

---

# **ahk Documentation**

*Release 0.8.1*

**Spencer Phillip Young**

**Jan 07, 2021**



# CONTENTS:

<b>1</b>	<b>Quickstart</b>	<b>3</b>
1.1	Installing AHK . . . . .	3
<b>2</b>	<b>API</b>	<b>5</b>
2.1	Autohotkey . . . . .	5
2.2	Directives . . . . .	6
2.3	Keyboard . . . . .	7
2.4	Keys . . . . .	10
2.5	Mouse . . . . .	15
2.6	Registry . . . . .	18
2.7	Screen . . . . .	18
2.8	The Script Engine . . . . .	20
2.9	Sound . . . . .	22
2.10	Utils . . . . .	24
2.11	Window . . . . .	24
<b>3</b>	<b>ahk</b>	<b>33</b>
<b>4</b>	<b>Installation</b>	<b>35</b>
<b>5</b>	<b>Usage</b>	<b>37</b>
<b>6</b>	<b>Examples</b>	<b>39</b>
6.1	Mouse . . . . .	39
6.2	Keyboard . . . . .	39
6.3	Windows . . . . .	40
6.3.1	Getting windows . . . . .	40
6.3.2	Working with windows . . . . .	40
6.4	Screen . . . . .	41
6.5	Sound . . . . .	41
6.6	GUI . . . . .	42
6.7	non-blocking modes . . . . .	42
6.8	Add directives . . . . .	43
6.9	Run arbitrary AutoHotkey scripts . . . . .	43
6.9.1	Communicating data from ahk to Python . . . . .	43
6.10	Preview features . . . . .	44
6.11	Async API . . . . .	44
6.11.1	Hotkeys . . . . .	45
6.11.2	ActionChain . . . . .	45
6.11.3	find_window/find_windows methods . . . . .	46
6.12	Errors and Debugging . . . . .	46

6.13 Non-Python dependencies . . . . .	46
<b>7 Contributing</b>	<b>47</b>
<b>8 Similar projects</b>	<b>49</b>
<b>9 Indices and tables</b>	<b>51</b>
<b>Python Module Index</b>	<b>53</b>
<b>Index</b>	<b>55</b>

GitHub



## QUICKSTART

This document assumes you have Python 3.6 or newer installed

### 1.1 Installing AHK

AHK requires the AutoHotkey software in addition to the Python package

1. Download and install AutoHotkey (1.1.x). It can be downloaded from the [autohotkey website](#).
2. Install the Python ahk package

```
py -m pip install ahk
```

3. Write your first script:

```
from ahk import AHK
ahk = AHK()
ahk.run_script('Run Notepad')
notepad_window = ahk.find_window_by_title(b'Untitled - Notepad')
notepad_window.send('Hello World')
```

Run the script!

If you get an *ExecutableNotFoundError* it's because AutoHotkey was installed to a location that is not on PATH or the default location (C:\Program Files\AutoHotkey\AutoHotkey.exe). You can either place the executable on PATH, in the default location, or specify the location manually in code:

```
ahk = AHK(executable_path='C:\\Path\\To\\AutoHotkey.exe')
```





This part of the documentation is intended for developers looking to contribute to this project or discover more about the programming interface. A lot of this is auto-generated documentation from the docstrings.

## 2.1 Autohotkey

**class** `ahk.autohotkey.AHK` (\*args, \*\*kwargs)  
Inherits its methods from the following classes:

```
ahk.window.WindowMixin  
ahk.mouse.MouseMixin  
ahk.keyboard.KeyboardMixin  
ahk.screen.ScreenMixin  
ahk.sound.SoundMixin  
ahk.registery.RegisteryMixin  
ahk.gui.GUIMixin
```

**class** `ahk.autohotkey.ActionChain` (\*args, \*\*kwargs)  
Reusable action chain to execute various actions in order

`__init__` (\*args, \*\*kwargs)

**Parameters**

- `mouse_speed` – default mouse speed
- `mode` –
- `kwargs` –

`perform` ()

`run_script` (\*args, \*\*kwargs)  
override `run_script` to defer to queue

`sleep` (n)

**Parameters** n – how long (in seconds) to sleep

**Returns**

**class** `ahk.autohotkey.AsyncAHK` (mouse\_speed=2, mode=None, \*\*kwargs)

## 2.2 Directives

Contains directive classes

**class** ahk.directives.**AllowSameLineComments** (*\*\*kwargs*)

**class** ahk.directives.**ClipboardTimeout** (*milliseconds=0, \*\*kwargs*)

**\_\_init\_\_** (*milliseconds=0, \*\*kwargs*)

Initialize self. See help(type(self)) for accurate signature.

**class** ahk.directives.**Directive** (*\*\*kwargs*)

Simple directive class They are designed to be hashable and comparable with string equivalent of AHK directive.

Directives that don't require arguments do not need to be instantiated.

**\_\_init\_\_** (*\*\*kwargs*)

Initialize self. See help(type(self)) for accurate signature.

**class** ahk.directives.**DirectiveMeta**

Overrides **\_\_str\_\_** so directives with no arguments can be used without instantiation Overrides **\_\_hash\_\_** to make objects 'unique' based upon a hash of the str representation

**class** ahk.directives.**ErrorStdOut** (*\*\*kwargs*)

**class** ahk.directives.**HotKeyInterval** (*milliseconds=0, \*\*kwargs*)

**class** ahk.directives.**HotKeyModifierTimeout** (*milliseconds=0, \*\*kwargs*)

**class** ahk.directives.**Include** (*include\_name, \*\*kwargs*)

**\_\_init\_\_** (*include\_name, \*\*kwargs*)

Initialize self. See help(type(self)) for accurate signature.

**class** ahk.directives.**IncludeAgain** (*include\_name, \*\*kwargs*)

**class** ahk.directives.**InputLevel** (*level, \*\*kwargs*)

**\_\_init\_\_** (*level, \*\*kwargs*)

Initialize self. See help(type(self)) for accurate signature.

**class** ahk.directives.**InstallKeybdHook** (*\*\*kwargs*)

**class** ahk.directives.**InstallMouseHook** (*\*\*kwargs*)

**class** ahk.directives.**KeyHistory** (*limit=40, \*\*kwargs*)

**\_\_init\_\_** (*limit=40, \*\*kwargs*)

Initialize self. See help(type(self)) for accurate signature.

**class** ahk.directives.**MaxHotkeysPerInterval** (*value, \*\*kwargs*)

**\_\_init\_\_** (*value, \*\*kwargs*)

Initialize self. See help(type(self)) for accurate signature.

**class** ahk.directives.**MaxMem** (*megabytes, \*\*kwargs*)

**\_\_init\_\_** (*megabytes, \*\*kwargs*)

Initialize self. See help(type(self)) for accurate signature.

```

class ahk.directives.MaxThreads

    __init__()
        Initialize self. See help(type(self)) for accurate signature.

class ahk.directives.MaxThreadsBuffer

    __init__()
        Initialize self. See help(type(self)) for accurate signature.

class ahk.directives.MaxThreadsPerHotkey

    __init__()
        Initialize self. See help(type(self)) for accurate signature.

class ahk.directives.MenuMaskKey

    __init__()
        Initialize self. See help(type(self)) for accurate signature.

class ahk.directives.NoEnv (**kwargs)
class ahk.directives.NoTrayIcon (**kwargs)
class ahk.directives.Persistent (**kwargs)
class ahk.directives.SingleInstance (**kwargs)
class ahk.directives.UseHook (**kwargs)
class ahk.directives.Warn (**kwargs)
class ahk.directives.WinActivateForce (**kwargs)

```

## 2.3 Keyboard

```

class ahk.keyboard.AsyncKeyboardMixin (executable_path="", directives=None, **kwargs)

    async key_press (key, release=True, blocking=True)
        Press and (optionally) release a single key

        Parameters
            • key –
            • release –

        Returns

    async key_state (key_name, mode=None)

        Return type bool

    async key_wait (key_name, timeout=None, logical_state=False, released=False)
        Wait for key to be pressed or released (default is pressed; specify released=True to wait for key
        release).

        https://autohotkey.com/docs/commands/KeyWait.htm

```

**Parameters**

- **key\_name** – The name of the key
- **timeout** (*Optional[int]*) – how long (in seconds) to wait for the key. If not specified, waits indefinitely
- **logical\_state** – Check the logical state of the key, which is the state that the OS and the active window believe the key to be in (not necessarily the same as the physical state). This option is ignored for joystick buttons.
- **released** – Set to True to wait for the key to be released rather than pressed

**Return type** None

**Returns** None

**Raises** **TimeoutError** – if the key was not pressed (or released, if specified) within timeout

**class** `ahk.keyboard.Hotkey` (*engine, hotkey, script*)

`__init__` (*engine, hotkey, script*)

**Parameters**

- **engine** (*ScriptEngine*) – an AHK instance
- **hotkey** (*str*) – The hotkey to use (AutoHotkey syntax)
- **script** (*str*) – The script to execute when the hotkey is activated (AutoHotkey code as a string)

**property** `running`

**start** ()

Starts an AutoHotkey process with the hotkey script

**stop** ()

Stops the process if it is running

**class** `ahk.keyboard.KeyboardMixin` (*executable\_path="", directives=None, \*\*kwargs*)

**hotkey** (*\*args, \*\*kwargs*)

Convenience function for creating Hotkey instance using current engine.

**Parameters**

- **hotkey** – The hotkey to use (AutoHotkey syntax)
- **script** – The script to execute when the hotkey is activated (AutoHotkey code as a string)

**Returns** an *Hotkey* instance

**key\_down** (*key, blocking=True*)

Press down a key (without releasing it)

**Parameters**

- **key** –
- **blocking** –

**Returns**

**key\_press** (*key*, *release=True*, *blocking=True*)

Press and (optionally) release a single key

**Parameters**

- **key** –
- **release** –

**Returns**

**key\_release** (*key*, *blocking=True*)

Release a key that is currently in pressed down state

**Parameters** **key** –

**Returns**

**key\_state** (*key\_name*, *mode=None*)

**Return type** `bool`

**key\_up** (*key*, *blocking=True*)

Alias for `:meth:`~KeyboardMixin.key_release``

**key\_wait** (*key\_name*, *timeout=None*, *logical\_state=False*, *released=False*)

Wait for key to be pressed or released (default is pressed; specify `released=True` to wait for key release).

<https://autohotkey.com/docs/commands/KeyWait.htm>

**Parameters**

- **key\_name** – The name of the key
- **timeout** (`Optional[int]`) – how long (in seconds) to wait for the key. If not specified, waits indefinitely
- **logical\_state** – Check the logical state of the key, which is the state that the OS and the active window believe the key to be in (not necessarily the same as the physical state). This option is ignored for joystick buttons.
- **released** – Set to `True` to wait for the key to be released rather than pressed

**Return type** `None`

**Returns** `None`

**Raises** **TimeoutError** – if the key was not pressed (or released, if specified) within timeout

**send** (*s*, *raw=False*, *delay=None*, *blocking=True*)

<https://autohotkey.com/docs/commands/Send.htm>

**Parameters**

- **s** –
- **raw** –
- **delay** –
- **blocking** – if `True`, waits until script finishes, else returns immediately.

