pysikuli

Oleg Smoliakov

Mar 11, 2024

TABLE OF CONTENTS

Pysikuli initially inspired by Sikuli Project and secondly by python automation tool pyautogui and the quick image search in python-imagesearch library. So if you already know about **SikuliX** or **pyautogui**, but want to speed up your scripts to the max, you're in the right place.

Note: This project is under development. And there may be some issues

---

Pysikuli helps to automate almost every user actions in Windows, Linux and MacOS.

To use Pysikuli, first install it using pip:

```
(.venv) $ pip install pysikuli
```

More info on *Installation* page

CHAPTER

ONE

INSTALLATION

On Windows use:

```
(.venv) $ pip install pysikuli
```

```
python -m pip install pysikuli
```

On MacOS and Linux use:

```
(.venv) $ pip3 install pysikuli
```

```
python3 -m pip install pysikuli
```

1.1 External dependencies

External dependencies mostly belong to the sounddevice module, which is used to user-friendly capture `Region` and `Location`. However, pysikuli can work normally without them.

1.1.1 Windows:

sounddevice dependence:

- Microsoft C++ build tools

1.1.2 Linux:

sounddevice dependencies:

```
$ sudo apt install libportaudio2
```

Also, pysikuli will try to install this package on its own the first time you run it

1.1.3 MacOS

You need to enable **Accessibility** and **Screen capture** permissions to terminal or python itself.

1.2 VS Code add-ons

I would also recommend installing these VS Code's add-ons:

- Paste Image, for pasting screenshots directly in the code from clipboard. You can also set up a specific folder to store your pics.
- Image preview, for preview captured photos
- luna-paint, useful tool for fast cropping captured images inside VS Code

CHAPTER

TWO

TUTORIALS

2.1 Calculator

The completed project is available here: `Download`.

This is a short script that will allow you to familiarize yourself with the basic pysikuli's features.

Firstly, import `pysikuli` and `Key` class:

```
import pysikuli as sik

from pysikuli import Key
```

Now, lets take 3 screenshot of calculator's `2`, `+` and `=` buttons and save them in `/pics` folder in project root directory.

At this step, it is more convenient and faster to use *Paste Image* for this purpose. However the tutorial can be repeated without it

Captured pics:

Fig. 1: pic_2.png

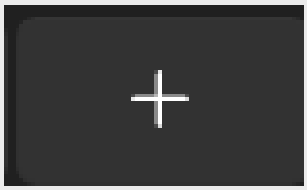

Fig. 2: pic_plus.png

Fig. 3: pic_equal.png

Then add them to the script:

```
if __name__ == "__main__":
    pic_2 = "pics/pic_2.png"
    pic_plus = "pics/pic_plus.png"
    pic_equal = "pics/pic_equal.png"
```

To run calculator let's use the most common way:

Press the `Win` key, write a `calculator` and press `Enter` to apply.

In code:

```
sik.tap(Key.win), sik.sleep(0.02)
sik.paste("calculator"), sik.sleep(0.3)
sik.tap(Key.enter)
```

Note: The `sik.sleep(...)` value can be various and depend on your PC reaction. You can test several values and choose your favorites

---

Now, we can start clicking on calculator's button

```
sik.click(pic_2, precision=0.9)
sik.click(pic_plus)
sik.click(pic_2, precision=0.9)
sik.click(pic_equal)
```

Note: if the `click()` function couldn't find the buttons or missed, you can increase the precision or take another screenshot with more details

---

Completed code below:

```
import pysikuli as sik

from pysikuli import Key

if __name__ == "__main__":
   pic_2 = "pics/pic_2.png"
   pic_plus = "pics/pic_plus.png"
   pic_equal = "pics/pic_equal.png"

   sik.tap(Key.win), sik.sleep(0.02)
   sik.paste("calculator"), sik.sleep(0.3)
   sik.tap(Key.enter)

   sik.click(pic_2, precision=0.9)
   sik.click(pic_plus)
   sik.click(pic_2, precision=0.9)
   sik.click(pic_equal)
```

Result:

CHAPTER

THREE

API

3.1 Mouse functions

pysikuli.**click**(*loc_or_pic=None*, *region=None*, *max_search_time: float = None*, *time_step: float = None*, *grayscale: bool = None*, *precision: float = None*, *button: Button = None*, *clicks: int = 1*, *interval: float = 0.0*)

Perform a mouse click operation.

