<http://codereview.stackexchange.com/questions/94290/classic-snake-game-using-python-tkinter-and-threading> snake

<http://code.activestate.com/recipes/578928-game-of-life-python-34-tkinter/> other

<https://gist.github.com/feelinc/5379013> ball

<https://gist.github.com/calebrob6/4022622> ball

<http://stackoverflow.com/questions/29278341/python-tkinter-bounce-ball-game-restart> ball

In Tkinter, event.type contains a numeric code. You can map back from int(event.type) to the event name:

{ 2: "KeyPress",

3: "KeyRelease",

4: "ButtonPress",

5: "ButtonRelease",

6: "Motion",

7: "Enter",

8: "Leave",

9: "FocusIn",

10: "FocusOut",

12: "Expose",

15: "Visibility",

17: "Destroy",

18: "Unmap",

19: "Map",

21: "Reparent",

22: "Configure",

24: "Gravity",

26: "Circulate",

28: "Property",

32: "Colormap",

36: "Activate",

37: "Deactivate",

38: "MouseWheel"

}

Tkinter matches Tk quite closely, and there's no way to get \*all\* bindings

for a standard widget with a single call at the Tk level (afaik).

to extract this information from a Tkinter widget, you should first call

bindtags() on the widget to get a list of binding classes used for this widget,

and you can then use bind\_class(cls) to get the events for that class.

to get all events, you can use something like:

>>> from Tkinter import Button

>>> b = Button()

>>> bindings = set()

>>> for cls in b.bindtags():

... bindings |= set(b.bind\_class(cls)) # s |= t means: update set s, adding elements from t

...

>>> bindings

set(['<Alt-KeyRelease>', '<Leave>', '<Enter>', '<KeyRelease-Alt\_L>',

'<Key-Alt\_R>', '<<PrevWindow>>', '<Key-F10>', '<KeyRelease-F10>',

'<Key-space>', '<Alt-Key>', '<Button-1>', '<ButtonRelease-1>',

'<KeyRelease-Alt\_R>', '<Key-Tab>', '<Key-Alt\_L>'])

</F>

<https://docs.python.org/3/library/tkinter.html>

The fundamental unit of a [Tkinter](https://tkinter.unpythonic.net/wiki/Tkinter) GUI is a Widget. You can use the "built-in" widgets that are provided by Tk (listed below), create your own, or install extension packages that define additional widgets.

Widgets provided by Tk include:

* [/Button](https://tkinter.unpythonic.net/wiki/Widgets/Button)
* [/Canvas](https://tkinter.unpythonic.net/wiki/Widgets/Canvas)
* [/Checkbutton](https://tkinter.unpythonic.net/wiki/Widgets/Checkbutton)
* [/Entry](https://tkinter.unpythonic.net/wiki/Widgets/Entry)
* [/Frame](https://tkinter.unpythonic.net/wiki/Widgets/Frame)
* [/Label](https://tkinter.unpythonic.net/wiki/Widgets/Label)
* [Listbox](https://tkinter.unpythonic.net/wiki/Listbox)
* [/Menu](https://tkinter.unpythonic.net/wiki/Widgets/Menu)
* [/Menubutton](https://tkinter.unpythonic.net/wiki/Widgets/Menubutton)
* [/Radiobutton](https://tkinter.unpythonic.net/wiki/Widgets/Radiobutton)
* [/Scale](https://tkinter.unpythonic.net/wiki/Widgets/Scale)
* [/Scrollbar](https://tkinter.unpythonic.net/wiki/Widgets/Scrollbar)
* [/Text](https://tkinter.unpythonic.net/wiki/Widgets/Text)
* [/TopLevel](https://tkinter.unpythonic.net/wiki/Widgets/TopLevel)

Widgets added in Tk 8.4 include:

* [/Spinbox](https://tkinter.unpythonic.net/wiki/Widgets/Spinbox)
* [/PanedWindow](https://tkinter.unpythonic.net/wiki/Widgets/PanedWindow)
* [/LabelFrame](https://tkinter.unpythonic.net/wiki/Widgets/LabelFrame)
* [/OptionMenu](https://tkinter.unpythonic.net/wiki/Widgets/OptionMenu)

Also compound buttons, labels, and menu items. the compound option allows you to display both the text and image. [ExampleCompoundButton](https://tkinter.unpythonic.net/wiki/ExampleCompoundButton)

(In the interest of WikiWikiNess, should we change the above to ButtonWidget, CanvasWidget, etc.?) [no. The above should be named like the Python classes. --jepler](I agree keep them the same as Tkinter class names - but do we need the '/'? MF)

BTW, you can get all configuration options for any widget with:

for k in widget.configure().keys():

print k, ':', widget.cget(k)

-john grayson

eg.

>>> from Tkinter import Button

>>> b = Button()

>>> for k in b.configure().keys():

print k, ':', b.cget(k)

<https://tkinter.unpythonic.net/wiki/Widgets>

# Tkinter.Button - create and manipulate button widgets

A button is a widget that displays a textual string, bitmap or image. If text is displayed, it must all be in a single font, but it can occupy multiple lines on the screen (if it contains newlines or if wrapping occurs because of the **wraplength=** option) and one of the characters may optionally be underlined using the **underline** option. It can display itself in either of three different ways, according to the **state**. to disable a button set **state = DISABLED**. To enable set **state = NORMAL**. Tk sets **state = ACTIVE** when mouse is over a non-**DISABLED** button) option; it can be made to appear raised, sunken, or flat; and it can be made to flash. When a user invokes the button (by pressing mouse button 1 with the cursor over the button), then the command specified in the **command=** option is invoked. If the command is a callable object, it is called with no arguments. If it is a string, it is executed as Tcl code.

# Base Classes

[../Widget](https://tkinter.unpythonic.net/wiki/Widgets/Widget), [../BaseWidget](https://tkinter.unpythonic.net/wiki/Widgets/BaseWidget), [../Pack](https://tkinter.unpythonic.net/wiki/Widgets/Pack), [../Place](https://tkinter.unpythonic.net/wiki/Widgets/Place), [../Grid](https://tkinter.unpythonic.net/wiki/Widgets/Grid), [../Misc](https://tkinter.unpythonic.net/wiki/Widgets/Misc)

# Default Bindings

Tk automatically creates class [bind](https://tkinter.unpythonic.net/wiki/bind)ings for buttons that give them default behavior:

1. A button activates whenever the mouse passes over it and deactivates whenever the mouse leaves the button. Under windows, this binding is only active when mouse button 1 has been pressed over the button.
2. A button's relief is changed to sunken whenever mouse button 1 is pressed over the button, and the relief is restored to its original value when button 1 is later released.
3. If mouse button 1 is pressed over a button and later released over the button, the button is invoked. However, if the mouse is not over the button when button 1 is released, then no invocation occurs.
4. When a button has the input focus, the space key causes the button to be invoked.

If the button's state is **disabled** then none of the above actions occur: the button is completely non-responsive. The behavior of buttons can be changed by defining new bindings for individual widgets or by redefining the class bindings.

# Gotchas

Just giving a button **default="active"** doesn't create a binding for <Return> (the enter key). See <http://wiki.tcl.tk/534> and search for "-default active" for some Tcl code related to this.

The standard Button widget does not allow parameters to be passed by the command method. This can be circumvented by re-writing the method as below:

[[des]activar nros. de línea](https://tkinter.unpythonic.net/wiki/Widgets/Button)

[1](https://tkinter.unpythonic.net/wiki/Widgets/Button#CA-4eca75bfb7e115051802c568a13eeb0012a13d2d_1) def command(func, \*args, \*\*kw):

[2](https://tkinter.unpythonic.net/wiki/Widgets/Button#CA-4eca75bfb7e115051802c568a13eeb0012a13d2d_2) def \_wrapper(\*wargs):

[3](https://tkinter.unpythonic.net/wiki/Widgets/Button#CA-4eca75bfb7e115051802c568a13eeb0012a13d2d_3) return func(\*(wargs + args), \*\*kw)

[4](https://tkinter.unpythonic.net/wiki/Widgets/Button#CA-4eca75bfb7e115051802c568a13eeb0012a13d2d_4) return \_wrapper

# Parameter-passing

Newcomers to Tkinter frequently ask about passing parameters through a button selection. The wrapper **command** above, supplied by Guilherme Polo, can be used as ... It's a clever alternative to Tkinter's more traditional style as expressed by [Bob Greschke](http://mail.python.org/pipermail/tkinter-discuss/2008-June/001452.html), for example. [Here](http://mail.python.org/pipermail/tkinter-discuss/2009-July/002022.html) and [here](http://mail.python.org/pipermail/tkinter-discuss/2009-July/002023.html) are two more variations that allow parameters to be passed to buttons [and explain %W some time]. Also, Francesco Bochicchio once noted in private correspondence, that, "[w]ith modern python, an alternative to use [of] lambda or a home-made callable is to use **functools.partial** :

b = Button( command = functools.partial( mycallback, myarg ))

# Tkinter.Button - create and manipulate button widgets

A button is a widget that displays a textual string, bitmap or image. If text is displayed, it must all be in a single font, but it can occupy multiple lines on the screen (if it contains newlines or if wrapping occurs because of the **wraplength=** option) and one of the characters may optionally be underlined using the **underline** option. It can display itself in either of three different ways, according to the **state**. to disable a button set **state = DISABLED**. To enable set **state = NORMAL**. Tk sets **state = ACTIVE** when mouse is over a non-**DISABLED** button) option; it can be made to appear raised, sunken, or flat; and it can be made to flash. When a user invokes the button (by pressing mouse button 1 with the cursor over the button), then the command specified in the **command=** option is invoked. If the command is a callable object, it is called with no arguments. If it is a string, it is executed as Tcl code.

# Base Classes

[../Widget](https://tkinter.unpythonic.net/wiki/Widgets/Widget), [../BaseWidget](https://tkinter.unpythonic.net/wiki/Widgets/BaseWidget), [../Pack](https://tkinter.unpythonic.net/wiki/Widgets/Pack), [../Place](https://tkinter.unpythonic.net/wiki/Widgets/Place), [../Grid](https://tkinter.unpythonic.net/wiki/Widgets/Grid), [../Misc](https://tkinter.unpythonic.net/wiki/Widgets/Misc)

# Default Bindings

Tk automatically creates class [bind](https://tkinter.unpythonic.net/wiki/bind)ings for buttons that give them default behavior:

1. A button activates whenever the mouse passes over it and deactivates whenever the mouse leaves the button. Under windows, this binding is only active when mouse button 1 has been pressed over the button.
2. A button's relief is changed to sunken whenever mouse button 1 is pressed over the button, and the relief is restored to its original value when button 1 is later released.
3. If mouse button 1 is pressed over a button and later released over the button, the button is invoked. However, if the mouse is not over the button when button 1 is released, then no invocation occurs.
4. When a button has the input focus, the space key causes the button to be invoked.

If the button's state is **disabled** then none of the above actions occur: the button is completely non-responsive. The behavior of buttons can be changed by defining new bindings for individual widgets or by redefining the class bindings.

# Gotchas

Just giving a button **default="active"** doesn't create a binding for <Return> (the enter key). See <http://wiki.tcl.tk/534> and search for "-default active" for some Tcl code related to this.

The standard Button widget does not allow parameters to be passed by the command method. This can be circumvented by re-writing the method as below:

[[des]activar nros. de línea](https://tkinter.unpythonic.net/wiki/Widgets/Button)

[1](https://tkinter.unpythonic.net/wiki/Widgets/Button#CA-4eca75bfb7e115051802c568a13eeb0012a13d2d_1) def command(func, \*args, \*\*kw):

[2](https://tkinter.unpythonic.net/wiki/Widgets/Button#CA-4eca75bfb7e115051802c568a13eeb0012a13d2d_2) def \_wrapper(\*wargs):

[3](https://tkinter.unpythonic.net/wiki/Widgets/Button#CA-4eca75bfb7e115051802c568a13eeb0012a13d2d_3) return func(\*(wargs + args), \*\*kw)

[4](https://tkinter.unpythonic.net/wiki/Widgets/Button#CA-4eca75bfb7e115051802c568a13eeb0012a13d2d_4) return \_wrapper

# Parameter-passing

Newcomers to Tkinter frequently ask about passing parameters through a button selection. The wrapper **command** above, supplied by Guilherme Polo, can be used as ... It's a clever alternative to Tkinter's more traditional style as expressed by [Bob Greschke](http://mail.python.org/pipermail/tkinter-discuss/2008-June/001452.html), for example. [Here](http://mail.python.org/pipermail/tkinter-discuss/2009-July/002022.html) and [here](http://mail.python.org/pipermail/tkinter-discuss/2009-July/002023.html) are two more variations that allow parameters to be passed to buttons [and explain %W some time]. Also, Francesco Bochicchio once noted in private correspondence, that, "[w]ith modern python, an alternative to use [of] lambda or a home-made callable is to use **functools.partial** :

b = Button( command = functools.partial( mycallback, myarg ))

# Tkinter.Canvas - create and manipulate canvas widgets

A canvas is a versatile widget that can display multiple graphics and text "items" within its boundaries. Each item can be manipulated and formatted separately.

# Base Classes

[../Widget](https://tkinter.unpythonic.net/wiki/Widgets/Widget), [../BaseWidget](https://tkinter.unpythonic.net/wiki/Widgets/BaseWidget), [../Pack](https://tkinter.unpythonic.net/wiki/Widgets/Pack), [../Place](https://tkinter.unpythonic.net/wiki/Widgets/Place), [../Grid](https://tkinter.unpythonic.net/wiki/Widgets/Grid), [../Misc](https://tkinter.unpythonic.net/wiki/Widgets/Misc)

# Items

Items that can be placed on a Canvas widget include arc, bitmap, image, line, oval, polygon, rectangle, text, and widget. Use arc, line, oval, polygon, and rectangle items to draw your own graphics on the canvas. Use bitmap and image (gif images are supported but for other formats the Python Imaging Library is required) to import graphics. Use text to place text items or labels anywhere on the canvas (each text item can have its own font, color, and other attributes set independently). Use a window item to hold other Tkinter widgets.

Each item on a Canvas can be given one or more tags. It is sometimes useful to give an item multiple tags as a way of grouping items for later operations. You can specify the tags either as a tuple (not a list) of strings, or as a single string with the tags separated by whitespace.

# Gotchas

Canvas items are not Python objects; they are only accessible through the methods of the Canvas object. The "handle" that is returned when you create an item is simply an integer that the Canvas uses to identify the item.

The Canvas module (not Tkinter.Canvas) allows you to create canvas items that are wrapped by Python objects. However, a comment at the top of this file notes that it is considered obsolete.

If you create a canvas without scrollbars, the origin might be hidden by the Canvas border. [FredrikLundh](https://tkinter.unpythonic.net/wiki/FredrikLundh) says:

by default, the coordinate system is aligned with the widget's upper

left corner, which means that things you draw will be covered by the

inner border.

to fix this, you can either set the border width to zero, add scrollbars

to the widget (this fixes the coordinate system), or explicitly reset the

coordinate system:

w.xview\_moveto(0)

w.yview\_moveto(0)

# kFileDialog

**(Note: for Python 3, the tkFileDialog module was renamed "filedialog."** For other tkinter modules' new names see <https://docs.python.org/3.4/library/tkinter.html#tkinter-modules>**)**

If you want to open or save a file or to choose a directory using a filedialog you dont need to implement it on your own. The module tkFileDialog is just for you. In most cases the seven convenience functions provided by the module will serve your needs.

## Functions

First you have to decide if you want to open a file or just want to get a filename in order to open the file on your own. In the first case you should use tkFileDialog.askopenfile() in the latter case tkFileDialog.askopenfilename(). Both functions come in a multiple file version with just the same parameters as the single file version which allow the user to select multiple files.

The multiple file versions return a list of open files or a list of file names. If no files were selected or the cancle button was pressed an empty list is returned.

Saving files works in a similar way. You also have two variants of the function, one to get an opened file which you can use to save your data and another to get a file name in order to open the file on your own. These functions are only provided in the single file version. A multiple file version would make no sense.

|  |  |  |  |
| --- | --- | --- | --- |
|  | single file | multiple files | available options |
| open a file | askopenfile(mode='r', \*\*options) | askopenfiles(mode='r', \*\*options) | defaultextension, filetypes, initialdir, initialfile, multiple, message, parent, title |
| get a filename to open | askopenfilename(\*\*options) | askopenfilenames(\*\*options) | defaultextension, filetypes, initialdir, initialfile, multiple, message, parent, title |
| save a file | asksaveasfile(mode='w', \*\*options) | n/a | defaultextension, filetypes, initialdir, initialfile, multiple, message, parent, title |
| get a filename to save | asksaveasfilename(\*\*options) | n/a | defaultextension, filetypes, initialdir, initialfile, multiple, message, parent, title |
| choose a directory | askdirectory(\*\*options) | n/a | initialdir, parent, title, mustexist |

## Options

-defaultextension extension

* Specifies a string that will be appended to the filename if the user enters a filename without an extension. The default value is the empty string, which means no extension will be appended to the filename in any case. This option is ignored on the Macintosh platform, which does not require extensions to filenames, and the UNIX implementation guesses reasonable values for this from the -filetypes option when this is not supplied.

-filetypes filePatternList

* If a File types listbox exists in the file dialog on the particular platform, this option gives the filetypes in this listbox. When the user choose a filetype in the listbox, only the files of that type are listed. If this option is unspecified, or if it is set to the empty list, or if the File types listbox is not supported by the particular platform then all files are listed regardless of their types. See the section SPECIFYING FILE PATTERNS below for a discussion on the contents of filePatternList.

-initialdir directory

* Specifies that the files in directory should be displayed when the dialog pops up. If this parameter is not specified, then the files in the current working directory are displayed. If the parameter specifies a relative path, the return value will convert the relative path to an absolute path. This option may not always work on the Macintosh. This is not a bug. Rather, the General Controls control panel on the Mac allows the end user to override the application default directory.

-initialfile filename

* Specifies a filename to be displayed in the dialog when it pops up. This option is ignored on the Macintosh platform.

-message string

* Specifies a message to include in the client area of the dialog. This is only available on the Macintosh, and only when Navigation Services are installed.

-multiple boolean

* Allows the user to choose multiple files from the Open dialog. On the Macintosh, this is only available when Navigation Services are installed.

-mustexist boolean

* Specifies whether the user may specify non-existent directories. If this parameter is true, then the user may only select directories that already exist. The default value is false.

-parent window

* Makes window the logical parent of the dialog. The dialog is displayed on top of its parent window.

-title titleString

* Specifies a string to display as the title of the dialog box. If this option is not specified, then a default title will be displayed.

[source:]

* <http://www.tcl.tk/man/tcl8.4/TkCmd/getOpenFile.htm>
* <http://www.tcl.tk/man/tcl8.4/TkCmd/chooseDirectory.htm>

## Example

This example just should give you an idea of the modules use.

[[des]activar nros. de línea](https://tkinter.unpythonic.net/wiki/tkFileDialog)

[1](https://tkinter.unpythonic.net/wiki/tkFileDialog#CA-87fa57b85635fb29da2b08946d341845aecff099_1) import Tkinter, Tkconstants, tkFileDialog

[2](https://tkinter.unpythonic.net/wiki/tkFileDialog#CA-87fa57b85635fb29da2b08946d341845aecff099_2)

[3](https://tkinter.unpythonic.net/wiki/tkFileDialog#CA-87fa57b85635fb29da2b08946d341845aecff099_3) class TkFileDialogExample(Tkinter.Frame):

[4](https://tkinter.unpythonic.net/wiki/tkFileDialog#CA-87fa57b85635fb29da2b08946d341845aecff099_4)

[5](https://tkinter.unpythonic.net/wiki/tkFileDialog#CA-87fa57b85635fb29da2b08946d341845aecff099_5) def \_\_init\_\_(self, root):

[6](https://tkinter.unpythonic.net/wiki/tkFileDialog#CA-87fa57b85635fb29da2b08946d341845aecff099_6)

[7](https://tkinter.unpythonic.net/wiki/tkFileDialog#CA-87fa57b85635fb29da2b08946d341845aecff099_7) Tkinter.Frame.\_\_init\_\_(self, root)

[8](https://tkinter.unpythonic.net/wiki/tkFileDialog#CA-87fa57b85635fb29da2b08946d341845aecff099_8)

[9](https://tkinter.unpythonic.net/wiki/tkFileDialog#CA-87fa57b85635fb29da2b08946d341845aecff099_9) # options for buttons

[10](https://tkinter.unpythonic.net/wiki/tkFileDialog#CA-87fa57b85635fb29da2b08946d341845aecff099_10) button\_opt = {'fill': Tkconstants.BOTH, 'padx': 5, 'pady': 5}

[11](https://tkinter.unpythonic.net/wiki/tkFileDialog#CA-87fa57b85635fb29da2b08946d341845aecff099_11)

[12](https://tkinter.unpythonic.net/wiki/tkFileDialog#CA-87fa57b85635fb29da2b08946d341845aecff099_12) # define buttons

[13](https://tkinter.unpythonic.net/wiki/tkFileDialog#CA-87fa57b85635fb29da2b08946d341845aecff099_13) Tkinter.Button(self, text='askopenfile', command=self.askopenfile).pack(\*\*button\_opt)