**Returns**

**send\_event** (*s*, *delay=None*)

<https://autohotkey.com/docs/commands/Send.htm>

**Parameters**

- **s** –
- **delay** –

**Returns**

**send\_input** (*s*, *blocking=True*)

**send\_play** (*s*)

<https://autohotkey.com/docs/commands/Send.htm>

**Parameters s** –**Returns**

**send\_raw** (*s*, *delay=None*)

<https://autohotkey.com/docs/commands/Send.htm>

**Parameters**

- **s** –
- **delay** –

**Returns**

**set\_capslock\_state** (*state=None*)

<https://www.autohotkey.com/docs/commands/SetNumScrollCapsLockState.htm>

Sets capslock state

**Parameters state** (*str*) – the desired state (“on” or “off”). Can also be True/False. If omitted, toggles capslock state.

**Returns**

**type** (*s*, *blocking=True*)

Sends keystrokes using `send_input`, also escaping the string for use in AHK.

**Parameters**

- **s** – the string to type
- **blocking** – if `True`, waits until script finishes, else returns immediately.

## 2.4 Keys

The `ahk.keys` module contains some useful constants and classes for working with keys.

**class** `ahk.keys.KEYS`

KEYS constants REF: <https://autohotkey.com/docs/KeyList.htm>

`ALT = KeyModifier(key_name='Alt')`

`Alt = KeyModifier(key_name='Alt')`

`BACKSPACE = Key(key_name='Backspace')`

`Backspace = Key(key_name='Backspace')`

`CAPS_LOCK = Key(key_name='CapsLock')`

`CONTROL = KeyModifier(key_name='Control')`

```
CTRL = KeyModifier(key_name='Control')
CapsLock = Key(key_name='CapsLock')
Control = KeyModifier(key_name='Control')
Ctrl = KeyModifier(key_name='Control')
DEL = Key(key_name='Delete')
DELETE = Key(key_name='Delete')
DOWN = Key(key_name='Down')
Del = Key(key_name='Delete')
Delete = Key(key_name='Delete')
Down = Key(key_name='Down')
ENTER = Key(key_name='Enter')
ESCAPE = Key(key_name='Escape')
Enter = Key(key_name='Enter')
F1 = Key(key_name='F1')
F10 = Key(key_name='F10')
F11 = Key(key_name='F11')
F12 = Key(key_name='F12')
F13 = Key(key_name='F13')
F14 = Key(key_name='F14')
F15 = Key(key_name='F15')
F16 = Key(key_name='F16')
F17 = Key(key_name='F17')
F18 = Key(key_name='F18')
F19 = Key(key_name='F19')
F2 = Key(key_name='F2')
F20 = Key(key_name='F20')
F21 = Key(key_name='F21')
F22 = Key(key_name='F22')
F23 = Key(key_name='F23')
F24 = Key(key_name='F24')
F3 = Key(key_name='F3')
F4 = Key(key_name='F4')
F5 = Key(key_name='F5')
F6 = Key(key_name='F6')
F7 = Key(key_name='F7')
F8 = Key(key_name='F8')
```

```
F9 = Key(key_name='F9')
JOY1 = Key(key_name='Joy1')
JOY10 = Key(key_name='Joy10')
JOY11 = Key(key_name='Joy11')
JOY12 = Key(key_name='Joy12')
JOY13 = Key(key_name='Joy13')
JOY14 = Key(key_name='Joy14')
JOY15 = Key(key_name='Joy15')
JOY16 = Key(key_name='Joy16')
JOY17 = Key(key_name='Joy17')
JOY18 = Key(key_name='Joy18')
JOY19 = Key(key_name='Joy19')
JOY2 = Key(key_name='Joy2')
JOY20 = Key(key_name='Joy20')
JOY21 = Key(key_name='Joy21')
JOY22 = Key(key_name='Joy22')
JOY23 = Key(key_name='Joy23')
JOY24 = Key(key_name='Joy24')
JOY25 = Key(key_name='Joy25')
JOY26 = Key(key_name='Joy26')
JOY27 = Key(key_name='Joy27')
JOY28 = Key(key_name='Joy28')
JOY29 = Key(key_name='Joy29')
JOY3 = Key(key_name='Joy3')
JOY30 = Key(key_name='Joy30')
JOY31 = Key(key_name='Joy31')
JOY32 = Key(key_name='Joy32')
JOY4 = Key(key_name='Joy4')
JOY5 = Key(key_name='Joy5')
JOY6 = Key(key_name='Joy6')
JOY7 = Key(key_name='Joy7')
JOY8 = Key(key_name='Joy8')
JOY9 = Key(key_name='Joy9')
Joy1 = Key(key_name='Joy1')
Joy10 = Key(key_name='Joy10')
Joy11 = Key(key_name='Joy11')
```



```
Joy12 = Key(key_name='Joy12')
Joy13 = Key(key_name='Joy13')
Joy14 = Key(key_name='Joy14')
Joy15 = Key(key_name='Joy15')
Joy16 = Key(key_name='Joy16')
Joy17 = Key(key_name='Joy17')
Joy18 = Key(key_name='Joy18')
Joy19 = Key(key_name='Joy19')
Joy2 = Key(key_name='Joy2')
Joy20 = Key(key_name='Joy20')
Joy21 = Key(key_name='Joy21')
Joy22 = Key(key_name='Joy22')
Joy23 = Key(key_name='Joy23')
Joy24 = Key(key_name='Joy24')
Joy25 = Key(key_name='Joy25')
Joy26 = Key(key_name='Joy26')
Joy27 = Key(key_name='Joy27')
Joy28 = Key(key_name='Joy28')
Joy29 = Key(key_name='Joy29')
Joy3 = Key(key_name='Joy3')
Joy30 = Key(key_name='Joy30')
Joy31 = Key(key_name='Joy31')
Joy32 = Key(key_name='Joy32')
Joy4 = Key(key_name='Joy4')
Joy5 = Key(key_name='Joy5')
Joy6 = Key(key_name='Joy6')
Joy7 = Key(key_name='Joy7')
Joy8 = Key(key_name='Joy8')
Joy9 = Key(key_name='Joy9')
LAlt = KeyModifier(key_name='LAlt')
LControl = KeyModifier(key_name='LControl')
LCtrl = KeyModifier(key_name='LControl')
LEFT = Key(key_name='Left')
LEFT_ALT = KeyModifier(key_name='LAlt')
LEFT_CONTROL = KeyModifier(key_name='LControl')
LEFT_SHIFT = KeyModifier(key_name='LShift')
```

```
LEFT_WIN = KeyModifier(key_name='LWin')
LShift = KeyModifier(key_name='LShift')
LWin = KeyModifier(key_name='LWin')
Left = Key(key_name='Left')
NUMPAD0 = Key(key_name='Numpad0')
NUMPAD1 = Key(key_name='Numpad1')
NUMPAD2 = Key(key_name='Numpad2')
NUMPAD3 = Key(key_name='Numpad3')
NUMPAD4 = Key(key_name='Numpad4')
NUMPAD5 = Key(key_name='Numpad5')
NUMPAD6 = Key(key_name='Numpad6')
NUMPAD7 = Key(key_name='Numpad7')
NUMPAD8 = Key(key_name='Numpad8')
NUMPAD9 = Key(key_name='Numpad9')
NUMPAD_ADD = Key(key_name='NumpadAdd')
NUMPAD_DEL = Key(key_name='NumpadDel')
NUMPAD_DIV = Key(key_name='NumpadDiv')
NUMPAD_DOT = Key(key_name='NumpadDot')
NUMPAD_ENTER = Key(key_name='NumpadEnter')
NUMPAD_MULT = Key(key_name='NumpadMult')
NUMPAD_SUB = Key(key_name='NumpadSub')
NUM_LOCK = Key(key_name='NumLock')
NumLock = Key(key_name='NumLock')
Numpad0 = Key(key_name='Numpad0')
Numpad1 = Key(key_name='Numpad1')
Numpad2 = Key(key_name='Numpad2')
Numpad3 = Key(key_name='Numpad3')
Numpad4 = Key(key_name='Numpad4')
Numpad5 = Key(key_name='Numpad5')
Numpad6 = Key(key_name='Numpad6')
Numpad7 = Key(key_name='Numpad7')
Numpad8 = Key(key_name='Numpad8')
Numpad9 = Key(key_name='Numpad9')
NumpadAdd = Key(key_name='NumpadAdd')
NumpadDel = Key(key_name='NumpadDel')
NumpadDiv = Key(key_name='NumpadDiv')
```

```

NumpadDot = Key(key_name='NumpadDot')
NumpadEnter = Key(key_name='NumpadEnter')
NumpadMult = Key(key_name='NumpadMult')
NumpadSub = Key(key_name='NumpadSub')
RAlt = KeyModifier(key_name='RAlt')
RControl = KeyModifier(key_name='RControl')
RCtrl = KeyModifier(key_name='RControl')
RIGHT = Key(key_name='Right')
RIGHT_ALT = KeyModifier(key_name='RAlt')
RIGHT_CONTROL = KeyModifier(key_name='RControl')
RIGHT_SHIFT = KeyModifier(key_name='RShift')
RIGHT_WIN = KeyModifier(key_name='RWin')
RShift = KeyModifier(key_name='RShift')
RWin = KeyModifier(key_name='RWin')
Right = Key(key_name='Right')
SCROLL_LOCK = Key(key_name='ScrollLock')
SHIFT = KeyModifier(key_name='Shift')
SPACE = Key(key_name='Space')
ScrollLock = Key(key_name='ScrollLock')
Shift = KeyModifier(key_name='Shift')
TAB = Key(key_name='Tab')
Tab = Key(key_name='Tab')
UP = Key(key_name='Up')
Up = Key(key_name='Up')
WIN = KeyModifier(key_name='Win')
Win = KeyModifier(key_name='Win')

```

## 2.5 Mouse

```
class ahk.mouse.AsyncMouseMixin(mouse_speed=2, mode=None, **kwargs)
```

```
    async get_mouse_position(mode=None)
```

```
    property mouse_position
```

```
class ahk.mouse.MouseMixin(mouse_speed=2, mode=None, **kwargs)
```

```
    Provides mouse functionality for the AHK class
```

```
    __init__(mouse_speed=2, mode=None, **kwargs)
```

**Parameters**

- **mouse\_speed** – default mouse speed
- **mode** –
- **kwargs** –

**click** (\*args, \*\*kwargs)

Click mouse button at a specified position. REF: <https://www.autohotkey.com/docs/commands/Click.htm>

**Parameters**

- **x** –
- **y** –
- **button** –
- **n** – number of times to click the button
- **direction** –
- **relative** –
- **blocking** –
- **mode** –

**Returns**

**double\_click** (\*args, \*\*kwargs)

Convenience function to double click, equivalent to `click` with `n=2`

**Parameters**

- **args** –
- **kwargs** –

**Returns**

**get\_mouse\_position** (mode=None)

**mouse\_drag** (\*args, \*\*kwargs)

Click and drag the mouse

**Parameters**

- **x** –
- **y** –
- **from\_position** – (x,y) tuple of an optional starting position. Current position is used if omitted
- **speed** –
- **button** – The button the click and drag; defaults to left mouse button
- **relative** – click and drag to a relative position rather than an absolute position
- **blocking** –
- **mode** –

**Returns**

**mouse\_move** (\*args, \*\*kwargs)

REF: <https://www.autohotkey.com/docs/commands/MouseMove.htm>

**Parameters**

- **x** – the x coordinate to move to. If omitted, current position is used
- **y** – the y coordinate to move to. If omitted, current position is used
- **speed** – 0 (fastest) to 100 (slowest). Can be a callable or string AHK expression
- **relative** – Move the mouse relative to current position rather than absolute x,y coordinates
- **mode** –
- **blocking** –

**Returns****property** `mouse_position`**property** `mouse_speed`**mouse\_wheel** (*direction*, \*args, \*\*kwargs)

Convenience function for ‘clicking’ the mouse wheel in a given direction.

