Урок 3. Метод pack()

Прежде чем продолжить знакомство с виджетами GUI остановимся на вопросе их расположения в окне. Это важный вопрос, так как от интуитивности интерфейса во многом зависит удобство использования программы. Организуя виджеты в пространстве, программист отчасти становится дизайнером, разработчиком интерфейсов.

В Tkinter существует три так называемых менеджера геометрии – упаковщик, сетка и размещение по координатам. В этом уроке будет рассмотрен первый как наиболее простой и часто используемый, остальные два упомянем позже.

Упаковщик (packer) вызывается методом pack(), который имеется у всех виджетов-объектов. Мы уже использовали его. Если к элементу интерфейса не применить какой-либо из менеджеров геометрии, то он не отобразится в окне. При этом в одном окне (или любом другом родительском виджете) нельзя комбинировать разные менеджеры. Если вы начали размещать виджеты методом pack(), то не надо тут же использовать методы grid() (сетка) и place() (место).

Если в упаковщики не передавать аргументы, то виджеты будут располагаться вертикально, друг над другом. Тот объект, который первым вызовет pack(), будет вверху. Который вторым – под первым, и так далее.

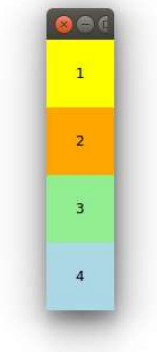
У метода pack() есть параметр side (сторона), который принимает одно из четырех значений-констант tkinter – TOP, BOTTOM, LEFT, RIGHT (верх, низ, лево, право). По умолчанию, когда в pack() не указывается side, его значение равняется TOP. Из-за этого виджеты располагаются вертикально.

Создадим четыре раскрашенные метки

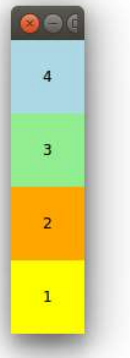
```
…
l1 = Label(width=7, height=4, bg='yellow', text="1")
l2 = Label(width=7, height=4, bg='orange', text="2")
l3 = Label(width=7, height=4, bg='lightgreen', text="3")
l4 = Label(width=7, height=4, bg='lightblue', text="4")
…
```

и рассмотрим разные комбинации значений сайда:

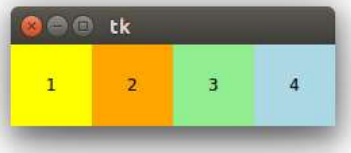
```
l1.pack()
l2.pack()
l3.pack()
l4.pack()
```



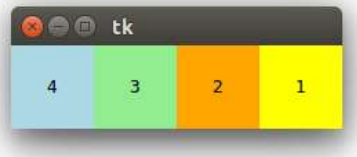
```
l1.pack(side=BOTTOM)
l2.pack(side=BOTTOM)
l3.pack(side=BOTTOM)
l4.pack(side=BOTTOM)
```



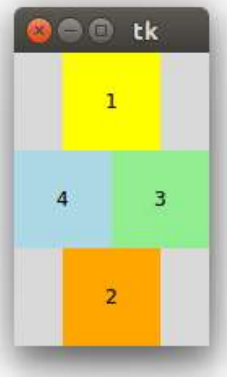

```
l1.pack(side=LEFT)
l2.pack(side=LEFT)
l3.pack(side=LEFT)
l4.pack(side=LEFT)
```



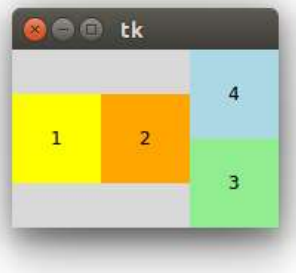

```
l1.pack(side=RIGHT)
l2.pack(side=RIGHT)
l3.pack(side=RIGHT)
l4.pack(side=RIGHT)
```




```
l1.pack(side=TOP)
l2.pack(side=BOTTOM)
l3.pack(side=RIGHT)
l4.pack(side=LEFT)
```




```
l1.pack(side=LEFT)
l2.pack(side=LEFT)
l3.pack(side=BOTTOM)
l4.pack(side=LEFT)
```



Проблема двух последних вариантов в том, что если надо разместить виджеты квадратом, т. е. два сверху, два снизу ровно под двумя верхними, то сделать это проблематично, если

вообще возможно. Поэтому прибегают к вспомогательному виджету – фрейму (рамке), который порождается от класса Frame.

