Библиотека tkinter

В Python есть довольно много GUI фреймворков (graphicaluserinterface), однако только Tkinter встроен в стандартную библиотеку языка. У Tkinter есть несколько преимуществ. Он кроссплатформенный, поэтому один и тот же код можно использовать на Windows, macOS и Linux.
Визуальные элементы отображаются через собственные элементы текущей операционной системы, поэтому приложения, созданные с помощью Tkinter, выглядят так, как будто они принадлежат той платформе, на которой они работают.
Хотя Tkinter является популярным GUI фреймворком на Python, у него есть свои недостатки. Один из них заключается в том, что графические интерфейсы, созданные с использованием Tkinter, выглядят устаревшими. Если вам нужен современный, броский интерфейс, то Tkinter может оказаться не совсем тем, для этого есть PyQt5 который развивается сильнее в данном плане.
Тем не менее, в плане использования, Tkinter является относительно легким по сравнению с другими библиотеками. Это отличный выбор для создания GUI приложений в Python, особенно если современный облик не в приоритете для программы, а большую роль играет функциональность и кроссплатформенная скорость.
Tk (от англ. Toolkit — «набор инструментов», «инструментарий») — кроссплатформенная библиотека базовых элементов графического интерфейса, распространяемая с открытыми исходными текстами.
Tk был разработан Джоном Оустерхаутом как расширение для интерпретируемого языка программирования Tcl. Также, с использованием специальных библиотек, Tk может использоваться другими языками программирования, например, Perl, Python, Ruby. Большинство из этих языков использует Tcl как мост для Tk.
Tk портирован на большинство реализаций Linux, Mac OS X, Unix и Microsoft Windows. Начиная с Tcl/Tk 8 графический интерфейс имеет «родной» для ОС вид, то есть графические элементы будут выглядеть так же, как и стандартные для данной ОС. В версии 8.5 в Tk появилась возможность использовать движок для отрисовки элементов с поддержкой тем оформления — Ttk. Кроме того, есть несколько расширений, обеспечивающих drag-and-drop с внешними приложениями, непрямоугольные окна и т. д.

Tk представляет разработчику набор Tcl-команд, предназначенных для создания компонентов (виджетов) и выполнения различных действий с ними. Компонент представляет собой окно в составе графического интерфейса, имеющее определённый внешний вид и выполняющее некоторые функции.
Компоненты организованы в иерархическую структуру. С точки зрения приложения существует главное, или первичное, окно, в котором создаются дочерние окна. Дочерние окна, в свою очередь, могут выступать в качестве родительских по отношению к другим окнам и т. д. Компонентами управляют диспетчеры компоновки (geometry manager), которые определяют размеры компонентов и их размещение на экране.
Tk-приложения, как и большинство оконных приложений, представляют собой программы, управляемые событиями. Компоненты Tk автоматически обрабатывают большинство событий, что упрощает задачу разработчиков по созданию приложений.
Как правило, выполнение Tk-сценария начинается с создания компонентов и размещения их с помощью диспетчера компоновки, после чего сценарий связывает обработчики событий с компонентами. После того, как интерпретатор оканчивает разбор команд, ответственных за инициализацию пользовательского интерфейса, он переходит в цикл обработки событий. С этого момента приложение начинает реагировать на действия пользователя.
Для Tk существуют наборы дополнительных компонентов, например, BWidget, Tix или incr Widgets. Особо стоит выделить BWidget, так как преимуществом его использования является отсутствие необходимости в компиляции под определённую платформу (код тулкита представляет собой «чистый» Tcl).

Общий порядок работы с tkinter:

1. Импортировать tkinter.
2. Создать главное окно.
3. Добавить виджеты в главном окне, выполнить их конфигурацию (при необходимости).
4. Определить и описать обработчики событий, т.е. на что и как будет реагировать программа.
5. Отобразить виджет в главном окне с помощью упаковщика.
6. Зациклить приложение.

