**Example Usage**

**Keyboard and Mouse Control**

The x, y coordinates used by PyAutoGUI has the 0, 0 origin coordinates in the top left corner of the screen. The x coordinates increase going to the right (just as in mathematics) but the y coordinates increase going down (the opposite of mathematics). On a screen that is 1920 x 1080 pixels in size, coordinates 0, 0 are for the top left while 1919, 1079 is for the bottom right.

Currently, PyAutoGUI only works on the primary monitor. PyAutoGUI isn't reliable for the screen of a second monitor (the mouse functions may or may not work on multi-monitor setups depending on your operating system and version).

All keyboard presses done by PyAutoGUI are sent to the window that currently has focus, as if you had pressed the physical keyboard key.

>>> import pyautogui

>>> screenWidth, screenHeight = pyautogui.size() # Returns two integers, the width and height of the screen. (The primary monitor, in multi-monitor setups.)

>>> currentMouseX, currentMouseY = pyautogui.position() # Returns two integers, the x and y of the mouse cursor's current position.

>>> pyautogui.moveTo(100, 150) # Move the mouse to the x, y coordinates 100, 150.

>>> pyautogui.click() # Click the mouse at its current location.

>>> pyautogui.click(200, 220) # Click the mouse at the x, y coordinates 200, 220.

>>> pyautogui.move(None, 10) # Move mouse 10 pixels down, that is, move the mouse relative to its current position.

>>> pyautogui.doubleClick() # Double click the mouse at the

>>> pyautogui.moveTo(500, 500, duration=2, tween=pyautogui.easeInOutQuad) # Use tweening/easing function to move mouse over 2 seconds.

>>> pyautogui.write('Hello world!', interval=0.25) # Type with quarter-second pause in between each key.

>>> pyautogui.press('esc') # Simulate pressing the Escape key.

>>> pyautogui.keyDown('shift')

>>> pyautogui.write(['left', 'left', 'left', 'left', 'left', 'left'])

>>> pyautogui.keyUp('shift')

>>> pyautogui.hotkey('ctrl', 'c')

**Display Message Boxes**

>>> import pyautogui

>>> pyautogui.alert('This is an alert box.')

'OK'

>>> pyautogui.confirm('Shall I proceed?')

'Cancel'

>>> pyautogui.confirm('Enter option.', buttons=['A', 'B', 'C'])

'B'

>>> pyautogui.prompt('What is your name?')

'Al'

>>> pyautogui.password('Enter password (text will be hidden)')

'swordfish'

**Screenshot Functions**

(PyAutoGUI uses Pillow for image-related features.)

>>> import pyautogui

>>> im1 = pyautogui.screenshot()

>>> im1.save('my\_screenshot.png')

>>> im2 = pyautogui.screenshot('my\_screenshot2.png')

You can also locate where an image is on the screen:

>>> import pyautogui

>>> button7location = pyautogui.locateOnScreen('button.png') # returns (left, top, width, height) of matching region

>>> button7location

(1416, 562, 50, 41)

>>> buttonx, buttony = pyautogui.center(button7location)

>>> buttonx, buttony

(1441, 582)

>>> pyautogui.click(buttonx, buttony) # clicks the center of where the button was found

The locateCenterOnScreen() function returns the center of this match region:

>>> import pyautogui

>>> buttonx, buttony = pyautogui.locateCenterOnScreen('button.png') # returns (x, y) of matching region

>>> buttonx, buttony

(1441, 582)

>>> pyautogui.click(buttonx, buttony) # clicks the center of where the button was found

**How Does PyAutoGUI Work?**

The three major operating systems (Windows, macOS, and Linux) each have different ways to programmatically control the mouse and keyboard. This can often involve confusing, obscure, and deeply technical details. The job of PyAutoGUI is to hide all of this complexity behind a simple API.

* On Windows, PyAutoGUI accesses the Windows API (also called the WinAPI or win32 API) through the built-in ctypes module. The nicewin module at <https://github.com/asweigart/nicewin> provides a demonstration for how Windows API calls can be made through Python.
* On macOS, PyAutoGUI uses the rubicon-objc module to access the Cocoa API.
* On Linux, PyAutoGUI uses the Xlib module to access the X11 or X Window System.