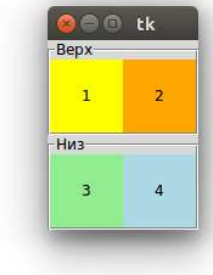
Фреймы размещают на главном окне, а уже в фреймах – виджеты:

```
from tkinter import *
root = Tk()
f_top = Frame(root) # root можно не указывать
f_bot = Frame(root)
l1 = Label(f_top, width=7, height=4, bg='yellow', text="1")
l2 = Label(f_top, width=7, height=4, bg='orange', text="2")
l3 = Label(f_bot, width=7, height=4, bg='lightgreen', text="3")
l4 = Label(f_bot, width=7, height=4, bg='lightblue', text="4")

f_top.pack()
f_bot.pack()
l1.pack(side=LEFT)
l2.pack(side=LEFT)
l3.pack(side=LEFT)
l4.pack(side=LEFT)

root.mainloop()
```

Результат:

Кроме Frame существует похожий класс LabelFrame – фрейм с подписью. В отличие от простого фрейма у него есть свойство text.

```
…
f_top = LabelFrame(text="Верх")
f_bot = LabelFrame(text="Низ")
…
```



Кроме side у pack() есть другие параметры-свойства. Можно задавать внутренние (ipadx и ipady) и внешние (padx и pady) отступы:

```
f_top.pack(padx=10, pady=10)

l1.pack(side=LEFT)
l2.pack(side=LEFT)
```


```
f_top.pack(ipadx=10, ipady=10)

l1.pack(side=LEFT)
l2.pack(side=LEFT)


root.mainloop()
```

Когда устанавливаются внутренние отступы, то из-за того, что side прибивает виджет к левой границе, справа получаем отступ в 20 пикселей, а слева – ничего. Можно частично решить проблему, заменив внутренние отступы рамки на внешние отступы у меток.

```
f_top.pack()
l1.pack(side=LEFT, padx=10, pady=10)
l2.pack(side=LEFT, padx=10, pady=10)
```

Но тут появляется промежуток между самими метками. Чтобы его убрать, пришлось бы каждый виджет укладывать в свой собственный фрейм. Отсюда делаем вывод, что упаковщик Tkinter удобен только для относительно простых интерфейсов.

Следующие два свойства – fill (заполнение) и expand (расширение). По-умолчанию expand равен нулю (другое значение – единица), а fill – NONE (другие значения BOTH, X, Y). Создадим окно с одной меткой:

```
from tkinter import *
root = Tk()
l1 = Label(bg="lightgreen", width=30, height=10, text="This is a label")
l1.pack()
root.mainloop()
```

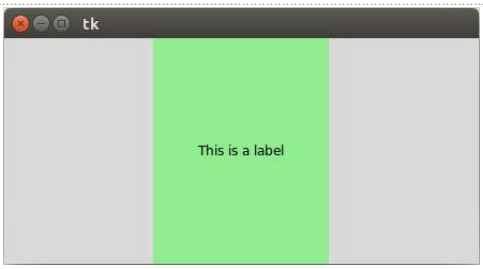
Если начать расширять окно или сразу раскрыть его на весь экран, то метка окажется вверху по вертикали и в середине по горизонтали. Причина, по которой метка не в середине по вертикали заключается в том, что side по-умолчанию равен TOP, и метку прибивает к верху.

Если установить свойство expand в 1, то при расширении окна метка будет всегда в середине:

```
…
l1.pack(expand=1)
…
```

Свойство fill заставляет виджет заполнять все доступное пространство. Заполнить его можно во всех направлениях или только по одной из осей:

```
…
l1.pack(expand=1, fill=Y)
…
```



Последняя опция метода pack() – anchor (якорь) – может принимать значения N (north – север), S (south – юг), W (west – запад), E (east – восток) и их комбинации:

```
…
l1.pack(expand=1, anchor=SE)
…
```

Практическая работа

Перепишите программу из практической работы предыдущего урока так, чтобы интерфейс выглядел примерно следующим образом: