Contents

[1. Introduction 2](#_Toc197614476)

[Why Was AutoItLibrary Developed? 8](#_Toc197614477)

[Quick Real-World Example 8](#_Toc197614478)

[Quick History Timeline 8](#_Toc197614479)

[Summary 8](#_Toc197614480)

[Why Was AutoItLibrary Developed? 9](#_Toc197614481)

[2. Installation 11](#_Toc197614482)

[3. AutoIt Function 14](#_Toc197614483)

[Real-Life Example: 20](#_Toc197614484)

[Extra Advanced 30](#_Toc197614485)

[4.Mouse Keyword 30](#_Toc197614486)

[5. Control Action 34](#_Toc197614487)

[Some Shortcuts 35](#_Toc197614488)

[When Should You Use Control Actions? 36](#_Toc197614489)

[How to Check If an App Is UWP: 37](#_Toc197614490)

[4. Advance AutoItLibrary automation techniques 38](#_Toc197614491)

# 1. Introduction

1. What is AutoIt?

AutoIt is a powerful and flexible scripting language designed for automating the windows operating system. It can simulates keystrokes, mouse movements and window commands (maximize, minimize, wait for etc.), making it useful for automating repetitive task and GUI interaction. It is commonly used in system automation, testing.

AutoIt was introduced by a developer named Jonathan Bennett.

It was first released in 1999.

Today AutoIt is maintained by Jonathan Bennett and the AutoIt Team.

Used Of AutoIt:

1. GUI Automation: Automates interactions with windows and controls in graphical user interfaces (GUI) on windows.

2. Keystrokes and Mouse Simulation: Simulates typing and mouse movements, click and dragging to automate repetitive tasks.