**Parameters**

- **direction** – the string ‘up’ or ‘down’
- **args** – args passed to `click`
- **kwargs** – keyword args passed to `click`

**Returns****right\_click** (\*args, \*\*kwargs)Convenience function clicking right mouse button. Equivalent to `click` with `button='R'`**Parameters**

- **args** –
- **kwargs** –

**Returns****wheel\_down** (\*args, \*\*kwargs)Convenience function for `click` with wheel down button**Parameters**

- **args** –
- **kwargs** –

**Returns****wheel\_up** (\*args, \*\*kwargs)Convenience function for `click` with wheel up button**Parameters**

- **args** –
- **kwargs** –

**Returns**`ahk.mouse.resolve_button` (*button*)

Resolve a string of a button name to a canonical name used for AHK script

**Parameters** `button` (*str*) –

Returns

## 2.6 Registry

## 2.7 Screen

**class** `ahk.screen.AsyncScreenMixin` (*executable\_path=""*, *directives=None*, *\*\*kwargs*)

**async image\_search** (*\*args*, *\*\*kwargs*)

[AutoHotkey ImageSearch reference](#)

### Parameters

- **image\_path** – path to the image file e.g. C:\locationofcats.png
- **upper\_bound** – a two-tuple of X,Y coordinates for the upper-left corner of the search area e.g. (200, 400) defaults to (0,0)
- **lower\_bound** – like `upper_bound` but for the lower-righthand corner of the search area e.g. (400, 800) defaults to screen width and height (lower right-hand corner; `%A_ScreenWidth%`, `%A_ScreenHeight%`).
- **color\_variation** – Shades of variation (up or down) for the intensity of RGB for each pixel. Equivalent of `*n` option. Defaults to 0.
- **coord\_mode** – the Pixel CoordMode to use. Default is ‘Screen’
- **scale\_height** – Scale height in pixels. Equivalent of `*hn` option
- **scale\_width** – Scale width in pixels. Equivalent of `*wn` option
- **transparent** – Specific color in the image that will be ignored during the search. Pixels with the exact color given will match any color. Can be used with color names e.g. (Black, Purple, Yellow) found at <https://www.autohotkey.com/docs/commands/Progress.htm#colors> or hexadecimal values e.g. (0xFFFFAA, 05FA15, 632511). Equivalent of `*TransN` option
- **icon** – Number of the icon group to use. Equivalent of `*Icon` option

**Returns** coordinates of the upper-left pixel of where the image was found on the screen; None if the image was not found

**Return type** Union[Tuple[int, int], None]

Note: when only `scale_height` or only `scale_width` are provided, aspect ratio is maintained by default.

**async pixel\_search** (*\*args*, *\*\*kwargs*)

[AutoHotkey PixelSearch reference](#)

### Parameters

- **color** –
- **variation** –
- **upper\_bound** –
- **lower\_bound** –
- **coord\_mode** –

- **fast** –
- **rgb** –

**Returns** the coordinates of the pixel; None if the pixel is not found

**class** `ahk.screen.ScreenMixin` (*executable\_path=""*, *directives=None*, *\*\*kwargs*)

**image\_search** (*image\_path*, *upper\_bound=(0, 0)*, *lower\_bound=None*, *color\_variation=None*, *coord\_mode='Screen'*, *scale\_height=None*, *scale\_width=None*, *transparent=None*, *icon=None*)

[AutoHotkey ImageSearch reference](#)

#### Parameters

- **image\_path** (*str*) – path to the image file e.g. C:\locationofcats.png
- **upper\_bound** (*Tuple[int, int]*) – a two-tuple of X,Y coordinates for the upper-left corner of the search area e.g. (200, 400) defaults to (0,0)
- **lower\_bound** (*Optional[Tuple[int, int]]*) – like `upper_bound` but for the lower-right-hand corner of the search area e.g. (400, 800) defaults to screen width and height (lower right-hand corner; %A\_ScreenWidth%, %A\_ScreenHeight%).
- **color\_variation** (*Optional[int]*) – Shades of variation (up or down) for the intensity of RGB for each pixel. Equivalent of `*n` option. Defaults to 0.
- **coord\_mode** (*str*) – the Pixel CoordMode to use. Default is 'Screen'
- **scale\_height** (*Optional[int]*) – Scale height in pixels. Equivalent of `*hn` option
- **scale\_width** (*Optional[int]*) – Scale width in pixels. Equivalent of `*wn` option
- **transparent** (*Optional[str]*) – Specific color in the image that will be ignored during the search. Pixels with the exact color given will match any color. Can be used with color names e.g. (Black, Purple, Yellow) found at <https://www.autohotkey.com/docs/commands/Progress.htm#colors> or hexadecimal values e.g. (0xFFFFAA, 05FA15, 632511). Equivalent of `*TransN` option
- **icon** (*Optional[int]*) – Number of the icon group to use. Equivalent of `*Icon` option

**Returns** coordinates of the upper-left pixel of where the image was found on the screen; None if the image was not found

**Return type** `Union[Tuple[int, int], None]`

Note: when only `scale_height` or only `scale_width` are provided, aspect ratio is maintained by default.

**pixel\_get\_color** (*\*args*, *\*\*kwargs*)

[AutoHotkey PixelGetColor reference](#)

#### Parameters

- **x** – x coordinate
- **y** – y coordinate
- **coord\_mode** –
- **alt** –
- **slow** –
- **rgb** – returns

**Returns** the color as an RGB hexadecimal string;

`pixel_search` (\*args, \*\*kwargs)  
AutoHotkey PixelSearch reference

#### Parameters

- `color` –
- `variation` –
- `upper_bound` –
- `lower_bound` –
- `coord_mode` –
- `fast` –
- `rgb` –

**Returns** the coordinates of the pixel; None if the pixel is not found

## 2.8 The Script Engine

The `script` module, most essentially, houses the `ScriptEngine` class.

The `ScriptEngine` is responsible for rendering autohotkey code from jinja templates and executing that code. This is the heart of how this package works. Every other major component either inherits from this class or utilizes an instance of this class.

The current implementation of how autohotkey code is executed is by calling the autohotkey executable with `subprocess`.

```
ahk.script.DEFAULT_EXECUTABLE_PATH = C:\Program Files\AutoHotkey\AutoHotkey.exe
```

The default path to look for AutoHotkey, if not specified some other way

```
class ahk.script.AsyncScriptEngine (executable_path="", directives=None, **kwargs)
```

```
async run_script (script_text, decode=True, blocking=True, **runkwargs)
```

async version of `run_script`

#### Parameters

- `script_text` (`str`) – a string containing AutoHotkey code
- `decode` – If `True`, attempt to decode the stdout of the completed process. If `False`, returns the completed process. Only has effect when `blocking=True`
- `blocking` – If `True`, script must finish before returning. If `False`, function returns a `asyncio.process` object immediately without blocking
- `runkwargs` – keyword arguments passed to `subprocess.Popen` or `subprocess.run`

#### Returns

A string of the decoded stdout if `blocking` and `decode` are `True`.  
A bytes object of stdout if `blocking` is `True` and `decode` is `False`.  
`asyncio.subprocess.Process` object if `blocking` is `False`.



```
ahk.script.DEFAULT_EXECUTABLE_PATH = 'C:\\Program Files\\AutoHotkey\\AutoHotkey.exe'
```

The default path to look for AutoHotkey, if not specified some other way

```
exception ahk.script.ExecutableNotFoundError
```

```
class ahk.script.ScriptEngine (executable_path="", directives=None, **kwargs)
```

```
__init__ (executable_path="", directives=None, **kwargs)
```

This class is typically not used directly. AHK components inherit from this class and the arguments for this class should usually be passed in to AHK.

#### Parameters

- **executable\_path** (`str`) – the path to the AHK executable. If not provided explicitly in this argument, the path to the AHK executable is resolved in the following order:
  - The `AHK_PATH` environment variable, if present
  - `DEFAULT_EXECUTABLE_PATH` if the file exists
 If environment variable not present, tries to look for ‘AutoHotkey.exe’ or ‘AutoHotkeyA32.exe’ with `shutil.which`
- **directives** (`Optional[Set]`) – a set of directives to apply to all generated AHK scripts

**Raises ExecutableNotFound** – if AHK executable cannot be found or the specified file does not exist

```
async a_run_script (script_text, decode=True, blocking=True, **kwargs)
```

async version of `run_script`

#### Parameters

- **script\_text** (`str`) – a string containing AutoHotkey code
- **decode** – If `True`, attempt to decode the stdout of the completed process. If `False`, returns the completed process. Only has effect when `blocking=True`
- **blocking** – If `True`, script must finish before returning. If `False`, function returns a `asyncio.process` object immediately without blocking
- **kwargs** – keyword arguments passed to `subprocess.Popen` or `subprocess.run`

#### Returns

A string of the decoded stdout if `blocking` and `decode` are `True`.  
 A bytes object of stdout if `blocking` is `True` and `decode` is `False`.  
`asyncio.subprocess.Process` object if `blocking` is `False`.

```
render_template (template_name, directives=None, blocking=True, **kwargs)
```

Renders a given jinja template and returns a string of script text

#### Parameters

- **template\_name** – the name of the jinja template to render
- **directives** – additional AHK directives to add to the resulting script
- **blocking** – whether the template should be rendered to block (use `#Persistent` directive)

- **kwargs** – keywords passed to template rendering

**Returns** An AutoHotkey script as a string

```
>>> from ahk import AHK
>>> ahk = AHK()
>>> ahk.render_template('keyboard/send_input.ahk', s='Hello')
'#NoEnv\n#Persistent\n\n\nSendInput Hello\n\nExitApp\n'
```

**run\_script** (*script\_text*, *decode=True*, *blocking=True*, *\*\*kwargs*)

Given an AutoHotkey script as a string, execute it

#### Parameters

- **script\_text** (*str*) – a string containing AutoHotkey code
- **decode** – If `True`, attempt to decode the stdout of the completed process. If `False`, returns the completed process. Only has effect when `blocking=True`
- **blocking** – If `True`, script must finish before returning. If `False`, function returns a `subprocess.Popen` object immediately without blocking
- **kwargs** – keyword arguments passed to `subprocess.Popen` or `subprocess.run`

#### Returns

A string of the decoded stdout if `blocking` and `decode` are `True`.