Parameters

- **max_search_time** (*float*) – Maximum time for searching, in seconds.
- **time_step** (*float*) – The location or picture to click.

pysikuli.**rightClick**(*loc_or_pic=None*, *region=None*, *max_search_time: float = None*, *time_step: float = None*, *grayscale: bool = None*, *precision: float = None*, *clicks=1*, *interval=0.0*)

Performs a right mouse button click.

This is a wrapper function for click('right', x, y). The x and y parameters detail where the mouse event happens. If None, the current mouse position is used. If a float value, it is rounded down. If outside the boundaries of the screen, the event happens at edge of the screen.

Parameters

- **x** (*int, float, None, tuple, optional*) – The x position on the screen where the
- **tuple** (*click happens. None by default. If*) –
- **y.** (*this is used for x and*) –
- **str** (*If x is a*) –
- **on** (*it's considered a filename of an image to find*) –
- **of.** (*the screen with locateOnScreen() and click the center*) –
- **y** (*int, float, None, optional*) – The y position on the screen where the
- **default.** (*click happens. None by*) –
- **interval** – (float, optional): The number of seconds in between each click,
- **default** (*if the number of clicks is greater than 1. 0.0 by*) –
- **no** (*for*) –
- **clicks.** (*pause in between*) –

Returns

None

pysikuli.**mouseDown**(*button: Button = None*, *speed: float = None*, *loc_or_pic=None*, *region=None*, *max_search_time: float = None*, *time_step: float = None*, *grayscale: bool = None*, *precision: float = None*)

pysikuli.**mouseUp**(*button: Button = None*, *speed: float = None*, *loc_or_pic=None*, *region=None*, *max_search_time: float = None*, *time_step: float = None*, *grayscale: bool = None*, *precision: float = None*)

pysikuli.**mouseMove**(*destination_loc*)

pysikuli.**mouseSmoothMove**(*destination_loc*, *speed: float = None*)

pysikuli.**mouseMoveRelative**(*xOffset*, *yOffset*, *speed: float = None*)

pysikuli.**mousePosition**()

pysikuli.**scroll**(*dx=0*, *dy=0*)

Sends scroll events.

Parameters

- **(int** (*dy*) –
- **optional)** (*The vertical scroll. The units of scrolling is*) –
- **undefined.** –
- **(int** –
- **optional)** –
- **undefined.** –

pysikuli.**hscroll**(*duration=0.1*, *speed=1*)

pysikuli.**vscroll**(*duration=0.1*, *speed=1*)

pysikuli.**dragDrop**(*destination_loc: list*, *start_location=None*, *speed: float = None*, *button: Button = None*)

summary

Parameters

- **start_location** (*list*) – x, y start location example: [1200, 700]
- **end_location** (*list*) – x, y end location example: [1400, 700]
- **butt** (*str, optional*) – Which button will be holded. Defaults to “left”.

3.2 Keyboard functions

pysikuli.**tap**(*key*, *presses=1*, *interval=0.0*, *time_step: float = None*)

taps a single key on keyboard

Parameters

- **key** (*_type_*) – _description_
- **presses** (*int, optional*) – number of times the entered key is pressed. Defaults to 1.
- **interval** (*float, optional*) – time interval after each tap. Defaults to 0.0.

- **time_step** (*float, optional*) – time interval after each key press and each key realese. Defaults to None.

pysikuli.**keyUp**(*key*)

pysikuli.**keyDown**(*key*)

pysikuli.**hotkey**(**keys*, *interval=0.0*)

pysikuli.**write**(*message*, *time_step: float = None*)

pysikuli.**pressedKeys**()

pysikuli.**paste**(*text: str*)

fast paste text into selected window. Equivalent copyToClip + (ctrl or cmd + v)

Parameters

text (*str*) – text, which one will be entered into active window

pysikuli.**copyToClip**(*text: str*)

pysikuli.**pasteFromClip**()

3.3 Pop-ups

pysikuli.**popupAlert**(*text=''*, *title=''*, *root: tuple[int, int] = None*, *timeout: float = None*)

Displays a simple message box with text and a single OK button.