Раздаточный материал № 161

1 `from tkinter import *` 2 `from tkinter import ttk` 3	В результате импортирования в пространстве имён программы (скрипта) появляются имена, встроенные в tkinter, к которым можно обращаться непосредственно. Массовое импортирование имён может привести к их конфликту. Кроме того, для интерпретатора требуется больше времени, чтобы в списке доступных имён найти нужное. Основные виджеты-**Раздаточный материал № 2** Подмодуль ttk предоставляет доступ к множеству стилизуемых виджетов tk, а также предоставляет дополнительные виджеты. Импорт ttk должен следовать за импортом tk
4 `root = Tk()` 5 `root.title("Привет мир!")` 6 `root.geometry('300x40')` 7	команда создаёт корневое (root) окно программы команда меняет заголовок окна команда устанавливает размеры окна
8 `def button_clicked():` 9 `  print("Hello World!")` 10 11 `def close():` 12 `  root.destroy()` 13 `  root.quit()`	определение функции-обработчика события «нажата кнопка мыши» Функция-обработчик события «закрытие главного окна». Она останавливает главный цикл приложения и разрушает главное окно. Без неё закрыть программу можно, лишь если завершить процесс интерпретатора Python. Поскольку функция использует глобальную переменную root, объявление самой функции должно следовать после объявления переменной root.
15 `button = ttk.Button(root,` 16 `  text="Press Me",` 17 `  command=button_clicked)`	Создание кнопки с текстом «Press Me» и привязка её к функции-обработчику. Root можно опускать, т.к. это значение по умолчанию.
18 `button.pack(fill=BOTH)`	«Упаковываем» созданную кнопку с помощью менеджера компоновки pack. fill=BOTH (также можно fill="both") указывает кнопке занимать все доступное пространство (по ширине и высоте) на родительском виджете root
19 20 `root.protocol('WM_DELETE_WINDOW', close)`	Привязываем событие закрытия главного окна с функцией-обработчиком close
22 `root.mainloop()`	Запускаем главный цикл приложения

Раздаточный материал № 162 - Основные виджеты (справочно)

Toplevel/root

Toplevel–окно верхнего уровня. Обычно используется для создания многооконных программ, а также для диалоговых окон.

Методы виджета

title - заголовок окна

overrideredirect - указание оконному менеджеру игнорировать это окно. Аргументом является True или False. В случае, если аргумент не указан - получаем текущее значение. Если аргумент равен True, то такое окно будет показано оконным менеджером без обрамления (без заголовка и бордюра). Может быть использовано, например, для создания splashscreen при старте программы.

iconify / deiconify - свернуть / развернуть окно

withdraw - "спрятать" (сделать невидимым) окно. Для того, чтобы снова показать его, надо использовать метод deiconify.

minsize и maxsize - минимальный / максимальный размер окна. Методы принимают два аргумента - ширина и высота окна. Если аргументы не указаны - возвращают текущее значение.

state - получить текущее значение состояния окна. Может возвращать следующие значения: normal (нормальное состояние), icon (показано в виде иконки), iconic (свёрнуто), withdrawn (не показано), zoomed (развёрнуто на полный экран, только для Windows и Mac OS X)

resizable - может ли пользователь изменять размер окна. Принимает два аргумента - возможность изменения размера по горизонтали и по вертикали. Без аргументов возвращает текущее значение.

geometry - устанавливает геометрию окна в формате ширина x высота+x+y (пример: geometry("600x400+40+80") - поместить окно в точку с координатам 40,80 и установить размер в 600x400). Размер или координаты могут быть опущены (geometry("600x400") - только изменить размер, geometry("+40+80") - только переместить окно).

transient - сделать окно зависимым от другого окна, указанного в аргументе. Будет сворачиваться вместе с указанным окном. Без аргументов возвращает текущее значение.

protocol - получает два аргумента: название события и функцию, которая будет вызываться при наступлении указанного события. События могут называться WM_TAKE_FOCUS (получение фокуса), WM_SAVE_YOURSELF (необходимо сохраниться, в настоящий момент является устаревшим), WM_DELETE_WINDOW (удаление окна).

tkraise (синоним lift) и lower - поднимает (размещает поверх всех других окон) или опускает окно. Методы могут принимать один необязательный аргумент: над/под каким окном разместить текущее.

grab_set - устанавливает фокус на окно, даже при наличии открытых других окон

grab_release - снимает монопольное владение фокусом ввода с окна

bg – цвет фона

bd – ширина рамки

Пример:

```
from Tkinter import *
def window_deleted():
print u'Окно закрыто'
   root.quit() # явное указание на выход из программы
root=Tk()
root.title(u'Пример приложения')
root.geometry('500x400+300+200') # ширина=500, высота=400, x=300, y=200
root.protocol('WM_DELETE_WINDOW', window_deleted) # обработчик закрытия окна
root.resizable(True, False) # размер окна может быть изменён только по горизонтали
```

```
root.mainloop()
```

Таким способом можно предотвратить закрытие окна (например, если закрытие окна приведёт к потере введённых пользователем данных).