3. Windows Control: Allows for windows manipulation like minimizing, maximizing and sending commands to active windows.

4. File Operations: Handles file operations like opening, closing or editing files.

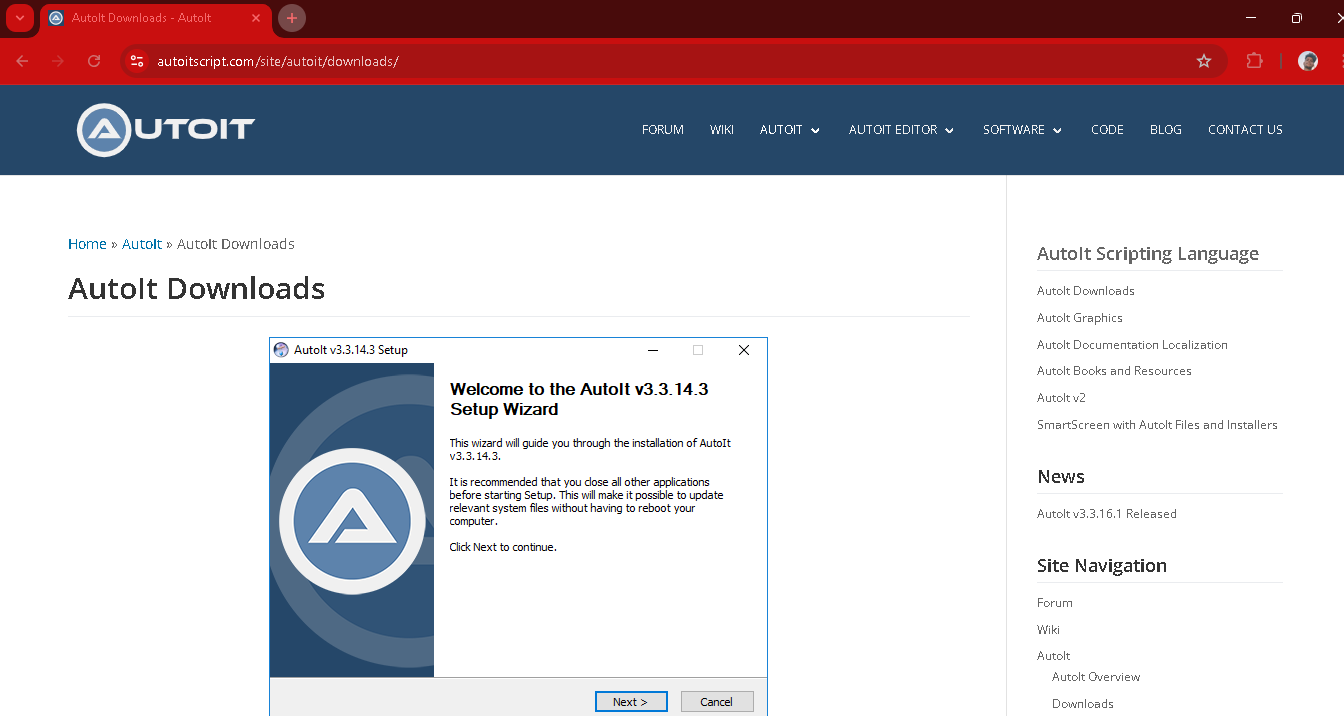
Example Code

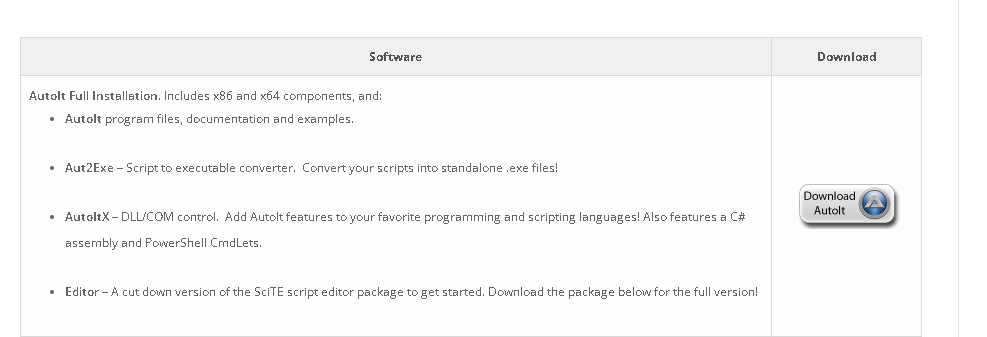