Parameters

- **(str** (*title*) –
- **optional)** (*time in seconds after which message box will be closed.*) –
- **(str** –
- **optional)** –
- **(tuple[int** (*root*) –
- **int]** (left top corner location. Defaults gets from *config.ROOT_WINDOW_POSITION*.) –
- **optional)** –
- **(float** (*timeout*) –
- **optional)** –

Returns

str | None

Return type

the text of the button clicked on, or "Timeout" if time has elapsed.

pysikuli.**popupPassword**(*text=''*, *title=''*, *default=''*, *mask='*'*, *root: tuple[int, int] = None*, *timeout: float = None*)

Displays a message box with text input, and OK & Cancel buttons. Typed characters appear as * by default.

Parameters

- **(str** (*mask*) –

- **optional)** (*time in seconds after which message box will be closed.*) –
- **(str** –
- **optional)** –
- **(str** –
- **optional)** –
- **(str** –
- **optional)** –
- **(tuple[int** (*root*) –
- **int]** (left top corner location. Defaults gets from *config.ROOT_WINDOW_POSITION*.) –
- **optional)** –
- **(float** (*timeout*) –
- **optional)** –

Returns
str | None

Return type
the entered text, None if Cancel was clicked, or "Timeout" if time has elapsed.

pysikuli.**popupConfirm**(*text=''*, *title=''*, *buttons=('OK', 'Cancel')*, *root: tuple[int, int] = None*, *timeout: float = None*)

Displays a message box with OK and Cancel buttons. Number and text of buttons can be customized.

Parameters

- **(str** (*title*) –
- **optional)** (*time in seconds after which message box will be closed.*) –
- **(str** –
- **optional)** –
- **(tuple** (*buttons*) –
- **optional)** –
- **(tuple[int** (*root*) –
- **int]** (left top corner location. Defaults gets from *config.ROOT_WINDOW_POSITION*.) –
- **optional)** –
- **(float** (*timeout*) –
- **optional)** –

Returns
str

Return type
the text of the button clicked on, or "Timeout" if time has elapsed.

pysikuli.**popupPrompt**(*text=''*, *title=''*, *default=''*, *root: tuple[int, int] = None*, *timeout: float = None*)

Displays a message box with text input, and OK & Cancel buttons.

Parameters

- **(str** (*default*) –
- **optional)** (*time in seconds after which message box will be closed.*) –
- **(str** –
- **optional)** –
- **(str** –
- **optional)** –
- **(tuple[int** (*root*) –
- **int]** (left top corner location. Defaults gets from *config.ROOT_WINDOW_POSITION*.) –
- **optional)** –
- **(float** (*timeout*) –
- **optional)** –

Returns
str | None

Return type
the entered text, None if Cancel was clicked, or "Timeout" if time has elapsed.

3.4 File management functions

pysikuli.**saveNumpyImg**(*image: ndarray*, *image_name: str = None*, *path: str = None*)

Save a numpy array into png image in root directory

Parameters

- **image** (*np.ndarray*) – the variable with pic, that you want to save
- **image_name** (*str*) – file's name
- **path** (*str*) – path where the pic will be saved

pysikuli.**saveScreenshot**(*region=None*)

pysikuli.**deleteFile**(*file_path: PathLike*)

func first tries to move the file to the bin, otherwise uses os.remove for permanently deletion

Parameters
(PathLike) (*file_path*) –

3.5 Utils

pysikuli.**getLocation**(*interval=0.5*)

pysikuli.**getRegion**(*interval=0.5*)

getRigion helps to determine the Region coordinates.

Simply by hovering your mouse over points on the screen. The region requires 2 points, left-top and right-bottom. If you hold the mouse on the same spot for 3 *intervals* the point will be captured and capture sound will be played by default

Parameters

- **interval** (*float*) – time in seconds, which uses for delay
- **0.5.** (*between each mouse posuition capture. Defaults to*) –

Returns
"Region(1, 2, 3, 4)" also print this region in console

Return type
updated clipboard with prepared region like

pysikuli.**cleanupPics**(*pics_folder_path='pics'*, *pics_folder_prefix_in_code='pics'*, *root_path='/home/docs/checkouts/readthedocs.org/user_builds/pysikuli/checkouts/latest/docs/source'*, *formats: list = ['bmp', 'jfif', 'jpeg', 'jpe', 'jpg', 'png', 'tiff', 'tif']*)

3.6 Screenshot functions

pysikuli.**exist**(*image*, *region=None*, *grayscale: bool = None*, *precision: float = None*, *rgb_diff: float = None*, *_tuple_region: tuple | list = None*)

Searchs for an image within an area or on the screen

input :

image : path to the image file (see opencv imread for supported types) region : (x1, y1, x2, y2) precision : the higher, the lesser tolerant and fewer false positives are found default is 0.8 numpy_region : a PIL or numpy image, usefull if you intend to search the same unchanging region for several elements, must be stored in *RGB format*

returns : the top left corner coordinates of the element if found as an array [x,y] or [-1,-1] if not

if exist find several patterns with same score, will return the most right and the most bottom match

pysikuli.**existAny**(*image_list*, *region=None*, *grayscale: bool = None*, *precision: float = None*, *rgb_diff: float = None*)

pysikuli.**existCount**(*image*, *region=None*, *precision: float = None*, *grayscale: bool = None*, *rgb_diff: float = None*, *tuple_region: tuple[int, int, int, int] = None*)

Searches for an image on the screen and counts the number of occurrences.

input : image : path to the target image file (see opencv imread for supported types) precision : the higher, the lesser tolerant and fewer false positives are found default is 0.9

returns : the number of times a given image appears on the screen. optionally an output image with all the occurances boxed with a red outline.

pysikuli.**existFromFolder**(*path*, *region=None*, *grayscale=None*, *precision=None*)

Get all pictures from the provided folder and search them on screen.

input : path : path of the folder with the images to be searched on screen like pics/ precision : the higher, the lesser tolerant and fewer false positives are found default is 0.8

returns : A dictionary where the key is the path to image file and the value is the position where it was found.

pysikuli.**find**(*image*, *region=None*, *max_search_time: float = None*, *time_step: float = None*, *grayscale: bool = None*, *precision: float = None*, *rgb_diff: float = None*)

find an image pattern on the screen or specific region

Parameters

- **image** (*_type_*) – _description_
- **region** (*_type_, optional*) – _description_. Defaults to None.
- **max_search_time** (*float, optional*) – _description_. Defaults to None.
- **time_step** (*float, optional*) – _description_. Defaults to None.
- **grayscale** (*bool, optional*) – _description_. Defaults to None.
- **precision** (*float, optional*) – _description_. Defaults to None.
- **rgb_diff** (*float, optional*) – _description_. Defaults to None.

Returns
description

Return type
type

pysikuli.**grab**(*region: tuple[int, int, int, int]*)

pysikuli.**getPixelRGB**(*rel_location: tuple[int, int] | list[int, int]*, *np_image: ndarray = None*) → tuple[int, int, int]

pysikuli.**wait**(*image: str*, *region=None*, *max_search_time: float = None*, *time_step: float = None*, *grayscale: bool = None*, *precision: float = None*, *rgb_diff: float = None*)

pysikuli.**waitWhileExist**(*image*, *region=None*, *max_search_time: float = None*, *time_step: float = None*, *grayscale: bool = None*, *precision: float = None*, *rgb_diff: float = None*)

3.7 Window control functions

pysikuli.**activateWindow**(*window_title: str*)

Activate entered window and make it the foreground (focused) window

Parameters

- **(str)** (*window_title*) –
- **titles** (*To get all available*) –
- **pysikuli.getAllWindowsTitle()** (*use*) –

Returns
bool

Return type
"True" if window activated

pysikuli.**activateWindowAt**(*location: tuple[int, int]*)

Activate the top window under entered location(x, y)

Parameters

- **(tuple[int** (*location*) –
- **int])** (*tuple with int coordinates (x, y)*) –

Returns
bool

Return type
"True" if window is activated

pysikuli.**activateWindowUnderMouse**()

Activate the top window under mouse position

Returns
bool

Return type
"True" if window is activated

pysikuli.**getWindowRegion**(*window_title: str*)

grab the window region window by it's name

Parameters

- **(str)** (*window_title*) –
- **titles** (*To get all available*) –
- **pysikuli.getAllWindowsTitle()** (*use*) –

Returns
Region

Return type
an object of class *Region*, if the window is within the screen boundaries and exists

pysikuli.**getWindowWithTitle**(*window_title: str*)

Get the window objects whose title match the given string

Parameters

- **(str)** (*window_title*) –
- **titles** (*To get all available*) –
- **pysikuli.getAllWindowsTitle()** (*use*) –

Return type
Window object | None

pysikuli.**getWindowUnderMouse**()