[14](https://tkinter.unpythonic.net/wiki/tkFileDialog#CA-87fa57b85635fb29da2b08946d341845aecff099_14) Tkinter.Button(self, text='askopenfilename', command=self.askopenfilename).pack(\*\*button\_opt)

[15](https://tkinter.unpythonic.net/wiki/tkFileDialog#CA-87fa57b85635fb29da2b08946d341845aecff099_15) Tkinter.Button(self, text='asksaveasfile', command=self.asksaveasfile).pack(\*\*button\_opt)

[16](https://tkinter.unpythonic.net/wiki/tkFileDialog#CA-87fa57b85635fb29da2b08946d341845aecff099_16) Tkinter.Button(self, text='asksaveasfilename', command=self.asksaveasfilename).pack(\*\*button\_opt)

[17](https://tkinter.unpythonic.net/wiki/tkFileDialog#CA-87fa57b85635fb29da2b08946d341845aecff099_17) Tkinter.Button(self, text='askdirectory', command=self.askdirectory).pack(\*\*button\_opt)

[18](https://tkinter.unpythonic.net/wiki/tkFileDialog#CA-87fa57b85635fb29da2b08946d341845aecff099_18)

[19](https://tkinter.unpythonic.net/wiki/tkFileDialog#CA-87fa57b85635fb29da2b08946d341845aecff099_19) # define options for opening or saving a file

[20](https://tkinter.unpythonic.net/wiki/tkFileDialog#CA-87fa57b85635fb29da2b08946d341845aecff099_20) self.file\_opt = options = {}

[21](https://tkinter.unpythonic.net/wiki/tkFileDialog#CA-87fa57b85635fb29da2b08946d341845aecff099_21) options['defaultextension'] = '.txt'

[22](https://tkinter.unpythonic.net/wiki/tkFileDialog#CA-87fa57b85635fb29da2b08946d341845aecff099_22) options['filetypes'] = [('all files', '.\*'), ('text files', '.txt')]

[23](https://tkinter.unpythonic.net/wiki/tkFileDialog#CA-87fa57b85635fb29da2b08946d341845aecff099_23) options['initialdir'] = 'C:\\'

[24](https://tkinter.unpythonic.net/wiki/tkFileDialog#CA-87fa57b85635fb29da2b08946d341845aecff099_24) options['initialfile'] = 'myfile.txt'

[25](https://tkinter.unpythonic.net/wiki/tkFileDialog#CA-87fa57b85635fb29da2b08946d341845aecff099_25) options['parent'] = root

[26](https://tkinter.unpythonic.net/wiki/tkFileDialog#CA-87fa57b85635fb29da2b08946d341845aecff099_26) options['title'] = 'This is a title'

[27](https://tkinter.unpythonic.net/wiki/tkFileDialog#CA-87fa57b85635fb29da2b08946d341845aecff099_27)

[28](https://tkinter.unpythonic.net/wiki/tkFileDialog#CA-87fa57b85635fb29da2b08946d341845aecff099_28) # This is only available on the Macintosh, and only when Navigation Services are installed.

[29](https://tkinter.unpythonic.net/wiki/tkFileDialog#CA-87fa57b85635fb29da2b08946d341845aecff099_29) #options['message'] = 'message'

[30](https://tkinter.unpythonic.net/wiki/tkFileDialog#CA-87fa57b85635fb29da2b08946d341845aecff099_30)

[31](https://tkinter.unpythonic.net/wiki/tkFileDialog#CA-87fa57b85635fb29da2b08946d341845aecff099_31) # if you use the multiple file version of the module functions this option is set automatically.

[32](https://tkinter.unpythonic.net/wiki/tkFileDialog#CA-87fa57b85635fb29da2b08946d341845aecff099_32) #options['multiple'] = 1

[33](https://tkinter.unpythonic.net/wiki/tkFileDialog#CA-87fa57b85635fb29da2b08946d341845aecff099_33)

[34](https://tkinter.unpythonic.net/wiki/tkFileDialog#CA-87fa57b85635fb29da2b08946d341845aecff099_34) # defining options for opening a directory

[35](https://tkinter.unpythonic.net/wiki/tkFileDialog#CA-87fa57b85635fb29da2b08946d341845aecff099_35) self.dir\_opt = options = {}

[36](https://tkinter.unpythonic.net/wiki/tkFileDialog#CA-87fa57b85635fb29da2b08946d341845aecff099_36) options['initialdir'] = 'C:\\'

