Дисциплина:
Программирование

Тема 2: Разработка графического интерфейса в «Python»

Преподаватель: канд. техн. наук, доцент
Кромина Людмила Александровна

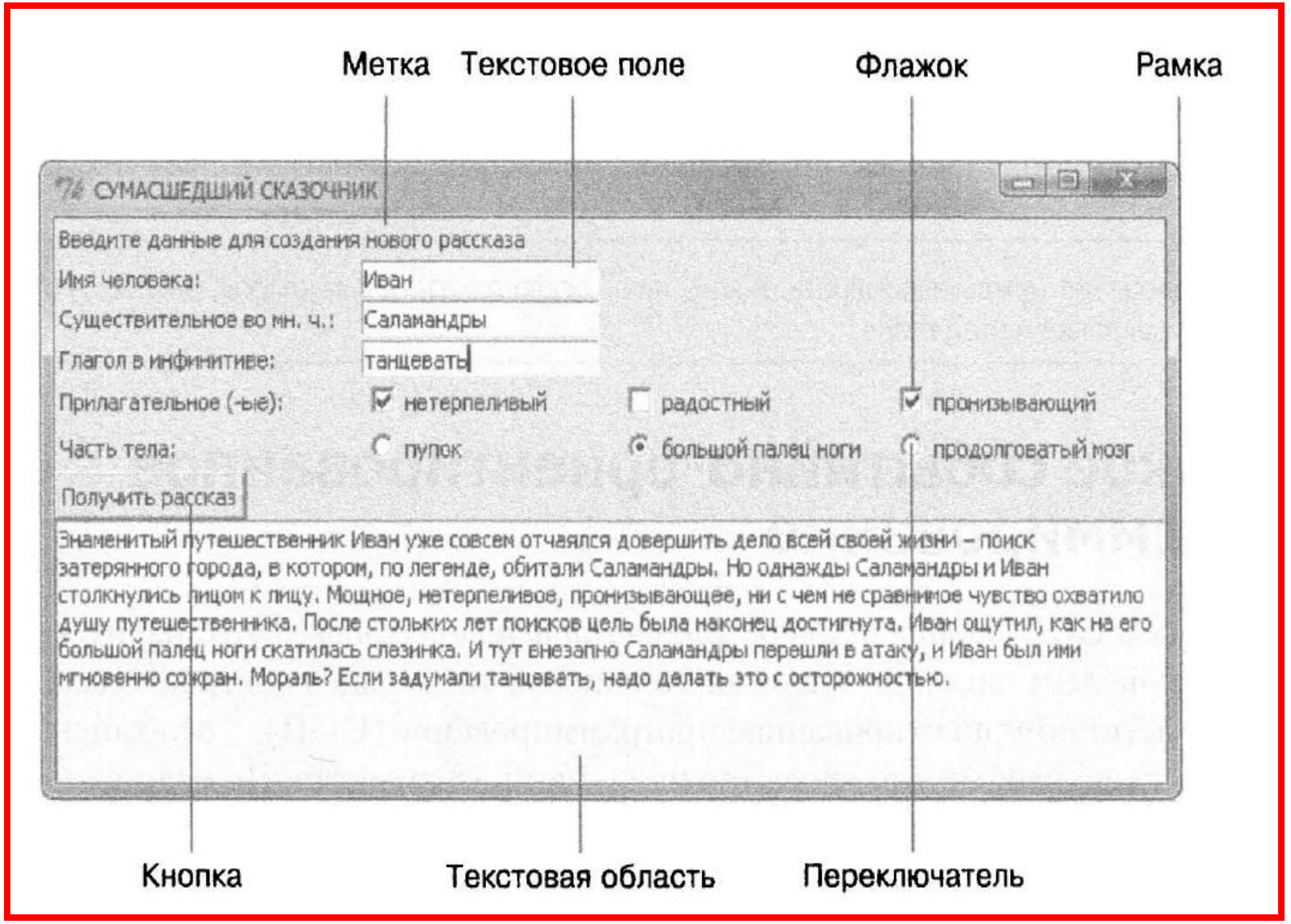
Понятие графического интерфейса пользователя , его назначение

