

---

# **pyscreenshot Documentation**

*Release 0.4.2*

**ponty**

**Sep 18, 2017**



---

## Contents

---

<b>1</b>	<b>About</b>	<b>1</b>
<b>2</b>	<b>Examples</b>	<b>3</b>
<b>3</b>	<b>Installation</b>	<b>5</b>
3.1	General . . . . .	5
3.2	Ubuntu 14.04 . . . . .	5
3.3	Uninstall . . . . .	5
<b>4</b>	<b>Command line interface</b>	<b>7</b>
<b>5</b>	<b>command line help</b>	<b>9</b>
<b>6</b>	<b>Hierarchy</b>	<b>11</b>
<b>7</b>	<b>API</b>	<b>13</b>
	<b>Python Module Index</b>	<b>15</b>



The `pyscreenshot` module can be used to copy the contents of the screen to a `PIL` or `Pillow` image memory using various back-ends. Replacement for the `ImageGrab` Module, which works on Windows only, so Windows users don't need this library. For handling image memory (e.g. saving to file, converting,..) please read `PIL` or `Pillow` documentation.

**Links:**

- home: <https://github.com/ponty/pyscreenshot>
- documentation: <http://pyscreenshot.readthedocs.org>
- PYPI: <https://pypi.python.org/pypi/pyscreenshot>

**Goal:** Pyscreenshot tries to allow to take screenshots without installing 3rd party libraries. It is cross-platform but useful for Linux based distributions. It is only a pure Python wrapper, a thin layer over existing back-ends. Its strategy should work on most Linux distributions: a lot of back-ends are wrapped, if at least one exists then it works, if not then one back-end should be installed. Performance and interactivity are not important for this library.

**Features:**

- Cross-platform wrapper
- Capturing the whole desktop
- Capturing an area
- saving to `PIL` or `Pillow` image memory
- some back-ends are based on this discussion: <http://stackoverflow.com/questions/69645/take-a-screenshot-via-a-python-script-linux>
- pure Python library
- supported python versions: 2.7, 3.3, 3.4, 3.5
- **Plugin based, it has wrappers for various back-ends:**
  - `scrot`

- ImageMagick
  - PyGTK
  - PIL or Pillow (only on windows)
  - PyQt4
  - PySide
  - wxPython
  - Quartz (Mac)
  - screencapture (Mac)
- time: 0.1s - 1.0s

**Known problems:**

- ImageMagick creates `blackbox` on some systems
- PyGTK back-end does not check `$DISPLAY` -> not working with Xvfb

**Similar projects:**

- <http://sourceforge.net/projects/gtkshots/>
- <http://pypi.python.org/pypi/autopy>

## CHAPTER 2

---

### Examples

---

grab and show the whole screen:

```
#!/usr/bin/env python
#-- include('examples/showgrabfullscreen.py') --#
import pyscreenshot as ImageGrab

if __name__ == '__main__':

    # grab fullscreen
    im = ImageGrab.grab()

    # save image file
    im.save('screenshot.png')

    # show image in a window
    im.show()
#--#
```

to start the example:

```
python -m pyscreenshot.examples.showgrabfullscreen
```

grab and show the part of the screen:

```
#!/usr/bin/env python
#-- include('examples/showgrabbox.py')--#
import pyscreenshot as ImageGrab

if __name__ == '__main__':
    # part of the screen
    im = ImageGrab.grab(bbox=(10, 10, 510, 510)) # X1,Y1,X2,Y2
    im.show()
#--#
```

to start the example:

```
python -m pyscreenshot.examples.showgrabbox
```



### General

- install pip
- install PIL or Pillow
- install at least one back-end
- install the program:

```
# as root
pip install pyscreenshot
```

### Ubuntu 14.04

```
sudo apt-get install python-pip
sudo apt-get install python-pil
sudo pip install pyscreenshot
# optional back-ends
sudo apt-get install scrot imagemagick python-gtk2 python-qt4 python-wxgtk2.8 python-
↳pySide
# optional for examples
sudo pip install entrypoint2
```

### Uninstall

```
# as root
pip uninstall pyscreenshot
```



---

## Command line interface

---

### Back-end performance:

The performance can be checked **with** `pyscreenshot.check.speedtest`.

Example:

```
## sh('python -m pyscreenshot.check.speedtest --virtual-display 2>/dev/null') --#  
  
n=10  
-----  
wx                1.2  sec      ( 120 ms per call)  
pygtk             1.2  sec      ( 124 ms per call)  
pyqt              1.4  sec      ( 136 ms per call)  
scrot             0.93 sec      (  93 ms per call)  
imagemagick       0.67 sec      (  67 ms per call)  
pyside            1.3  sec      ( 133 ms per call)  
##
```

### Print versions:

```
## sh('python -m pyscreenshot.check.versions 2> /dev/null ')--#  
pyscreenshot      0.4.2  
wx                2.8.12.1  
pygtk             2.28.6  
pyqt              4.10.4  
scrot             0.8  
imagemagick       6.7.7  
pyside            1.2.1  
##
```



