pyahk Documentation

Release 0.2.2

Ed Blake

May 03, 2017

Contents

Description Python ctypes wrapper around AutoHotKey dll.

License BSD-style, see LICENSE.txt

Author Ed Blake <kitsu.eb@gmail.com>

Date Dec 30 2014

Revision 14

CHAPTER 1

Introduction

AutoHotKey is a powerful task automator with a terrible scripting language built-in. Python is a powerful scripting language with old, unmaintained, and incomplete automation modules. Combining the two creates a powerful automation tool with a powerful scripting back-end with access to all the power of the Python standard library.

This is made possible by the amazing Python ctypes module and the AutoHotKey.dll project. Together they allow exchange of data between both scripting engines, and the execution of AHK commands from Python.

CHAPTER 2

Requirements

- Written for Python2.7, not currently py3k compatible.
- Written against AutoHotkey_H ANSI 32-bit Version 1.1.8.1 (on WinXP).
- A copy of the ANSI 32-bit dll must be provided either in the system location of your version of Windows, or in the same folder as the ahk.py file. (The required dll is not provided as part of this distribution, see the AutoHotKey.dll site for download instructions (alternate link AutoHotKey_H).)

CHAPTER 3

Installation

Get the version from PyPI using pip:

```
pip install pyahk
```

Or download the latest commit from bitbucket and:

```
`python setup.py install`
```

CHAPTER 4

Testing

A helper script “runtests.py” is provided in the project root to run the entire test suite. Runnable test scripts are provided for each sub-module in the test folder. Tests require Michael Foord’s mock library.

CHAPTER 5

Usage

First import the ahk module:

```
import ahk
```

Lower level function wrappers provide more direct access to the underlaying dll:

```
ahk.start() # Ititializes a new script thread
ahk.ready() # Waits until status is True
ahk.execute('a := WinActive("Notepad")') # Sets a to the return value of WinActive
print ahk.get('a') # prints the HWND of the found window or 0x0 as a string
```

Object wrappers are also provided which have a somewhat cleaner interface:

```
script = ahk.Script() # Initializes with start and ready commands as above
a = script.winActive("Notepad") # Sets a to the return value of the method
print a # prints the HWND of the found window as an int, or None
script.variable('b', float) # Creates a transparent variable attribute
ahk.execute("Random, b, 0.0, 1.0") # Stores value in `b`
print 100*script.b # b is retrieved and converted to its saved type (float)
```

See the Docs for further details.

CHAPTER 6

ToDo

- Re-write script.Function to use the now working ahk.call function.
- Extend setup script with options to download/install the correct dll?
- Add remaining Control commands to Control class.
- Add doc-tests?
- Extend Script class with something to replace ahk.execute (maybe some kind of subroutine wrapper?).
- Add examples directory.
- Add optional unicode support.

CHAPTER 7

Sections

Function wrappers

These are lower level wrappers around the raw dll functions. Functions have been combined and simplified where reasonable. Only the minimal extraction/conversion has been preformed on c return types.

functions

- *start()*
- *ready()*
- *add_lines()*
- *execute()*
- *jump()*
- *call()*
- *post()*
- *set()*
- *get()*
- *terminate()*
- *reload()*
- *find_func()*
- *find_label()*
- *pause()*
- *exec_line()*

ahk.**start** (*filename=None*, *script=''*, *options=''*, *params=''*)
Wrapper around ahkdll and ahktextdll.

Start a new ahk thread from file or a string. Defaults to an empty script with no options or params. Filename is preferred over script if provided.

Note: Using any option besides ahk.start() seems not to work. Specifying a file doesn't cause it to run, passing a string doesn't cause it to be executed?

Returns Thread handle for created instance (see thread functions).

---

ahk.**ready** (*nowait=False*, *retries=None*)
Wrapper around ahkReady.

Returns True if ahk is ready to use. By default this polls the dll function until it is ready. By calling with nowait=True the immediate result is returned instead. By calling with retries > 1 state will be checked at most retries times.

---

ahk.**add_lines** (*script=''*, *filename=None*, *duplicates=False*, *ignore=True*)
Wrapper around addFile and addScript.

