

---

# **psidialogs Documentation**

***Release 0.0.5***

**ponty**

February 08, 2012

# CONTENTS

<b>1</b>	<b>Basic usage</b>	<b>2</b>
<b>2</b>	<b>Installation</b>	<b>3</b>
2.1	General . . . . .	3
2.2	Ubuntu . . . . .	3
2.3	Uninstall . . . . .	3
2.4	similar projects . . . . .	3
<b>3</b>	<b>Hierarchy</b>	<b>4</b>
<b>4</b>	<b>API</b>	<b>5</b>
4.1	ask_file() . . . . .	5
4.2	ask_folder() . . . . .	5
4.3	ask_ok_cancel() . . . . .	5
4.4	ask_string() . . . . .	6
4.5	ask_yes_no() . . . . .	6
4.6	choice() . . . . .	6
4.7	error() . . . . .	7
4.8	message() . . . . .	7
4.9	multi_choice() . . . . .	7
4.10	text() . . . . .	8
4.11	warning() . . . . .	8
<b>5</b>	<b>Screenshots</b>	<b>9</b>
5.1	ask_ok_cancel() . . . . .	9
5.2	ask_string() . . . . .	12
5.3	ask_yes_no() . . . . .	16
5.4	choice() . . . . .	19
5.5	error() . . . . .	24
5.6	message() . . . . .	28
5.7	multi_choice() . . . . .	31
5.8	text() . . . . .	36
5.9	warning() . . . . .	47
<b>6</b>	<b>Demo</b>	<b>52</b>
6.1	command line help . . . . .	53
<b>7</b>	<b>Development</b>	<b>54</b>
7.1	Tools . . . . .	54
7.2	Install on ubuntu . . . . .	54
7.3	Tasks . . . . .	54
<b>8</b>	<b>Indices and tables</b>	<b>56</b>

<b>Python Module Index</b>	<b>57</b>
<b>Index</b>	<b>58</b>

## psidialogs

**Date** February 08, 2012

**PDF** [psidialogs.pdf](#)

Contents:

psidialogs (Python Simple Dialogs) is a common API for different standard dialogs like message, ask\_string,...

### Links:

- home: <https://github.com/ponty/psidialogs>
- documentation: <http://ponty.github.com/psidialogs>

### Backends:

- PyGTK
- Zenity
- easygui
- gMessage
- PyQt
- TkInter
- wxPython
- PythonDialog
- console
- EasyDialogs

Some dialogs are too simple, because a common basic implementation is used where implementation is missing.

# BASIC USAGE

```
>>> from psidialogs import message
>>> message('Hello!')
```

# INSTALLATION

## 2.1 General

- install `pip`
- install the program:

```
# as root
pip install psidialogs
```

## 2.2 Ubuntu

```
sudo apt-get install python-pip
sudo pip install psidialogs
```

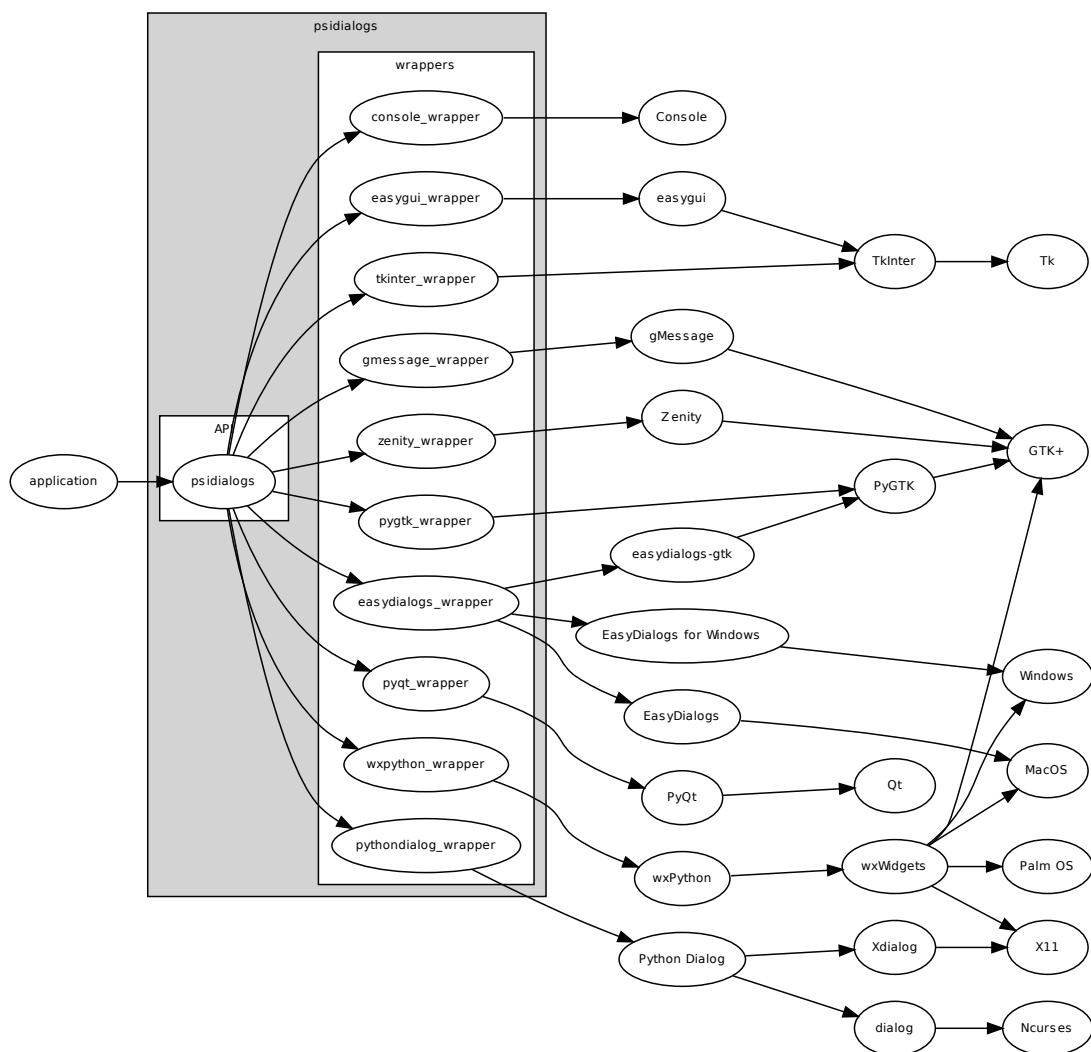
## 2.3 Uninstall

```
# as root
pip uninstall psidialogs
```

## 2.4 similar projects

- `anygui`: multiple backends
- `easygui`: tk backend
- `PyZenity`: Zenity backend
- `vsgui`: Zenity backend
- `dlg`: dialog/Xdialog/gdialog backend
- `python-dialog`: dialog/Xdialog/gdialog backend
- `easydialogs-gtk`: EasyDialogs API, PyGTK backend
- `EasyDialogs`: EasyDialogs API, Mac backend
- `EasyDialogs for Windows`: EasyDialogs API, Windows backend

# HIERARCHY



# API

## 4.1 ask\_file()

`psidialogs.ask_file(message='Select file for open.', default='', title='', save=False)`

A dialog to get a file name. The “default” argument specifies a file path.

save=False -> file for loading save=True -> file for saving

Return the file path that the user entered, or None if he cancels the operation.

### Parameters

- **message** – message to be displayed.
- **save** – bool 0 -> load , 1 -> save
- **title** – window title
- **default** – default file path

**Return type** None or string

## 4.2 ask\_folder()

`psidialogs.ask_folder(message='Select folder.', default='', title='')`

A dialog to get a directory name. Returns the name of a directory, or None if user chose to cancel. If the “default” argument specifies a directory name, and that directory exists, then the dialog box will start with that directory.

### Parameters

- **message** – message to be displayed.
- **title** – window title
- **default** – default folder path
- **ok** – label of the ok button
- **cancel** – label of the cancel button

**Return type** None or string

## 4.3 ask\_ok\_cancel()

`psidialogs.ask_ok_cancel(message='', default=0, title='')`

Display a message with choices of OK and Cancel.

**returned value:** OK -> True Cancel -> False



*screenshots*

#### Parameters

- **message** – message to be displayed.
- **title** – window title
- **default** – default button as boolean (OK=True, Cancel=False)

**Return type** bool

## 4.4 ask\_string()

`psidialogs.ask_string(message='Enter something.', default='', title='')`

Show a box in which a user can enter some text.

You may optionally specify some default text, which will appear in the entry-box when it is displayed.

Returns the text that the user entered, or None if he cancels the operation

*screenshots*

#### Parameters

- **message** – message to be displayed.
- **title** – window title
- **default** – entry-box default string
- **ok** – label of the ok button
- **cancel** – label of the cancel button

**Return type** None or string

## 4.5 ask\_yes\_no()

`psidialogs.ask_yes_no(message='', default=0, title='')`

Display a message with choices of Yes and No.

**returned value:** Yes -> True No -> False

*screenshots*

#### Parameters

- **message** – message to be displayed.
- **title** – window title
- **default** – default button as boolean (YES=True, NO=False)

**Return type** bool

## 4.6 choice()

`psidialogs.choice(choices=[], message='Pick something.', default=None, title='')`

Present the user with a list of choices. return the choice that he selects. return None if he cancels the selection selection.

*screenshots*

#### Parameters

- **choices** – a list of the choices to be displayed
- **message** – message to be displayed.
- **title** – window title
- **default** – default string of choice

**Return type** None or string

## 4.7 error()

`psidialogs.error(message='Error!', title='')`

Display a warning message

*screenshots*

### Parameters

- **message** – message to be displayed.
- **title** – window title

**Return type** None

## 4.8 message()

`psidialogs.message(message, title='')`

Display a message

*screenshots*

### Parameters

- **message** – message to be displayed.
- **title** – window title

**Return type** None

## 4.9 multi\_choice()

`psidialogs.multi_choice(choices=[], message='Pick as many items as you like.', default=None, title='')`

Present the user with a list of choices. allow him to select multiple items and return them in a list. if the user doesn't choose anything from the list, return the empty list. return None if he cancelled selection.

*screenshots*

### Parameters

- **choices** – a list of the choices to be displayed
- **message** – message to be displayed.
- **title** – window title
- **default** – default list of strings

**Return type** None or list of strings

## 4.10 text()

`psidialogs.text(text, message='', title='')`

This function is suitable for displaying general text, which can be longer than in `message()`

*screenshots*

### Parameters

- **text** – (long) text to be displayed
- **message** – (short) message to be displayed.
- **title** – window title

**Return type** None

## 4.11 warning()

`psidialogs.warning(message='Warning!', title='')`

Display an error message

*screenshots*

### Parameters

- **message** – message to be displayed.
- **title** – window title

**Return type** None


# SCREENSHOTS

## 5.1 ask\_ok\_cancel()

API

### 5.1.1 console

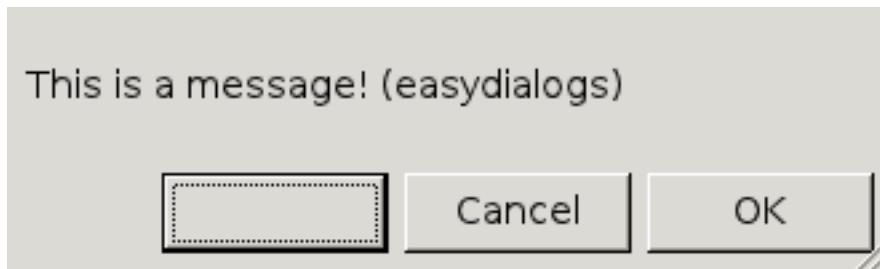
```
$ xterm -e "python -m psidialogs.examples.demo -b console -f ask_ok_cancel"
```



```
True  
This is a message! (console) [Yes/No] ☐
```

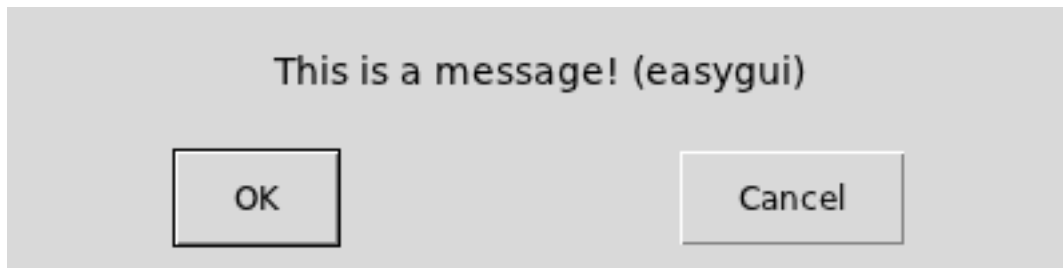
### 5.1.2 easydialogs

```
$ python -m psidialogs.examples.demo -b easydialogs -f ask_ok_cancel
```



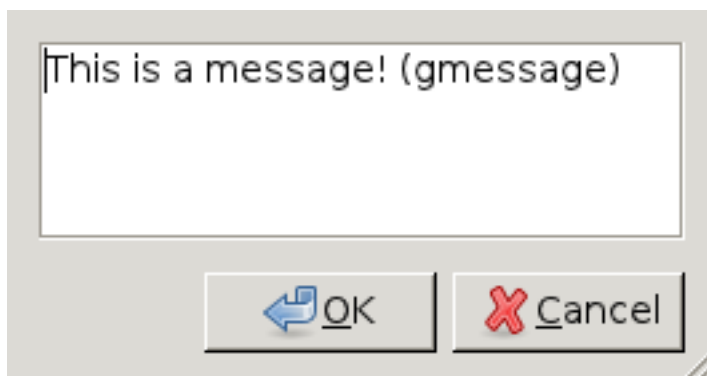
### 5.1.3 easygui

```
$ python -m psidialogs.examples.demo -b easygui -f ask_ok_cancel
```



### 5.1.4 gmessage

```
$ python -m psidialogs.examples.demo -b gmessage -f ask_ok_cancel
```



### 5.1.5 pygtk

```
$ python -m psidialogs.examples.demo -b pygtk -f ask_ok_cancel
```



### 5.1.6 pyqt

```
$ python -m psidialogs.examples.demo -b pyqt -f ask_ok_cancel
```



### 5.1.7 pythondialog

```
$ xterm -e "python -m psidialogs.examples.demo -b pythondialog -f ask_ok_cancel"
```



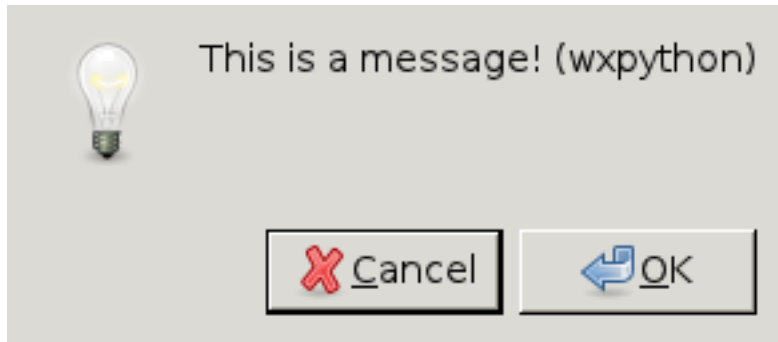
### 5.1.8 tkinter

```
$ python -m psidialogs.examples.demo -b tkinter -f ask_ok_cancel
```



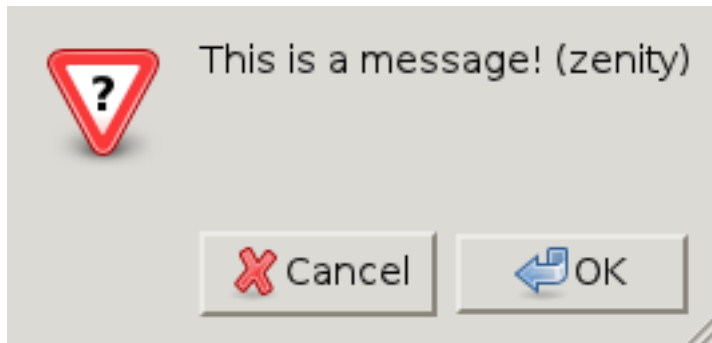
### 5.1.9 wxpython

```
$ python -m psidialogs.examples.demo -b wxpython -f ask_ok_cancel
```



### 5.1.10 zenity

```
$ python -m psidialogs.examples.demo -b zenity -f ask_ok_cancel
```



## 5.2 ask\_string()

API

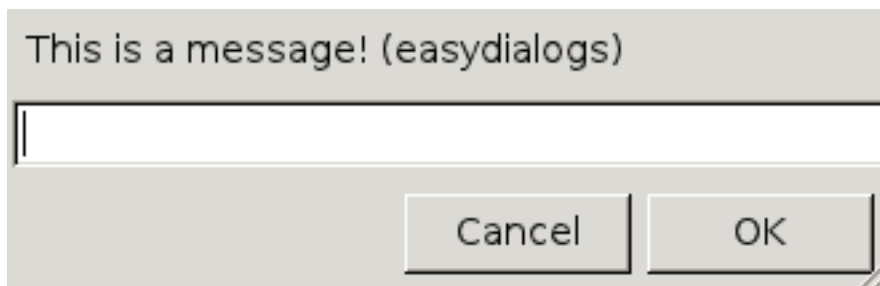
### 5.2.1 console

```
$ xterm -e "python -m psidialogs.examples.demo -b console -f ask_string"
```

```
True  
This is a message! (console)[]
```

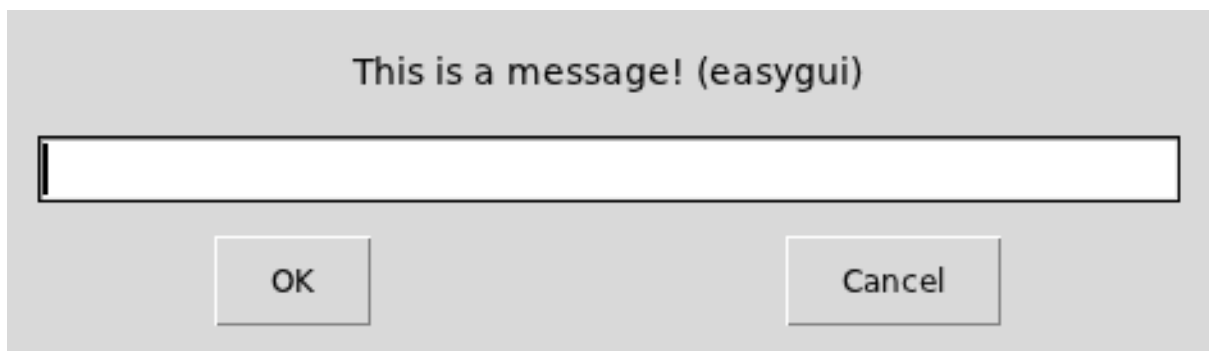
### 5.2.2 easydialogs

```
$ python -m psidialogs.examples.demo -b easydialogs -f ask_string
```



### 5.2.3 easygui

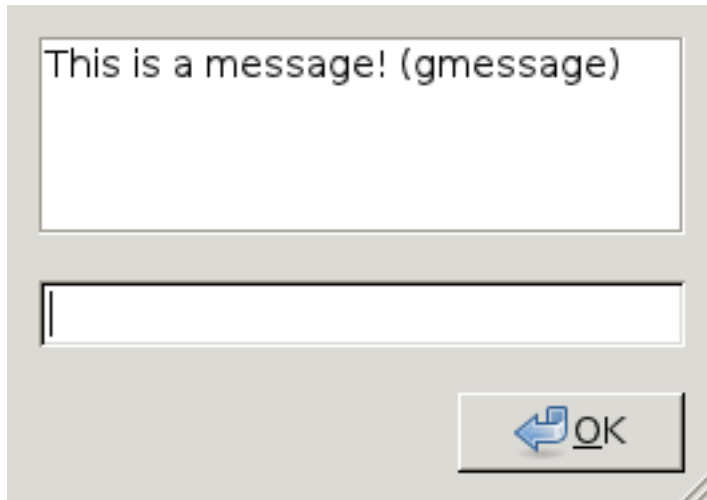
```
$ python -m psidialogs.examples.demo -b easygui -f ask_string
```





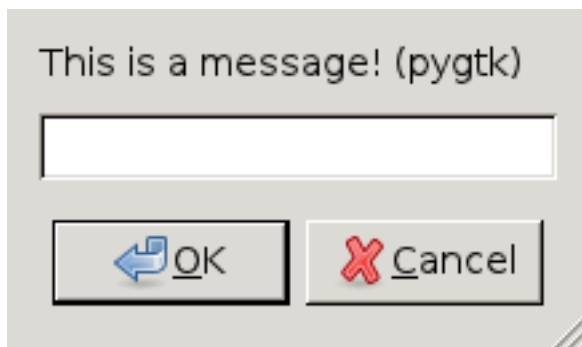
### 5.2.4 gmessage

```
$ python -m psidialogs.examples.demo -b gmessage -f ask_string
```



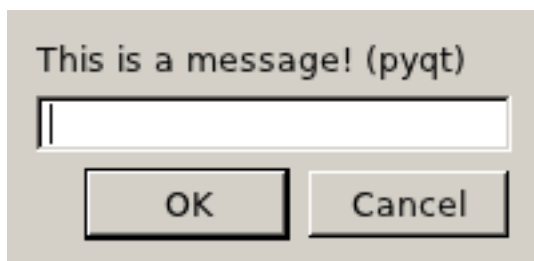
### 5.2.5 pygtk

```
$ python -m psidialogs.examples.demo -b pygtk -f ask_string
```



### 5.2.6 pyqt

```
$ python -m psidialogs.examples.demo -b pyqt -f ask_string
```



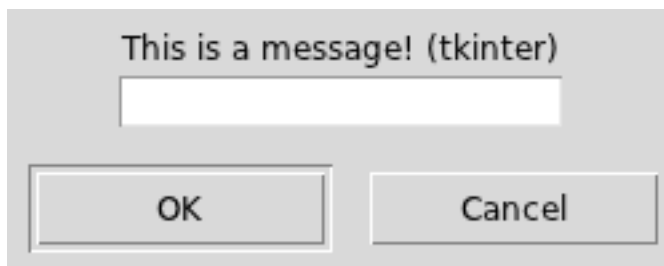
### 5.2.7 pythondialog

```
$ xterm -e "python -m psidialogs.examples.demo -b pythondialog -f ask_string"
```



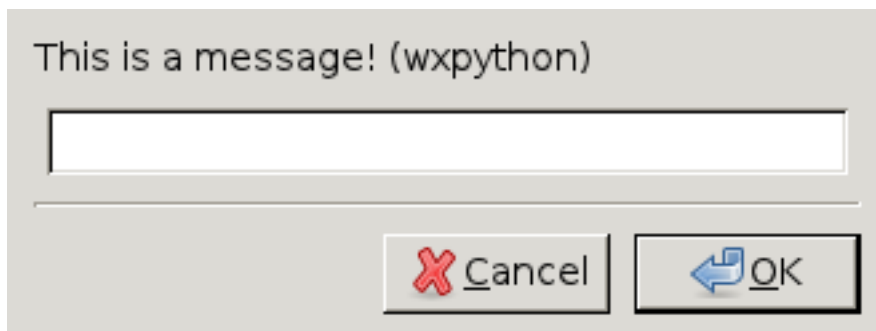
### 5.2.8 tkinter

```
$ python -m psidialogs.examples.demo -b tkinter -f ask_string
```



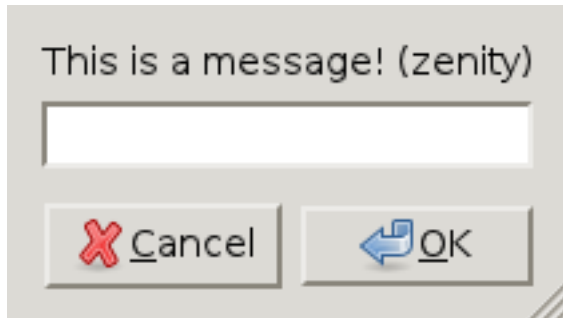
### 5.2.9 wxpython

```
$ python -m psidialogs.examples.demo -b wxpython -f ask_string
```



### 5.2.10 zenity

```
$ python -m psidialogs.examples.demo -b zenity -f ask_string
```



## 5.3 ask\_yes\_no()

API

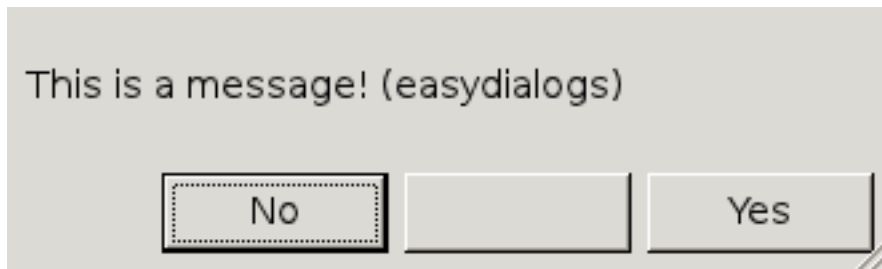
### 5.3.1 console

```
$ xterm -e "python -m psidialogs.examples.demo -b console -f ask_yes_no"
```



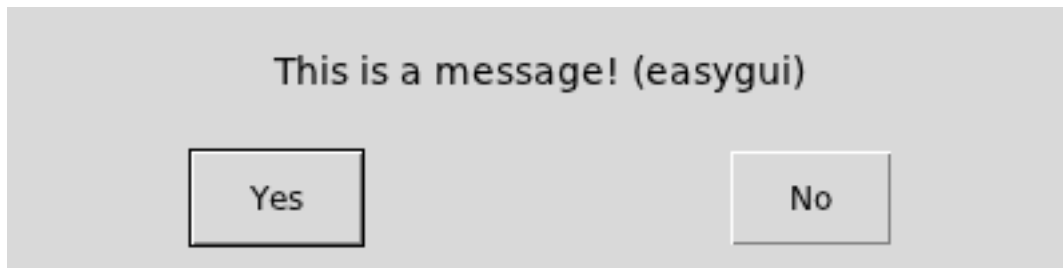
### 5.3.2 easydialogs

```
$ python -m psidialogs.examples.demo -b easydialogs -f ask_yes_no
```



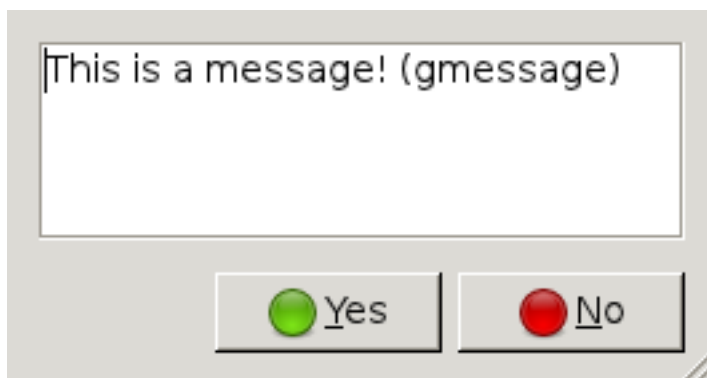
### 5.3.3 easygui

```
$ python -m psidialogs.examples.demo -b easygui -f ask_yes_no
```



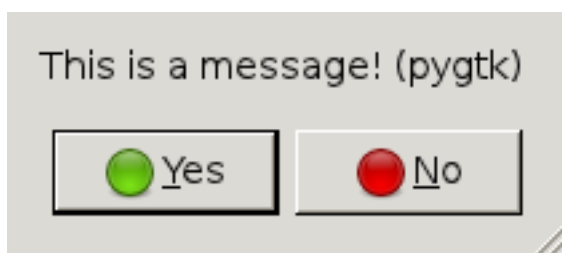
### 5.3.4 gmessage

```
$ python -m psidialogs.examples.demo -b gmessage -f ask_yes_no
```



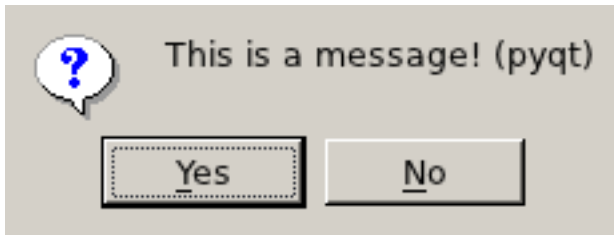
### 5.3.5 pygtk

```
$ python -m psidialogs.examples.demo -b pygtk -f ask_yes_no
```



### 5.3.6 pyqt

```
$ python -m psidialogs.examples.demo -b pyqt -f ask_yes_no
```



### 5.3.7 pythondialog

```
$ xterm -e "python -m psidialogs.examples.demo -b pythondialog -f ask_yes_no"
```



### 5.3.8 tkinter

```
$ python -m psidialogs.examples.demo -b tkinter -f ask_yes_no
```



### 5.3.9 wxpython

```
$ python -m psidialogs.examples.demo -b wxpython -f ask_yes_no
```



### 5.3.10 zenity

```
$ python -m psidialogs.examples.demo -b zenity -f ask_yes_no
```



## 5.4 choice()

API

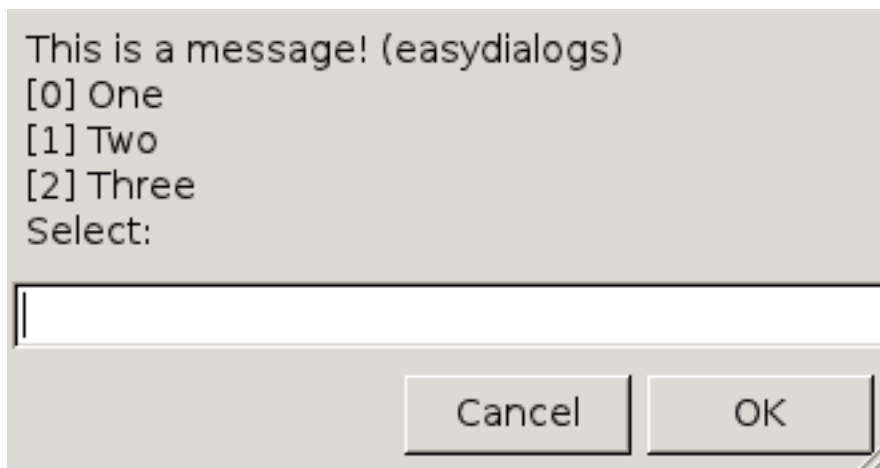
### 5.4.1 console

```
$ xterm -e "python -m psidialogs.examples.demo -b console -f choice"
```

```
True
This is a message! (console)
[0] One
[1] Two
[2] Three
Select:[]
```

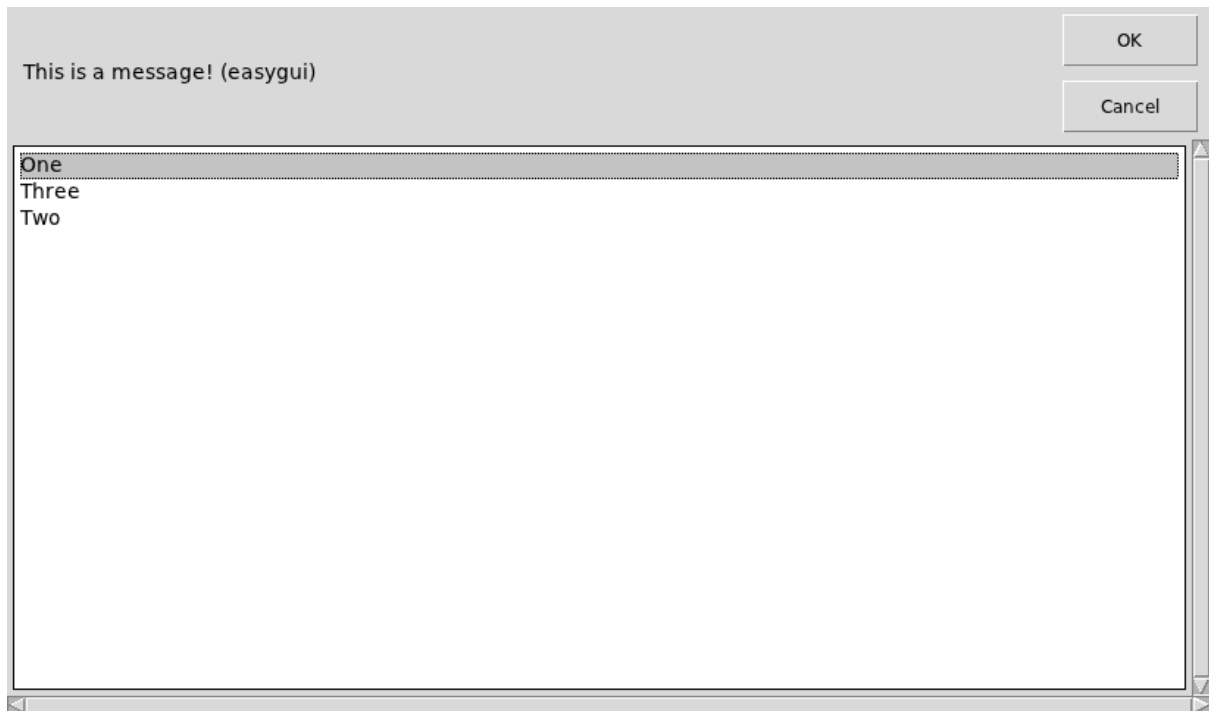
### 5.4.2 easydialogs

```
$ python -m psialogs.examples.demo -b easydialogs -f choice
```



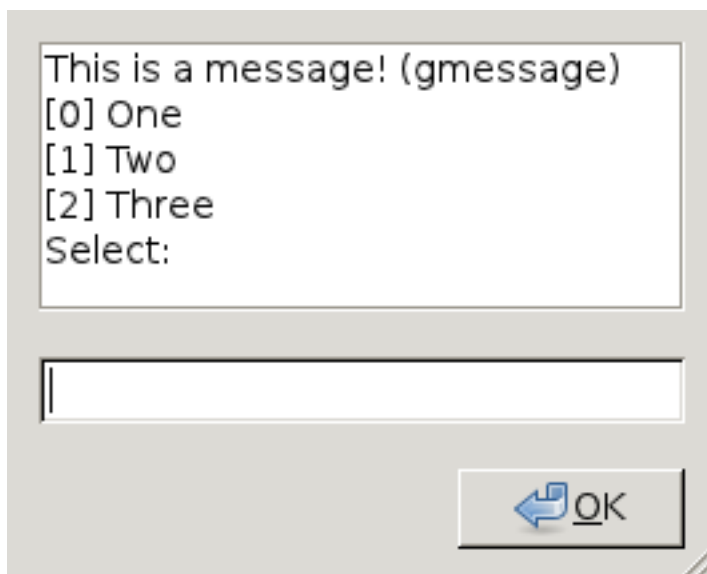
### 5.4.3 easygui

```
$ python -m psialogs.examples.demo -b easygui -f choice
```



#### 5.4.4 gmessage

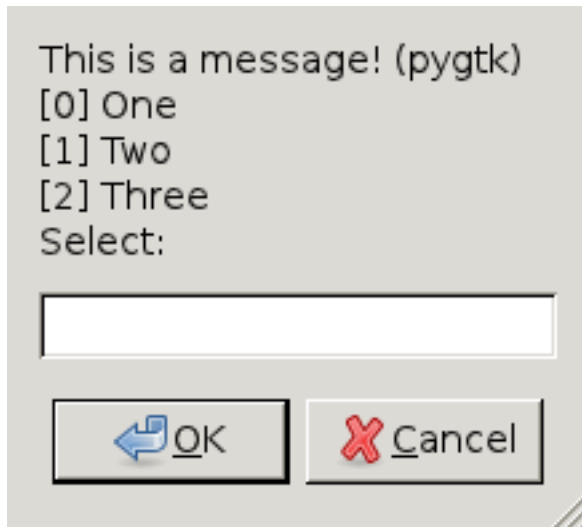
```
$ python -m psidialogs.examples.demo -b gmessage -f choice
```



#### 5.4.5 pygtk

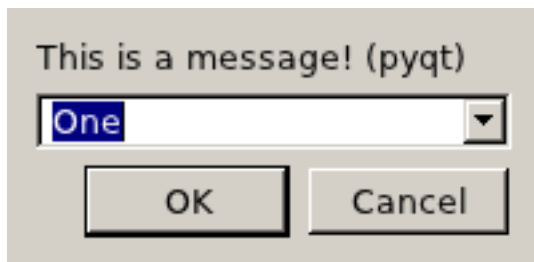


```
$ python -m psidualogs.examples.demo -b pygtk -f choice
```



#### 5.4.6 pyqt

```
$ python -m psidualogs.examples.demo -b pyqt -f choice
```



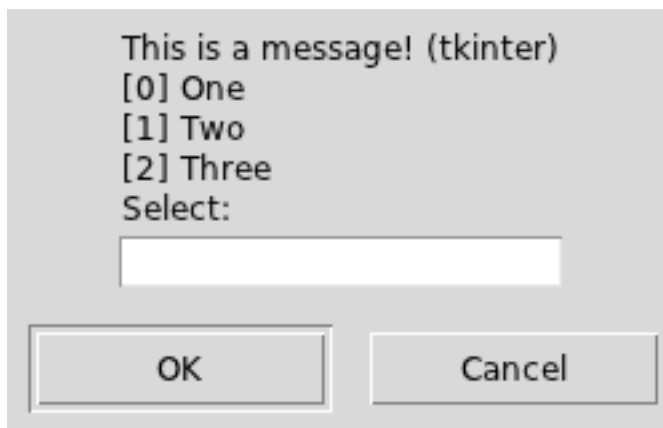
#### 5.4.7 pythondialog

```
$ xterm -e "python -m psidualogs.examples.demo -b pythondialog -f choice"
```



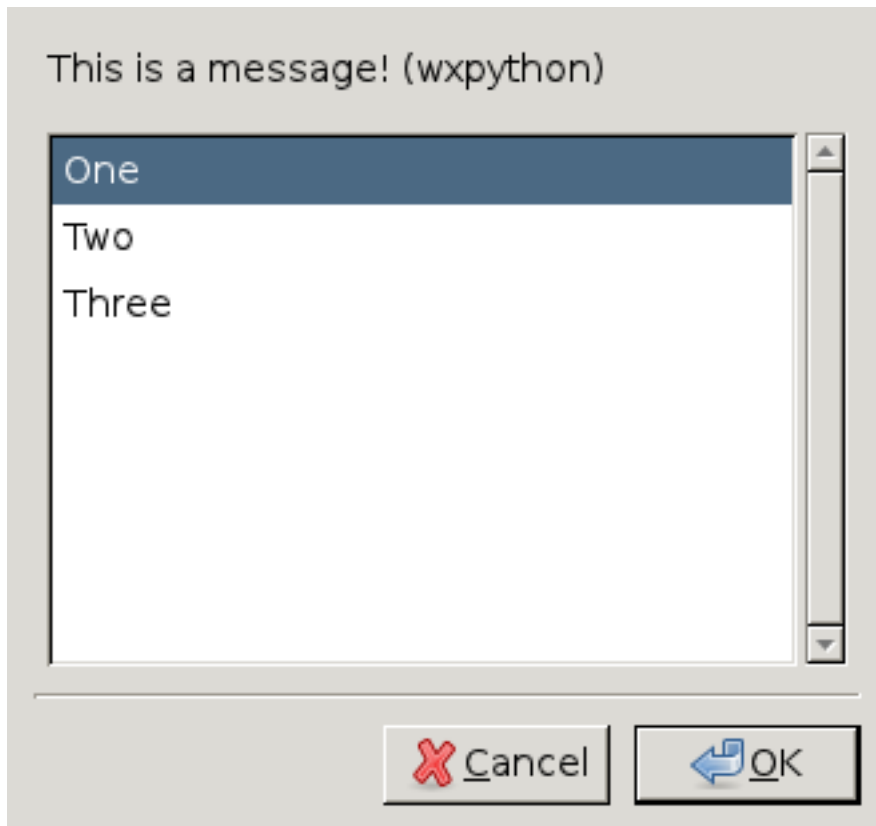
#### 5.4.8 tkinter

```
$ python -m psidialogs.examples.demo -b tkinter -f choice
```



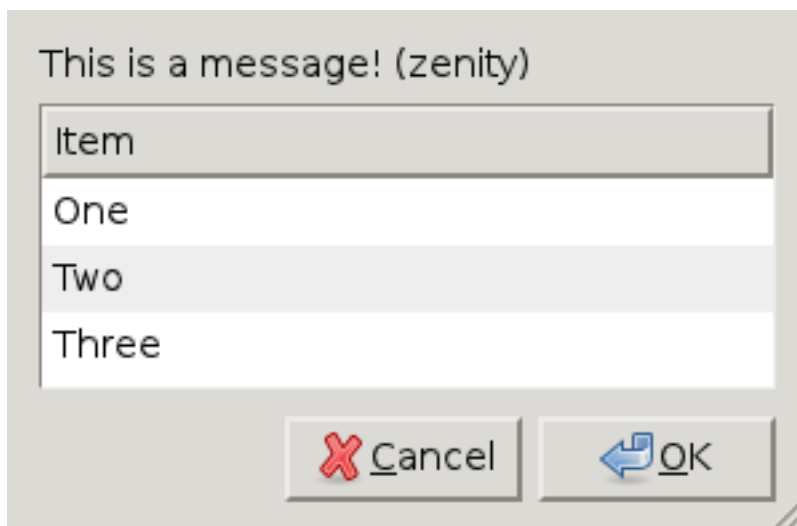
#### 5.4.9 wxpython

```
$ python -m psidialogs.examples.demo -b wxpython -f choice
```



#### 5.4.10 zenity

```
$ python -m psidialogs.examples.demo -b zenity -f choice
```



## 5.5 error()

API

### 5.5.1 console

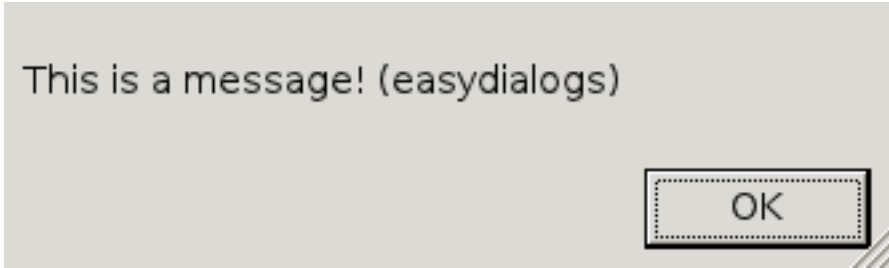
```
$ xterm -e "python -m psidialogs.examples.demo -b console -f error"
```



```
True  
[ERROR] This is a message! (console)[ENTER]
```

### 5.5.2 easydialogs

```
$ python -m psidialogs.examples.demo -b easydialogs -f error
```




This is a message! (easydialogs)

OK

### 5.5.3 easygui

```
$ python -m psidialogs.examples.demo -b easygui -f error
```

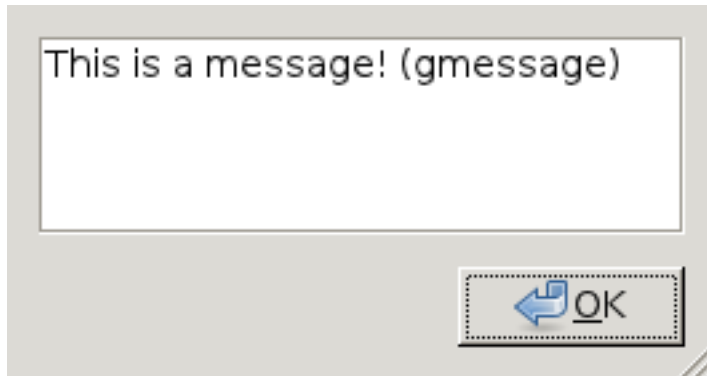


This is a message! (easygui)

OK

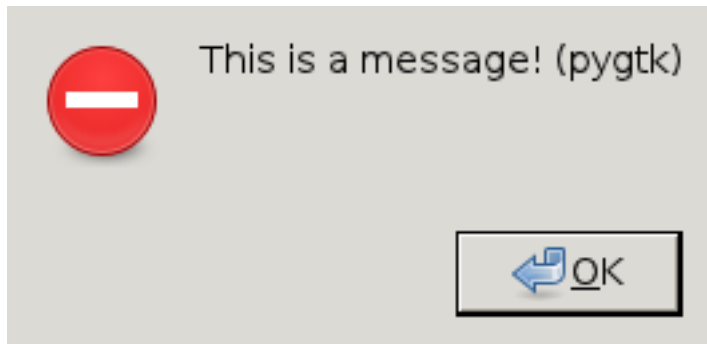
### 5.5.4 gmessage

```
$ python -m psidialogs.examples.demo -b gmessage -f error
```



### 5.5.5 pygtk

```
$ python -m psidialogs.examples.demo -b pygtk -f error
```



### 5.5.6 pyqt

```
$ python -m psidialogs.examples.demo -b pyqt -f error
```



### 5.5.7 pythondialog

```
$ xterm -e "python -m psidialogs.examples.demo -b pythondialog -f error"
```



### 5.5.8 tkinter

```
$ python -m psidialogs.examples.demo -b tkinter -f error
```



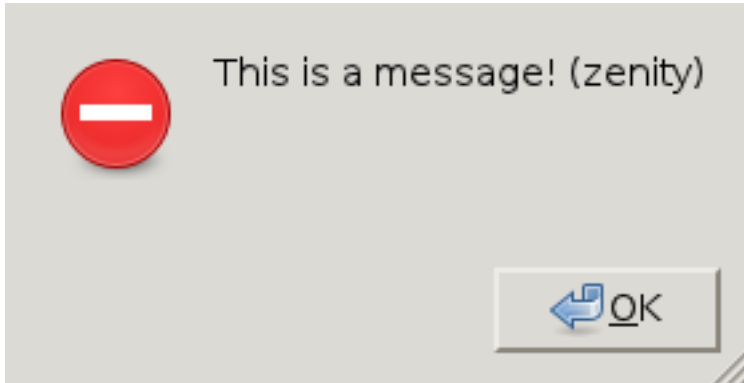
### 5.5.9 wxpython

```
$ python -m psidialogs.examples.demo -b wxpython -f error
```



### 5.5.10 zenity

```
$ python -m psidialogs.examples.demo -b zenity -f error
```



## 5.6 message()

API

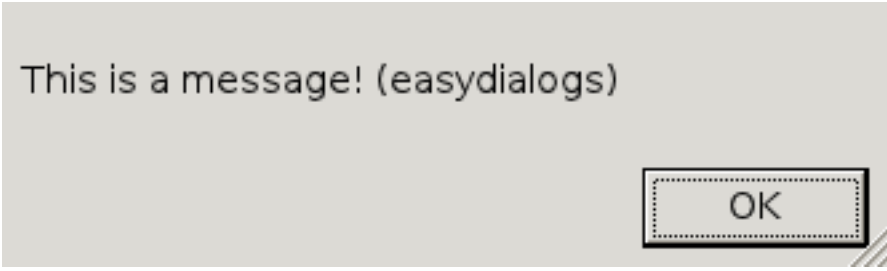
### 5.6.1 console

```
$ xterm -e "python -m psidialogs.examples.demo -b console -f message"
```

```
True  
This is a message! (console)[ENTER]
```

### 5.6.2 easydialogs

```
$ python -m psidialogs.examples.demo -b easydialogs -f message
```




This is a message! (easydialogs)

OK

### 5.6.3 easygui

```
$ python -m psidialogs.examples.demo -b easygui -f message
```

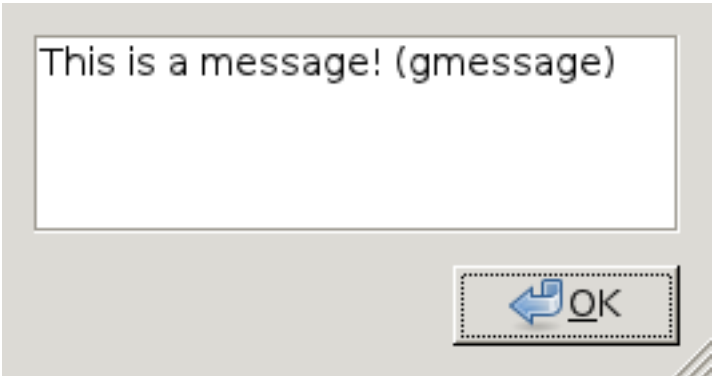


This is a message! (easygui)

OK

### 5.6.4 gmessage

```
$ python -m psidialogs.examples.demo -b gmessage -f message
```



This is a message! (gmessage)

OK

### 5.6.5 pygtk

```
$ python -m psidialogs.examples.demo -b pygtk -f message
```



This is a message! (pygtk)

OK



### 5.6.6 pyqt

```
$ python -m psidialogs.examples.demo -b pyqt -f message
```



### 5.6.7 pythondialog

```
$ xterm -e "python -m psidialogs.examples.demo -b pythondialog -f message"
```



### 5.6.8 tkinter

```
$ python -m psidialogs.examples.demo -b tkinter -f message
```



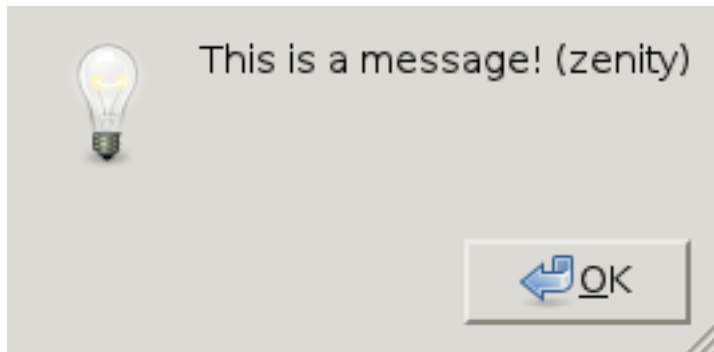
### 5.6.9 wxpython

```
$ python -m psidialogs.examples.demo -b wxpython -f message
```



### 5.6.10 zenity

```
$ python -m psidialogs.examples.demo -b zenity -f message
```



## 5.7 multi\_choice()

API

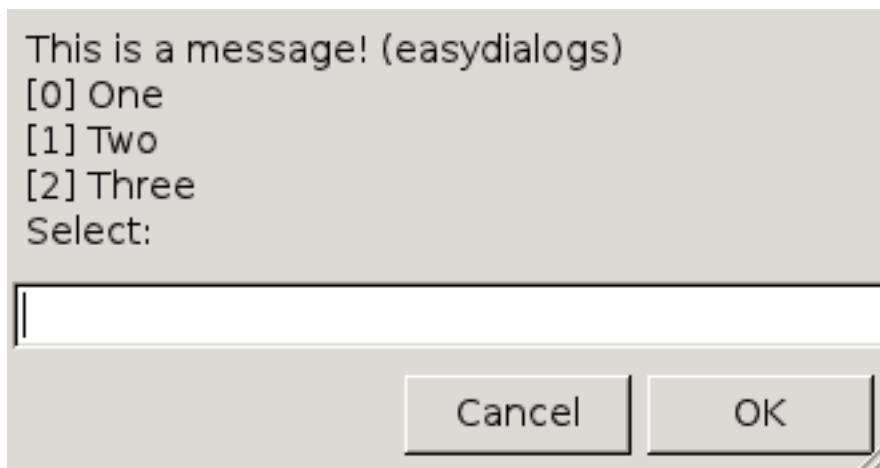
### 5.7.1 console

```
$ xterm -e "python -m psidialogs.examples.demo -b console -f multi_choice"
```

```
True
This is a message! (console)
[0] One
[1] Two
[2] Three
Select:[]
```

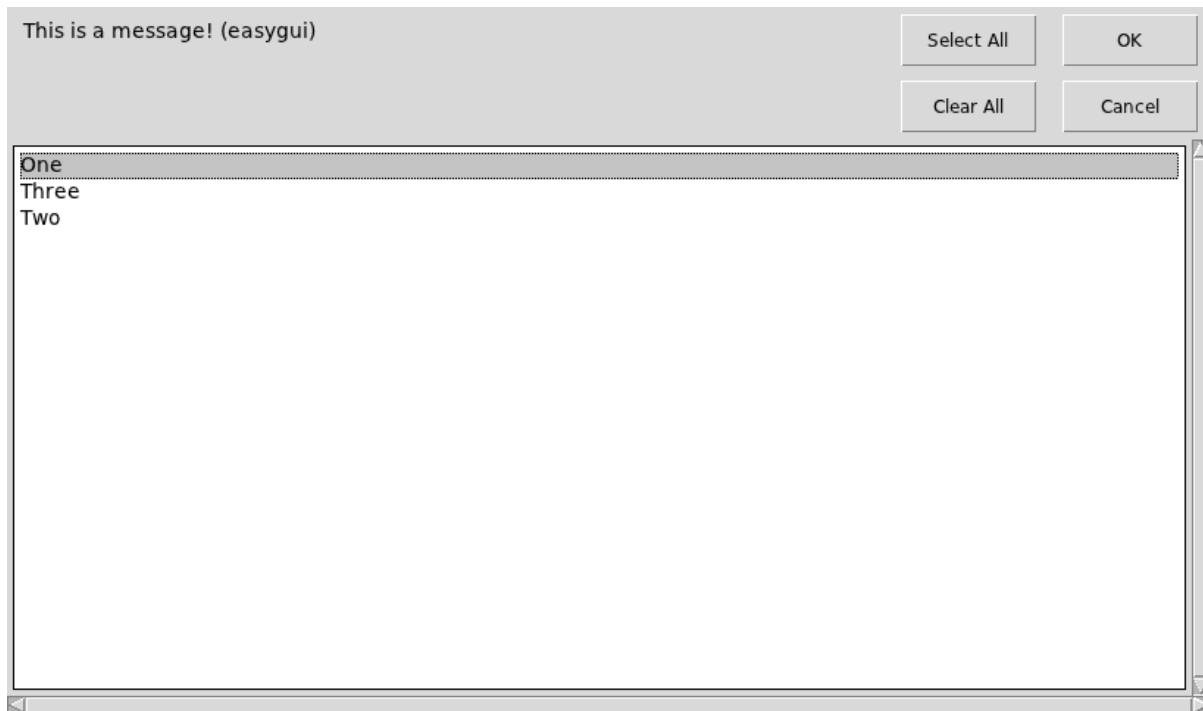
### 5.7.2 easydialogs

```
$ python -m psialogs.examples.demo -b easydialogs -f multi_choice
```



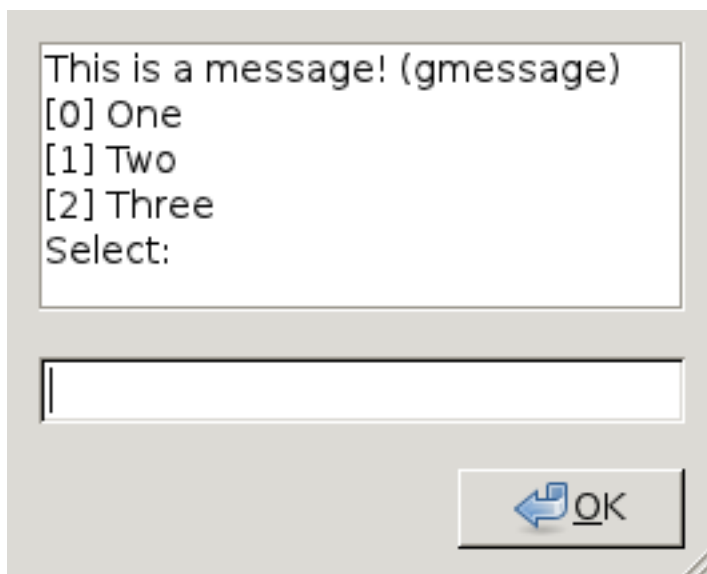
### 5.7.3 easygui

```
$ python -m psialogs.examples.demo -b easygui -f multi_choice
```



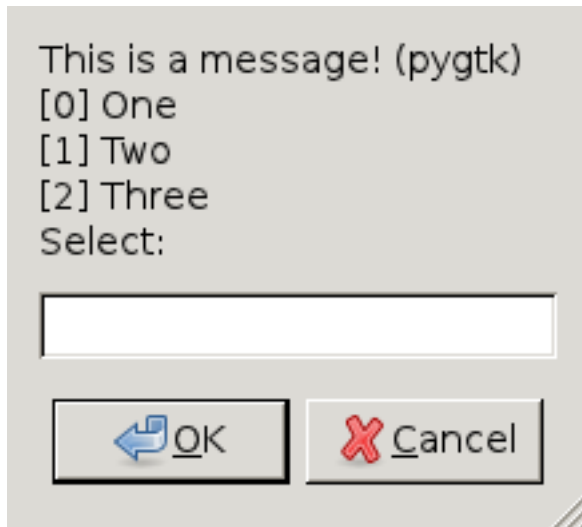
### 5.7.4 gmessage

```
$ python -m psidialogs.examples.demo -b gmessage -f multi_choice
```



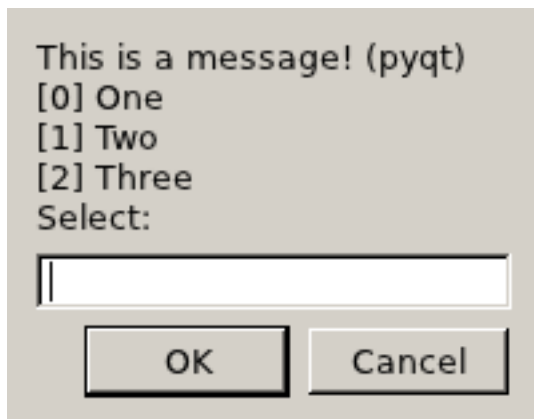
### 5.7.5 pygtk

```
$ python -m psidialogs.examples.demo -b pygtk -f multi_choice
```



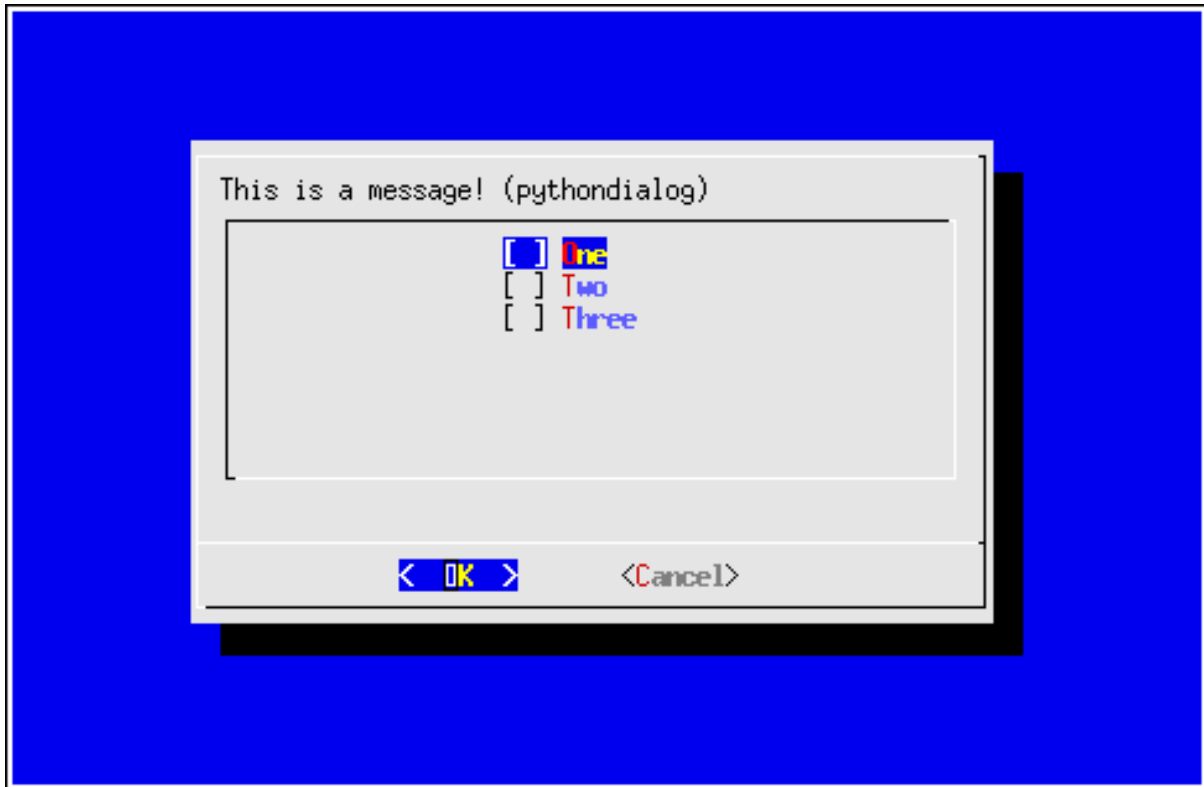
### 5.7.6 pyqt

```
$ python -m psidialogs.examples.demo -b pyqt -f multi_choice
```



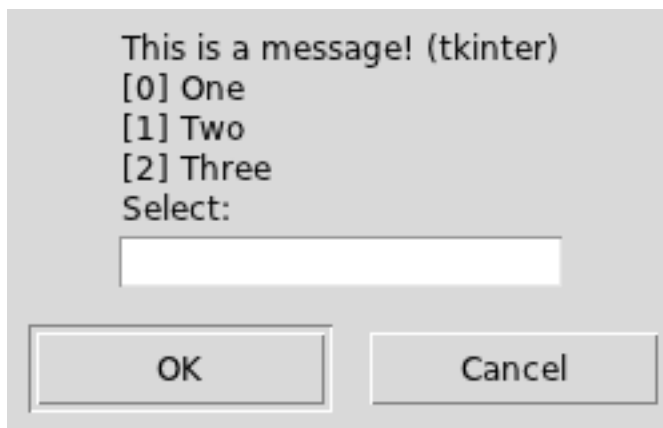
### 5.7.7 pythondialog

```
$ xterm -e "python -m psidialogs.examples.demo -b pythondialog -f multi_choice"
```



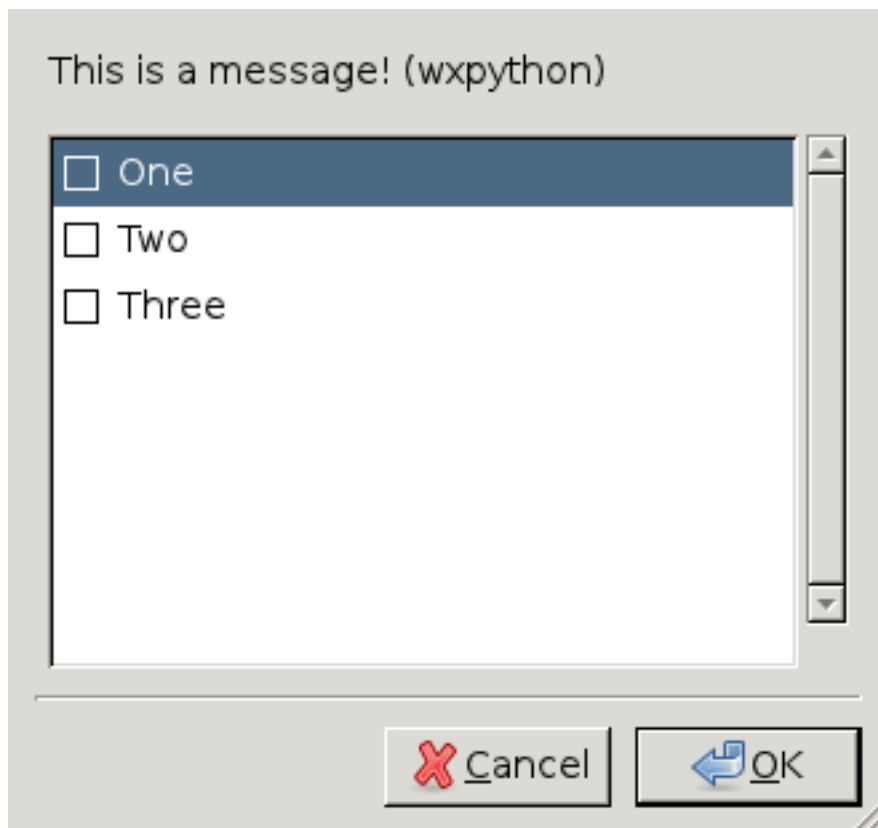
### 5.7.8 tkinter

```
$ python -m psidialogs.examples.demo -b tkinter -f multi_choice
```



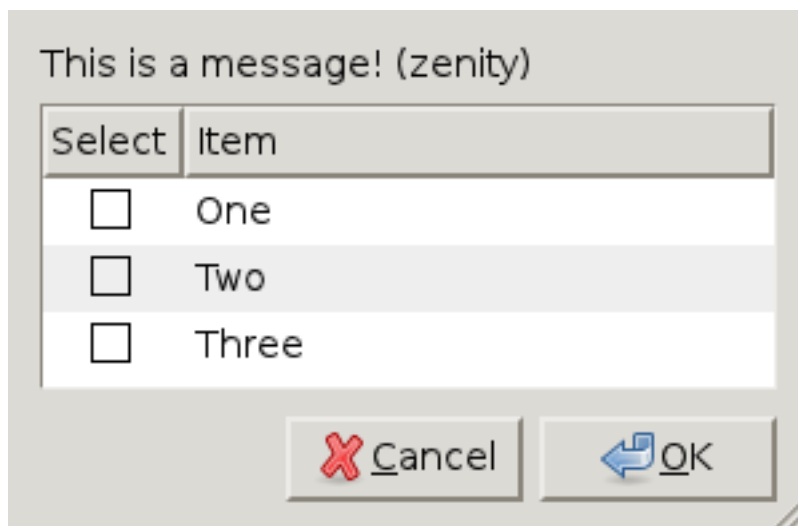
### 5.7.9 wxpython

```
$ python -m psidialogs.examples.demo -b wxpython -f multi_choice
```



### 5.7.10 zenity

```
$ python -m psidialogs.examples.demo -b zenity -f multi_choice
```



## 5.8 text()

API

### 5.8.1 console

```
$ xterm -e "python -m psidialogs.examples.demo -b console -f text"
```

```

    while 1:
        #d = dict([(x.backend, x.name) for x in psidialogs.all_backends()])
        #names=sorted(d.keys())
        names=sorted(BackendLoader().all_names)
        b = psidialogs.choice(names, 'Select backend!', title=title)
        if not b:
            break
        BackendLoader().force(b)
        try:
            BackendLoader().selected()
        except Exception, detail:
            BackendLoader().force(None)
            psidialogs.text('Exception:\n' + str(detail))
            continue

        #psidialogs.set_backend(force_backend=d[b])
        selectfunc(title, **kwargs)

@entrypoint
def demo(backend=None, function=None, title=''):
    print os.isatty(sys.stdout.fileno())
    selectbackend(backend=backend, function=function, title=title)

[ENTER]
```

### 5.8.2 easydialogs

```
$ python -m psidialogs.examples.demo -b easydialogs -f text
```



```
This is a message! (easydialogs)
from ent
rypoint2 import entrypoint
from psidialog
s.backendloader import BackendLoader
impo
rt inspect
import logging
import os
impor
t psidialogs
import sys

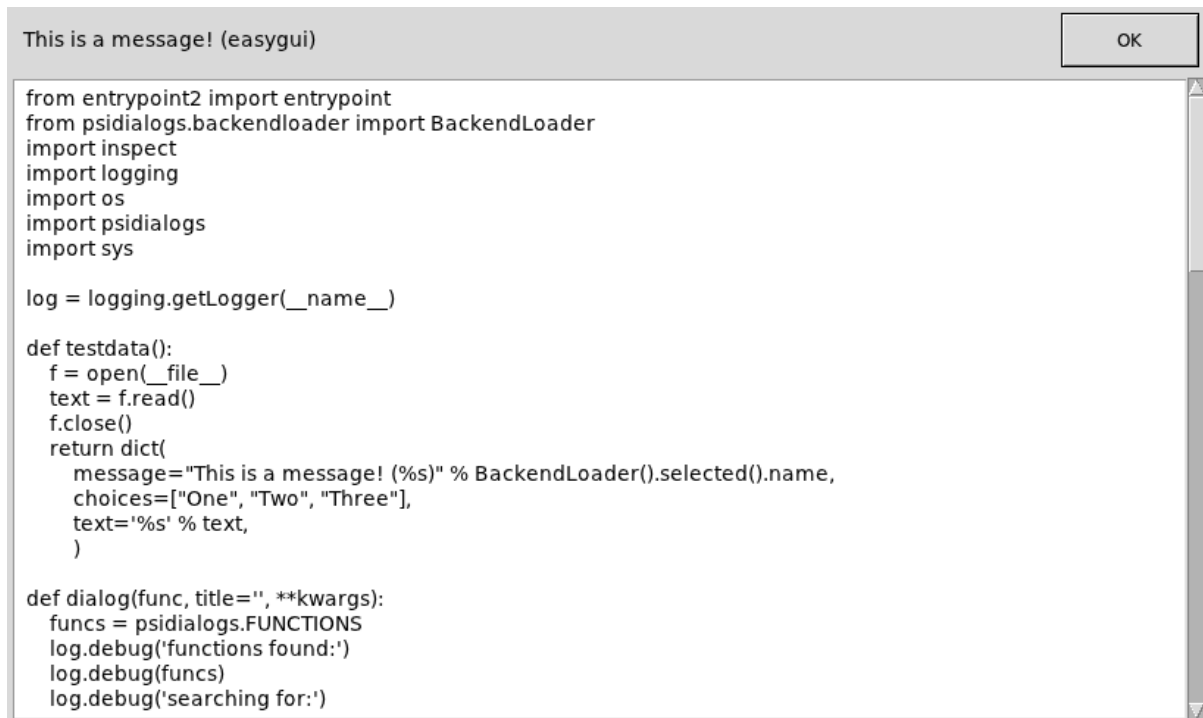
log = logging.ge
tLogger(__name__)

def testdata():
    f
    = open(☒
```



### 5.8.3 easygui

```
$ python -m psidialogs.examples.demo -b easygui -f text
```



### 5.8.4 gmessage

```
$ python -m psidialogs.examples.demo -b gmessage -f text
```

```
This is a message! (gmessage)
from entrypoint2 import entrypoint
from psidialogs.backendloader import BackendLoader
import inspect
import logging
import os
import psidialogs
import sys

log = logging.getLogger(__name__)

def testdata():
    f = open(__file__)
    text = f.read()
    f.close()
    return dict(
        message="This is a message! (%s)" % BackendLoader().selected().name,
        choices=["One", "Two", "Three"],
        text='%s' % text,
    )

def dialog(func, title="", **kwargs):
    funcs = psidialogs.FUNCTIONS
    log.debug('functions found:')
    log.debug(funcs)
    log.debug('searching for:')
    log.debug(func)
    f = None
    for x in funcs:
        if x.__name__ == func:
```