`subprocess.CompletedProcess` if `blocking` is `True` and `decode` is `False`.

`subprocess.Popen` object if `blocking` is `False`.

```
>>> from ahk import AHK
>>> ahk = AHK()
>>> ahk.run_script('FileAppend, Hello World, *')
'Hello World'
>>> ahk.run_script('FileAppend, Hello World, *', decode=False)
CompletedProcess(args=['C:\\path\\AutoHotkey.exe', '/ErrorStdOut', '*'],
↳returncode=0, stdout=b'Hello World', stderr=b'')
>>> ahk.run_script('FileAppend, Hello World, *', blocking=False)
<subprocess.Popen at 0x18a599cde10>
```

## 2.9 Sound

**class** `ahk.sound.AsyncSoundMixin` (*executable\_path=""*, *directives=None*, *\*\*kwargs*)

**class** `ahk.sound.SoundMixin` (*executable\_path=""*, *directives=None*, *\*\*kwargs*)

**get\_volume** (*device\_number=1*)

REF: <https://autohotkey.com/docs/commands/SoundGetWaveVolume.htm>

**Parameters** *device\_number* –

**Returns**

**set\_volume** (*value*, *device\_number=1*)

REF: <https://autohotkey.com/docs/commands/SoundSetWaveVolume.htm>

**Parameters**

- **value** – percent volume to set volume to
- **device\_number** –

**Returns**

**sound\_beep** (*frequency=523, duration=150*)

REF: <https://autohotkey.com/docs/commands/SoundBeep.htm>

**Parameters**

- **frequency** – number between 37 and 32767
- **duration** – how long in milliseconds to play the beep

**Returns** None

**sound\_get** (*device\_number=1, component\_type='MASTER', control\_type='VOLUME'*)

REF: <https://autohotkey.com/docs/commands/SoundGet.htm>

**Parameters**

- **device\_number** –
- **component\_type** –
- **control\_type** –

**Returns**

**sound\_play** (*filename, blocking=True*)

REF: <https://autohotkey.com/docs/commands/SoundPlay.htm>

**Parameters**

- **filename** –
- **blocking** –
- **wait** –

**Returns**

**sound\_set** (*value, device\_number=1, component\_type='MASTER', control\_type='VOLUME'*)

REF: <https://autohotkey.com/docs/commands/SoundSet.htm>

**Parameters**

- **value** –
- **device\_number** –
- **component\_type** –
- **control\_type** –

**Returns**

## 2.10 Utils

`ahk.utils.async_filter` (*async\_pred, iterable*)

`ahk.utils.escape_sequence_replace` (*s*)

Replace Python escape sequences with AHK equivalent escape sequences. Additionally escapes some other characters for AHK escape sequences. Intended for use with AHK Send command functions.

Note: This DOES NOT provide ANY assurances against accidental or malicious injection. Does NOT escape quotes.

```
>>> escape_sequence_replace('Hello, World!')
'Hello`, World{!}'
```

`ahk.utils.make_logger` (*name*)

## 2.11 Window

**class** `ahk.window.AsyncWindow` (*engine, ahk\_id, encoding=None*)

**property** `always_on_top`

Return type `bool`

**async classmethod** `from_mouse_position` (*engine, \*\*kwargs*)

**async classmethod** `from_pid` (*engine, pid, \*\*kwargs*)

**async** `get_pos` (*info=None*)

Returns

**async** `get_transparency` ()

Return type `int`

**property** `height`

**async** `is_always_on_top` ()

**async** `is_maximized` ()

**async** `is_minimized` ()

**async** `is_minmax` ()

**property** `non_max_non_min`

**property** `position`

**property** `rect`

**async** `set_transparency` (*value*)

**property** `title`

**property** `transparent`

Return type `int`

**property** `width`

```
class ahk.window.AsyncWindowMixin (*args, **kwargs)
```

```
property active_window
```

```
async find_window (func=None, **kwargs)
```

Like `find_windows` but only returns the first found window

**Parameters**

- **func** –
- **kwargs** –

**Returns** a `Window` object or `None` if no matching window is found

```
async find_window_by_class (*args, **kwargs)
```

**Parameters**

- **args** –
- **kwargs** –

**Returns** a `Window` object or `None` if no matching window is found

```
async find_window_by_text (*args, **kwargs)
```

**Parameters**

- **args** –
- **kwargs** –

**Returns** a `Window` object or `None` if no matching window is found

```
async find_window_by_title (title)
```

Like `find_windows_by_title` but only returns the first result.

**Returns** a `Window` object or `None` if no matching window is found

```
find_windows (func=None, **kwargs)
```

Find all matching windows

**Parameters**

- **func** – a callable to filter windows
- **exact** (*bool*) – if `False` (the default) partial matches are found. If `True`, only exact matches are returned
- **kwargs** – keywords of attributes of the window (has no effect if `func` is provided)

**Returns** a generator containing any matching `Window` objects.

```
find_windows_by_class (class_name, exact=False)
```

**Parameters**

- **class\_name** –
- **exact** –

**Returns** a generator containing any matching `Window` objects.

```
find_windows_by_text (text, exact=False)
```

**Parameters**

- **text** –

- **exact** –

Returns a generator containing any matching *Window* objects.

**find\_windows\_by\_title** (*title*, *exact=False*)

Equivalent to `find_windows (title=title)`

Note that *title* is a bytes object

**Parameters**

- **title** (*bytes*) –
- **exact** –

**Returns**

**async win\_get** (*\*args*, *\*\*kwargs*)

**async windows** ()

Returns a list of windows

**Returns**

**class** `ahk.window.Control`

**\_\_init\_\_** ()

Initialize self. See `help(type(self))` for accurate signature.

**click** ()

REF: <https://www.autohotkey.com/docs/commands/ControlClick.htm>

**Returns**

**focus** ()

REF: <https://www.autohotkey.com/docs/commands/ControlFocus.htm>

**Returns**

**get** (*key*)

REF: <https://www.autohotkey.com/docs/commands/ControlGet.htm>

**Parameters** *key* –

**Returns**

**has\_focus** ()

**move** ()

REF: <https://www.autohotkey.com/docs/commands/ControlMove.htm>

**Returns**

**property position**

REF: <https://www.autohotkey.com/docs/commands/ControlGetPos.htm>

**Returns**

**send** (*raw=False*)

REF: <https://www.autohotkey.com/docs/commands/ControlSend.htm>

**Parameters** *raw* –

**Returns**

**property text**

REF: <https://www.autohotkey.com/docs/commands/ControlGetText.htm>

**Returns**

**class** ahk.window.**Window** (*engine, ahk\_id, encoding=None*)

**MAXIMIZED** = '1'

**MINIMIZED** = '-1'

**NON\_MIN\_NON\_MAX** = '0'

**\_\_init\_\_** (*engine, ahk\_id, encoding=None*)

Initialize self. See help(type(self)) for accurate signature.

**activate** ()

Activate the window.

See also: [WinActivate](#)

**Returns**

**activate\_bottom** ()

Calls [WinActivateBottom](#) on the window

**Returns**

**property active**

**property always\_on\_top**

**Return type** bool

**property class\_name**

**click** (*\*args, \*\*kwargs*)

Click at an x/y location on the screen. Uses [ControlClick](https://autohotkey.com/docs/commands/ControlClick.htm) <https://autohotkey.com/docs/commands/ControlClick.htm>

x/y position params may also be specified as a 2-item sequence

**Parameters**

- **x** – x offset relative to topleft corner of the window
- **y** – y offset relative to the top of the window
- **button** – the button to press (default is left mouse)
- **n** – number of times to click
- **options** – per [ControlClick](#) documentation
- **blocking** –

**Returns**

**close** (*seconds\_to\_wait=""*)

Closes the Window. See also: [WinClose](#)

**Parameters** **seconds\_to\_wait** –

**Returns**

**disable** ()

Distable the window

**Returns**

**enable ()**

Enable the window

**Returns**

**property exist**

**exists ()**

**classmethod from\_mouse\_position (engine, \*\*kwargs)**

**classmethod from\_pid (engine, pid, \*\*kwargs)**

**get (subcommand)**

**get\_class\_name ()**

**get\_pos (info=None)**

**Returns**

**get\_text ()**

**get\_title ()**

**get\_transparency ()**

**Return type** int

**property height**

**hide ()**

Hides the window. See also: [WinHide](#)

**Returns**

**is\_active ()**

**is\_always\_on\_top ()**

**Return type** bool

**is\_maximized ()**

**is\_minimized ()**

**is\_minmax ()**

**kill (seconds\_to\_wait="")**

**maximize ()**

maximize the window

**Returns**

**property maximized**

**minimize ()**

minimize the window

**Returns**

**property minimized**

**move (x=", y=", width=None, height=None)**

Move the window to a position and/or change its geometry

**Parameters**

- **x** –



- **y** -
- **width** -
- **height** -

**Returns****property non\_max\_non\_min****property position****property rect****redraw ()****restore ()**

restore the window

**Returns****send** (*keys, delay=10, raw=False, blocking=False, escape=False, press\_duration=-1*)Send keystrokes directly to the window. Uses ControlSend <https://autohotkey.com/docs/commands/Send.htm>**set** (*subcommand, value*)**set\_always\_on\_top** (*value*)**set\_position** (*new\_position*)**set\_title** (*value*)**set\_transparency** (*value*)**show ()**

show the window

**Returns****property text****property title****to\_bottom ()**

Send window to bottom (behind other windows) :return:

**to\_top ()**

Bring the window to the foreground (above other windows)

**Returns****property transparent****Return type** int**wait** (*seconds\_to\_wait=""*)**Parameters** *seconds\_to\_wait* -**Returns****wait\_active** (*seconds\_to\_wait=""*)**Parameters** *seconds\_to\_wait* -**Returns****wait\_close** (*seconds\_to\_wait=""*)

Parameters `seconds_to_wait` –

Returns

`wait_not_active` (*seconds\_to\_wait*="")

Parameters `seconds_to_wait` –

Returns

property `width`

`class` `ahk.window.WindowMixin` (\*args, \*\*kwargs)

`__init__` (\*args, \*\*kwargs)

This class is typically not used directly. AHK components inherit from this class and the arguments for this class should usually be passed in to AHK.