Adds lines to the running script from file or string. Lines added from string are evaluated immediately, lines added from file are not evaluated. Defaults to an empty script, no duplicates, and ignore errors. Filename is preferred over script if provided.

Returns Pointer address to first line in added script (see execute_line).

---

ahk.**execute** (*script*)
Wrapper around ahkExec.

Execute provided ahk commands. No lines are added to the active script.

Returns True if successful, else False.

---

ahk.**jump** (*label*, *nowait=False*)
Wrapper around ahkLabel.

GoSub/GoTo like function, branch to labeled location. Defaults to nowait=False, i.e. GoSub mode. Using nowait=True, i.e. GoTo mode, is unreliable and may fail entirely.

Returns True if label exists, else False.

---

ahk.**call** (*func*, **args*)
Wrapper around ahkFunction.

Call the indicated function.

Returns Result of function call as a string.

---

ahk.**post**(*func*, **args*)

Wrapper around ahkPostFunction.

Call the indicated function but discard results.

Returns True if function exists, else False.

ahk.**set**(*name*, *value*)

Wrapper around ahkassign.

Assigns the string *value* to the variable *name*.

Returns True for success and False for failure.

ahk.**get**(*name*, *pointer=False*)

Wrapper around ahkgetvar.

Get the string value of a variable from ahk. Call with pointer=True to request a reference to the variable.

Returns A string representing the value, or a c_char_p.

ahk.**terminate**(*timeout=1*)

Wrapper around ahkTerminate.

Terminate the script, removing all hotkeys and hotstrings. The default timeout is 1ms, must be positive > 0.

ahk.**reload**()

Wrapper around ahkReload.

Terminates and restarts the script.

ahk.**find_func**(*name*)

Wrapper around ahkFindFunc.

Warning: The following doesn't seem to work, any advice welcome.

Get a pointer address to the named function. To use this you must first create a ctypes CFUNCTYPE prototype:

```
proto = ctypes.CFUNCTYPE(ctypes.c_int, ctypes.c_char_p)
```

Then call the prototype with the address as an argument to get a function:

```
func = proto(address)
```

Now it can be called:

```
result = func(5)
```

Returns The address of the function as an integer.

`ahk.`**`find_label`**(*name*)

Wrapper around ahkFindLabel.

Get a pointer address to a label...

Returns The address of the label as an integer.

`ahk.`**`pause`**(*pause_=True*)

Wrapper around ahkPause.

Pause or unpause the script. Calling with pause_=True pauses, pause_=False un-pauses. Calling with pause_=None just reports current state without changing it.

Returns True if the script is paused, else False.

`ahk.`**`exec_line`**(*line=None*, *mode=3*, *wait=False*)

Wrapper around ahkExecuteLine.

Execute starting from the provided line address. If line=None the address of the first line will be returned. Four modes of execution are available:

0.No execution, but the next line address is returned.

1.Run until a return statement is found.

2.Run until the end of the current block.

3.(default) execute only one line.

Setting wait=True will block until end of execution.

Returns A line pointer address.

Function and Script wrappers

These are higher-level wrappers representing an AHK function and an AHK script. A Function object instance is transparently callable from either Python or AHK. A Script object handles initialization automatically, provides transparent variable access, and convenience wrappers around some AHK commands.

classes

- *`Function`*
- ***`Script`***
  - *`Script.variable()`*
  - *`Script.function()`*
  - *`Script.send()`*
  - *`Script.click()`*
  - *`Script.winActivate()`*
  - *`Script.winActive()`*
  - *`Script.winExist()`*

- *Script.waitActive()*
- *Script.waitWindow()*
- *Script.convert_color()*
- *Script.getPixel()*
- *Script.waitPixel()*
- *Script.message()*
- *Script.msgResult()*

Function

class ahk.**Function**(*name*, *result*, *args*, *body*)

Object wrapper around ahk functions

Called Functions are automatically transform to their result by calling the provided result function on the return value from the ahk function call.