### 5.8.5 pygtk

```
$ python -m psidialogs.examples.demo -b pygtk -f text
```



```

This is a message! (pygtk)
from entrypoint2 import entrypoint
from psidualogs.backendloader import BackendLoader
import inspect
import logging
import os
import psidualogs
import sys

log = logging.getLogger(__name__)

def testdata():
    f = open(__file__)
    text = f.read()
    f.close()
    return dict(
        message="This is a message! (%s)" % BackendLoader
        ().selected().name,
        choices=["One", "Two", "Three"],
        text='%s' % text,
    )

def dialog(func, title="", **kwargs):
    funcs = psidualogs.FUNCTIONS
    log.debug('functions found:')
    log.debug(funcs)
    log.debug('searching for:')
    log.debug(func)
    f = None
    for x in funcs:
        if x.__name__ == func:
            f = x
    assert f
    argnames, varargs, varkw, defaults = inspect.getargspec(f)
    #argnames = psidualogs.argnames(func)
    args = testdata()
    if title:
        args['title'] = title
    args = dict([(k, v) for (k, v) in args.items() if k in argnames])
    result=None
    exec 'result = psidualogs.%s(**args)' % (func)
    #result = psidualogs.__dict__[func](**args)
    #print 'result: ', result
    log.debug('result:'+str(result))
    if result is not None:

```

## 5.8.6 pyqt

```
$ python -m psidialogs.examples.demo -b pyqt -f text
```



```

This is a message! (pyqt)
from entrypoint2 import entrypoint
from psidialogs.backendloader import BackendLoader
import inspect
import logging
import os
import psidialogs
import sys

log = logging.getLogger(__name__)

def testdata():
    f = open(__file__)
    text = f.read()
    f.close()
    return dict(
        message="This is a message! (%s)" %
BackendLoader().selected().name,
        choices=["One", "Two", "Three"],
        text='%s' % text,
    )

def dialog(func, title="", **kwargs):
    funcs = psidialogs.FUNCTIONS
    log.debug('functions found:')
    log.debug(funcs)
    log.debug('searching for:')
    log.debug(func)
    f = None
    for x in funcs:
        if x.__name__ == func:
            f = x
    assert f
    argnames, varargs, varkw, defaults = inspect.getargspec(f)
    #argnames = psidialogs.argnames(func)
    args = testdata()
    if title:
        args['title'] = title
    args = dict([(k, v) for (k, v) in args.items() if k in argnames])
    result=None
    exec 'result = psidialogs.%s(**args)' % (func)
    #result = psidialogs.__dict__[func](**args)
    #print 'result: ', result
    log.debug('result:'+str(result))
    if result is not None:
        psidialogs.text('Return value="%s"' % result)

def selectfunc(title="", function=None, **kwargs):
    if function:
        dialog(function, title, **kwargs)
    else:

```

### 5.8.7 pythondialog

```
$ xterm -e "python -m psialogs.examples.demo -b pythondialog -f text"
```



```
from entrypoint2 import entrypoint
from psialogs.backendloader import BackendLoader
import inspect
import logging
import os
import psialogs
import sys

log = logging.getLogger(__name__)

def testdata():
    f = open(__file__)
    text = f.read()
    f.close()
    return dict(
        message="This is a message! (%s)" % BackendLoader().selected().nam
    )
```

(+) 13%

< EXIT >

### 5.8.8 tkinter

```
$ python -m psialogs.examples.demo -b tkinter -f text
```

```

    assert f
    argnames, varargs, varkw, defaults
= inspect.getargspec(f)
    #argnames =
psialogs.argnames(func)
    args = testdata()
    if title:
        args['title'] = title
        args = dict([(k, v) for (k, v) in
args.items() if k in argnames])
        result=None
        exec 'result =
psialogs.%s(**args)' % (func)
        #result =
psialogs.__dict__[func](**args)
        #print 'result: ', result
        log.debug('result:'+str(result))
        if result is not None:
            psialogs.text('Return
value="%s"' % result)

def selectfunc(title="",
function=None, **kwargs):
    if function:
        dialog(function, title, **kwargs)
    else:
        while 1:
            funcs =
psialogs.FUNCTION_NAMES
            funcs.sort()
            func =
psialogs.choice(funcs, 'Select
function!', title=title)
            if not func:
                break
            dialog(func, title, **kwargs)

def selectbackend(backend=None,
title="", **kwargs):
    if backend:
        BackendLoader().force(backend)

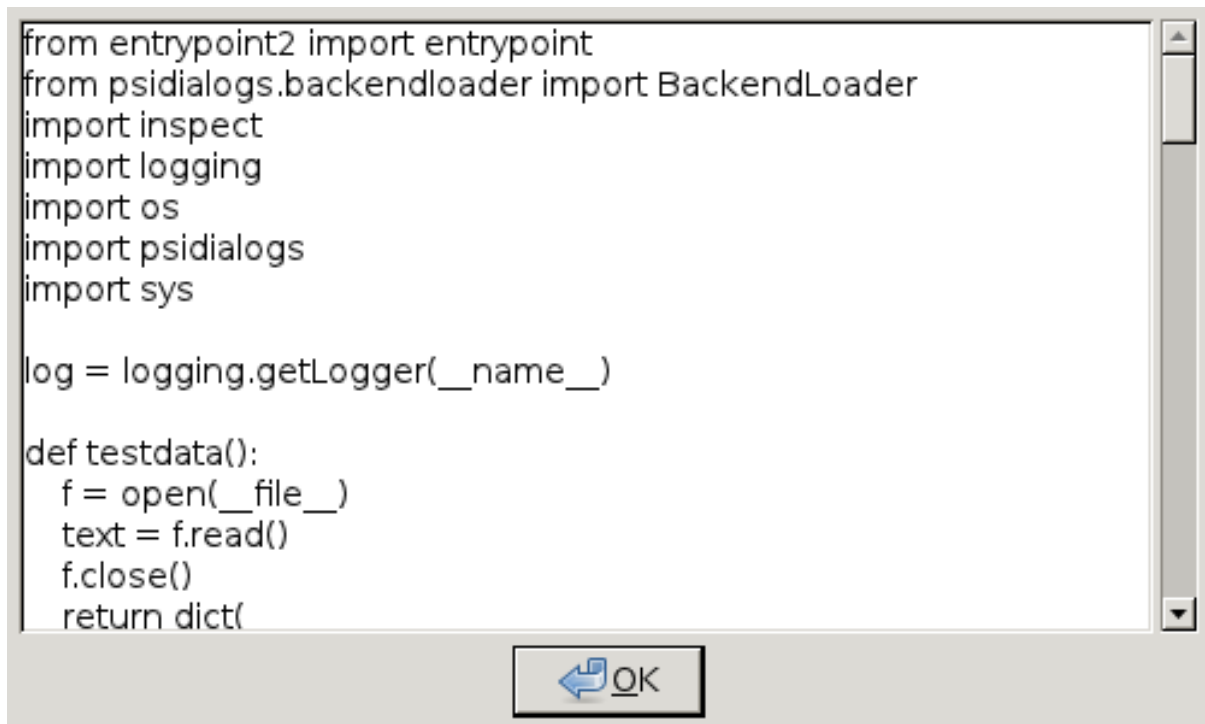
selectfunc(title, **kwargs)
else:
    while 1:

```



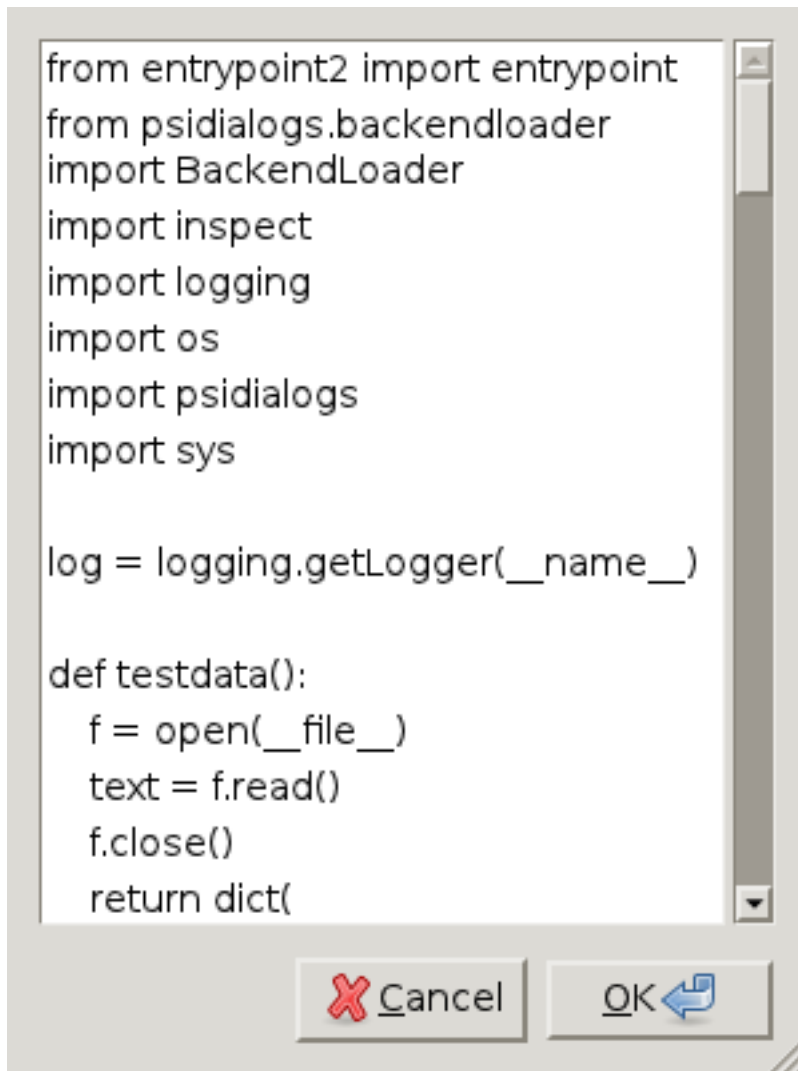
### 5.8.9 wxpython

```
$ python -m psialogs.examples.demo -b wxpython -f text
```



### 5.8.10 zenity

```
$ python -m psialogs.examples.demo -b zenity -f text
```



## 5.9 warning()

API

### 5.9.1 console

```
$ xterm -e "python -m psidialogs.examples.demo -b console -f warning"
```

```
True  
[WARNING] This is a message! (console)[ENTER]
```

### 5.9.2 easydialogs

```
$ python -m psidialogs.examples.demo -b easydialogs -f warning
```

This is a message! (easydialogs)

OK

### 5.9.3 easygui

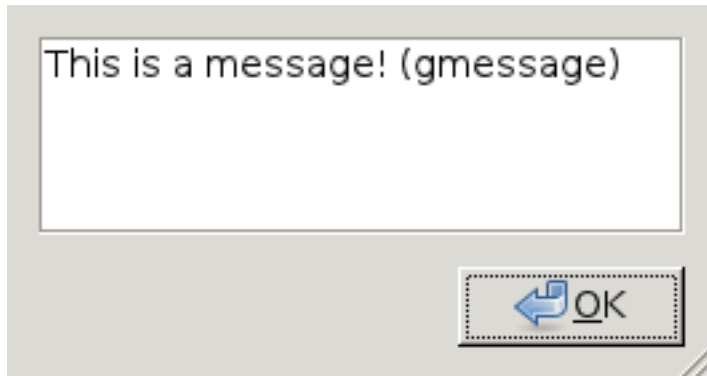
```
$ python -m psidialogs.examples.demo -b easygui -f warning
```

This is a message! (easygui)