Parameters

- `executable_path` – the path to the AHK executable. If not provided explicitly in this argument, the path to the AHK executable is resolved in the following order:
  - The `AHK_PATH` environment variable, if present
  - `DEFAULT_EXECUTABLE_PATH` if the file exists
 If environment variable not present, tries to look for ‘AutoHotkey.exe’ or ‘AutoHotkeyA32.exe’ with `shutil.which`
- `directives` – a set of directives to apply to all generated AHK scripts

Raises `ExecutableNotFound` – if AHK executable cannot be found or the specified file does not exist

property `active_window`

`find_window` (*func*=None, \*\*kwargs)

Like `find_windows` but only returns the first found window

Parameters

- `func` –
- `kwargs` –

Returns a `Window` object or `None` if no matching window is found

`find_window_by_class` (\*args, \*\*kwargs)

Parameters

- `args` –
- `kwargs` –

Returns a `Window` object or `None` if no matching window is found

`find_window_by_text` (\*args, \*\*kwargs)

Parameters

- `args` –
- `kwargs` –

Returns a `Window` object or `None` if no matching window is found

**find\_window\_by\_title** (\*args, \*\*kwargs)

Like `find_windows_by_title` but only returns the first result.

**Returns** a `Window` object or `None` if no matching window is found

**find\_windows** (func=None, \*\*kwargs)

Find all matching windows

**Parameters**

- **func** – a callable to filter windows
- **exact** (*bool*) – if `False` (the default) partial matches are found. If `True`, only exact matches are returned
- **kwargs** – keywords of attributes of the window (has no effect if `func` is provided)

**Returns** a generator containing any matching `Window` objects.

**find\_windows\_by\_class** (class\_name, exact=False)

**Parameters**

- **class\_name** –
- **exact** –

**Returns** a generator containing any matching `Window` objects.

**find\_windows\_by\_text** (text, exact=False)

**Parameters**

- **text** –
- **exact** –

**Returns** a generator containing any matching `Window` objects.

**find\_windows\_by\_title** (title, exact=False)

Equivalent to `find_windows(title=title)`

Note that `title` is a `bytes` object

**Parameters**

- **title** (*bytes*) –
- **exact** –

**Returns**

**get\_active\_window** ()

**win\_get** (title="", text="", exclude\_title="", exclude\_text="", encoding=None)

**win\_set** (subcommand, \*args, blocking=True)

**windows** ()

Returns a list of windows

**Returns**

**exception** `ahk.window.WindowNotFoundError`



**AHK**

A Python wrapper around AHK.



## INSTALLATION

```
pip install ahk
```

Requires Python 3.6+

*Async API* requires Python 3.8+

See also *Non-Python dependencies*





## USAGE

```
from ahk import AHK

ahk = AHK()

ahk.mouse_move(x=100, y=100, blocking=True) # Blocks until mouse finishes moving,
↳ (the default)
ahk.mouse_move(x=150, y=150, speed=10, blocking=True) # Moves the mouse to x, y,
↳ taking 'speed' seconds to move
print(ahk.mouse_position) # (150, 150)
```



## EXAMPLES

Non-exhaustive examples of some of the functions available with this package. Full documentation coming soon!

### 6.1 Mouse

```
from ahk import AHK

ahk = AHK()

ahk.mouse_position # Returns a tuple of mouse coordinates (x, y)
ahk.mouse_move(100, 100, speed=10, relative=True) # Moves the mouse reletave to the_
↳current position
ahk.mouse_position = (100, 100) # Moves the mouse instantly to absolute screen_
↳position
ahk.click() # Click the primary mouse button
ahk.double_click() # Clicks the primary mouse button twice
ahk.click(200, 200) # Moves the mouse to a particular position and clicks
ahk.right_click() # Clicks the secondary mouse button
ahk.mouse_drag(100, 100, relative=True) # Holds down primary button and moves the_
↳mouse
```

### 6.2 Keyboard

```
from ahk import AHK

ahk = AHK()

ahk.type('hello, world!') # Send keys, as if typed (performs ahk string escapes)
ahk.send_input('Hello`, World{!}') # Like AHK SendInput, must escape strings_
↳yourself!
ahk.key_state('Control') # Return True or False based on whether Control key is_
↳pressed down
ahk.key_state('CapsLock', mode='T') # Check toggle state of a key (like for NumLock,_
↳CapsLock, etc)
ahk.key_press('a') # Press and release a key
ahk.key_down('Control') # Press down (but do not release) Control key
ahk.key_up('Control') # Release the key
ahk.key_wait('a', timeout=3) # Wait up to 3 seconds for the "a" key to be pressed._
↳NOTE: This throws
```

(continues on next page)

(continued from previous page)

```

                                # a TimeoutError if the key isn't pressed within the_
↪timeout window
ahk.set_capslock_state("on") # Turn CapsLock on

```

## 6.3 Windows

You can do stuff with windows, too.

### 6.3.1 Getting windows

```

from ahk import AHK
from ahk.window import Window

ahk = AHK()

win = ahk.active_window           # Get the active window
win = ahk.win_get(title='Untitled - Notepad') # by title
win = list(ahk.windows())        # list of all windows
win = Window(ahk, ahk_id='0xabc123') # by ahk_id
win = Window.from_mouse_position(ahk) # the window under the mouse cursor
win = Window.from_pid('20366')    # by process ID

```

### 6.3.2 Working with windows

```

from ahk import AHK

ahk = AHK()

ahk.run_script('Run Notepad') # Open notepad
win = ahk.find_window(title=b'Untitled - Notepad') # Find the opened window

win.send('hello') # Send keys directly to the window (does not need focus!)
win.move(x=200, y=300, width=500, height=800)

win.activate()           # Give the window focus
win.activate_bottom()   # Give the window focus
win.close()              # Close the window
win.hide()               # Hide the window
win.kill()               # Kill the window
win.maximize()           # Maximize the window
win.minimize()           # Minimize the window
win.restore()            # Restore the window
win.show()                # Show the window
win.disable()            # Make the window non-interactable
win.enable()              # Enable it again
win.to_top()              # Move the window on top of other windows
win.to_bottom()          # Move the window to the bottom of the other windows

win.always_on_top = True # Make the window always on top

```

(continues on next page)

(continued from previous page)

```

for window in ahk.windows():
    print(window.title)

    # Some more attributes
    print(window.text)
    print(window.rect)    # (x, y, width, height)
    print(window.id)     # ahk_id
    print(window.pid)
    print(window.process)

if window.active:      # Check if window active
    window.minimize()

if window.exist:     # Check if window exist
    window.maximize()

```

## 6.4 Screen

```

from ahk import AHK

ahk = AHK()

ahk.image_search('C:\\path\\to\\image.jpg') # Find an image on screen

# Find an image within a boundary on screen
ahk.image_search('C:\\path\\to\\image.jpg', upper_bound=(100, 100), # upper-left_
↳corner of search area
                                     lower_bound=(400, 400)) # lower-right_
↳corner of search area
ahk.pixel_get_color(100, 100) # Get color of pixel located at coords (100, 100)
ahk.pixel_search('0x9d6346') # Get coords of the first pixel with specified color

```

## 6.5 Sound

```

from ahk import AHK

ahk = AHK()

ahk.sound_play('C:\\path\\to\\sound.wav') # Play an audio file
ahk.sound_beep(frequency=440, duration=1000) # Play a beep for 1 second (duration in_
↳microseconds)
ahk.get_volume(device_number=1) # Get volume of a device
ahk.set_volume(50, device_number=1) # Set volume of a device
ahk.sound_get(device_number=1, component_type='MASTER', control_type='VOLUME') # Get_
↳sound device property
ahk.sound_set(50, device_number=1, component_type='MASTER', control_type='VOLUME') #_
↳Set sound device property

```

## 6.6 GUI

```
from ahk import AHK

ahk = AHK()
ahk.show_tooltip("hello4", second=2, x=10, y=10) # ↵
↳ToolTip
ahk.show_info_traytip("Info", "It's also info", slient=False, blocking=True) # ↵
↳Default info traytip
ahk.show_warning_traytip("Warning", "It's warning") # ↵
↳Warning traytip
ahk.show_error_traytip("Error", "It's error") # Error ↵
↳trytip
```

## 6.7 non-blocking modes

For some functions, you can also opt for a non-blocking interface, so you can do other stuff while AHK scripts run.

```
import time

from ahk import AHK

ahk = AHK()

ahk.mouse_position = (200, 200) # Moves the mouse instantly to the start position
start = time.time()
ahk.mouse_move(x=100, y=100, speed=30, blocking=False)
while True: # report mouse position while it moves
    t = round(time.time() - start, 4)
    position = ahk.mouse_position
    print(t, position)
    if position == (100, 100):
        break
```

You should see an output something like

```
0.032 (187, 187)
0.094 (173, 173)
0.137 (164, 164)
...
0.788 (100, 103)
0.831 (100, 101)
0.873 (100, 100)
```

## 6.8 Add directives

You can add directives that will be added to all generated scripts. For example, to prevent the AHK trayicon from appearing, you can add the NoTrayIcon directive.

```
from ahk import AHK
from ahk.directives import NoTrayIcon

ahk = AHK(directives=[NoTrayIcon])
```

By default, some directives are automatically added to ensure functionality and are merged with any user-provided directives.

## 6.9 Run arbitrary AutoHotkey scripts

```
from ahk import AHK

ahk = AHK()

ahk_script = 'Run Notepad'
ahk.run_script(ahk_script, blocking=False)
```

### 6.9.1 Communicating data from ahk to Python

If you're writing your own ahk scripts to use with this library, you can use FileAppend with the \* parameter to get data from your ahk script into Python.

Suppose you have a script like so

```
#Persistent
data := "Hello Data!"
FileAppend, %data%, * ; send data var to stdout
ExitApp
```

```
result = ahk.run_script(my_script)
print(result) # Hello Data!
```

If your autohotkey returns something that can't be decoded, add the keyword argument decode=False in which case you'll get back a CompletedProcess object where stdout (and stderr) will be bytes and you can handle it however you choose.

```
result = ahk.run_script(my_script, decode=False)
print(result.stdout) # b'Hello Data!'
```

## 6.10 Preview features

Preview features are experimental features that are may not be fully functional. These features are (even more) likely to have breaking changes without warning.

Github issues are provided for convenience to collect feedback on these features.

## 6.11 Async API

An async API is provided so functions can be called using `async/await`. All the same methods from the synchronous API are available in the async API.

```
from ahk import AsyncAHK
import asyncio
ahk = AsyncAHK()

async def main():
    await ahk.mouse_move(100, 100)
    x, y = await ahk.get_mouse_position()
    print(x, y)

asyncio.run(main())
```

For the most part, the async API is identical to that of the normal API, with a few exceptions:

While properties (like `.mouse_position` or `.title` for windows) can be awaited, additional methods (like `get_mouse_position()` and `get_title()`) have been added for a more intuitive API.

Property setters have different (probably undesired) behavior in the async API. Instead, you should use a comparable method. If you *do* use the property setters, the invocation is created using `asyncio.create_task()`, which means that the task won't run until control is yielded back to the event loop. For now, this will also raise a warning to the same.