Get the Window object at the top of the stack under mouse position

Return type
Window object | None

pysikuli.**getAllWindowsTitle**()

Get the list of titles of all visible windows

Returns
list[str]

Return type
list of titles as strings

pysikuli.**minimizeWindow**(*window_title: str*)

Minimizes entered window

Parameters

- **(str)** (*window_title*) –

- **titles** (*To get all available*) –
- **pysikuli.getAllWindowsTitle()** (*use*) –

Returns
bool

Return type
"True" if window minimized

pysikuli.**maximizeWindow**(*window_title: str*)

Maximizes entered window

Parameters

- **(str)** (*window_title*) –
- **titles** (*To get all available*) –
- **pysikuli.getAllWindowsTitle()** (*use*) –

Returns
bool

Return type
"True" if window is maximized

pysikuli.**closeWindow**(*window_title: str*)

Closes this window. This is identical to clicking the X button on the window.

This may trigger "Are you sure you want to quit?" dialogs or other actions that prevent the window from actually closing.

Parameters

- **(str)** (*window_title*) –
- **titles** (*To get all available*) –
- **pysikuli.getAllWindowsTitle()** (*use*) –

Returns
bool

Return type
"True" if window is closed

pysikuli.**windowExist**(*window_title: str*)

check if entered window exists

Parameters

- **window_title** (*str*) – full window name, e.g. "Mozilla Firefox".
- **titles** (*To get all available*) –
- **pysikuli.getAllWindowsTitle()** (*use*) –

Returns
full window title

Return type
str | None

3.8 Classes

class pysikuli.**Region**(*x1: int*, *y1: int*, *x2: int*, *y2: int*)

Represent specific part of the screen

e.g. Region(0, 0, 100, 100) define up left part of the screen with heigh and width equal to 100 pixels.

Parameters
Region (*x1, y1, x2, y2) (int*) – class initialization

Returns
description

Return type
type

Uses the same functions that are declared separately, but sets the region and time_step parameters by default.

reg tuple(int, int, int, int)

x1 (int)

y1 (int)

x2 (int)

y2 (int)

time_step (float)

Common parameter for all search function in Class `Region` Sets the time between each iteration of the search. Default values is `config.TIME_STEP`

click(*loc_or_pic*, *max_search_time: float = None*, *grayscale: bool = None*, *precision: float = None*, *clicks=1*, *interval=0.0*, *button=Button.left*)

rightClick(*loc_or_pic*, *max_search_time: float = None*, *grayscale: bool = None*, *precision: float = None*, *clicks=1*, *interval=0.0*)

has(*image: str*, *grayscale: bool = None*, *precision: float = None*, *rgb_diff: float = None*)

find(*image*, *max_search_time: float = None*, *grayscale: bool = None*, *precision: float = None*, *rgb_diff: float = None*)

find an image pattern

Parameters

- **image** (*_type_*) – _description_
- **max_search_time** (*float, optional*) – _description_. Defaults to None.
- **grayscale** (*bool, optional*) – _description_. Defaults to None.
- **precision** (*float, optional*) – _description_. Defaults to None.
- **rgb_diff** (*float, optional*) – _description_. Defaults to None.

Returns
description

Return type
type

existAny(*image*, *grayscale: bool = None*, *precision: float = None*)

wait(*image: str*, *max_search_time: float = None*, *grayscale: bool = None*, *precision: float = None*, *rgb_diff: float = None*)

waitWhileExist(*image: str*, *max_search_time: float = None*, *grayscale: bool = None*, *precision: float = None*, *rgb_diff: float = None*)

class pysikuli.**Button**

- Button.left
- Button.middle
- Button.right

class pysikuli.**Key**

The class **Key** attributes are platform-dependent, but they are the same in Linux and Windows.

Common attributes:

- Key.alt
- Key.alt_r
- Key.alt_gr
- Key.shift
- Key.shift_r
- Key.ctrl, Key.ctrl_r
- Key.caps_lock
- Key.left
- Key.right
- Key.up
- Key.down
- Key.page_down
- Key.page_up
- Key.home
- Key.end
- Key.esc
- Key.space
- Key.tab
- Key.f1
- Key.f2
- Key.f3
- Key.f4
- Key.f5
- Key.f6