Run ("notepad.exe") ; Run Notepad

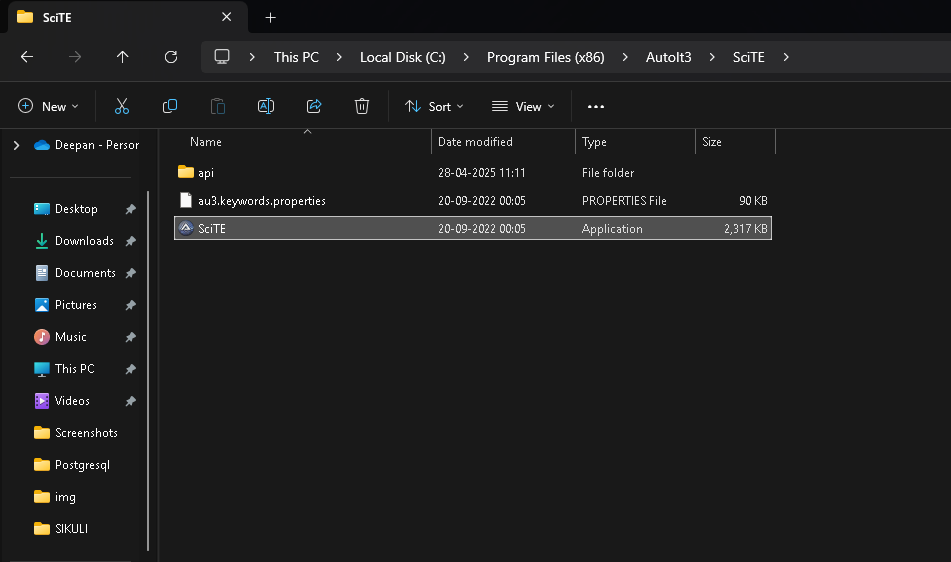
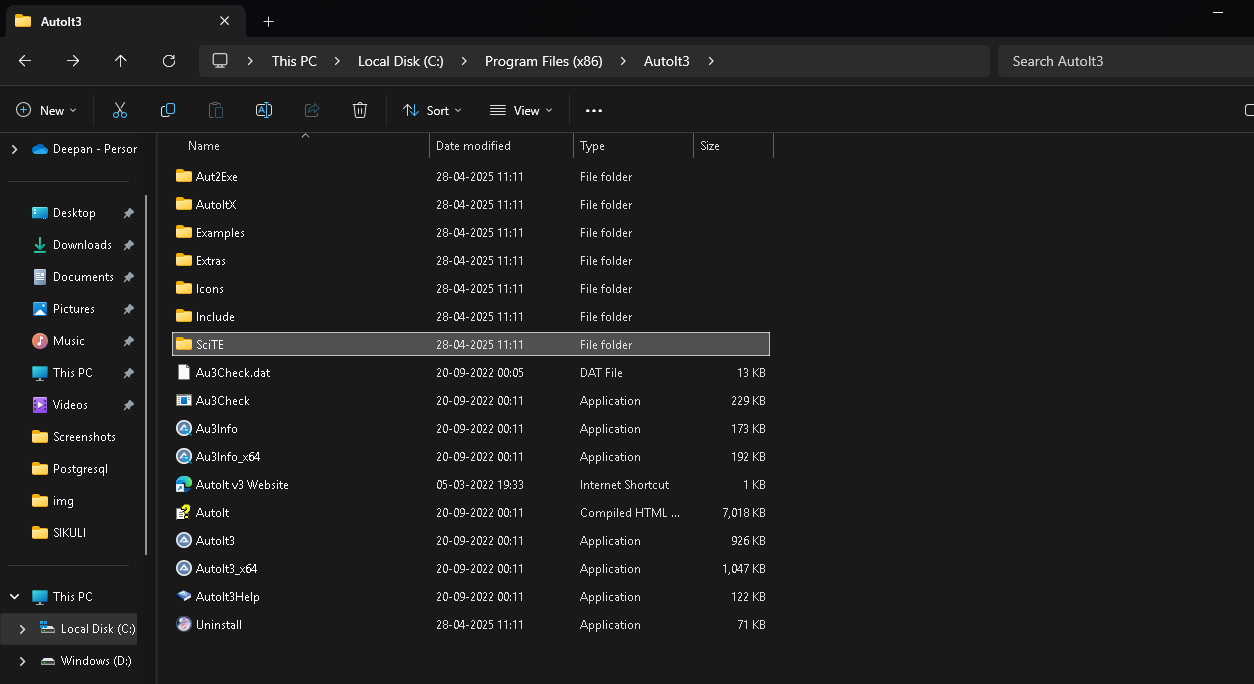
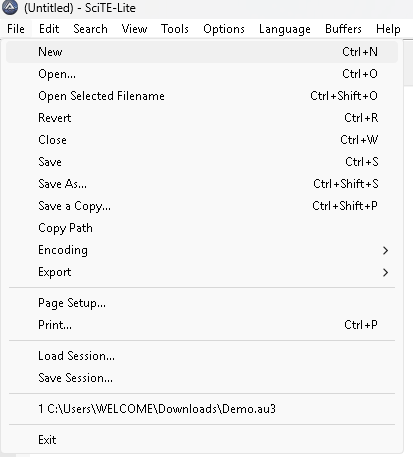
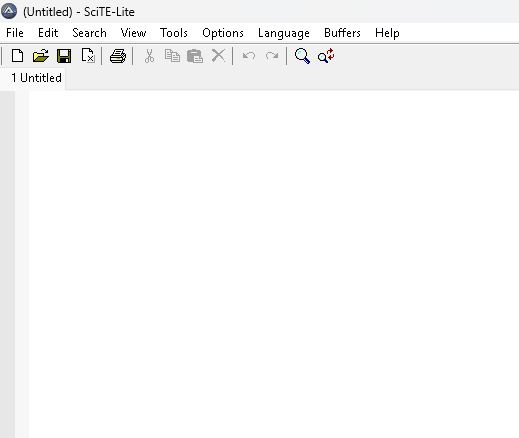
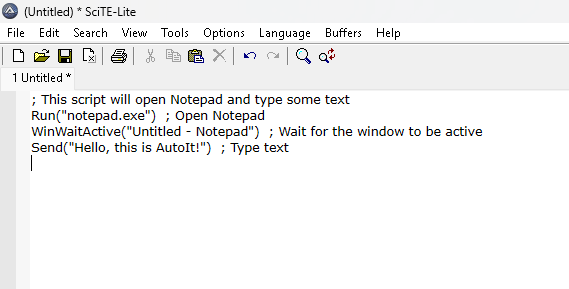
WinWaitActive ("Untitled - Notepad"); Wait for the Notepad window to be active