I'm trying a simple python script, it clicks on a screen coordinate.

I've tried with Pyautogui, pynput, pydirectinput, pywinauto... But in none of them the click is actually made, the only thing that works is to move the mouse to the coordinate.

the scripts are simple, but it still doesn't work, by deduction I think it's a win10 related problem.

Does anyone know how I can solve this?

Do I need to install anything else, maybe a driver?

Is it missing to give some kind of permission?

is there a way for me to give command to the mouse hardware to make the click, instead of being a virtualized click?

**Some of my attempts below**

**OBS:** In all attempts the mouse moves, but does not click.

Pyautogui:

import pyautogui

pyautogui.moveTo(35, 240)

pyautogui.click()

Pydirectinput:

import pyautogui

import pydirectinput

pydirectinput.moveTo(35, 240)

pydirectinput.click()

pywinauto:

import pywinauto

from pywinauto import Desktop, Application, mouse, findwindows

pywinauto.mouse.move(coords=(160, 400))

pywinauto.mouse.double\_click(button='left', coords=(160, 400))

Direct windows click:

import win32api, win32con

def click(x,y):

win32api.SetCursorPos((x,y))

win32api.mouse\_event(win32con.MOUSEEVENTF\_LEFTDOWN,x,y,0,0)

win32api.mouse\_event(win32con.MOUSEEVENTF\_LEFTUP,x,y,0,0)

click(10,200)

Autoclicker using pynput:

import time

import threading

from pynput.mouse import Button, Controller

from pynput.keyboard import Listener, KeyCode

delay = 0.001

button = Button.left

start\_stop\_key = KeyCode(char='s')

exit\_key = KeyCode(char='e')

class ClickMouse(threading.Thread):

def \_\_init\_\_(self, delay, button):

super(ClickMouse, self).\_\_init\_\_()

self.delay = delay

self.button = button

self.running = False

self.program\_running = True

def start\_clicking(self):

self.running = True

def stop\_clicking(self):

self.running = False

def exit(self):

self.stop\_clicking()

self.program\_running = False

def run(self):

while self.program\_running:

while self.running:

mouse.click(self.button)

time.sleep(self.delay)

time.sleep(0.1)

mouse = Controller()

click\_thread = ClickMouse(delay, button)

click\_thread.start()

def on\_press(key):

if key == start\_stop\_key:

if click\_thread.running:

click\_thread.stop\_clicking()

else:

click\_thread.start\_clicking()

elif key == exit\_key:

click\_thread.exit()

listener.stop()

with Listener(on\_press=on\_press) as listener:

listener.join()

* [**python**](https://stackoverflow.com/questions/tagged/python)
* [**windows**](https://stackoverflow.com/questions/tagged/windows)
* [**click**](https://stackoverflow.com/questions/tagged/click)
* [**mouseevent**](https://stackoverflow.com/questions/tagged/mouseevent)
* [**mouse**](https://stackoverflow.com/questions/tagged/mouse)

[Share](https://stackoverflow.com/q/66282631)

Follow

[edited Feb 19, 2021 at 18:36](https://stackoverflow.com/posts/66282631/revisions)

asked Feb 19, 2021 at 17:52

[[](https://stackoverflow.com/users/9336128/ridjin)](https://stackoverflow.com/users/9336128/ridjin)

[Ridjin](https://stackoverflow.com/users/9336128/ridjin)

**33**11 gold badge11 silver badge44 bronze badges

* Does the click only not work in certain software, like a game? If so it could be that the game uses raw mouse hardware input, and so would bypass a programmatic click done via this method. Or, is it not working anywhere?

– [Random Davis](https://stackoverflow.com/users/6273251/random-davis)

[CommentedFeb 19, 2021 at 18:07](https://stackoverflow.com/questions/66282631/how-to-make-python-script-that-performs-a-mouse-click-in-windows-10#comment117184359_66282631)

* Yes, my focus is to click on a game. But he doesn’t click on anything, not even in tests on vsCode I managed to make him click! There is software like Blue Eye Macro that clicks on any application. But I can't depend on it. Is there any way I can give a command to click the mouse hardware?

– [Ridjin](https://stackoverflow.com/users/9336128/ridjin)

[CommentedFeb 19, 2021 at 18:24](https://stackoverflow.com/questions/66282631/how-to-make-python-script-that-performs-a-mouse-click-in-windows-10#comment117184792_66282631)

* What person do you mean by "him"? I'm not sure what you're saying. Are you saying that this script doesn't cause clicks to be registered in VSCode? But the application "Blue Eye Macro" does work, for both VSCode and the game?

– [Random Davis](https://stackoverflow.com/users/6273251/random-davis)

[CommentedFeb 19, 2021 at 18:36](https://stackoverflow.com/questions/66282631/how-to-make-python-script-that-performs-a-mouse-click-in-windows-10#comment117185091_66282631)

* Sorry, "him" is the script haha. I tried click in everything. vsCode, Google Chrome, notepad, but doenst work in anything.

– [Ridjin](https://stackoverflow.com/users/9336128/ridjin)

[CommentedFeb 19, 2021 at 18:46](https://stackoverflow.com/questions/66282631/how-to-make-python-script-that-performs-a-mouse-click-in-windows-10#comment117185317_66282631)

* Maybe give ctypes a try: [stackoverflow.com/a/1181539/1698225](https://stackoverflow.com/a/1181539/1698225)

– [Alex Jadczak](https://stackoverflow.com/users/1698225/alex-jadczak)

[CommentedFeb 19, 2021 at 21:08](https://stackoverflow.com/questions/66282631/how-to-make-python-script-that-performs-a-mouse-click-in-windows-10#comment117188170_66282631)

[Add a comment](https://stackoverflow.com/questions/66282631/how-to-make-python-script-that-performs-a-mouse-click-in-windows-10)

1 Answer

Sorted by:

                                              Highest score (default)                                                                   Trending (recent votes count more)                                                                   Date modified (newest first)                                                                   Date created (oldest first)

**3**

Try this way:

import pywinauto

app = pywinauto.Application().connect(path='your\_process\_name.exe')

app.MainDialog.click\_input(coords=(x, y))

For click method to work you need to specify the process/dialog on which the coordinate is present. Use connect() method to connect to a existing method else use start() to open new instance.

**Mouse and keyboard automation using Python**

Last Updated : 21 Jan, 2021

This article illustrates how to automate movements of mouse and keyboard using **pyautogui** module in python. This module is not preloaded with python. So to install it run the following command: 

pip3 install pyautogui

**Controlling mouse movements using pyautogui module**

Python tracks and controls mouse using the coordinate system of the screen. Suppose the resolution of your screen is 1920X1080, then your screen’s coordinate system looks like this: 

gui in python

* **size():** This function is used to get Screen resolution.
* Python

|  |
| --- |
| **import** pyautogui  print(pyautogui.size()) |

Save this file with .py extension, and then run the file.   
This python code use size() function to output your screen resolution in x, y format:   
Output: 

(1920, 1080)

Note: Some of the codes provided in this article might not run on geeksforgeeks IDE, since geeksforgeeks IDE doesn’t have the required modules to run these codes. But these codes can be easily run locally on your PC by installing python and following the instructions provided in the article. 

* **moveTo():** use this function to move the mouse in pyautogui module.
* Python

|  |
| --- |
| **import** pyautogui  pyautogui.moveTo(100, 100, duration **=** 1) |

This code uses moveTo() function, which takes x and y coordinates, and an optional duration argument. This function moves your mouse pointer from it’s current location to x, y coordinate, and takes time as specified by duration argument to do so. Save and run this python script to see your mouse pointer magically moving from its current location to coordinates (100, 100), taking 1 second in this process. 

* **moveRel() function:** moves the mouse pointer relative to its previous position.
* Python

|  |
| --- |
| **import** pyautogui  pyautogui.moveRel(0, 50, duration **=** 1) |

This code will move mouse pointer at (0, 50) relative to its original position. For example, if mouse position before running the code was (1000, 1000), then this code will move the pointer to coordinates (1000, 1050) in duration 1 second. 