Lastly, while it's possible to pass `blocking=False` in the async API, this sometimes will cause problems with certain functions. For now, a warning is raised in this case.

```
ahk = AsyncAHK()
async def main():
    pos = ahk.mouse_position # BAD! Does not work!
    pos = await ahk.mouse_position # OK. Works, but looks kind of weird
    pos = await ahk.get_mouse_position() # GOOD!

    # BAD: You probably don't want to do this
    ahk.mouse_position = (100, 100) # won't do anything right away. Raises warning
    print(await ahk.get_mouse_position()) # probably won't be 100,100

    # GOOD: Instead, do this:
    await ahk.mouse_move(100, 100, speed=0)
    assert await ahk.get_mouse_position() == (100, 100)
```



## 6.11.1 Hotkeys

GH-9

Hotkeys now have a primitive implementation. You give it a hotkey (a string the same as in an ahk script, without the : :) and the body of an AHK script to execute as a response to the hotkey.

Right now, only AHK code is supported as callbacks for hotkeys. Support for Python callbacks via the Async API is planned.

```
from ahk import AHK, Hotkey

ahk = AHK()

key_combo = '#n' # Define an AutoHotkey key combination
script = 'Run Notepad' # Define an ahk script
hotkey = Hotkey(ahk, key_combo, script) # Create Hotkey
hotkey.start() # Start listening for hotkey
```

At this point, the hotkey is active. If you press + , the script Run Notepad will execute.

There is no need to add return to the provided script, as it is provided by the template.

To stop the hotkey call the stop() method.

```
hotkey.stop()
```

See also the [relevant AHK documentation](#)

## 6.11.2 ActionChain

GH-25

ActionChains let you define a set of actions to be performed in order at a later time.

They work just like the AHK class, except the actions are deferred until the perform method is called.

An additional method sleep is provided to allow for waiting between actions.

```
from ahk import ActionChain

ac = ActionChain()

# An Action Chain doesn't perform the actions until perform() is called on the chain

ac.mouse_move(100, 100, speed=10) # nothing yet
ac.sleep(1) # still nothing happening
ac.mouse_move(500, 500, speed=10) # not yet
ac.perform() # *now* each of the actions run in order
```

Just like anywhere else, scripts running simultaneously may conflict with one another, so using blocking interfaces is generally recommended. Currently, there is limited support for interacting with windows in actionchains, you may want to use win\_set)

### 6.11.3 find\_window/find\_windows methods

GH-26

Right now, these are implemented by iterating over all window handles and filtering with Python. They may be optimized in the future.

AHK.`find_windows` returns a generator filtering results based on attributes provided as keyword arguments. AHK.`find_window` is similar, but returns the first matching window instead of all matching windows.

There are couple convenience functions, but not sure if these will stay around or maybe we'll add more, depending on feedback.

- `find_windows_by_title`
- `find_window_by_title`
- `find_windows_by_text`
- `find_window_by_text`

## 6.12 Errors and Debugging

You can enable debug logging, which will output script text before execution, and some other potentially useful debugging information.

```
import logging
logging.basicConfig(level=logging.DEBUG)
```

(See the [logging module documentation](#) for more information)

Also note that, for now, errors with running AHK scripts will often pass silently. In the future, better error handling will be added.

## 6.13 Non-Python dependencies

To use this package, you need the [AutoHotkey executable](#).

It's expected to be on PATH by default. You can also use the `AHK_PATH` environment variable to specify the executable location.

Alternatively, you may provide the path in code

```
from ahk import AHK

ahk = AHK(executable_path='C:\\path\\to\\AutoHotkey.exe')
```

## CONTRIBUTING

All contributions are welcomed and appreciated.

Please feel free to open a GitHub issue or PR for feedback, ideas, feature requests or questions.

There's still some work to be done in the way of implementation. The ideal interfaces are still yet to be determined and *your* help would be invaluable.

The vision is to provide access to the most useful features of the AutoHotkey API in a Pythonic way.



## SIMILAR PROJECTS

These are some similar projects that are commonly used for automation with Python.

- [Pyautogui](#) - Al Sweigart's creation for cross-platform automation
- [Pywinauto](#) - Automation on Windows platforms with Python.
- [keyboard](#) - Pure Python cross-platform keyboard hooks/control and hotkeys!
- [mouse](#) - From the creators of [keyboard](#), Pure Python *mouse* control!
- [pynput](#) - Keyboard and mouse control



## INDICES AND TABLES

- genindex
- modindex
- search





## PYTHON MODULE INDEX

### a

ahk.autohotkey, 5  
ahk.directives, 6  
ahk.keyboard, 7  
ahk.keys, 10  
ahk.mouse, 15  
ahk.screen, 18  
ahk.script, 20  
ahk.sound, 22  
ahk.utils, 24  
ahk.window, 24



## Symbols

- `__init__()` (*ahk.autohotkey.ActionChain* method), 5
  - `__init__()` (*ahk.directives.ClipboardTimeout* method), 6
  - `__init__()` (*ahk.directives.Directive* method), 6
  - `__init__()` (*ahk.directives.Include* method), 6
  - `__init__()` (*ahk.directives.InputLevel* method), 6
  - `__init__()` (*ahk.directives.KeyHistory* method), 6
  - `__init__()` (*ahk.directives.MaxHotkeysPerInterval* method), 6
  - `__init__()` (*ahk.directives.MaxMem* method), 6
  - `__init__()` (*ahk.directives.MaxThreads* method), 7
  - `__init__()` (*ahk.directives.MaxThreadsBuffer* method), 7
  - `__init__()` (*ahk.directives.MaxThreadsPerHotkey* method), 7
  - `__init__()` (*ahk.directives.MenuMaskKey* method), 7
  - `__init__()` (*ahk.keyboard.Hotkey* method), 8
  - `__init__()` (*ahk.mouse.MouseMixin* method), 15
  - `__init__()` (*ahk.script.ScriptEngine* method), 21
  - `__init__()` (*ahk.window.Control* method), 26
  - `__init__()` (*ahk.window.Window* method), 27
  - `__init__()` (*ahk.window.WindowMixin* method), 30
- A**
- `a_run_script()` (*ahk.script.ScriptEngine* method), 21
  - ActionChain* (class in *ahk.autohotkey*), 5
  - `activate()` (*ahk.window.Window* method), 27
  - `activate_bottom()` (*ahk.window.Window* method), 27
  - `active()` (*ahk.window.Window* property), 27
  - `active_window()` (*ahk.window.AsyncWindowMixin* property), 25
  - `active_window()` (*ahk.window.WindowMixin* property), 30
  - AHK (class in *ahk.autohotkey*), 5
  - ahk.autohotkey* (module), 5
  - ahk.directives* (module), 6
  - ahk.keyboard* (module), 7
  - ahk.keys* (module), 10
  - ahk.mouse* (module), 15
  - ahk.screen* (module), 18
  - ahk.script* (module), 20
  - ahk.sound* (module), 22
  - ahk.utils* (module), 24
  - ahk.window* (module), 24
  - AllowSameLineComments* (class in *ahk.directives*), 6
  - ALT (*ahk.keys.KEYS* attribute), 10
  - Alt (*ahk.keys.KEYS* attribute), 10
  - `always_on_top()` (*ahk.window.AsyncWindow* property), 24
  - `always_on_top()` (*ahk.window.Window* property), 27
  - `async_filter()` (in module *ahk.utils*), 24
  - AsyncAHK* (class in *ahk.autohotkey*), 5
  - AsyncKeyboardMixin* (class in *ahk.keyboard*), 7
  - AsyncMouseMixin* (class in *ahk.mouse*), 15
  - AsyncScreenMixin* (class in *ahk.screen*), 18
  - AsyncScriptEngine* (class in *ahk.script*), 20
  - AsyncSoundMixin* (class in *ahk.sound*), 22
  - AsyncWindow* (class in *ahk.window*), 24
  - AsyncWindowMixin* (class in *ahk.window*), 24
- B**
- BACKSPACE (*ahk.keys.KEYS* attribute), 10
  - Backspace (*ahk.keys.KEYS* attribute), 10
- C**
- CAPS\_LOCK (*ahk.keys.KEYS* attribute), 10
  - CapsLock (*ahk.keys.KEYS* attribute), 11
  - `class_name()` (*ahk.window.Window* property), 27
  - `click()` (*ahk.mouse.MouseMixin* method), 16
  - `click()` (*ahk.window.Control* method), 26
  - `click()` (*ahk.window.Window* method), 27
  - ClipboardTimeout* (class in *ahk.directives*), 6
  - `close()` (*ahk.window.Window* method), 27
  - CONTROL (*ahk.keys.KEYS* attribute), 10
  - Control (*ahk.keys.KEYS* attribute), 11
  - Control (class in *ahk.window*), 26
  - CTRL (*ahk.keys.KEYS* attribute), 10
  - Ctrl (*ahk.keys.KEYS* attribute), 11

## D

DEFAULT\_EXECUTABLE\_PATH (in module *ahk.script*), 20  
 DEL (*ahk.keys.KEYS* attribute), 11  
 Del (*ahk.keys.KEYS* attribute), 11  
 DELETE (*ahk.keys.KEYS* attribute), 11  
 Delete (*ahk.keys.KEYS* attribute), 11  
 Directive (class in *ahk.directives*), 6  
 DirectiveMeta (class in *ahk.directives*), 6  
 disable() (*ahk.window.Window* method), 27  
 double\_click() (*ahk.mouse.MouseMixin* method), 16  
 DOWN (*ahk.keys.KEYS* attribute), 11  
 Down (*ahk.keys.KEYS* attribute), 11

## E

enable() (*ahk.window.Window* method), 27  
 ENTER (*ahk.keys.KEYS* attribute), 11  
 Enter (*ahk.keys.KEYS* attribute), 11  
 ErrorStdOut (class in *ahk.directives*), 6  
 ESCAPE (*ahk.keys.KEYS* attribute), 11  
 escape\_sequence\_replace() (in module *ahk.utils*), 24  
 ExecutableNotFoundError, 21  
 exist() (*ahk.window.Window* property), 28  
 exists() (*ahk.window.Window* method), 28