Parameters

- **name** (*str*) – The name of the function to wrap.
- **result** (*callable type (default=str)*) – Type of the expected result.
- **args** (*str (default='()')*) – The argument definition of the function.
- **body** (*str (default='')*) – The body of the function (excluding braces).

Script

class ahk.**Script**(*script=''*, *filename=None*)

Wrapper around ahk script commands.

Initializes the ahk script engine just like calling the low-level function ahk.start followed by ahk.ready.

variable(*name*, *kind=<type 'str'>*, *value=''*)

Create a new ahk variable wrapper.

Wrapped variables are tracked by the instance and can be accessed as object attributes. Variables are automatically transformed to their declared kind by calling their kind with their ahk string value as the first argument (just like normal python type conversions).

Variable wrappers are already provided for the special ahk Clipboard and ErrorLevel variables (Note the case).

Note: Variables are not allowed to start with '_' the underscore char!

Parameters

- **name** (*str*) – Then name of the new variable.
- **kind** (*callable type (default=str)*) – Type of the variable (cast on get).
- **value** (*match kind or str (default='')*) – Initial value of the variable.

Raises AttributeError if the provided name already exists in the instance as either a variable or an attribute.

function (*name*, *result=<type 'str'>*, *args='()'*, *body=''*)

Create a new ahk function wrapper.

Wrapped functions are tracked by the instance and can be accessed as object attributes. Functions are automatically transform to their result by calling the provided result function on the return value from the ahk function call.

Note: Function names are not allowed to start with '_' the underscore char!

Parameters

- **name** (*str*) – The name of the function to wrap.
- **result** (*callable type (default=str)*) – Type of the expected result.
- **args** (*str (default='()')*) – The argument definition of the function.
- **body** (*str (default='')*) – The body of the function (excluding braces).

Raises AttributeError if the indicated name is already used.

Returns Function wrapper object.

send (*keys*, *mode='SendInput'*)

Convenience wrapper to send input to the active window.

Sends a ahk formatted series of keystrokes to the active window. See AHK docs for more details.

Parameters

- **keys** (*str*) – The keys to be sent.
- **mode** (*str*) – The ahk command used to send keys. Valid modes:
  - Send
  - SendRaw
  - SendInput
  - SendPlay
  - SendEvent

click (*button=''*, *count=1*, *x=None*, *y=None*)

Convenience wrapper to the ahk click function.

Send a mouse click of any type to any coordinate with optional repeats. See AHK docs for more details. The button argument can actually take a number of options:

- *blank* - Simple primary click.
- right - Auxiliary button click.
- wheelup - Send mouse wheel scroll event.
- down - Press but don't release the primary button.
- rel[ative] - Interpret coordinates in relative mode.
- etc.

Button options can be composed when not mutually exclusive (e.g. "right down relative").

Use count=0 with x and y to move the mouse without clicking.

Parameters

- **button** (*str (default="")*) – The mouse button(s) to click.
- **count** (*int (default 1)*) – Number of times to repeat the click.
- **x** (*int*) – Mouse x-coord.
- **y** (*int*) – Mouse y-coord.

winActivate (*title='', text='', extitle='', extext='', bottom=False*)

Convenience wrapper for ahk WinActivate commands.

Used to set a window with provided parameters as active. If all parameters are empty the *last found* window is activated. See AHK docs for more details.

Parameters

- **title** (*str (default="")*) – Partial window title text to match.
- **text** (*str (default="")*) – Partial window text to match.
- **extitle** (*str (default="")*) – Partial window title text to avoid.
- **extext** (*str (default="")*) – Partial window text to avoid.
- **bottom** (*bool (default=False)*) – Whether to act on the bottom-most window.

winActive (*title='', text='', extitle='', extext=''*)

Convenience wrapper for ahk IfWinActive command.

Used to check if a window with provided parameters is active. If all parameters are empty the *last found* window is checked. See AHK docs for more details.

Parameters

- **title** (*str (default="")*) – Partial window title text to match.
- **text** (*str (default="")*) – Partial window text to match.
- **extitle** (*str (default="")*) – Partial window title text to avoid.
- **extext** (*str (default="")*) – Partial window text to avoid.