* **position():** function to get current position of the mouse pointer.
* Python

|  |
| --- |
| **import** pyautogui  print(pyautogui.position()) |

Output: coordinates where your mouse was residing at the time of executing the program. 

* **click():**Function used for clicking and dragging the mouse.
* Python

|  |
| --- |
| **import** pyautogui  pyautogui.click(100, 100) |

This code performs a typical mouse click at the location (100, 100).   
We have two functions associated with the drag operation of the mouse, **dragTo and dragRel**. They perform similar to moveTo and moveRel functions, except they hold the left mouse button while moving, thus initiating a drag.   
This functionality can be used at various places, like moving a dialog box, or drawing something automatically using a pencil tool in MS Paint. To draw a square in paint: 

* Python

|  |
| --- |
| **import** time    # a module which has functions related to time.  # It can be installed using cmd command:  # pip install time, in the same way as pyautogui.  **import** pyautogui  time.sleep(10)    # makes program execution pause for 10 sec  pyautogui.moveTo(1000, 1000, duration **=** 1)    # moves mouse to 1000, 1000.  pyautogui.dragRel(100, 0, duration **=** 1)    # drags mouse 100, 0 relative to its previous position,  # thus dragging it to 1100, 1000  pyautogui.dragRel(0, 100, duration **=** 1)  pyautogui.dragRel(**-**100, 0, duration **=** 1)  pyautogui.dragRel(0, **-**100, duration **=** 1) |

Before running the code, open MS paint in the background with the pencil tool selected. Now run the code, quickly switch to MS paint before 10 seconds (since we have given 10 second pause time using sleep() function before running the program).   
After 10 seconds, you will see a square being drawn in MS paint, with its top-left edge at 1000, 1000, and edge length 100 pixels.

* **scroll():** scroll function takes no. of pixels as an argument, and scrolls the screen up to a given number of pixels.
* Python

|  |
| --- |
| **import** pyautogui  pyautogui.scroll(200) |

This code scrolls the active screen up to 200 pixels. 

* **typewrite():** You can automate typing of the string by using typewrite() function. just pass the string which you want to type as an argument of this function.
* Python

|  |
| --- |
| **import** pyautogui  pyautogui.click(100, 100)  pyautogui.typewrite("hello Geeks !") |

Suppose a text field was present at coordinates 100, 100 on-screen, then this code will click the text field to make it active and type “hello Geeks!” in it. 

* **Passing key names:** You can pass key names separately through typewrite() function.
* Python

|  |
| --- |
| **import** pyautogui  pyautogui.typewrite(["a", "left", "ctrlleft"]) |

This code is the automatic equivalent of typing “a”, pressing the left arrow key, and pressing the left control key. 

* **Pressing hotkey combinations:** Use hotkey() function to press the combination of keys like ctrl-c, ctrl-a, etc.
* Python

|  |
| --- |
| **import** pyautogui  pyautogui.hotkey("ctrlleft", "a") |

This code is the automatic equivalent of pressing left ctrl and “a” simultaneously. Thus in windows, this will result in the selection of all text present on the screen.

**Example:**

To send a message in WhatsApp and delete it for everyone automatically. You need to have Whatsapp already opened in chrome, to run this. After running this code, open the WhatsApp tab on chrome.

* Python3

|  |
| --- |
| **import** pyautogui as pg  **import** time    **def** delete\_for\_everyone():      pg.click(807, 979)      pg.typewrite("hello")      pg.typewrite(["enter"])      time.sleep(2)      pg.click(1621, 896)      pg.click(1621, 896)        # time.sleep(1)      pg.click(1693, 859)        # time.sleep(1)      pg.click(1014, 669)        # time.sleep(1)      pg.click(1111, 605)    a**=**20  time.sleep(10)  **while**(a!**=**0):      delete\_for\_everyone()      a**=**a**-**1 |

**T**

**tkkhhaarree**