- Key.f7
- Key.f8
- Key.f9
- Key.f10
- Key.f11
- Key.f12
- Key.f13
- Key.f14
- Key.f15
- Key.f16
- Key.f17
- Key.f18
- Key.f19
- Key.f20
- Key.media_play_pause
- Key.media_volume_mute
- Key.media_volume_down
- Key.media_volume_up
- Key.media_previous
- Key.media_next

Windows/Linux specific keys:

- Key.win
- Key.enter
- Key.delete
- Key.backspace
- Key.insert
- Key.menu
- Key.num_lock
- Key.pause
- Key.print_screen
- Key.scroll_lock

MacOS specific keys:

- Key.cmd
- Key.cmd_r
- Key.delete
- Key.option

- Key.option_r
- Key.return_r

pysikuli.**config**

The config variable contain Config class with all global pysikuli settings. The full setting list is provided below.

General parameters:

Parameter	Type	Default value	Description
OSX, WIN, UNIX	bool	False	Sets to True value if you use this **OS**
MIN_SLEEP_T	float	0.02	If the time is less than this value, the sleep time isn't accurate enough
PAUSE_BETW	float	0	The number of seconds of pause after EVERY public function call. Useful for debugging
FAILSAFE	bool	True	If the mouse is within FAILSAFE_REGIONS or FAILSAFE_HOTKEY is pressed and FAILSAFE is True, a FailSafeException will be raised.
FAIL-SAFE_REGION	list	[(0, 0, 0, 0)]	You can add another regions using the .append() method.
FAIL-SAFE_HOTKE	list	MacOS: [Key.alt, Key.shift, "c"] Windows/Linux: [Key.ctrl, Key.alt, "z"]	Currently, only one hotkey can be configured
WIN-DOW_WAITIN	bool	True	If active, pysikuli will wait for the window management functions to complete their execution
RE-FRESH_RATE	int	set automatically	Left in case of incorrect detection. Uses in scroll functions and DEBUG SETTINGS
MONI-TOR_REGION	tuple (int, int, int, int)	the screen region of your main monitor	Left in case of incorrect detection. Uses as default Region in search functions and for validating user Region s
MONI-TOR_RESOLU	tuple(int, int)	the resolution of your main monitor	Uses in popup functions
DEBUG	bool	False	If True, DEBUG_SETTINGS will be applied

DEBUG_SETTINGS list:

Parameter	Default value
PAUSE_BETWEEN_ACTION	0.5
TIME_STEP	time per frame
MOUSE_SPEED	1

Search functions parameters:

Parameter	Type	Default value	Description
TIME_STEP	float	`0`	TIME_STEP is the time in seconds between each step in function loop, such as `find`, `wait`, `tap`, `write`, `click` and etc.
MAX_SEARCI	float	`2.0`	The time limit for search functions. If it is exceeded, `None` is returned
GRAYSCALE	bool	`True`	This parameter increases speed by about 30%, but degrades unambiguous image recognition
COMPRES-SION_RATIO	float	`2`	This option almost doubles the speed by using compressed images in image recognition, but degrades unambiguous image recognition
MIN_PRECISI	float	`0.8`	The value after which all search functions return positive

Mouse functions parameters:

Parameter	Type	Default value	Description
MOUSE_PRIMARY_B	`Butto`	`Button.left`	The primary button used for mouse actions by default
MOUSE_SECONDARY	`Butto`	`Button.right`	The secondary button used for mouse actions by default
MOUSE_SPEED	float	`2`	The speed of the mouse movement. This is abstract value and if equal to 1 it 1000px per second

Utils functions parameters:

Parameter	Type	Default value	Description
SOUND_ON	bool	`True`	Determines if sound is enabled
SOUND_CAPTURE_PATH	str	`"tools_data/_capture.mp3"`	Path to the capture sound file
SOUND_FINISH_PATH	str	`"tools_data/_finish.mp3"`	Path to the finish sound file

Popup functions parameters:

Parameter	Type	Default value	Description
OK_TEXT	str	`"OK"`	Text for the “OK” button
CANCEL_TEXT	str	`"Cancel"`	Text for the “Cancel” button
ROOT_WINDOW_POSIT	tuple(int, int)	center of the screen	Position of the root window (the top left corner of the window)
PROPOR-TIONAL_FONT_SIZE	int	`10`	Font size for proportional text
MONOSPACE_FONT_SI	int	`9`	Font size for monospace text
TEXT_ENTRY_FONT_SI	int	`12`	Font size for text entry fields

INDEX

V

W

X

Y