	j_4321