## CHAPTER 5

---

### command line help

---

```
## sh('python -m pyscreenshot.check.speedtest --help')--#
usage: speedtest.py [-h] [-v] [--debug]

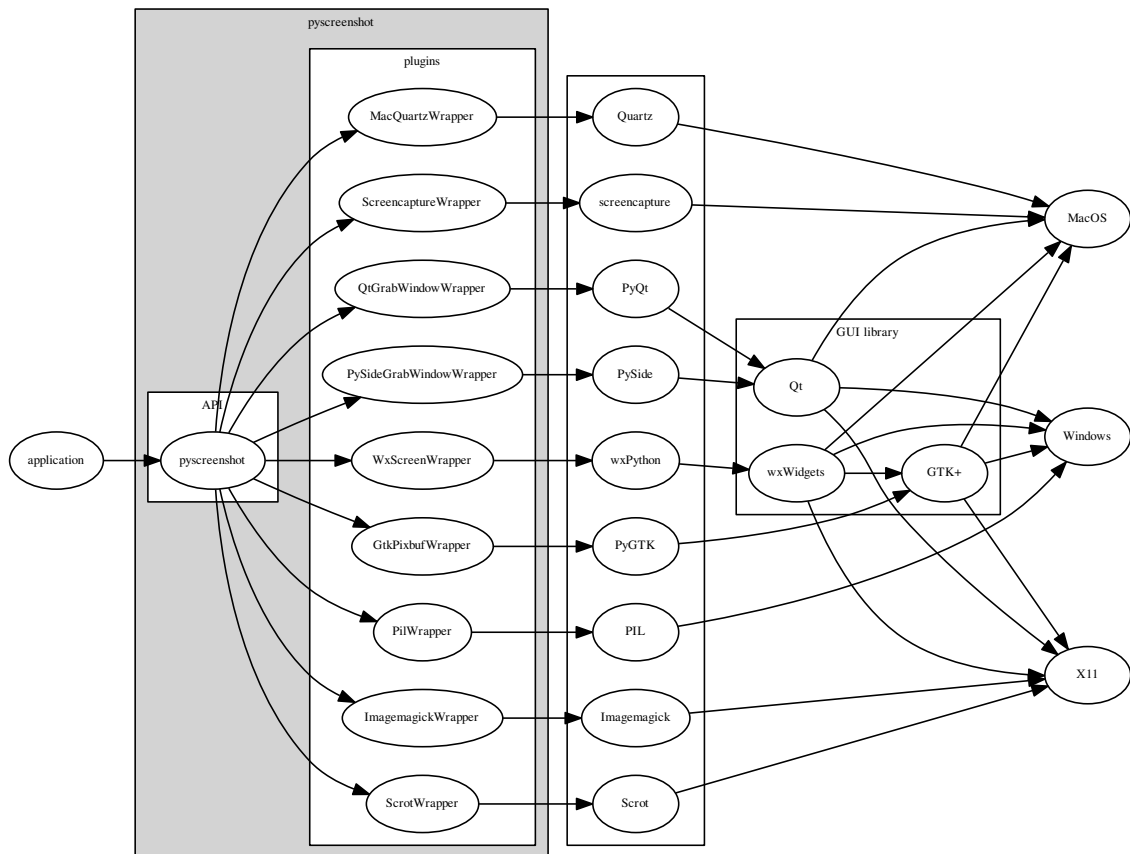
optional arguments:
  -h, --help            show this help message and exit
  -v, --virtual-display
  --debug               set logging level to DEBUG
##
```

```
## sh('python -m pyscreenshot.check.versions --help')--#
usage: versions.py [-h] [--debug]

optional arguments:
  -h, --help            show this help message and exit
  --debug               set logging level to DEBUG
##
```



Hierarchy







`pyscreenshot.backend_version (backend, childprocess=None)`

Back-end version

**Parameters**

- **backend** – back-end (examples:scrot, wx,..)
- **childprocess** – see `grab()`

**Returns** version as string

`pyscreenshot.backends ()`

Back-end names as a list

**Returns** back-ends as string list

`pyscreenshot.childprocess_default_value ()`

IDLE has problem with multiprocessing. Therefore the default is False for childprocess if the program was started inside IDLE.

`pyscreenshot.grab (bbox=None, childprocess=None, backend=None)`

Copy the contents of the screen to PIL image memory.

**Parameters**

- **bbox** – optional bounding box (x1,y1,x2,y2)
- **childprocess** – pyscreenshot can cause an error, if it is used on more different virtual displays and back-end is not in different process. Some back-ends are always different processes: scrot, imagemagick The default is False if the program was started inside IDLE, otherwise it is True.
- **backend** – back-end can be forced if set (examples:scrot, wx,..), otherwise back-end is automatic

`pyscreenshot.grab_to_file (filename, childprocess=None, backend=None)`

Copy the contents of the screen to a file. Internal function! Use `PIL.Image.save ()` for saving image to file.

**Parameters**

---

- **filename** – file for saving
- **childprocess** – see *grab()*
- **backend** – see *grab()*

**p**

pyscreenshot, 13



## B

backend\_version() (in module pyscreenshot), 13

backends() (in module pyscreenshot), 13

## C

childprocess\_default\_value() (in module pyscreenshot),  
13

## G

grab() (in module pyscreenshot), 13

grab\_to\_file() (in module pyscreenshot), 13

## P

pyscreenshot (module), 13