ocvbot

Release 0.1

Austin Kelley

May 21, 2020

Table of Contents

ocvbot

1.1 ocvbot package

1.1.1 Submodules

1.1.2 ocvbot.behavior module

ocvbot.behavior.**drop_item_rapid** (*item*)
Drops an instance of the given inventory item.

ocvbot.behavior.**login** ( )
Automatically logs the client in using credentials specified in a file.

ocvbot.behavior.**wait_rand** (*chance*, *wait_min*, *wait_max*)
Random chance to do nothing for the specified period of time.

1.1.3 ocvbot.input module

ocvbot.input.**click** (*button='left'*, *sleep_before_min=0*, *sleep_before_max=500*, *sleep_after_min=0*, *sleep_after_max=500*, *click_duration_min=0*, *click_duration_max=100*)
Clicks the left or right mouse button, waiting before and after for a randomized period of time.

Parameters
- **button** (*str*) – Which mouse button to click, default is left.
- **sleep_before_min** (*int*) – Minimum number of miliseconds to wait before clicking, default is 0.
- **sleep_before_max** (*int*) – Maximum number of miliseconds to wait before clicking, default is 500.
- **sleep_after_min** (*int*) – Minimum number of miliseconds to wait after clicking, default is 0.
- **sleep_after_max** (*int*) – Maximum number of miliseconds to wait after clicking, default is 500.
- **click_duration_min** (*int*) – Minimum number of miliseconds to hold down the mouse button, default is 0.
- **click_duration_max** (*int*) – Maximum number of miliseconds to hold down the mouse button, default is 100.

Returns Always returns 0.

ocvbot.input.**click_coord** (*left*, *top*, *width*, *height*, *button='left'*)

Clicks within the provided coordinates. If width and height are both 0, then this function will click in the exact same location every time.

Parameters
- **left** (*int*) – The left edge (x) of the coordinate space to click within.
- **top** (*int*) – The top edge (y) of the coordinate space to click within.
- **width** (*int*) – The x width of the coordinate space to randomize the click within.
- **height** (*int*) – The y height of the coordinate space to randomize the click within.
- **button** (*str*) – The mouse button to click with, default is 'left'.

Returns Always returns '0'.

ocvbot.input.**double_hotkey_press** (*key1*, *key2*, *timedown_min=5*, *timedown_max=190*, *before_min=500*, *before_max=1000*, *after_min=500*, *after_max=1000*)

Performs a two-key hotkey shortcut, such as Ctrl-c for copying text.

Parameters
- **key1** (*str*) – The first hotkey used in the two-hotkey shortcut, sometimes also called the modifier key.
- **key2** (*str*) – The second hotkey used in the two-hotkey shortcut.
- **timedown_min** (*int*) – See keypress()'s docstring, default is 5.
- **timedown_max** (*int*) – See keypress()'s docstring, default is 190.
- **before_min** (*int*) – See keypress()'s docstring, default is 500.
- **before_max** (*int*) – See keypress()'s docstring, default is 1000.
- **after_min** (*int*) – See keypress()'s docstring, default is 500.
- **after_max** (*int*) – See keypress()'s docstring, default is 1000.

Returns Always returns 0.

ocvbot.input.**keypress** (*key*, *timedown_min=1*, *timedown_max=180*, *sleep_before_min=50*, *sleep_before_max=1000*, *sleep_after_min=50*, *sleep_after_max=1000*)

Holds down the specified key for a random period of time. All values are in miliseconds.

Parameters
- **key** (*str*) – The key on the keyboard to press, according to PyAutoGUI.
- **timedown_min** (*int*) – The shortest time the key can be down, default is 1.
- **timedown_max** (*int*) – The longest time the key can be down, default is 180.
- **sleep_before_min** (*int*) – The shortest time to wait before pressing the key down, default is 50.
- **sleep_before_max** (*int*) – The longest time to wait before pressing the key down, default is 1000.
- **sleep_after_min** (*int*) – The shortest time to wait after releasing the key, default is 50.
- **sleep_after_max** (*int*) – The longest time to wait after releasing the key, default is 1000.