## F

F1 (*ahk.keys.KEYS* attribute), 11  
 F10 (*ahk.keys.KEYS* attribute), 11  
 F11 (*ahk.keys.KEYS* attribute), 11  
 F12 (*ahk.keys.KEYS* attribute), 11  
 F13 (*ahk.keys.KEYS* attribute), 11  
 F14 (*ahk.keys.KEYS* attribute), 11  
 F15 (*ahk.keys.KEYS* attribute), 11  
 F16 (*ahk.keys.KEYS* attribute), 11  
 F17 (*ahk.keys.KEYS* attribute), 11  
 F18 (*ahk.keys.KEYS* attribute), 11  
 F19 (*ahk.keys.KEYS* attribute), 11  
 F2 (*ahk.keys.KEYS* attribute), 11  
 F20 (*ahk.keys.KEYS* attribute), 11  
 F21 (*ahk.keys.KEYS* attribute), 11  
 F22 (*ahk.keys.KEYS* attribute), 11  
 F23 (*ahk.keys.KEYS* attribute), 11  
 F24 (*ahk.keys.KEYS* attribute), 11  
 F3 (*ahk.keys.KEYS* attribute), 11  
 F4 (*ahk.keys.KEYS* attribute), 11  
 F5 (*ahk.keys.KEYS* attribute), 11  
 F6 (*ahk.keys.KEYS* attribute), 11  
 F7 (*ahk.keys.KEYS* attribute), 11  
 F8 (*ahk.keys.KEYS* attribute), 11  
 F9 (*ahk.keys.KEYS* attribute), 11  
 find\_window() (*ahk.window.AsyncWindowMixin* method), 25

find\_window() (*ahk.window.WindowMixin* method), 30  
 find\_window\_by\_class() (*ahk.window.AsyncWindowMixin* method), 25  
 find\_window\_by\_class() (*ahk.window.WindowMixin* method), 30  
 find\_window\_by\_text() (*ahk.window.AsyncWindowMixin* method), 25  
 find\_window\_by\_text() (*ahk.window.WindowMixin* method), 30  
 find\_window\_by\_title() (*ahk.window.AsyncWindowMixin* method), 25  
 find\_window\_by\_title() (*ahk.window.WindowMixin* method), 30  
 find\_windows() (*ahk.window.AsyncWindowMixin* method), 25  
 find\_windows() (*ahk.window.WindowMixin* method), 31  
 find\_windows\_by\_class() (*ahk.window.AsyncWindowMixin* method), 25  
 find\_windows\_by\_class() (*ahk.window.WindowMixin* method), 31  
 find\_windows\_by\_text() (*ahk.window.AsyncWindowMixin* method), 25  
 find\_windows\_by\_text() (*ahk.window.WindowMixin* method), 31  
 find\_windows\_by\_title() (*ahk.window.AsyncWindowMixin* method), 26  
 find\_windows\_by\_title() (*ahk.window.WindowMixin* method), 31  
 focus() (*ahk.window.Control* method), 26  
 from\_mouse\_position() (*ahk.window.AsyncWindow* class method), 24  
 from\_mouse\_position() (*ahk.window.Window* class method), 28  
 from\_pid() (*ahk.window.AsyncWindow* class method), 24  
 from\_pid() (*ahk.window.Window* class method), 28

## G

get() (*ahk.window.Control* method), 26  
 get() (*ahk.window.Window* method), 28  
 get\_active\_window() (*ahk.window.WindowMixin* method), 31  
 get\_class\_name() (*ahk.window.Window* method), 28  
 get\_mouse\_position()

- (*ahk.mouse.AsyncMouseMixin* method), 15
- `get_mouse_position()` (*ahk.mouse.MouseMixin* method), 16
- `get_pos()` (*ahk.window.AsyncWindow* method), 24
- `get_pos()` (*ahk.window.Window* method), 28
- `get_text()` (*ahk.window.Window* method), 28
- `get_title()` (*ahk.window.Window* method), 28
- `get_transparency()` (*ahk.window.AsyncWindow* method), 24
- `get_transparency()` (*ahk.window.Window* method), 28
- `get_volume()` (*ahk.sound.SoundMixin* method), 22
- ## H
- `has_focus()` (*ahk.window.Control* method), 26
- `height()` (*ahk.window.AsyncWindow* property), 24
- `height()` (*ahk.window.Window* property), 28
- `hide()` (*ahk.window.Window* method), 28
- `Hotkey` (class in *ahk.keyboard*), 8
- `hotkey()` (*ahk.keyboard.KeyboardMixin* method), 8
- `HotKeyInterval` (class in *ahk.directives*), 6
- `HotKeyModifierTimeout` (class in *ahk.directives*), 6
- ## I
- `image_search()` (*ahk.screen.AsyncScreenMixin* method), 18
- `image_search()` (*ahk.screen.ScreenMixin* method), 19
- `Include` (class in *ahk.directives*), 6
- `IncludeAgain` (class in *ahk.directives*), 6
- `InputLevel` (class in *ahk.directives*), 6
- `InstallKeybdHook` (class in *ahk.directives*), 6
- `InstallMouseHook` (class in *ahk.directives*), 6
- `is_active()` (*ahk.window.Window* method), 28
- `is_always_on_top()` (*ahk.window.AsyncWindow* method), 24
- `is_always_on_top()` (*ahk.window.Window* method), 28
- `is_maximized()` (*ahk.window.AsyncWindow* method), 24
- `is_maximized()` (*ahk.window.Window* method), 28
- `is_minimized()` (*ahk.window.AsyncWindow* method), 24
- `is_minimized()` (*ahk.window.Window* method), 28
- `is_minmax()` (*ahk.window.AsyncWindow* method), 24
- `is_minmax()` (*ahk.window.Window* method), 28
- ## J
- `JOY1` (*ahk.keys.KEYS* attribute), 12
- `Joy1` (*ahk.keys.KEYS* attribute), 12
- `JOY10` (*ahk.keys.KEYS* attribute), 12
- `Joy10` (*ahk.keys.KEYS* attribute), 12
- `JOY11` (*ahk.keys.KEYS* attribute), 12
- `Joy11` (*ahk.keys.KEYS* attribute), 12
- `JOY12` (*ahk.keys.KEYS* attribute), 12
- `Joy12` (*ahk.keys.KEYS* attribute), 12
- `JOY13` (*ahk.keys.KEYS* attribute), 12
- `Joy13` (*ahk.keys.KEYS* attribute), 13
- `JOY14` (*ahk.keys.KEYS* attribute), 12
- `Joy14` (*ahk.keys.KEYS* attribute), 13
- `JOY15` (*ahk.keys.KEYS* attribute), 12
- `Joy15` (*ahk.keys.KEYS* attribute), 13
- `JOY16` (*ahk.keys.KEYS* attribute), 12
- `Joy16` (*ahk.keys.KEYS* attribute), 13
- `JOY17` (*ahk.keys.KEYS* attribute), 12
- `Joy17` (*ahk.keys.KEYS* attribute), 13
- `JOY18` (*ahk.keys.KEYS* attribute), 12
- `Joy18` (*ahk.keys.KEYS* attribute), 13
- `JOY19` (*ahk.keys.KEYS* attribute), 12
- `Joy19` (*ahk.keys.KEYS* attribute), 13
- `JOY2` (*ahk.keys.KEYS* attribute), 12
- `Joy2` (*ahk.keys.KEYS* attribute), 13
- `JOY20` (*ahk.keys.KEYS* attribute), 12
- `Joy20` (*ahk.keys.KEYS* attribute), 13
- `JOY21` (*ahk.keys.KEYS* attribute), 12
- `Joy21` (*ahk.keys.KEYS* attribute), 13
- `JOY22` (*ahk.keys.KEYS* attribute), 12
- `Joy22` (*ahk.keys.KEYS* attribute), 13
- `JOY23` (*ahk.keys.KEYS* attribute), 12
- `Joy23` (*ahk.keys.KEYS* attribute), 13
- `JOY24` (*ahk.keys.KEYS* attribute), 12
- `Joy24` (*ahk.keys.KEYS* attribute), 13
- `JOY25` (*ahk.keys.KEYS* attribute), 12
- `Joy25` (*ahk.keys.KEYS* attribute), 13
- `JOY26` (*ahk.keys.KEYS* attribute), 12
- `Joy26` (*ahk.keys.KEYS* attribute), 13
- `JOY27` (*ahk.keys.KEYS* attribute), 12
- `Joy27` (*ahk.keys.KEYS* attribute), 13
- `JOY28` (*ahk.keys.KEYS* attribute), 12
- `Joy28` (*ahk.keys.KEYS* attribute), 13
- `JOY29` (*ahk.keys.KEYS* attribute), 12
- `Joy29` (*ahk.keys.KEYS* attribute), 13
- `JOY3` (*ahk.keys.KEYS* attribute), 12
- `Joy3` (*ahk.keys.KEYS* attribute), 13
- `JOY30` (*ahk.keys.KEYS* attribute), 12
- `Joy30` (*ahk.keys.KEYS* attribute), 13
- `JOY31` (*ahk.keys.KEYS* attribute), 12
- `Joy31` (*ahk.keys.KEYS* attribute), 13
- `JOY32` (*ahk.keys.KEYS* attribute), 12
- `Joy32` (*ahk.keys.KEYS* attribute), 13
- `JOY4` (*ahk.keys.KEYS* attribute), 12
- `Joy4` (*ahk.keys.KEYS* attribute), 13
- `JOY5` (*ahk.keys.KEYS* attribute), 12
- `Joy5` (*ahk.keys.KEYS* attribute), 13
- `JOY6` (*ahk.keys.KEYS* attribute), 12
- `Joy6` (*ahk.keys.KEYS* attribute), 13

JOY7 (*ahk.keys.KEYS attribute*), 12  
 Joy7 (*ahk.keys.KEYS attribute*), 13  
 JOY8 (*ahk.keys.KEYS attribute*), 12  
 Joy8 (*ahk.keys.KEYS attribute*), 13  
 JOY9 (*ahk.keys.KEYS attribute*), 12  
 Joy9 (*ahk.keys.KEYS attribute*), 13

## K

key\_down() (*ahk.keyboard.KeyboardMixin method*), 8  
 key\_press() (*ahk.keyboard.AsyncKeyboardMixin method*), 7  
 key\_press() (*ahk.keyboard.KeyboardMixin method*), 8  
 key\_release() (*ahk.keyboard.KeyboardMixin method*), 9  
 key\_state() (*ahk.keyboard.AsyncKeyboardMixin method*), 7  
 key\_state() (*ahk.keyboard.KeyboardMixin method*), 9  
 key\_up() (*ahk.keyboard.KeyboardMixin method*), 9  
 key\_wait() (*ahk.keyboard.AsyncKeyboardMixin method*), 7  
 key\_wait() (*ahk.keyboard.KeyboardMixin method*), 9  
 KeyboardMixin (*class in ahk.keyboard*), 8  
 KeyHistory (*class in ahk.directives*), 6  
 KEYS (*class in ahk.keys*), 10  
 kill() (*ahk.window.Window method*), 28

## L

LAlt (*ahk.keys.KEYS attribute*), 13  
 LControl (*ahk.keys.KEYS attribute*), 13  
 LCtrl (*ahk.keys.KEYS attribute*), 13  
 LEFT (*ahk.keys.KEYS attribute*), 13  
 Left (*ahk.keys.KEYS attribute*), 14  
 LEFT\_ALT (*ahk.keys.KEYS attribute*), 13  
 LEFT\_CONTROL (*ahk.keys.KEYS attribute*), 13  
 LEFT\_SHIFT (*ahk.keys.KEYS attribute*), 13  
 LEFT\_WIN (*ahk.keys.KEYS attribute*), 13  
 LShift (*ahk.keys.KEYS attribute*), 14  
 LWin (*ahk.keys.KEYS attribute*), 14