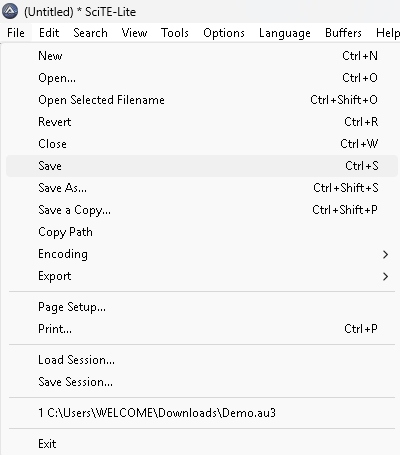
Send ("Hello, this is AutoIt!"); Type the message in Notepad

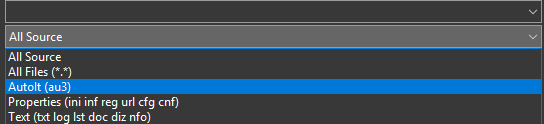
Sleep (2000) ; Wait for 2 seconds

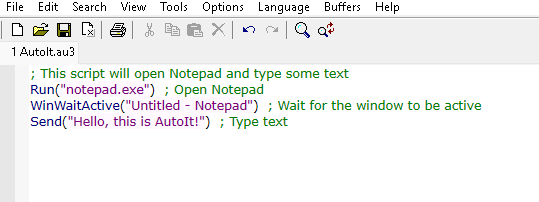
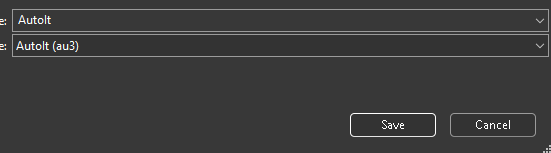
WinClose ("Untitled - Notepad"); Close the Notepad window

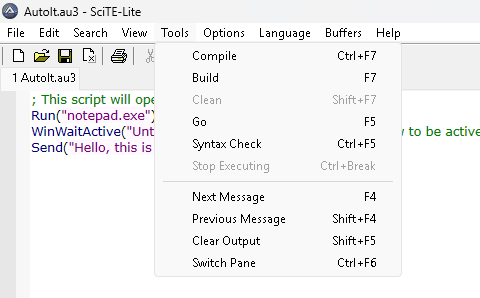


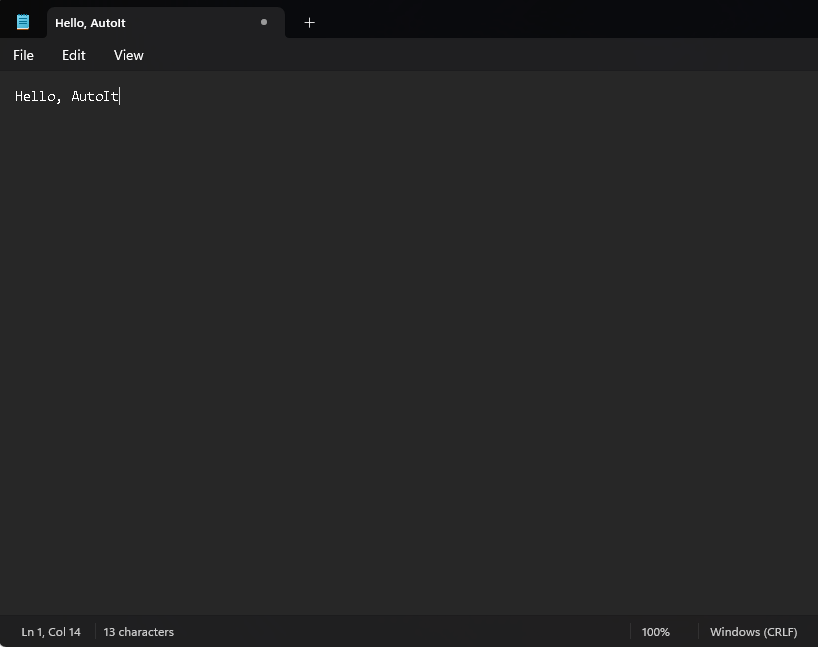
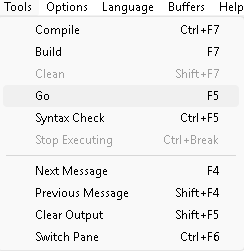
   











