**Python Events**

|  |  |  |
| --- | --- | --- |
| Event | Bind Keyword | Description |
| <Button> |  | A mouse button is pressed with the mouse pointer over the widget. The detail part specifies which button - |
| <Button-1> | left mouse button is defined by the event |
| <Button-2> | middle mouse button is defined by the event |
| <Button-3> | rightmost mouse button |
| <Button-4> | defines the scroll up event |
| <Button-5> | defines the scroll down event |
| <Motion> |  | The mouse is moved with a mouse button being held down. The current position of the mouse pointer is provided in the x and y members of the event object passed to the callback, i.e. event.x, event.y |
| <B1-Motion> | specify the left |
| <B2-Motion> | specify the middle |
| <B2-Motion> | specify the right |
| <ButtonRelease> |  | Event, if a button is released. To specify the left, middle or right mouse button use <ButtonRelease-1>, <ButtonRelease-2>, and <ButtonRelease-3> respectively. The current position of the mouse pointer is provided in the x and y members of the event object passed to the callback, i.e. event.x, event.y |
| <ButtonRelease-1> | specify the left |
| <ButtonRelease-2> | specify the middle |
| <ButtonRelease-3> | specify the right |
| <Enter> | <Return> | The mouse pointer entered the widget. Attention: This doesn't mean that the user pressed the Enter key!. <Return> is used for this purpose. |
| <Leave> | <Leave> | The mouse pointer left the widget. |
| <FocusIn> | <FocusIn> | Keyboard focus was moved to this widget, or to a child of this widget. |
| <FocusOut> | <FocusOut> | Keyboard focus was moved from this widget to another widget. |
| <Return> | The user pressed the Enter key. You can bind to virtually all keys on the keyboard: | |
| Cancel (the Break key) | |
| BackSpace, Tab, Return(the Enter key) | |
| 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> |  | The user pressed any key. The key is provided in the char member of the event object passed to the callback (this is an empty string for special keys). |
| a |  | The user typed an "a" key. Most printable characters can be used as is. The exceptions are space (<space>) and less than (<less>). Note that 1 is a keyboard binding, while <1> is a button binding. |
| <Shift-Up> |  | The user pressed the Up arrow, while holding the Shift key pressed. You can use prefixes like Alt, Shift, and Control. |
| <Configure> |  | The size of the widget changed. The new size is provided in the width and height attributes of the event object passed to the callback. On some platforms, it can mean that the location changed. |

**Message Boxes**

The message dialogues are provided by the tkMessageBox module.   
  
The tkMessageBox consists of the following functions, which correspond to dialog windows:

* askokcancel(title=None, message=None, \*\*options)  
  Ask if operation should proceed; return true if the answer is ok
* askquestion(title=None, message=None, \*\*options)  
  Ask a question
* askretrycancel(title=None, message=None, \*\*options)  
  Ask if operation should be retried; return true if the answer is yes
* askyesno(title=None, message=None, \*\*options)  
  Ask a question; return true if the answer is yes
* askyesnocancel(title=None, message=None, \*\*options)  
  Ask a question; return true if the answer is yes, None if cancelled.
* showerror(title=None, message=None, \*\*options)  
  Show an error message
* showinfo(title=None, message=None, \*\*options)  
  Show an info message
* showwarning(title=None, message=None, \*\*options)  
  Show a warning message

**Open File Dialogue**

There is hardly any serious application, which doesn't need a way to read from a file or write to a file. Furthermore, such an application might have to choose a directory. Tkinter provides the module tkFileDialog for these purposes.

from Tkinter import \*

from tkFileDialog import askopenfilename

def callback():

name= askopenfilename()

print name

errmsg = 'Error!'

Button(text='File Open', command=callback).pack(fill=X)

mainloop()

**Choosing a Colour**

There are applications where the user should have the possibility to select a colour. Tkinter provides a pop-up menu to choose a colour. To this purpose we have to import the tkColorChooser module and have to use the method askColor:

result = tkColorChooser.askColor ( color, option=value, ...)

If the user clicks the OK button on the pop-up window, respectively, the return value of askColor() is a tuple with two elements, both a representation of the chosen colour, e.g. ((106, 150, 98), '#6a9662')   
The first element return[0] is a tuple (R, G, B) with the RGB representation in decimal values (from 0 to 255). The second element return[1] is a hexadecimal representation of the chosen colour.   
If the user clicks "Cancel" the method returns the tuple (None, None).   
  
The optional keyword parameters are:

|  |  |
| --- | --- |
| color | The variable color is used to set the default colour to be displayed. If color is not set, the initial colour will be grey. |
| title | The text assigned to the variable title will appear in the pop-up window's title area. The default title is "Color". |
| parent | Make the pop-up window appear over window W. The default behaviour is that it appears over the root window. |

Let's have a look at an example:

from Tkinter import \*

from tkColorChooser import askcolor

def callback():

result = askcolor(color="#6A9662",

title = "Bernd's Colour Chooser")

print result

root = Tk()

Button(root,

text='Choose Color',

fg="darkgreen",

command=callback).pack(side=LEFT, padx=10)

Button(text='Quit',

command=root.quit,

fg="red").pack(side=LEFT, padx=10)

mainloop()