ГРАФИЧЕСКИЙ ИНТЕРФЕЙС ПОЛЬЗОВАТЕЛЯ (GRAPHICAL USER INTERFACE, GUI) представляет собой разновидность интерфейса, в котором пользователь имеет произвольный доступ ко всем элементам интерфейса и осуществляет непосредственное манипулирование ими

Почти все графические интерфейсы общего назначения строятся по модели **WIMP – WINDOW, ICON, MENU, POINTER (ОКНО, ИКОНКА, МЕНЮ, УКАЗАТЕЛЬ)**. Внутри окон размещаются элементы графического интерфейса, называемые «**ВИДЖЕТАМИ**»

Основное предназначение графического интерфейса заключается в упрощение взаимодействия пользователя с программой

Понятие графического интерфейса пользователя , его назначение



Работа с модулем tkinter в языке программирования «Python»

Одной из стандартных библиотек, позволяющих создавать приложения на языке программирования Python, с графическим интерфейсом является библиотека *tkinter*

РАЗРАБОТКА ГРАФИЧЕСКОГО ИНТЕРФЕЙСА НАЧИНАЕТСЯ С СОЗДАНИЯ ГЛАВНОГО ОКНА И ПРЕДУСМАТРИВАЕТ ВЫПОЛНЕНИЕ СЛЕДУЮЩИХ ДЕЙСТВИЙ:

1. Подключение модуля tkinter, что может быть выполнено одним из двух способов:

1) import tkinter

2) from tkinter import *

в дальнейшем будем пользоваться только вторым способом, поскольку это позволит не указывать каждый раз имя модуля при обращении к объектам, которые в нем содержатся;

2. Создание графического окна программы, для этого применяется следующая инструкция:

имя окна=Tk()

в качестве имени окна принято указывать название root (хотя его можно назвать как угодно).

3. Указание инструкций, задающих значения свойств окна.

4. Отображение графического окна на экране, которое может быть выполнено через применение следующей инструкции:

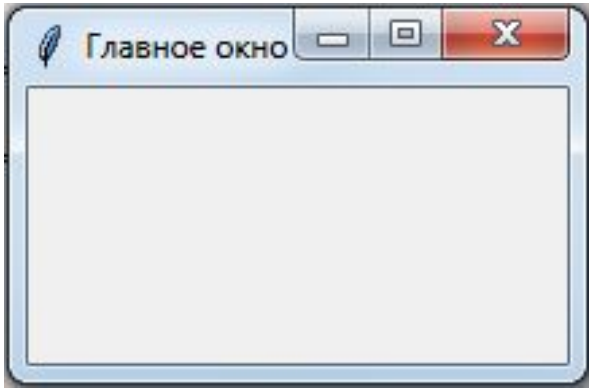
root.mainloop()

данный метод запускает цикл обработки событий окна для взаимодействия с пользователем

Работа с модулем tkinter в языке программирования «Python»

ТАКИМ ОБРАЗОМ, ПРИМЕР ПРОГРАММНОГО КОДА ДЛЯ СОЗДАНИЯ ГЛАВНОГО ОКНА МОЖЕТ ИМЕТЬ ВИД:

```
from tkinter import* # подключение модуля tkinter
root=Tk() # создание графического окна программы
root.title("Главное окно") # указание названия заголовка
root.geometry("200x100") # указание размеров окна
root.mainloop() # отображение графического окна на экране
```



Работа с модулем tkinter в языке программирования «Python»