[37](https://tkinter.unpythonic.net/wiki/tkFileDialog#CA-87fa57b85635fb29da2b08946d341845aecff099_37) options['mustexist'] = False

[38](https://tkinter.unpythonic.net/wiki/tkFileDialog#CA-87fa57b85635fb29da2b08946d341845aecff099_38) options['parent'] = root

[39](https://tkinter.unpythonic.net/wiki/tkFileDialog#CA-87fa57b85635fb29da2b08946d341845aecff099_39) options['title'] = 'This is a title'

[40](https://tkinter.unpythonic.net/wiki/tkFileDialog#CA-87fa57b85635fb29da2b08946d341845aecff099_40)

[41](https://tkinter.unpythonic.net/wiki/tkFileDialog#CA-87fa57b85635fb29da2b08946d341845aecff099_41) def askopenfile(self):

[42](https://tkinter.unpythonic.net/wiki/tkFileDialog#CA-87fa57b85635fb29da2b08946d341845aecff099_42)

[43](https://tkinter.unpythonic.net/wiki/tkFileDialog#CA-87fa57b85635fb29da2b08946d341845aecff099_43) """Returns an opened file in read mode."""

[44](https://tkinter.unpythonic.net/wiki/tkFileDialog#CA-87fa57b85635fb29da2b08946d341845aecff099_44)

[45](https://tkinter.unpythonic.net/wiki/tkFileDialog#CA-87fa57b85635fb29da2b08946d341845aecff099_45) return tkFileDialog.askopenfile(mode='r', \*\*self.file\_opt)

[46](https://tkinter.unpythonic.net/wiki/tkFileDialog#CA-87fa57b85635fb29da2b08946d341845aecff099_46)

[47](https://tkinter.unpythonic.net/wiki/tkFileDialog#CA-87fa57b85635fb29da2b08946d341845aecff099_47) def askopenfilename(self):

[48](https://tkinter.unpythonic.net/wiki/tkFileDialog#CA-87fa57b85635fb29da2b08946d341845aecff099_48)

[49](https://tkinter.unpythonic.net/wiki/tkFileDialog#CA-87fa57b85635fb29da2b08946d341845aecff099_49) """Returns an opened file in read mode.

[50](https://tkinter.unpythonic.net/wiki/tkFileDialog#CA-87fa57b85635fb29da2b08946d341845aecff099_50) This time the dialog just returns a filename and the file is opened by your own code.

[51](https://tkinter.unpythonic.net/wiki/tkFileDialog#CA-87fa57b85635fb29da2b08946d341845aecff099_51) """

[52](https://tkinter.unpythonic.net/wiki/tkFileDialog#CA-87fa57b85635fb29da2b08946d341845aecff099_52)

[53](https://tkinter.unpythonic.net/wiki/tkFileDialog#CA-87fa57b85635fb29da2b08946d341845aecff099_53) # get filename

[54](https://tkinter.unpythonic.net/wiki/tkFileDialog#CA-87fa57b85635fb29da2b08946d341845aecff099_54) filename = tkFileDialog.askopenfilename(\*\*self.file\_opt)

[55](https://tkinter.unpythonic.net/wiki/tkFileDialog#CA-87fa57b85635fb29da2b08946d341845aecff099_55)

[56](https://tkinter.unpythonic.net/wiki/tkFileDialog#CA-87fa57b85635fb29da2b08946d341845aecff099_56) # open file on your own

[57](https://tkinter.unpythonic.net/wiki/tkFileDialog#CA-87fa57b85635fb29da2b08946d341845aecff099_57) if filename:

[58](https://tkinter.unpythonic.net/wiki/tkFileDialog#CA-87fa57b85635fb29da2b08946d341845aecff099_58) return open(filename, 'r')

[59](https://tkinter.unpythonic.net/wiki/tkFileDialog#CA-87fa57b85635fb29da2b08946d341845aecff099_59)

[60](https://tkinter.unpythonic.net/wiki/tkFileDialog#CA-87fa57b85635fb29da2b08946d341845aecff099_60) def asksaveasfile(self):

[61](https://tkinter.unpythonic.net/wiki/tkFileDialog#CA-87fa57b85635fb29da2b08946d341845aecff099_61)

[62](https://tkinter.unpythonic.net/wiki/tkFileDialog#CA-87fa57b85635fb29da2b08946d341845aecff099_62) """Returns an opened file in write mode."""