Returns Always returns 0.

ocvbot.input.**move_away** (*direction='left'*)

Moves the mouse to a random spot on right half or the left half of the client window, away from wherever it clicked, to prevent tooltips from interfering with the script.

Parameters **direction** – Which side of the screen to move the mouse, by default this function randomly selects between left and right.

Returns Always returns 0.

ocvbot.input.**move_duration** (*move_duration_min=50*, *move_duration_max=1500*)

Randomizes the amount of time the mouse cursor takes to move to a new location. Input arguments are in miliseconds but return value is in seconds.

Parameters

- **move_duration_min** (*int*) – Minimum number of miliseconds the mouse pointer will take to move to its destination, default is 50.
- **move_duration_max** (*int*) – Maximum number of miliseconds the mouse pointer will take to move to its destination, default is 1500.

Returns Returns a float.

ocvbot.input.**move_path** ( )

Randomizes the movement behavior of the mouse cursor as it moves to a new location. One of 22 different movement patters is chosen at random.

Returns Returns a random PyAutoGUI function for different mouse movement.

ocvbot.input.**move_to** (*x*, *y*, *xmax*, *ymax*, *xmin=0*, *ymin=0*, *duration_min=50*, *duration_max=1500*)

Moves the mouse pointer to the specified coordinates. Coordinates are relative to the display's dimensions. Units are in pixels.

Parameters

- **x** – The X coordinate to move the mouse to.
- **y** – The Y coordinate to move the mouse to.
- **xmax** – The maximum random pixel offset from x.
- **ymax** – The maximum random pixel offset from y.
- **xmin** (*default = 0*) –
- **ymin** (*default = 0*) –

Returns Always returns 0.

ocvbot.input.**move_to_neutral** (*x*, *y*, *xmin=50*, *xmax=300*, *ymin=300*, *ymax=500*)

Moves the mouse to a 'neutral zone', away from any buttons or tooltop icons that could get in the way of the script. Units are in pixels.

Parameters

- **xmin** (*int*) – The minimum X-distance to move, default is 50.
- **xmax** (*int*) – The maximum X-distance to move, default is 300.
- **ymin** (*int*) – The minimum Y-distance to move, default is 300.
- **ymax** (*int*) – The maximum X-distance to move, default is 500.

Returns Always returns 0.

1.1.4 ocvbot.main module

1.1.5 ocvbot.main_test module

ocvbot.main_test.**kill** (*procname*)
Kills the provided process by name.

ocvbot.main_test.**test_cannonball_smelter** ( )
Full simulation of the cannonball_smelter script using screenshots.

1.1.6 ocvbot.misc module

ocvbot.misc.**rand_seconds** (*rmin=0*, *rmax=100*)
Gets a random integer between two values. Input arguments are in miliseconds but output is in seconds. For example, if this function generates a random value of 391, it will return a value of 0.391.

Parameters
- **rmin** (*default = 0*) – The minimum number of miliseconds.
- **rmax** (*default = 100*) – The maximum number of miliseconds.

Returns Returns a float.

ocvbot.misc.**sleep_rand** (*rmin=0*, *rmax=100*)
Does nothing for a random period of time. Input arguments are in miliseconds.

Parameters
- **rmin** (*default = 0*) – The minimum number of miliseconds to wait.
- **rmax** (*default = 100*) – The maximum number of miliseconds to wait.

Returns Always returns 0.

1.1.7 ocvbot.skilling module

ocvbot.skilling.**miner_double_drop** (*rock1*, *rock2*, *ore*)
A 2-rock drop mining script. The player alternates mining between two different rocks containing the same ore. Ore is dropped when inventory is full.