Создаваемое окно присваивается переменной ***root***, позволяющей осуществлять управление атрибутами окна.
Так, например, метод ***title()*** служит для указания заголовка окна, а метод ***geometry()*** – для указания размера окна (ширины и высоты соответственно), следует отметить, что если при создании окна метод ***geometry()*** не вызывается, то окно занимает пространство, необходимое и достаточное для размещения внутреннего содержимого.
По умолчанию окно разрабатываемого приложения размещается в левом верхнем углу экрана, однако его местоположение можно изменить, указав необходимые значения в методе ***geometry()***, например:

root.geometry("200x100+500+250")

Теперь окно будет размещаться на 500 пикселей вправо и на 250 пикселей вниз от верхнего левого угла экрана

Виджеты графического интерфейса

Главное окно приложения может содержать все остальные элементы или, как уже было отмечено ранее – **ВИДЖЕТЫ**, к которым относятся:

1. ***Label*** – надпись, содержащая текст или графическое изображение;
2. ***Canvas*** – рисунок, являющийся основой для вывода графических примитивов;
3. ***Checkbutton*** – кнопка (флажок), которая, при ее нажатии, переключается между двумя состояниями;
4. ***Entry*** – горизонтальное поле для ввода текста;
5. ***Frame*** – рамка, содержащая в себе другие визуальные компоненты;
6. ***Button*** – простая кнопка для выполнения определенных действий;
7. ***Listbox*** – прямоугольная рамка со списком, предназначенная для выбора одного или несколько элементов;
8. ***Menu*** – элемент, с помощью которого можно создавать всплывающие (**popup**) и ниспадающие (***pulldown***) меню;

Виджеты графического интерфейса

Главное окно приложения может содержать все остальные элементы или, как уже было отмечено ранее – **ВИДЖЕТЫ**, к которым относятся:

9. ***Menubutton*** – кнопка с ниспадающим меню;
10. ***Message*** – сообщение, аналогичное надписи, и позволяющее заворачивать длинные строки и менять размер по требованию менеджера расположения;
11. ***Radiobutton*** – кнопка для представления одного из нескольких альтернативных значений;
12. ***Scale*** – шкала, служащая для задания числового значения путем перемещения движка в определенном диапазоне;
13. ***Scrollbar*** – полоса прокрутки, служащая для отображения величины прокрутки в других виджетах, может быть как вертикальной, так и горизонтальной;
14. ***Text*** – форматированный текст, представленный в виде прямоугольного виджета и позволяющий редактировать и форматировать текст с использованием различных стилей, внедрять в текст рисунки и даже окна;
15. ***Toplevel*** – окно верхнего уровня, представленное в виде отдельного окна и содержащее внутри себя другие виджеты

Методы виджетов

1. Виджет ***Toplevel*** – окно верхнего уровня. Обычно используется для создания многоконных программ, а также для диалоговых окон

ИМЕЕТ СЛЕДУЮЩИЕ МЕТОДЫ:

1) ***title()*** – заголовок окна;
2) ***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);
3) ***iconify() / deiconify()*** – свернуть / развернуть окно;
4) ***withdraw()*** – "спрятать" (сделать невидимым) окно. Для того, чтобы снова показать его надо использовать метод ***deiconify()***;
5) ***minsize()*** *и* ***maxsize()*** – минимальный / максимальный размер окна. Методы принимают два аргумента – ширина и высота окна. Если аргументы не указаны – возвращают текущее значение;
6) ***state()*** – получить текущее значение состояния окна. Может возвращать следующие значения: normal (нормальное состояние), icon (показано в виде иконки), iconic (свернуто), withdrawn (не показано), zoomed (развернуто на полный экран);
7) ***resizable()*** – может ли пользователь изменять размер окна. Принимает два аргумента – возможность изменения размера по горизонтали и по вертикали. Без аргументов возвращает текущее значение;

Методы виджетов

8) ***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);
9) ***transient()*** – сделать окно зависимым от другого окна, указанного в аргументе. Будет сворачиваться вместе с указанным окном. Без аргументов возвращает текущее значение;
10) ***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);
11) ***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;
12) ***grab_set()*** – устанавливает фокус на окно, даже при наличии открытых других окон;
13) ***grab_release()*** – снимает монопольное владение фокусом ввода с окна