Button

Виджет Button - самая обыкновенная кнопка, которая используется в тысячах программ. Пример кода:

```
from Tkinter import *
root=Tk()
button1=Button(root,text='ok',width=25,height=5,bg='black',fg='red',font='arial 14')
button1.pack()
root.mainloop()
```

За создание, собственно, окна, отвечает класс Tk(), и первым делом нужно создать экземпляр этого класса. Этот экземпляр принято называть root, хотя вы можете назвать его как угодно. Далее создаётся кнопка, при этом мы указываем её свойства (начинать нужно с указания окна, в примере - root). Здесь перечислены некоторые из них:

text - какой текст будет отображён на кнопке (в примере - ок)
width,height - соответственно, ширина и длина кнопки.
bg - цвет кнопки (сокращенно от background, в примере цвет - чёрный)
fg - цвет текста на кнопке (сокращённо от foreground, в примере цвет - красный)
font - шрифт и его размер (в примере - arial, размер - 14)
compound – определяет ориентацию текста по отношению к изображению
image–добавляет изображение на кнопку

Label

Label - это виджет, предназначенный для отображения какой-либо надписи без возможности редактирования пользователем. Имеет те же свойства, что и перечисленные свойства кнопки.

Entry

Entry - это виджет, позволяющий пользователю ввести одну строку текста. Имеет дополнительное свойство bd (сокращённо от borderwidth), позволяющее регулировать ширину границы.

borderwidth - ширина бордюра элемента
bd - сокращение от borderwidth
width - задаёт длину элемента в знакоместах.
show - задает отображаемый символ.

Text

Text - это виджет, который позволяет пользователю ввести любое количество текста. Имеет дополнительное свойство wrap, отвечающее за перенос (чтобы, например, переносить по словам, нужно использовать значение WORD).Например,

```
from Tkinter import *
root=Tk()
text1=Text(root,height=7,width=7,font='Arial 14',wrap=WORD)
text1.pack()
root.mainloop()
```

Методыinsert, deleteиgetдобавляют, удаляютилиизвлекаюттекст. Первыйаргумент - местовставкиввиде 'x.y', где x – этострока, а y – столбец. Например,

```
text1.insert(1.0,'Добавить Текст\n\ в начало первой строки')
text1.delete('1.0', END)   # Удалитьвсе
text1.get('1.0', END)      # Извлечьвсе
```

Listbox

Listbox - это виджет, который представляет собой список, из элементов которого пользователь может выбирать один или несколько пунктов. Имеет дополнительное свойство selectmode, которое, при значении SINGLE, позволяет пользователю выбрать только один элемент списка, а при значении EXTENDED - любое количество. Пример:

```
from Tkinter import *
root=Tk()
listbox1=Listbox(root,height=5,width=15,selectmode=EXTENDED)
listbox2=Listbox(root,height=5,width=15,selectmode=SINGLE)
list1=[u"Москва",u"Санкт-Петербург",u"Саратов",u"Омск"]
list2=[u"Канберра",u"Сидней",u"Мельбурн",u"Аделаида"]
for i in list1:
listbox1.insert(END,i)
for i in list2:
listbox2.insert(END,i)
listbox1.pack()
listbox2.pack()
root.mainloop()
```

Frame

Виджет Frame (рамка) предназначен для организации виджетов внутри окна. Рассмотримпример:

```
from tkinter import *
root=Tk()
frame1=Frame(root,bg='green',bd=5)
frame2=Frame(root,bg='red',bd=5)
button1=Button(frame1,text=u'Перваякнопка')
button2=Button(frame2,text=u'Втораякнопка')
frame1.pack()
frame2.pack()
button1.pack()
button2.pack()
root.mainloop()
```

Свойство bd отвечает за толщину края рамки.