[63](https://tkinter.unpythonic.net/wiki/tkFileDialog#CA-87fa57b85635fb29da2b08946d341845aecff099_63)

[64](https://tkinter.unpythonic.net/wiki/tkFileDialog#CA-87fa57b85635fb29da2b08946d341845aecff099_64) return tkFileDialog.asksaveasfile(mode='w', \*\*self.file\_opt)

[65](https://tkinter.unpythonic.net/wiki/tkFileDialog#CA-87fa57b85635fb29da2b08946d341845aecff099_65)

[66](https://tkinter.unpythonic.net/wiki/tkFileDialog#CA-87fa57b85635fb29da2b08946d341845aecff099_66) def asksaveasfilename(self):

[67](https://tkinter.unpythonic.net/wiki/tkFileDialog#CA-87fa57b85635fb29da2b08946d341845aecff099_67)

[68](https://tkinter.unpythonic.net/wiki/tkFileDialog#CA-87fa57b85635fb29da2b08946d341845aecff099_68) """Returns an opened file in write mode.

[69](https://tkinter.unpythonic.net/wiki/tkFileDialog#CA-87fa57b85635fb29da2b08946d341845aecff099_69) This time the dialog just returns a filename and the file is opened by your own code.

[70](https://tkinter.unpythonic.net/wiki/tkFileDialog#CA-87fa57b85635fb29da2b08946d341845aecff099_70) """

[71](https://tkinter.unpythonic.net/wiki/tkFileDialog#CA-87fa57b85635fb29da2b08946d341845aecff099_71)

[72](https://tkinter.unpythonic.net/wiki/tkFileDialog#CA-87fa57b85635fb29da2b08946d341845aecff099_72) # get filename

[73](https://tkinter.unpythonic.net/wiki/tkFileDialog#CA-87fa57b85635fb29da2b08946d341845aecff099_73) filename = tkFileDialog.asksaveasfilename(\*\*self.file\_opt)

[74](https://tkinter.unpythonic.net/wiki/tkFileDialog#CA-87fa57b85635fb29da2b08946d341845aecff099_74)

[75](https://tkinter.unpythonic.net/wiki/tkFileDialog#CA-87fa57b85635fb29da2b08946d341845aecff099_75) # open file on your own

[76](https://tkinter.unpythonic.net/wiki/tkFileDialog#CA-87fa57b85635fb29da2b08946d341845aecff099_76) if filename:

[77](https://tkinter.unpythonic.net/wiki/tkFileDialog#CA-87fa57b85635fb29da2b08946d341845aecff099_77) return open(filename, 'w')

[78](https://tkinter.unpythonic.net/wiki/tkFileDialog#CA-87fa57b85635fb29da2b08946d341845aecff099_78)

[79](https://tkinter.unpythonic.net/wiki/tkFileDialog#CA-87fa57b85635fb29da2b08946d341845aecff099_79) def askdirectory(self):

[80](https://tkinter.unpythonic.net/wiki/tkFileDialog#CA-87fa57b85635fb29da2b08946d341845aecff099_80)

[81](https://tkinter.unpythonic.net/wiki/tkFileDialog#CA-87fa57b85635fb29da2b08946d341845aecff099_81) """Returns a selected directoryname."""

[82](https://tkinter.unpythonic.net/wiki/tkFileDialog#CA-87fa57b85635fb29da2b08946d341845aecff099_82)

[83](https://tkinter.unpythonic.net/wiki/tkFileDialog#CA-87fa57b85635fb29da2b08946d341845aecff099_83) return tkFileDialog.askdirectory(\*\*self.dir\_opt)

[84](https://tkinter.unpythonic.net/wiki/tkFileDialog#CA-87fa57b85635fb29da2b08946d341845aecff099_84)

[85](https://tkinter.unpythonic.net/wiki/tkFileDialog#CA-87fa57b85635fb29da2b08946d341845aecff099_85) if \_\_name\_\_=='\_\_main\_\_':

[86](https://tkinter.unpythonic.net/wiki/tkFileDialog#CA-87fa57b85635fb29da2b08946d341845aecff099_86) root = Tkinter.Tk()

[87](https://tkinter.unpythonic.net/wiki/tkFileDialog#CA-87fa57b85635fb29da2b08946d341845aecff099_87) TkFileDialogExample(root).pack()

[88](https://tkinter.unpythonic.net/wiki/tkFileDialog#CA-87fa57b85635fb29da2b08946d341845aecff099_88) root.mainloop()