Следует отметить, что рассмотренные методы также могут быть применимы для корневого (*root*) окна

Методы виджетов

2. Виджет ***Button*** – обыкновенная кнопка

ИМЕЕТ СЛЕДУЮЩИЕ МЕТОДЫ:

1) ***text*** – текст, отображенный на кнопке;
2) ***width, height*** – соответственно, ширина и высота кнопки;
3) ***bg*** – цвет кнопки (сокращенно от *background*);
4) ***fg*** – цвет текста на кнопке (сокращенно от *foreground*);
5) ***font*** – шрифт и его размер

3. Виджет ***Label*** – это виджет, предназначенный для отображения какой-либо надписи без возможности редактирования пользователем

ИМЕЕТ СЛЕДУЮЩИЕ МЕТОДЫ:

1) ***text*** – текст, отображенный на виджете Label;
2) ***width, height*** – соответственно, ширина и высота виджета Label;
3) ***bg*** – цвет виджета Label (сокращенно от background);
4) ***fg*** – цвет текста на виджете Label (сокращенно от foreground);
5) ***font*** – шрифт и его размер

Методы виджетов

4. Виджет ***Entry*** – это виджет, позволяющий пользователю ввести (или вывести с помощью метода ***insert()***) одну строку текста. Имеет дополнительное свойство ***bd*** (сокращенно от ***borderwidth***), позволяющее регулировать ширину границы

ИМЕЕТ СЛЕДУЮЩИЕ МЕТОДЫ:

1) ***borderwidth*** – ширина бордюра элемента (bd – сокращение от borderwidth);
2) ***width*** – задает длину элемента в знакоместах

5. Виджет ***Text*** – это виджет, который позволяет пользователю ввести (или вывести с помощью метода ***insert()***) многострочный текст. Имеет дополнительное свойство wrap, отвечающее за перенос чтобы, например, переносить по словам, нужно использовать значение WORD

6. Виджет ***Listbox*** – это виджет, который представляет из себя 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

Методы виджетов

7. Виджет ***Scale*** (шкала) – это виджет, позволяющий выбрать какое-либо значение из заданного диапазона

ИМЕЕТ СЛЕДУЮЩИЕ МЕТОДЫ:

1) ***orient*** – как расположена шкала на окне. Возможные значения: *HORIZONTAL, VERTICAL* (горизонтально, вертикально);
2) ***length*** – длина шкалы;
3) ***from_*** – с какого значения начинается шкала;
4) ***to*** – каким значением заканчивается шкала;
5) ***tickinterval*** – интервал, через который отображаются метки шкалы;
6) ***resolution*** – шаг передвижения (минимальная длина, на которую можно передвинуть движок)

Менеджеры расположения виджетов в tkinter

МЕНЕДЖЕР РАСПОЛОЖЕНИЯ (МЕНЕДЖЕР ГЕОМЕТРИИ ИЛИ УПАКОВЩИК) – представляет собой специальный механизм, который размещает (упаковывает) виджеты на окне

В модуле tkinter языка программирования «Python» имеют место три упаковщика:

1) pack();
2) place();
3) grid().

Необходимо отметить, что в одном виджете можно использовать только один тип упаковки, при смешивании разных типов упаковки программа будет выдавать ошибку

Менеджеры расположения виджетов в tkinter

1. Менеджер ***pack()*** – имеет максимальные функциональные возможности.