Parameters
- **rock1** – Filepath to a needle showing the first rock is full.
- **rock1_empty** – Filepath to a needle showing the first rock is empty.
- **rock2** – Filepath to a needle showing the second rock is full.
- **rock2_empty** – Filepath to a needle showing the second rock is empty.
- **ore** – Filepath to a needle of the item icon of the ore being mined.

Reutrns:
Returns 0 after emptying inventory.

1.1.8 ocvbot.vision module

class ocvbot.vision.**Vision** (*left*, *top*, *width*, *height*, *haystack=0*, *grayscale=False*)
Bases: object
The primary object method for locating images on the display. All coordinates are relative to the display's dimensions. Coordinate units are in pixels.

Parameters
- **self.haystack** (*default = 0*) – The image within which to search

- **the needle. By default this will cause mlocate() to search** (*for*) –
- **the coordinates provided by left/top/width/height.** (*within*) –
- **self.grayscale** (*default = False*) – Converts the haystack to
- **before searching within it. Speeds up searching by** (*grayscale*) –
- **30%.** (*about*) –
- **self.left** – The left edge (x) of the coordinate space to search
- **for the needle.** (*within*) –
- **self.top** – The top edge (y) of the coordinate space to search
- **for the needle.** –
- **self.width** – The total x width of the coordinate space to search
- **for the needle** (*within*) –
- **self.height** – The total y height of the coordinate space to
- **within for the needle** (*search*) –

click_image (*needle*, *button='left'*, *conf=0.95*, *loop_num=25*, *loop_sleep_min=10*, *loop_sleep_max=1000*, *click_sleep_before_min=0*, *click_sleep_before_max=500*, *click_sleep_after_min=0*, *click_sleep_after_max=500*, *move_duration_min=50*, *move_duration_max=1500*)

Moves the mouse to the provided needle image and clicks on it. If a haystack is provided, searches for the provided needle image within the haystack. If a haystack or set of coordinates is not provided, searches within the entire display.

Parameters

- **button** – The mouse button to use when clicking on the needle.
- **needle** – See mlocate()'s docstring.
- **conf** (*default = 0.95*) – See mlocate()'s docstring.
- **loop_num** (*default = 10*) – See wait_for_image()'s docstring.
- **loop_sleep_min** (*default = 10*) – See wait_for_image()'s
- **docstring.** –
- **loop_sleep_max** (*default = 1000*) – See wait_for_image()'s
- **docstring.** –

Returns See mlocate()'s docstring.

drop_all_item (*item*)

Drops all instances of the given inventory item.

mlocate (*needle*, *loctype='regular'*, *conf=0.95*)

Searches the haystack image for the needle image, returning a tuple containing the needle's XY coordinates within the haystack. If a haystack image is not provided, this function searches the entire client window.

Parameters

Returns If the needle is found, returns the needle as a PIL image object. If the needle is not found, returns 1.

wait_for_image (*needle*, *loctype='regular'*, *conf=0.95*, *loop_num=10*, *loop_sleep_min=10*, *loop_sleep_max=1000*)

Repeatedly searches within the haystack or coordinate space for the needle.

Parameters

- **needle** – See mlocate()'s docstring.
- **loctype** (*default = 'regular'*) – see mlocate()'s docstring.
- **conf** (*default = 0.95*) – See mlocate()'s docstring.
- **loop_num** (*default = 10*) – The number of times to search for
- **needle before giving up.** (*the*) –
- **loop_sleep_min** (*default = 10*) – The minimum number of
- **to wait after each search attempt.** (*miliseconds*) –
- **loop_sleep_max** (*default = 1000*) – The maximum number of
- **to wait after wach search attempt.** (*miliseconds*) –

Returns See mlocate()'s docstring.

ocvbot.vision.**find_anchor** (*display_width*, *display_height*)

Look for an icon to orient the client. If it's found, use its location within the game client to determine the coordinates of the game client relative to the display's coordinates.

This function is also used to determine if the client is logged out.

ocvbot.vision.**init_vision** ( )

1.1.9 Module contents

- *Index*
- *Module Index*
- *Search Page*

O

W