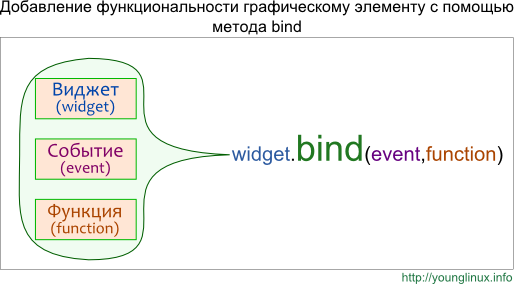
В tkinter с помощью метода bind между собой связываются виджет, событие и действие. Например, виджет – кнопка, событие – клик по ней левой кнопкой мыши, действие – отправка сообщения. Другой пример: виджет – текстовое поле, событие – нажатие Enter, действие – получение текста из поля методом get для последующей обработки программой. Действие оформляют как функцию или метод, которые вызываются при наступлении события.



У функций-обработчиков, которые вызываются через bind, а не через command, должен быть обязательный параметр event, через который передается событие. Имя event – соглашение, идентификатор может иметь другое имя, но обязательно должен стоять на первом месте в функции, или может быть вторым в методе.

Что делать, если в функцию надо передать дополнительные аргументы? Например, клик левой кнопкой мыши по метке устанавливает для нее один шрифт, а клик правой кнопкой мыши – другой. Можно написать две разные функции:

|  |
| --- |
| **from** tkinter **import** \* root = Tk() **def** font1(event):  l[**'font'**] = **"Verdana"  def** font2(event):  l[**'font'**] = **"Times"** l = Label(text=**"Hello World"**) l.bind(**'<Button-1>'**, font1) *# ЛКМ* l.bind(**'<Button-3>'**, font2) *# ПКМ* l.pack() root.mainloop() |

## Виды событий

Можно выделить три основных типа событий:

* производимые мышью,
* нажатиями клавиш на клавиатуре,
* события, возникающие в результате изменения виджетов.

Часто используемые события, производимые мышью:

* <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.

|  |
| --- |
| **from** tkinter **import** \* **def** b1(event):  root.title(**"Левая кнопка мыши"**) **def** b3(event):  root.title(**"Правая кнопка мыши"**) **def** move(event):  x = event.x  y = event.y  s = **"Движение мышью {}x{}"**.format(x, y)  root.title(s)  root = Tk() root.minsize(width = 500, height=400)  root.bind(**'<Button-1>'**, b1) root.bind(**'<Button-3>'**, b3) root.bind(**'<Motion>'**, move)  root.mainloop() |

## [#](https://learn4kid-python.firebaseapp.com/tkinter_2/tkinter_events/" \l "события-клавиатуры) События клавиатуры

Клавиатура в стиле ПК, специальные клавиши: Cancel (the Break key), BackSpace, Tab, Return(the Enter key), space Shift\_L (any Shift key), Control\_L (any Control key), Alt\_L (any Alt key), Pause, Caps\_Lock, Escape, Prior (Page Up), Next (Page Down), End, Home, Left, Up, Right, Down, Print, Insert, Delete, F1, F2, F3, F4, F5, F6, F7, F8, F9, F10, F11, F12, Num\_Lock, and Scroll\_Lock.

<Key> Пользователь нажал любую клавишу. Ключ предоставляется в символе объекта события, переданного в обратный вызов (это пустая строка для специальных ключей).

<a> Пользователь нажал "а". Можно использовать большинство печатных символов как есть. Исключения составляют пробел (<space>) и меньше (<Меньше>). Обратите внимание, что 1 - это привязка клавиатуры, а <1> - это привязка мыши.

<Motion> Пользователь полностью переместил указатель мыши внутри виджета.

Используя следующий пример кода можно узнать информацию о клавишах клавиатуры которые нажаты:

|  |
| --- |
| **from** tkinter **import** \* **def** show\_key(event):  root.title(str(event)) root = Tk() root.bind(**'<Key>'**, show\_key) root.mainloop() |

# Canvas - Холст

Виджет Canvas() - Холст используется для добавления структурированной графики в приложение.

Синтаксис создания виджета подобен, другим объектам tkinter:

widget\_name = Canvas(parent, options)

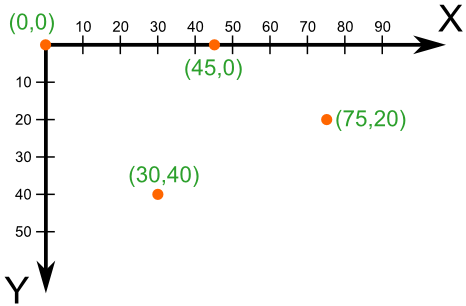
* parent - окно или фрейм (рамка), в котором будет размещаться холст;
* options - параметры настройки холста.

**Базовые параметры**

1. bd - представляет ширину границы. Ширина по умолчанию - 2.
2. bg - он представляет собой цвет фона холста.
3. confine - он настроен на то, чтобы холст нельзя было прокручивать за пределами области прокрутки.
4. cursor - курсор используется как стрелка, круг, точка и т.д. На холсте.
5. height - представляет размер холста в вертикальном направлении.
6. highlightcolor - представляет цвет выделения, когда виджет находится в фокусе.
7. relief - представляет собой тип бордюра. Возможные значения: SUNKEN, RAISED, GROOVE и RIDGE.
8. scrollregion - представляет координаты, заданные как кортеж, содержащий область холста.
9. width Он представляет собой ширину холста.
10. xscrollincrement - если задано положительное значение. Холст размещается только с кратным этому значению.
11. xscrollcommand - если холст можно прокручивать, этот атрибут должен быть методом .set() горизонтальной полосы прокрутки.
12. yscrollincrement - работает как xscrollincrement, но управляет вертикальным перемещением.
13. yscrollcommand - если холст можно прокручивать, этот атрибут должен быть методом .set() вертикальной полосы прокрутки.