ПРИ ЕГО ПРИМЕНЕНИИ МОЖНО УКАЗАТЬ СЛЕДУЮЩИЕ АРГУМЕНТЫ:

1) ***side ("left"/"right"/"top"/"bottom")*** – к какой стороне должен примыкать размещаемый виджет;

2) ***fill (None/"x"/"y"/"both")*** – необходимо ли расширять пространство предоставляемое виджету;

3) ***expand (True/False)*** – необходимо ли расширять сам виджет, чтобы он занял все предоставляемое ему пространство;

4) ***in_*** – явное указание в какой родительский виджет должен быть помещен

ДОПОЛНИТЕЛЬНЫЕ МЕТОДЫ ДЛЯ РАБОТЫ С МЕНЕДЖЕРОМ РАСПОЛОЖЕНИЯ *pack*():

1) ***pack_configure()*** – синоним для ***pack()***;

2) ***pack_slaves()*** (синоним ***slaves()***) – возвращает список всех дочерних упакованных виджетов;

3) ***pack_info*** – возвращает информацию о конфигурации упаковки;

4) ***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;

5) ***pack_forget()*** (синоним ***forget()***) – удаляет виджет и всю информацию о его расположении из упаковщика

Менеджеры расположения виджетов в tkinter

2. Менеджер ***place()*** представляет собой простой упаковщик, позволяющий размещать виджет в фиксированном месте с фиксированным размером. Также он позволяет указывать координаты размещения. При использовании данного упаковщика, необходимо указывать координаты каждого виджета.

ИМЕЕТ АРГУМЕНТЫ:

1) ***anchor ("n", "s", "e", "w", "ne", "nw", "se", "sw" или "center")*** – какой угол или какая сторона размещаемого виджета будет указана в аргументах x/y/relx/rely. По умолчанию "nw" – левый верхний угол;
2) ***bordermode ("inside", "outside", "ignore")*** – определяет в какой степени будут учитываться границы при размещении виджета;
3) ***in_*** – явное указание в какой родительский виджет должен быть помещен;
4) ***x*** и ***y*** – абсолютные координаты (в пикселях) размещения виджета;
5) ***width*** и ***height*** – абсолютные ширина и высота виджета;
6) ***relx*** и ***rely*** – относительные координаты (от 0.0 до 1.0) размещения виджета;
7) ***relwidth*** и ***relheight*** – относительные ширина и высота виджета

Следует отметить, что применение менеджера расположения *place()* позволяет комбинировать относительные и абсолютные координаты (а также ширину и высоту). 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

Менеджеры расположения виджетов в tkinter

ДОПОЛНИТЕЛЬНЫЕ МЕТОДЫ ДЛЯ РАБОТЫ С МЕНЕДЖЕРОМ РАСПОЛОЖЕНИЯ *place()*:

1) ***place _slaves()*** (синоним ***slaves()***) – возвращает список всех дочерних упакованных виджетов;
2) ***place _info*** – возвращает информацию о конфигурации упаковки;
3) ***place _forget()*** (синоним ***forget()***) – удаляет виджет и всю информацию о его расположении из упаковщика

Менеджеры расположения виджетов в tkinter

3. Менеджер ***grid()*** – упаковщик, представляющий собой таблицу с ячейками, в которые помещаются виджеты

ИМЕЕТ АРГУМЕНТЫ:

1) ***row*** – номер строки, в который помещаем виджет;
2) ***rowspan*** – сколько строк занимает виджет;
3) ***column*** – номер столбца, в который помещаем виджет;
4) ***columnspan*** – сколько столбцов занимает виджет;
5) ***padx / pady*** – размер внешней границы (бордюра) по горизонтали и вертикали;
6) ***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;
7) ***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;
8) ***in_*** – явное указание в какой родительский виджет должен быть помещен

Для каждого виджета следует указывать строку, и столбец его местоположения. При этом, если необходимо, нужно указывать количество ячеек, которые он занимает (если, например, нужно разместить три виджета под одним, необходимо «растянуть» верхний на три ячейки)

Менеджеры расположения виджетов в tkinter

ДОПОЛНИТЕЛЬНЫЕ МЕТОДЫ ДЛЯ РАБОТЫ С МЕНЕДЖЕРОМ РАСПОЛОЖЕНИЯ *grid()*:

1) ***grid_configure()*** – синоним для ***grid()***;
2) ***grid_slaves()*** (синоним ***slaves()***) – возвращает список всех дочерних упакованных виджетов;
3) ***grid_info()*** – возвращает информацию о конфигурации упаковки;
4) ***grid_propagate()*** (синоним ***propagate()***) – см. pack_propagate();
5) ***grid_forget()*** (синоним ***forget()***) – удаляет виджет и всю информацию о его расположении из упаковщика;
6) ***grid_remove()*** – удаляет виджет из-под управления упаковщиком, но сохраняет информацию об упаковке. Этот метод удобно использовать для временного удаления виджета;
7) ***grid_bbox()*** (синоним ***bbox()***) – возвращает координаты (в пикселях) указанных столбцов и строк;

Менеджеры расположения виджетов в tkinter

ДОПОЛНИТЕЛЬНЫЕ МЕТОДЫ ДЛЯ РАБОТЫ С МЕНЕДЖЕРОМ РАСПОЛОЖЕНИЯ *grid()*:

8) ***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;

9) ***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;

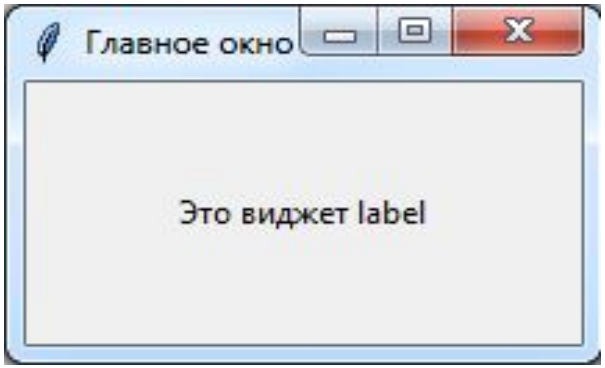
– ***weight*** – «вес» строки/столбца при увеличении размера виджета, 0 означает, что строка/столбец не будет расширяться; строка/столбец с «весом», равным 2 будет расширяться вдвое быстрее, чем с весом 1;

– ***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

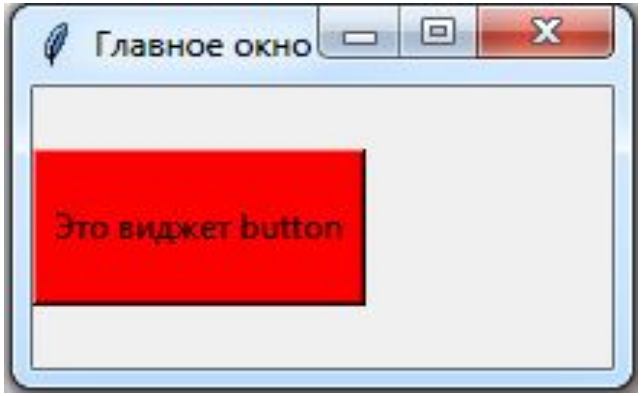
Пример применения менеджеров размещения к виджету Label

```
from tkinter import* # подключение модуля tkinter
root=Tk() # создание графического окна программы
root.title("Главное окно") # указание названия заголовка
root.geometry("200x100") # указание размеров окна
label1=Label(text="Это виджет label", width=15, height=20)# создание виджета Label и задание параметров
label1.pack()# отображение виджета с помощью менеджера pack
label1.pack(side=BOTTOM)# указание стороны, к которой должен примыкать виджет
root.mainloop() # отображение графического окна на экране
```



Пример применения менеджеров размещения к виджету Button

```
from tkinter import* # подключение модуля tkinter
root=Tk() # создание графического окна программы
root.title("Главное окно") # указание названия заголовка
root.geometry("200x100") # указание размеров окна
but1=Button(text="Это виджет Button", width=15, height=3, bg="#ff0000")#
создание виджета Button и задание параметров
but1.pack()# отображение виджета с помощью менеджера pack
but1.pack(side=LEFT)# указание стороны, к которой должен примыкать
виджет
root.mainloop() # отображение графического окна на экране
```