Checkbutton

Checkbutton - это виджет, который позволяет отметить „галочкой“ определенный пункт в окне. При использовании нескольких пунктов нужно каждому присвоить свою переменную. Разберем пример:

```
from tkinter import *
root=Tk()
var1=IntVar()
var2=IntVar()
check1=Checkbutton(root,text=u'1 пункт',variable=var1,onvalue=1,offvalue=0)
```

```
check2=Checkbutton(root,text=u'2 пункт',variable=var2,onvalue=1,offvalue=0)
check1.pack()
check2.pack()
root.mainloop()
```

IntVar() - специальный класс библиотеки для работы с целыми числами. variable - свойство, отвечающее за прикрепление к виджету переменной. onvalue, offvalue - свойства, которые присваивают прикреплённой к виджету переменной значение, которое зависит от состояния(onvalue - при выбранном пункте, offvalue - при невыбранном пункте).

Radiobutton

Виджет Radiobutton выполняет функцию, схожую с функцией виджета Checkbutton. Разница в том, что в виджете Radiobutton пользователь может выбрать лишь один из пунктов. Реализация этого виджета несколько иная, чем виджета Checkbutton:

```
from tkinter import *
root=Tk()
var=IntVar()
rbutton1=Radiobutton(root,text='1',variable=var,value=1)
rbutton2=Radiobutton(root,text='2',variable=var,value=2)
rbutton3=Radiobutton(root,text='3',variable=var,value=3)
rbutton1.pack()
rbutton2.pack()
rbutton3.pack()
root.mainloop()
```

В этом виджете используется уже одна переменная. В зависимости от того, какой пункт выбран, она меняет своё значение. Если присвоить этой переменной какое-либо значение, поменяется и выбранный виджет.

Scale

Scale (шкала) - это виджет, позволяющий выбрать какое-либо значение из заданного диапазона. Свойства:

orient - как расположена шкала на окне. Возможные значения: HORIZONTAL, VERTICAL (горизонтально, вертикально).

length - длина шкалы.

from_ - с какого значения начинается шкала.

to - каким значением заканчивается шкала.

tickinterval - интервал, через который отображаются метки шкалы.

resolution - шаг передвижения (минимальная длина, на которую можно передвинуть движок)

Примеркода:

```
from tkinter import *
root = Tk()
def getV(root):
   a = scale1.get()
print "Значение", a
scale1 = Scale(root,orient=HORIZONTAL,length=300,from_=50,to=80,tickinterval=5,
resolution=5)
button1 = Button(root,text=u"Получитьзначение")
scale1.pack()
button1.pack()
button1.bind("<Button-1>",getV)
root.mainloop()
```

Здесь используется специальный метод get(), который позволяет снять с виджета определенное значение, и используется не только в Scale.

Scrollbar

Этот виджет даёт возможность пользователю "прокрутить" другой виджет (например, текстовое поле). Необходимо сделать две привязки: command полосы прокрутки привязываем к методу xview/yview виджета, а xscrollcommand/yscrollcommand виджета привязываем к методу set полосы прокрутки.

Пример:

```
from tkinter import *
root = Tk()
text = Text(root, height=3, width=60)
text.pack(side='left')
scrollbar = Scrollbar(root)
scrollbar.pack(side='left')
# перваяпривязка
scrollbar['command'] = text.yview
# втораяпривязка
text['yscrollcommand'] = scrollbar.set
root.mainloop()
```

Упаковщики

Упаковщик (менеджер геометрии, менеджер расположения) это специальный механизм, который размещает (упаковывает) виджеты на окне. В Tkinter есть три упаковщика: pack, place, grid. В одном виджете можно использовать только один тип упаковки, при смешивании разных типов упаковки программа, скорее всего, не будет работать.

Раздаточный материал № 163 - Упаковщики (справочно)

pack()

Упаковщик pack() является самым интеллектуальным (и самым непредсказуемым). При использовании этого упаковщика с помощью свойства side нужно указать к какой стороне родительского виджета он должен примыкать. Как правило этот упаковщик используют для размещения виджетов друг за другом (слева направо или сверху вниз). Пример:

```
from tkinter import *
root=Tk()
button1 = Button(text="1")
button2 = Button(text="2")
button3 = Button(text="3")
button4 = Button(text="4")
button5 = Button(text="5")
button1.pack(side='left')
button2.pack(side='top')
button3.pack(side='left')
button4.pack(side='bottom')
button5.pack(side='right')
root.mainloop()
```

Для создания сложной структуры с использованием этого упаковщика обычно используют Frame, вложенные друг в друга.

При применении этого упаковщика можно указать следующие аргументы:

side ("left"/"right"/"top"/"bottom") - к какой стороне должен примыкать размещаемый виджет.

fill (None/"x"/"y"/"both") - необходимо ли расширять пространство предоставляемое виджету.

expand (True/False) - необходимо ли расширять сам виджет, чтобы он занял всё предоставляемое ему пространство.

in_ - явное указание в какой родительский виджет должен быть помещён.

Дополнительные функции

pack_configure - синонимдлярack.

pack_slaves (синоним slaves) - возвращает список всех дочерних упакованных виджетов.

pack_info - возвращает информацию о конфигурации упаковки.

pack_propagate (синоним propagate) (True/False) - включает/отключает распространении информации о геометрии дочерних виджетов. По умолчанию виджет изменяет свой размер в соответствии с размером своих потомков. Этот метод может отключить такое поведение (pack_propagate(False)). Это может быть полезно, если необходимо, чтобы виджет имел фиксированный размер и не изменял его по прихоти потомков.

pack_forget (синоним forget) - удаляет виджет и всю информацию о его расположении из упаковщика. Позднее этот виджет может быть снова размещён.

grid()

Этот упаковщик представляет собой таблицу с ячейками, в которые помещаются виджеты.

Аргументы

row - номер строки, в который помещаем виджет.

rowspan - сколько строк занимает виджет

column - номер столбца, в который помещаем виджет.

columnspan - сколько столбцов занимает виджет.

padx / pady - размер внешней границы (бордюра) по горизонтали и вертикали.

ipadx / ipady - размер внутренней границы (бордюра) по горизонтали и вертикали. Разница между pad и ipad в том, что при указании pad расширяется свободное пространство, а при ipad расширяется помещаемый виджет.

sticky ("n", "s", "e", "w" или их комбинация) - указывает к какой границе "приклеивать" виджет. Позволяет расширять виджет в указанном направлении. Границы названы в соответствии со сторонами света. "n" (север) - верхняя граница, "s" (юг) - нижняя, "w" (запад) - левая, "e" (восток) - правая.

in_ - явное указание в какой родительский виджет должен быть помещён.

Для каждого виджета указываем, в какой он находится строке, и в каком столбце. Если нужно, указываем, сколько ячеек он занимает (если, например, нам нужно разместить три виджета под одним, необходимо "растянуть" верхний на три ячейки). Пример:

```
entry1.grid(row=0,column=0,columnspan=3)
button1.grid(row=1,column=0)
button2.grid(row=1,column=1)
button3.grid(row=1,column=2)
```

Дополнительные функции

grid_configure - синонимдлягrid.

grid_slaves (синоним slaves) - см. pack_slaves.

grid_info - см.pack_info.

grid_propagate (синоним propagate) - см. pack_propagate.

grid_forget (синоним forget) - см. pack_forget.

grid_remove - удаляет виджет из-под управления упаковщиком, но сохраняет информацию об упаковке. Этот метод удобно использовать для временного удаления виджета.

grid_bbox (синоним bbox) - возвращает координаты (в пикселях) указанных столбцов и строк.

grid_location (синоним location) - принимает два аргумента: x и y (в пикселях). Возвращает номер строки и столбца в которые попадают указанные координаты, либо -1 если координаты попали вне виджета.

grid_size (синоним size) - возвращает размер таблицы в строках и столбцах.

grid_columnconfigure (синоним columnconfigure) и grid_rowconfigure (синоним rowconfigure) - важные функции для конфигурирования упаковщика. Методы принимают номер строки/столбца и аргументы конфигурации. Список возможных аргументов:

minsize - минимальная ширина/высота строки/столбца.

weight - "вес" строки/столбца при увеличении размера виджета. 0 означает, что строка/столбец не будет расширяться. Строка/столбец с "весом" равным 2 будет расширяться вдвое быстрее, чем с весом 1.

uniform - объединение строк/столбцов в группы. Строки/столбцы имеющие одинаковый параметр uniform будут расширяться строго в соответствии со своим весом.

pad - размер бордюра. Указывает, сколько пространства будет добавлено к самому большому виджету в строке/столбце.