## M

make\_logger() (*in module ahk.utils*), 24  
 MaxHotkeysPerInterval (*class in ahk.directives*), 6  
 maximize() (*ahk.window.Window method*), 28  
 MAXIMIZED (*ahk.window.Window attribute*), 27  
 maximized() (*ahk.window.Window property*), 28  
 MaxMem (*class in ahk.directives*), 6  
 MaxThreads (*class in ahk.directives*), 6  
 MaxThreadsBuffer (*class in ahk.directives*), 7  
 MaxThreadsPerHotkey (*class in ahk.directives*), 7  
 MenuMaskKey (*class in ahk.directives*), 7  
 minimize() (*ahk.window.Window method*), 28

MINIMIZED (*ahk.window.Window attribute*), 27  
 minimized() (*ahk.window.Window property*), 28  
 mouse\_drag() (*ahk.mouse.MouseMixin method*), 16  
 mouse\_move() (*ahk.mouse.MouseMixin method*), 16  
 mouse\_position() (*ahk.mouse.AsyncMouseMixin property*), 15  
 mouse\_position() (*ahk.mouse.MouseMixin property*), 17  
 mouse\_speed() (*ahk.mouse.MouseMixin property*), 17  
 mouse\_wheel() (*ahk.mouse.MouseMixin method*), 17  
 MouseMixin (*class in ahk.mouse*), 15  
 move() (*ahk.window.Control method*), 26  
 move() (*ahk.window.Window method*), 28

## N

NoEnv (*class in ahk.directives*), 7  
 non\_max\_non\_min() (*ahk.window.AsyncWindow property*), 24  
 non\_max\_non\_min() (*ahk.window.Window property*), 29  
 NON\_MIN\_NON\_MAX (*ahk.window.Window attribute*), 27  
 NoTrayIcon (*class in ahk.directives*), 7  
 NUM\_LOCK (*ahk.keys.KEYS attribute*), 14  
 NumLock (*ahk.keys.KEYS attribute*), 14  
 NUMPAD0 (*ahk.keys.KEYS attribute*), 14  
 Numpad0 (*ahk.keys.KEYS attribute*), 14  
 NUMPAD1 (*ahk.keys.KEYS attribute*), 14  
 Numpad1 (*ahk.keys.KEYS attribute*), 14  
 NUMPAD2 (*ahk.keys.KEYS attribute*), 14  
 Numpad2 (*ahk.keys.KEYS attribute*), 14  
 NUMPAD3 (*ahk.keys.KEYS attribute*), 14  
 Numpad3 (*ahk.keys.KEYS attribute*), 14  
 NUMPAD4 (*ahk.keys.KEYS attribute*), 14  
 Numpad4 (*ahk.keys.KEYS attribute*), 14  
 NUMPAD5 (*ahk.keys.KEYS attribute*), 14  
 Numpad5 (*ahk.keys.KEYS attribute*), 14  
 NUMPAD6 (*ahk.keys.KEYS attribute*), 14  
 Numpad6 (*ahk.keys.KEYS attribute*), 14  
 NUMPAD7 (*ahk.keys.KEYS attribute*), 14  
 Numpad7 (*ahk.keys.KEYS attribute*), 14  
 NUMPAD8 (*ahk.keys.KEYS attribute*), 14  
 Numpad8 (*ahk.keys.KEYS attribute*), 14  
 NUMPAD9 (*ahk.keys.KEYS attribute*), 14  
 Numpad9 (*ahk.keys.KEYS attribute*), 14  
 NUMPAD\_ADD (*ahk.keys.KEYS attribute*), 14  
 NUMPAD\_DEL (*ahk.keys.KEYS attribute*), 14  
 NUMPAD\_DIV (*ahk.keys.KEYS attribute*), 14  
 NUMPAD\_DOT (*ahk.keys.KEYS attribute*), 14  
 NUMPAD\_ENTER (*ahk.keys.KEYS attribute*), 14  
 NUMPAD\_MULT (*ahk.keys.KEYS attribute*), 14  
 NUMPAD\_SUB (*ahk.keys.KEYS attribute*), 14  
 NumpadAdd (*ahk.keys.KEYS attribute*), 14

NumpadDel (*ahk.keys.KEYS attribute*), 14  
 NumpadDiv (*ahk.keys.KEYS attribute*), 14  
 NumpadDot (*ahk.keys.KEYS attribute*), 14  
 NumpadEnter (*ahk.keys.KEYS attribute*), 15  
 NumpadMult (*ahk.keys.KEYS attribute*), 15  
 NumpadSub (*ahk.keys.KEYS attribute*), 15

## P

perform() (*ahk.autohotkey.ActionChain method*), 5  
 Persistent (*class in ahk.directives*), 7  
 pixel\_get\_color() (*ahk.screen.ScreenMixin method*), 19  
 pixel\_search() (*ahk.screen.AsyncScreenMixin method*), 18  
 pixel\_search() (*ahk.screen.ScreenMixin method*), 19  
 position() (*ahk.window.AsyncWindow property*), 24  
 position() (*ahk.window.Control property*), 26  
 position() (*ahk.window.Window property*), 29

## R

RAlt (*ahk.keys.KEYS attribute*), 15  
 RControl (*ahk.keys.KEYS attribute*), 15  
 RCtrl (*ahk.keys.KEYS attribute*), 15  
 rect() (*ahk.window.AsyncWindow property*), 24  
 rect() (*ahk.window.Window property*), 29  
 redraw() (*ahk.window.Window method*), 29  
 render\_template() (*ahk.script.ScriptEngine method*), 21  
 resolve\_button() (*in module ahk.mouse*), 17  
 restore() (*ahk.window.Window method*), 29  
 RIGHT (*ahk.keys.KEYS attribute*), 15  
 Right (*ahk.keys.KEYS attribute*), 15  
 RIGHT\_ALT (*ahk.keys.KEYS attribute*), 15  
 right\_click() (*ahk.mouse.MouseMixin method*), 17  
 RIGHT\_CONTROL (*ahk.keys.KEYS attribute*), 15  
 RIGHT\_SHIFT (*ahk.keys.KEYS attribute*), 15  
 RIGHT\_WIN (*ahk.keys.KEYS attribute*), 15  
 RShift (*ahk.keys.KEYS attribute*), 15  
 run\_script() (*ahk.autohotkey.ActionChain method*), 5  
 run\_script() (*ahk.script.AsyncScriptEngine method*), 20  
 run\_script() (*ahk.script.ScriptEngine method*), 22  
 running() (*ahk.keyboard.Hotkey property*), 8  
 RWin (*ahk.keys.KEYS attribute*), 15

## S

ScreenMixin (*class in ahk.screen*), 19  
 ScriptEngine (*class in ahk.script*), 21  
 SCROLL\_LOCK (*ahk.keys.KEYS attribute*), 15  
 ScrollLock (*ahk.keys.KEYS attribute*), 15  
 send() (*ahk.keyboard.KeyboardMixin method*), 9  
 send() (*ahk.window.Control method*), 26

send() (*ahk.window.Window method*), 29  
 send\_event() (*ahk.keyboard.KeyboardMixin method*), 9  
 send\_input() (*ahk.keyboard.KeyboardMixin method*), 10  
 send\_play() (*ahk.keyboard.KeyboardMixin method*), 10  
 send\_raw() (*ahk.keyboard.KeyboardMixin method*), 10  
 set() (*ahk.window.Window method*), 29  
 set\_always\_on\_top() (*ahk.window.Window method*), 29  
 set\_capslock\_state() (*ahk.keyboard.KeyboardMixin method*), 10  
 set\_position() (*ahk.window.Window method*), 29  
 set\_title() (*ahk.window.Window method*), 29  
 set\_transparency() (*ahk.window.AsyncWindow method*), 24  
 set\_transparency() (*ahk.window.Window method*), 29  
 set\_volume() (*ahk.sound.SoundMixin method*), 22  
 SHIFT (*ahk.keys.KEYS attribute*), 15  
 Shift (*ahk.keys.KEYS attribute*), 15  
 show() (*ahk.window.Window method*), 29  
 SingleInstance (*class in ahk.directives*), 7  
 sleep() (*ahk.autohotkey.ActionChain method*), 5  
 sound\_beep() (*ahk.sound.SoundMixin method*), 23  
 sound\_get() (*ahk.sound.SoundMixin method*), 23  
 sound\_play() (*ahk.sound.SoundMixin method*), 23  
 sound\_set() (*ahk.sound.SoundMixin method*), 23  
 SoundMixin (*class in ahk.sound*), 22  
 SPACE (*ahk.keys.KEYS attribute*), 15  
 start() (*ahk.keyboard.Hotkey method*), 8  
 stop() (*ahk.keyboard.Hotkey method*), 8

## T

TAB (*ahk.keys.KEYS attribute*), 15  
 Tab (*ahk.keys.KEYS attribute*), 15  
 text() (*ahk.window.Control property*), 26  
 text() (*ahk.window.Window property*), 29  
 title() (*ahk.window.AsyncWindow property*), 24  
 title() (*ahk.window.Window property*), 29  
 to\_bottom() (*ahk.window.Window method*), 29  
 to\_top() (*ahk.window.Window method*), 29  
 transparent() (*ahk.window.AsyncWindow property*), 24  
 transparent() (*ahk.window.Window property*), 29  
 type() (*ahk.keyboard.KeyboardMixin method*), 10

## U

UP (*ahk.keys.KEYS attribute*), 15  
 Up (*ahk.keys.KEYS attribute*), 15  
 UseHook (*class in ahk.directives*), 7

## W

`wait()` (*ahk.window.Window* method), 29  
`wait_active()` (*ahk.window.Window* method), 29  
`wait_close()` (*ahk.window.Window* method), 29  
`wait_not_active()` (*ahk.window.Window* method),  
30  
`Warn` (*class in ahk.directives*), 7  
`wheel_down()` (*ahk.mouse.MouseMixin* method), 17  
`wheel_up()` (*ahk.mouse.MouseMixin* method), 17  
`width()` (*ahk.window.AsyncWindow* property), 24  
`width()` (*ahk.window.Window* property), 30  
`WIN` (*ahk.keys.KEYS* attribute), 15  
`Win` (*ahk.keys.KEYS* attribute), 15  
`win_get()` (*ahk.window.AsyncWindowMixin* method),  
26  
`win_get()` (*ahk.window.WindowMixin* method), 31  
`win_set()` (*ahk.window.WindowMixin* method), 31  
`WinActivateForce` (*class in ahk.directives*), 7  
`Window` (*class in ahk.window*), 27  
`WindowMixin` (*class in ahk.window*), 30  
`WindowNotFoundError`, 31  
`windows()` (*ahk.window.AsyncWindowMixin* method),  
26  
`windows()` (*ahk.window.WindowMixin* method), 31