Returns The found window's HWND or None.

winExist (*title='', text='', extitle='', extext=''*)

Convenience wrapper for ahk IfWinExist command.

Used to check if a window with provided parameters exists. If all parameters are empty the *last found* window is checked. See AHK docs for more details.

Parameters

- **title** (*str (default="")*) – Partial window title text to match.
- **text** (*str (default="")*) – Partial window text to match.
- **extitle** (*str (default="")*) – Partial window title text to avoid.
- **extext** (*str (default="")*) – Partial window text to avoid.

Returns The found window's HWND or None.

waitActive (*title='', text='', timeout=5, extitle='', extext='', deactivate=False*)

Convenience wrapper for ahk WinWaitActive command.

Used to wait until a window matching the given parameters is activated. See AHK docs for more details.

Parameters

- **title** (*str (default="")*) – Partial window title text to match.
- **text** (*str (default="")*) – Partial window text to match.
- **timeout** (*int or None (default=5)*) – How long in seconds to wait for the window to activate. Use None to wait indefinitely.
- **extitle** (*str (default="")*) – Partial window title text to avoid.
- **extext** (*str (default="")*) – Partial window text to avoid.
- **deactivate** (*bool (default=False)*) – Toggle between WinWaitActive and WinWaitNotActive.

Returns The True if matching window is activated, else False.

waitWindow(*title='', text='', timeout=5, extitle='', extext='', closed=False*)

Convenience wrapper for ahk WinWait command.

Used to wait until a window matching the given parameters exists. See AHK docs, AHK docs for more details.

Parameters

- **title** (*str (default="")*) – Partial window title text to match.
- **text** (*str (default="")*) – Partial window text to match.
- **timeout** (*int or None (default=5)*) – How long in seconds to wait for the window to activate. Use None to wait indefinitely.
- **extitle** (*str (default="")*) – Partial window title text to avoid.
- **extext** (*str (default="")*) – Partial window text to avoid.
- **closed** (*bool (default=False)*) – Toggle between WinWait and WinWaitClose.

Returns The True if matching window is activated, else False.

convert_color(*color*)

Convert ahk color returned as a hex string to tuple of ints.

getPixel(*x=0, y=0, opt='RGB', screen=True*)

Convenince wrapper around ahk PixelGetColor

Gets the pixel color at the indicated coordinates. See AHK docs for more details.

Parameters

- **x** (*int*) – The pixel x coordinate (relative to screen).
- **y** (*int*) – The pixel y coordinate (relative to screen).
- **opt** (*str (default='RGB')*) – Space separated color picking options.
- **screen** (*bool (default=True)*) – Flag to use screen or relative coordinates.

waitPixel(*x=0, y=0, color=None, threshold=0.01, interval=0.5, timeout=False*)

Wait until the pixel at given coords changes color.

This function can wait until the indicated pixel *is* a color, or until it changes color. If a color is not provided the current color of the pixel when the function starts is stored and compared at a regular interval until it changes or until timeout.

Parameters

- **x** (*int*) – The pixel x coordinate (relative to screen).
- **y** (*int*) – The pixel y coordinate (relative to screen).
- **color** (*tuple(int r, int g, int b) or None*) – The color to wait for.
- **threshold** (*float*) – Error factor allowed for determining color match.
- **interval** (*float*) – How often the pixel color is checked.
- **timeout** (*float or None*) – How long to wait for a color match.

Returns True if pixel changed, False if timeout.

message (*text='Alert'*, *title='Alert'*, *options=0*, *timeout=None*)

Convenience wrapper to the ahk msgbox function.

Displays a message box with buttons. See AHK docs for more details.

Parameters

- **text** (*str (default='Alert')*) – The body text of the msgbox.
- **title** (*str (default='Alert')*) – The title text of the msgbox.
- **options** (*int (default=0)*) – Button flags bit-field.
- **timeout** (*int or None (default=None)*) – Optional timeout, dialog is closed after timeout.

msgResult (*name='OK'*)

Convenience wrapper to the ahk ifMsgBox function.

Get the clicked status of a button on the last MsgBox. See AHK docs for more details.

Parameters **name** (*str (default='OK')*) – The name of the button to check.