Пример, текстовый виджет с двумя полосами прокрутки:

```
from tkinter import *
root=Tk()
text = Text(wrap=NONE)
vscrollbar = Scrollbar(orient='vert', command=text.yview)
text['yscrollcommand'] = vscrollbar.set
hscrollbar = Scrollbar(orient='hor', command=text.xview)
text['xscrollcommand'] = hscrollbar.set
# размещаемвиджеты
text.grid(row=0, column=0, sticky='nsew')
vscrollbar.grid(row=0, column=1, sticky='ns')
hscrollbar.grid(row=1, column=0, sticky='ew')
# конфигурируем упаковщик, чтобы текстовый виджет расширялся
root.rowconfigure(0, weight=1)
root.columnconfigure(0, weight=1)
root.mainloop()
```

place()

place представляет собой простой упаковщик, позволяющий размещать виджет в фиксированном месте с фиксированным размером. Также он позволяет указывать координаты размещения в относительных единицах для реализации "резинового" размещения. При использовании этого упаковщика, нам необходимо указывать координаты каждого виджета. Например:

```
button1.place(x=0,y=0)
```

Этот упаковщик, хоть и кажется неудобным, предоставляет полную свободу в размещении виджетов на окне.

Аргументы

anchor ("n", "s", "e", "w", "ne", "nw", "se", "sw" или "center") - какой угол или сторона размещаемого виджета будет указана в аргументах x/y/relx/rely. По умолчанию "nw" - левый верхний

bordermode ("inside", "outside", "ignore") - определяет в какой степени будут учитываться границы при размещении виджета.

in_ - явное указание в какой родительский виджет должен быть помещён.

x и y - абсолютные координаты (в пикселях) размещения виджета.

width и height - абсолютные ширина и высота виджета.

relx и rely - относительные координаты (от 0.0 до 1.0) размещения виджета.
relwidth и relheight - относительные ширина и высота виджета.

Относительные и абсолютные координаты (а также ширину и высоту) можно комбинировать. Так например, relx=0.5, x=-2 означает размещение виджета в двух пикселях слева от центра родительского виджета, relheight=1.0, height=-2 - высота виджета на два пикселя меньше высоты родительского виджета.

Дополнительные функции
place_slaves, place_forget, place_info - см. описание аналогичных методов упаковщика pack.

Привязка событий

Обычно, чтобы приложение с графическим интерфейсом что-то делало, должны происходить те или иные события, чаще всего представляющие собой воздействие человека на элементы GUI.

Можно выделить три основных типа событий: производимые мышью, нажатиями клавиш на клавиатуре, а также события, возникающие в результате изменения виджетов. Нередко обрабатываются сочетания.

bind()

Метод bind() привязывает событие к какому-либо действию (нажатие кнопки мыши, нажатие клавиши на клавиатуре и т.д.). bind принимает три аргумента:

-название события

-функцию, которая будет вызвана при наступлении события

-третий аргумент (необязательный) - строка "+" - означает, что эта привязка добавляется к уже существующим. Если третий аргумент опущен или равен пустой строке - привязка замещает все другие привязки данного события к виджету.

Метод bind возвращает идентификатор привязки, который может быть использован в функции unbind.

Если bind привязан к окну верхнего уровня, то Tkinter будет обрабатывать события всех виджетов этого окна.

При вызове метода bind событие передается в качестве первого аргумента. Название события заключается в кавычки, а также в угловые скобки < и >. События описывается с помощью зарезервированных ключевых слов.

Часто используемые события, производимые мышью:

Раздаточный материал № 164

<Button-1> – клик левой кнопкой мыши
<Button-2> – клик средней кнопкой мыши
<Button-3> – клик правой кнопкой мыши
<Double-Button-1> – двойной клик левой кнопкой мыши
<Motion> – движение мыши
и т. д.