Обработка событий модуля Tkinter

Графическое приложение обычно ждет каких-либо внешних воздействий (щелчков кнопкой мыши, нажатий клавиш на клавиатуре, изменения виджетов) и затем выполняет заложенное программистом действие.
Из такого принципа работы можно вывести следующую схему настройки функциональности GUI:

НА ВИДЖЕТ ЧТО-ТО «ВЛИЯЕТ» ИЗ ВНЕ → ВЫПОЛНЯЕТСЯ КАКАЯ-ТО ФУНКЦИЯ (ДЕЙСТВИЕ)

Внешнее воздействие на графический компонент называется **СОБЫТИЕМ**

Обработка событий модуля Tkinter

Для большинства виджетов, реагирующих на действие пользователя, активацию виджета (например, нажатие кнопки) можно привязать к функции обработчику этого события, используя опцию ***command***

К таким виджетам относятся: ***Button, Checkbutton, Radiobutton, Spinbox, Scrollbar, Scale***

```
from tkinter import *
root = Tk()
Button(root, text="Выход", command=root.destroy).pack()
root.mainloop()
```

Способ, применяющий опцию ***command***, является предпочтительным и наиболее удобным способом привязки

command=root.destroy
Применяется для создания кнопки закрытия формы

Обработка событий модуля Tkinter

Другим способом привязки является использование метода ***bind()***, который привязывает событие к какому-либо действию (нажатие кнопки мыши, нажатие клавиши на клавиатуре и т.д.).
Для обработки событий, не зависимо от применяемого виджета, необходимо применять следующий синтаксис:

a.bind(e,f)

где, ***a*** – имя виджета, у которого настраивается реакция на событие ***e***, при наступлении события будет выполниться функция ***f***).
Выполняемая функция (если она «привязывается» к некоторому виджету методом ***bind***) должна быть описана в следующем виде:

```
def f(event):
    #тело функции
```

В случае, когда для этого используется метод ***command*** (функция будет срабатывать только про событии ***"<Button-1>"***), пишется без ***event***:

```
def f():
    #тело функции
```

Обработка событий модуля Tkinter

ОПИСАНИЕ СОБЫТИЯ:

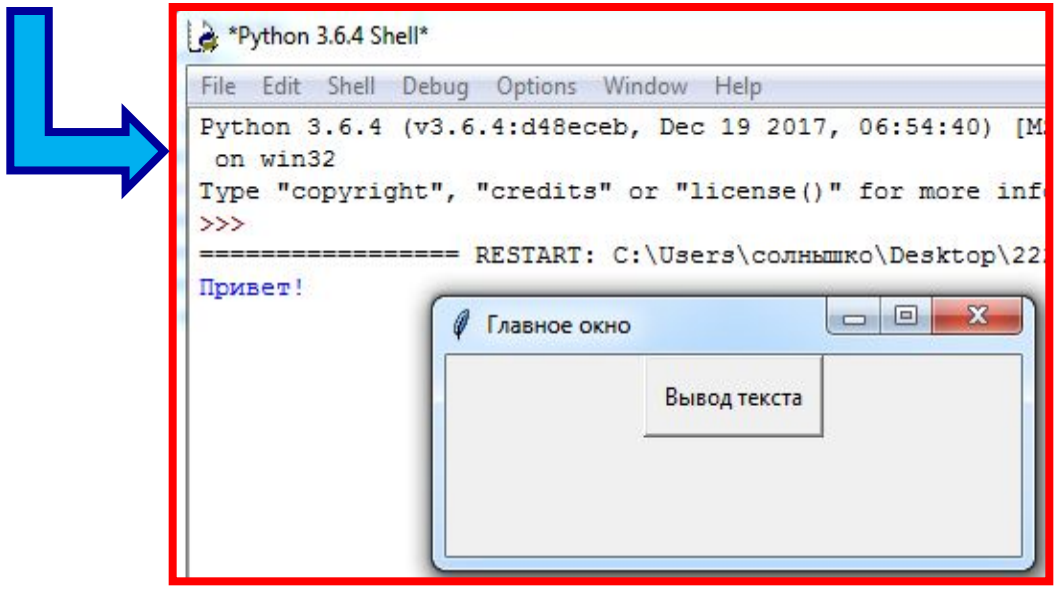
1. ***"<Button-m>"*** – события нажатия одной из кнопок мыши (левой, правой и колесика), при этом переменная ***m*** должна быть равна одному из значений:
1) ***1*** – нажата левая кнопка мыши (курсор над виджетом);
2) ***2*** – нажата правая кнопка мыши (курсор над виджетом);
3) ***3*** – нажато колесико мыши (курсор над виджетом)