Returns True if button was pressed, None if timeout, False otherwise.

Control wrapper

This wrapper allows simple access to another application's controls (i.e. buttons and other widgets).

classes

- *Control*
  - *Control.set_delay()*
  - *Control.click()*
  - *Control.send()*
  - *Control.setText()*
  - *Control.get_choices()*
  - *Control.get_chosen()*
  - *Control.choose()*
  - *Control.is_checked()*
  - *Control.check()*

Control

class `ahk.`**`Control`**(*script*, *title=''*, *text=''*, *extitle=''*, *extext=''*, *store=True*)

Wrapper around ahk window control commands.

Initialize a new Control object.

Stores information about the window who's controls you will manipulate.

Parameters

- **`script`** (*`ahk.Script instance`*) – Script object to use internally.
- **`title`** (*`str (default="")`*) – Partial window title text to match.
- **`text`** (*`str (default="")`*) – Partial window text to match.
- **`extitle`** (*`str (default="")`*) – Partial window title text to avoid.
- **`extext`** (*`str (default="")`*) – Partial window text to avoid.
- **`store`** (*`bool`*) – Whether to cache the found window.

Raises NameError if store=True but a matching window can't be found.

`set_delay`(*control=''*, *key=''*)

Set the control or key delay used by this Control object.

Either the control delay, key delay, or both may be set. Setting either to *None* will disable the local delay setting. See AHK docs for more details.

Parameters

- **`control`** (*`float or None`*) – The delay to use when modifying a control.
- **`key`** (*`float or None`*) – The delay to use for `send`-ing key events.

`click`(**args*, ***kwargs*)

Sends mouse input to a control.

The control arg can be a class name, window handle, or text string. Alternately pos as a (x, y) tuple may be provided indicating the mouse coordinates relative to the top-left corner of the application window in which to click. If both control and pos are provided pos is interpreted relative to the indicated control.

Default is a single left click, but you can send left/right/middle, wheel up or down left or right with a repeat count. See AHK docs for more details.

Parameters

- **`control`** (*`str`*) – The class, HWND, or text of the control to click.
- **`pos`** (*`tuple (indexable)`*) – The relative coordinates of at which to click.
- **`button`** (*`str`*) – The name of the button to be clicked.
- **`count`** (*`int`*) – Number of times it should be clicked.
- **`options`** (*`str`*) – Optional arguments.

`send`(**args*, ***kwargs*)

Send keystrokes to a control.

See AHK docs for more details.

Parameters

- **`control`** (*`str`*) – The class, HWND, or text of the control to send to.

- **keys** (*str*) – Keystrokes in the same format as ahk.Script.send.
- **raw** (*bool (default False)*) – Toggle raw input mode.

setText (**args*, ***kwargs*)

Set the text content of a control.

See AHK docs for more details.

Parameters

- **control** (*str*) – The class, HWND, or text of the control to be changed.
- **value** (*str*) – New text to substitute into the control.

get_choices (*control=''*)

Retrieve the available values from a ComboBox.

Parameters **control** (*str*) – The class, HWND, or text of the control to be checked.

Returns A list of value strings (in order).

get_chosen (*control=''*)

Retrieve the selected value from a ComboBox.

Parameters **control** (*str*) – The class, HWND, or text of the control to be checked.

Returns Selected value as a string.

choose (**args*, ***kwargs*)

Pick an item from a listbox or combobox.

See AHK docs for more details.

Parameters

- **control** (*str*) – The class, HWND, or text of the control to be changed.
- **value** (*str or int (default="")*) – Partial text of choice or choice index (starting from 1).

is_checked (*control=''*)

Check if a checkbox control is checked.

Parameters **control** (*str*) – The class, HWND, or text of the control to be checked.

Returns True if it was checked, else False.

check (**args*, ***kwargs*)

Pick an item from a listbox or combobox.

The state can be set to checked (True), unchecked (False), or toggle (None), default is toggle. See AHK docs for more details.

Parameters

- **control** (*str*) – The class, HWND, or text of the control to be changed.
- **state** (*bool or None*) – The desired checked state.

Index