OK

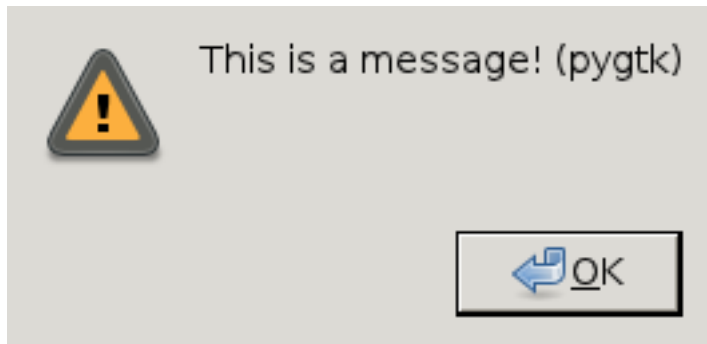
### 5.9.4 gmessage

```
$ python -m psidialogs.examples.demo -b gmessage -f warning
```



### 5.9.5 pygtk

```
$ python -m psidialogs.examples.demo -b pygtk -f warning
```



### 5.9.6 pyqt

```
$ python -m psidialogs.examples.demo -b pyqt -f warning
```



### 5.9.7 pythondialog

```
$ xterm -e "python -m psidialogs.examples.demo -b pythondialog -f warning"
```



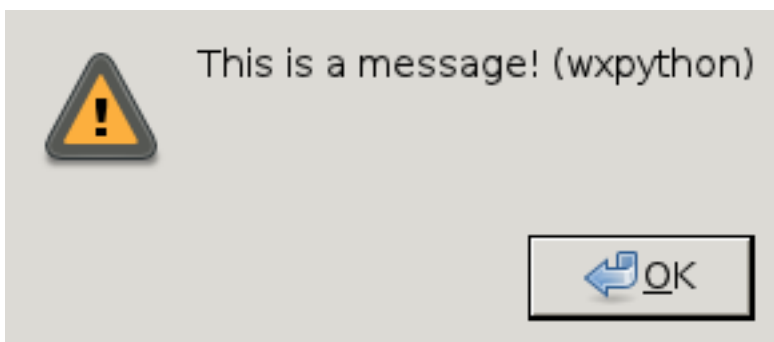
### 5.9.8 tkinter

```
$ python -m psidialogs.examples.demo -b tkinter -f warning
```



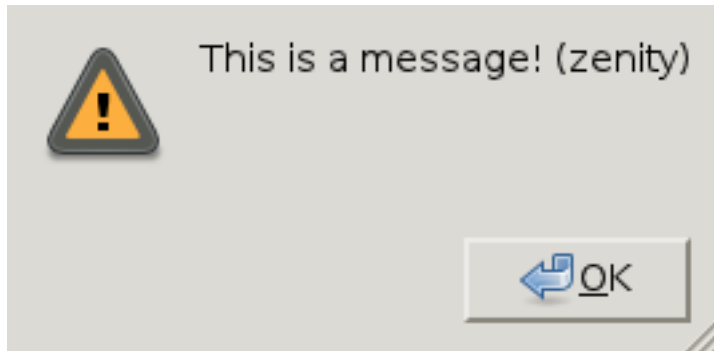
### 5.9.9 wxpython

```
$ python -m psidialogs.examples.demo -b wxpython -f warning
```



### 5.9.10 zenity

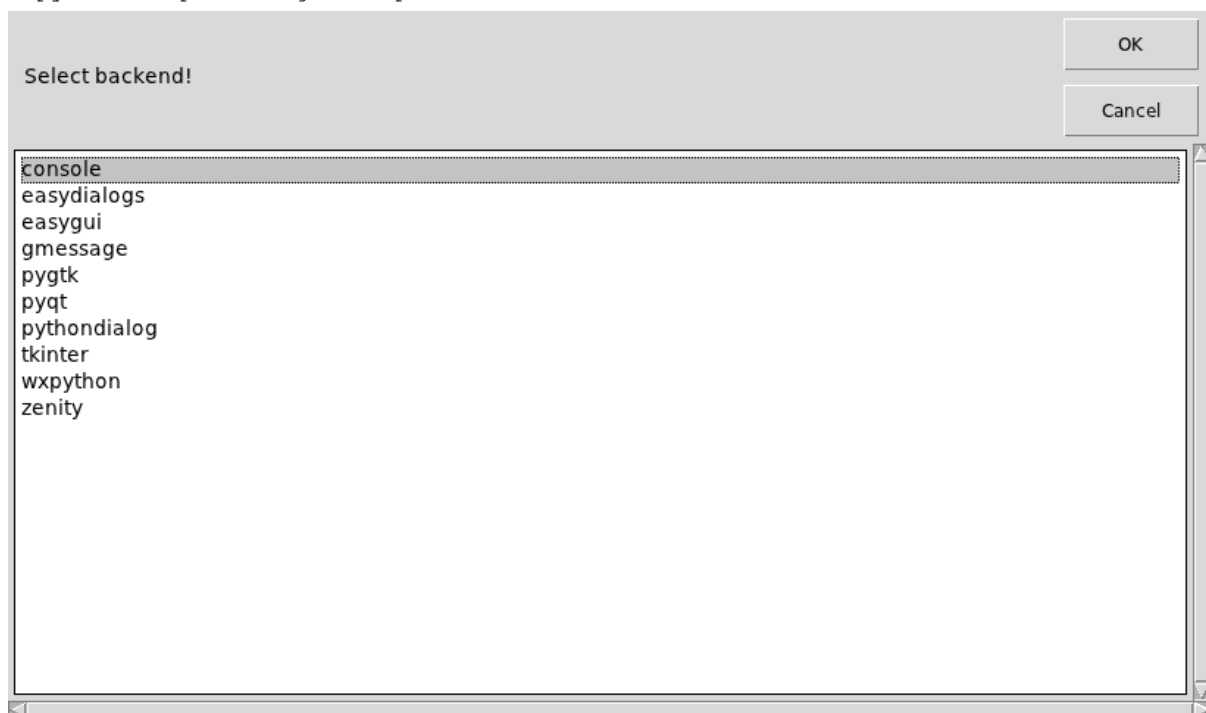
```
$ python -m psidialogs.examples.demo -b zenity -f warning
```



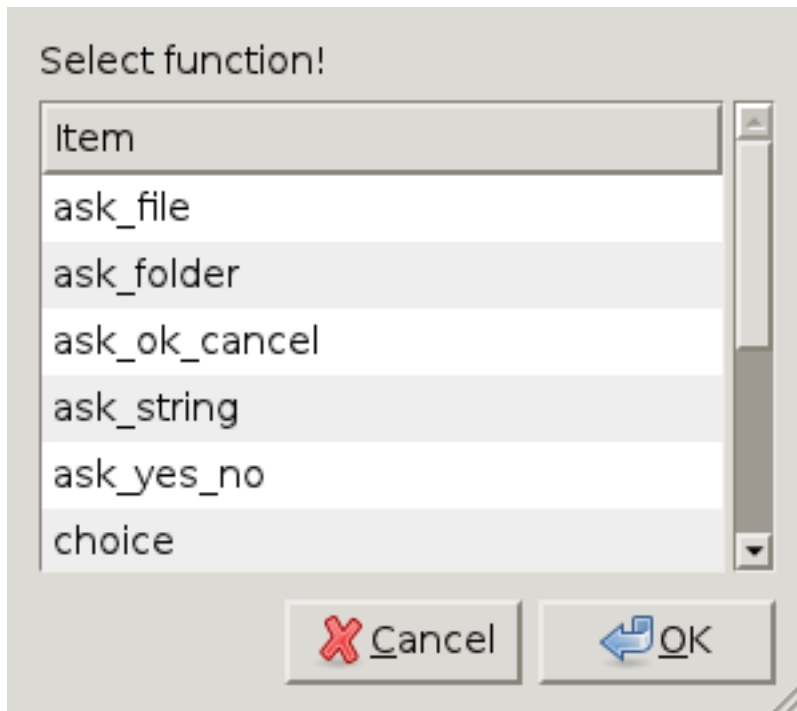
# DEMO

Backends and functions can be selected from list or as command line parameter

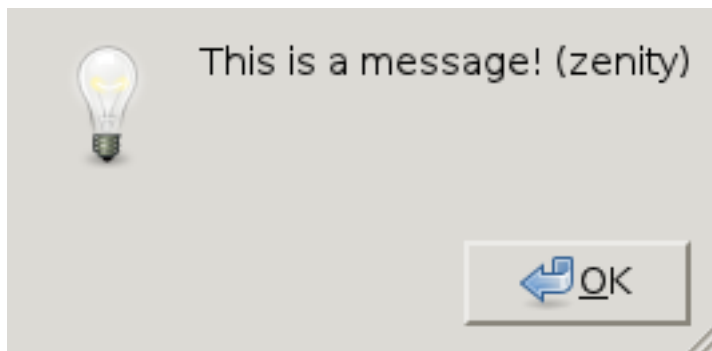
```
$ python -m psidialogs.examples.demo
```



```
$ python -m psidialogs.examples.demo --backend zenity
```



```
$ python -m psialogs.examples.demo --backend zenity --function message
```



## 6.1 command line help

```
$ python -m psialogs.examples.demo --help
usage: demo.py [-h] [-b BACKEND] [-f FUNCTION] [-t TITLE] [--debug]
```

optional arguments:

```
-h, --help            show this help message and exit
-b BACKEND, --backend BACKEND
-f FUNCTION, --function FUNCTION
-t TITLE, --title TITLE
--debug              set logging level to DEBUG
```



# DEVELOPMENT

## 7.1 Tools

1. `setuptools`
2. `Paver`
3. `nose`
4. `ghp-import`
5. `pyflakes`
6. `pychecker`
7. `paved fork`
8. `Sphinx`
9. `sphinxcontrib-programsscreenshot`
10. `sphinxcontrib-paverutils`
11. `autorun` from `sphinx-contrib` (there is no simple method, you have to download/unpack/setup)

## 7.2 Install on ubuntu

```
sudo apt-get install python-setuptools
sudo apt-get install python-paver
sudo apt-get install python-nose
sudo easy_install ghp-import
sudo apt-get install pyflakes
sudo apt-get install pychecker
sudo easy_install https://github.com/ponty/paved/zipball/master
sudo apt-get install scrot
sudo apt-get install xvfb
sudo apt-get install xserver-xephyr
sudo apt-get install python-imaging
sudo apt-get install python-sphinx
sudo easy_install sphinxcontrib-programsscreenshot
sudo easy_install sphinxcontrib-programoutput
sudo easy_install sphinxcontrib-paverutils
```

## 7.3 Tasks

`Paver` is used for task management, settings are saved in `pavement.py`. `Sphinx` is used to generate documentation.

print [paver](#) settings:

```
paver printoptions
```

clean generated files:

```
paver clean
```

generate documentation under *docs/\_build/html*:

```
paver cog pdf html
```

upload documentation to [github](#):

```
paver ghpages
```

run unit tests:

```
paver nose
#or
nosetests --verbose
```

check python code:

```
paver pyflakes
paver pychecker
```

generate python distribution:

```
paver sdist
```

upload python distribution to [PyPI](#):

```
paver upload
```

# INDICES AND TABLES

- *genindex*
- *modindex*
- *search*

# PYTHON MODULE INDEX

## p

`psidialogs`, 8

# INDEX

## A

`ask_file()` (in module `psialogs`), 5  
`ask_folder()` (in module `psialogs`), 5  
`ask_ok_cancel()` (in module `psialogs`), 5  
`ask_string()` (in module `psialogs`), 6  
`ask_yes_no()` (in module `psialogs`), 6

## C

`choice()` (in module `psialogs`), 6

## E

`error()` (in module `psialogs`), 7

## M

`message()` (in module `psialogs`), 7  
`multi_choice()` (in module `psialogs`), 7

## P

`psialogs` (module), 5–8

## T

`text()` (in module `psialogs`), 8

## W

`warning()` (in module `psialogs`), 8