2. ***"<k>"*** – событие нажатия клавиши, при этом ***k*** определяет название клавиши, которая нажата;

3. ***"<Bn-Motion>"*** – событие одновременного движения курсора мыши и нажатия на одну из кнопок мыши, при этом переменная n должна быть равна одному из значений:
1) 1 – нажата левая кнопка мыши во время движения курсора над виджетом;
2) 2 – нажата правая кнопка мыши во время движения курсора над виджетом;
3) 3 – нажато колесико мыши во время движения курсора над виджетом

Пример графического приложения, с активной кнопкой:

```
from tkinter import *
root=Tk()
root.title("Главное окно")
root.geometry('300x100')
def printText(event): # функция, выводящая приветствие print('Привет!')
button1=Button(root,text='Вывод текста',width=12, height=2)# создаем объект типа кнопка (Button)
button1.bind('<Button-1>',printText)# связываем нажатие ЛКМ по кнопке с функцией вывода приветствия
button1.pack()# отображаем кнопку в главном окне (это самый простой способ)
root.mainloop()
```



Пример графического приложения, с активной кнопкой:

```
from tkinter import * # подключение модуля tkinter
root = Tk()# создание графического окна программы
root.title('Сумма двух чисел') # указание названия заголовка

# первая метка в строке 0
label1=Label(text='Первое число') # создание виджета label1
label1.grid(row=0, sticky=W)# row – номер строки, в который помещаем виджет; sticky указывает к
какой границе "приклеивать" виджет

# вторая метка в строке 1
label2=Label(text='Второе число')
label2.grid(row=1, sticky=W)

# создание виджетов текстовых полей
Entry1 = Entry(width=10, font='Arial 14')
Entry2 = Entry(width=10, font='Arial 14')
Entry3 = Entry(width=20, font='Arial 14')

# размещение первых двух полей справа от меток, второй столбец (отсчет от нуля)
Entry1.grid(row=0, column=1, sticky=E)
Entry2.grid(row=1, column=1, sticky=E)

# третье текстовое поле ввода занимает всю ширину строки 2
# columnspan — объединение ячеек по столбцам; rowspan — по строкам
Entry3.grid(row=2, columnspan=2)
```

Пример графического приложения, с активной кнопкой:

```
def sum():
    a = Entry1.get() # взятие текста из первого поля
    a = int(a) # преобразование в число целого типа

    b = Entry2.get() # взятие текста из второго поля
    b = int(b) # преобразование в число целого типа

    result = str(a + b) # результат переведен в строку для дальнейшего вывода
    Entry3.delete(0, END) # очистка текстового поля полностью
    Entry3.insert(0, result) # вставка результата в начало


# размещение кнопки в строке 3 во втором столбце
but = Button(text='Сумма')
but = Button(text='Сумма', command=sum)
but.grid(row=3, column=1, sticky=E)

root.mainloop() # отображение графического окна на экране
```