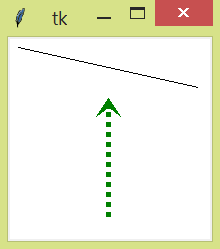
В tkinter от класса Canvas() создаются объекты-холсты, на которых можно «рисовать», размещая различные фигуры и объекты. Делается это с помощью вызовов соответствующих методов.

При создании экземпляра Canvas необходимо указать его ширину и высоту. При размещении геометрических примитивов и других объектов указываются их координаты на холсте. Точкой отсчёта является верхний левый угол.



В программе ниже создается холст. На нем с помощью метода **create\_line()** рисуются отрезки. Сначала указываются координаты начала (x1, y1), затем – конца (x2, y2):

|  |
| --- |
| **from** tkinter **import** \* root = Tk() c = Canvas(root, width=200, height=200, bg=**'white'**) c.pack() c.create\_line(10, 10, 190, 50) c.create\_line(100, 180, 100, 60,  fill=**'green'**, width=5, arrow=LAST,  dash=(10,2), activefill=**'lightgreen'**,  arrowshape=**"10 20 10"**) root.mainloop() |



Остальные свойства являются необязательными. Так activefill определяет цвет отрезка при наведении на него курсора мыши.

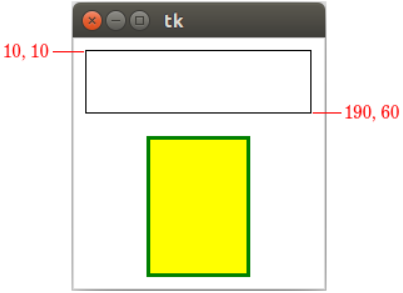
Создание прямоугольников методом **create\_rectangle():**

…

c.create\_rectangle(10, 10, 190, 60)

c.create\_rectangle(60, 80, 140, 190, fill='yellow', outline='green', width=3, activedash=(5, 4))

…



Первые координаты – верхний левый угол, вторые – правый нижний. В приведенном примере, когда на второй прямоугольник попадает курсор мыши, его рамка становится пунктирной, что определяется свойством activedash.

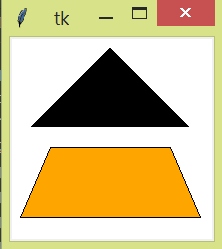
Методом **create\_polygon()** рисуется произвольный многоугольник путем задания координат каждой его точки:

…

c.create\_polygon(100, 10, 20, 90, 180, 90)

c.create\_polygon(40, 110, 160, 110, 190, 180, 10, 180, fill='orange', outline='black')

…



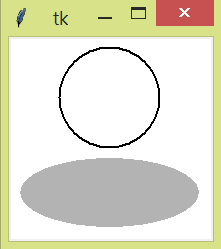
Метод **create\_oval()** создает эллипсы. При этом задаются координаты гипотетического прямоугольника, описывающего эллипс. Если нужно получить круг, то соответственно описываемый прямоугольник должен быть квадратом:

…

c.create\_oval(50, 10, 150, 110, width=2)

c.create\_oval(10, 120, 190, 190, fill='grey70', outline='white')

…



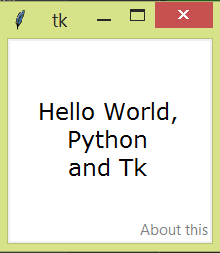
На холсте можно разместить текст. Делается это с помощью метода **create\_text()**:

…

c.create\_text(100, 100, text="Hello World,\nPython\nand Tk", justify=CENTER, font="Verdana 14")

c.create\_text(200, 200, text="About this", anchor=SE, fill="grey")

…



По умолчанию в заданной координате располагается центр текстовой надписи. Чтобы изменить это и, например, разместить по указанной координате левую границу текста, используется якорь со значением W (от англ. **west** – запад). Другие значения: **N**, **NE**, **E**, **SE**, **S**, **SW**, **W**, **NW**. Если букв, задающих сторону привязки, две, то вторая определяет вертикальную привязку (вверх или вниз «уйдет» текст от заданной координаты). Свойство justify определяет лишь выравнивание текста относительно себя самого.

**Анимация в tkinter**.

В данной программе создается анимация круга, который движется от левой границы холста до правой:

|  |
| --- |
| **from** tkinter **import** \*  root = Tk() c = Canvas(root, width=300, height=200, bg=**"white"**) c.pack()  ball = c.create\_oval(0, 100, 40, 140, fill=**'green'**) **def** motion():  c.move(ball, 1, 0)  **if** c.coords(ball)[2] < 300:  root.after(20, motion) motion() root.mainloop() |

Выражение c.coords(ball) возвращает список текущих координат объекта (в данном случае это ball). Третий элемент списка соответствует его второй координате x.

Метод after() вызывает функцию, переданную вторым аргументом, через количество миллисекунд, указанных первым аргументом.