2. AutoIt Library

AutoItLibrary is a Robot Framework library that integrates AutoIt functionality into Robot Framework. It enables Robot Framework (a test automation framework) to perform windows GUI automation tasks using the AutoIt scripting language.

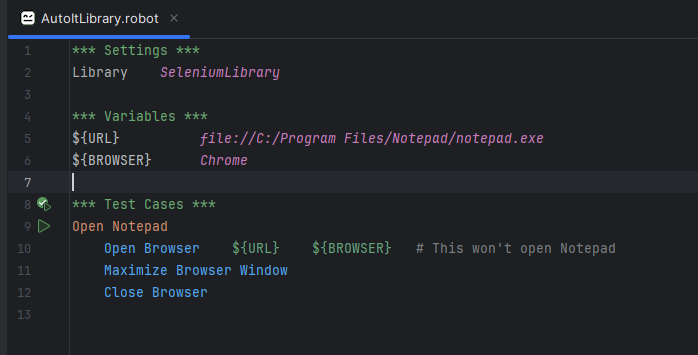
Why AutoItLibrary was created

Robot Framework is commonly used for testing web applications, but it doesn’t have built-in functionality to interact with Windows desktop applications.

Since AutoIt can automate windows GUI tasks, the AutoItLibrary was developed to bridge this gap and allow Robot Framework to automate windows desktop applications.

Build in functionality not open in notepad.exe in selenium Library

Not Work



What is AutoLibrary?

AutoLibrary is a library made for Robot Framework.

It allows Robot Framework to control windows desktop applications.

Example:

Opening Notepad

Clicking buttons

Typing text

Basically, AutoItlibrary helps Robot Framework “talk” to windows operating system GUI elements.

Who Developed AutoItLibrary?

AutoItLibrary was created by Jukka Rahko

He developed it as an open-source project.

AutoItLibrary is bridge between Robot Framework and AutoIt.

# Why Was AutoItLibrary Developed?

It was developed because:

| **Problem** | **Solution: AutoItLibrary** |
| --- | --- |
| Robot Framework originally automates **web apps** very well (using SeleniumLibrary), but **not desktop apps** | AutoItLibrary adds the ability to automate **desktop Windows apps** |
| Uploading files in web automation triggers Windows File Upload dialogs that Selenium **cannot control (**<input type="file">)**)** | AutoItLibrary can control these OS-level dialogs |
| People needed a way to **click, type, select** inside Windows apps in automation scripts | AutoItLibrary gives ready-made keywords to do it easily |
| Manual testing of desktop applications was **time-consuming** | Automation using AutoItLibrary makes it faster |

# Quick Real-World Example

Suppose you're testing a web app and you need to upload a file.

* Selenium can click the "Upload" button ✅
* But the file selection dialog is a **Windows window** ❌
* Selenium **cannot** handle it.
* **AutoItLibrary** can interact with the Windows upload window ➡ Select a file ➡ Click "Open"!

Problem solved! 🎯

# Quick History Timeline

| **Year** | **Event** |
| --- | --- |
| 1999 | AutoIt scripting language created (Jonathan Bennett) |
| 2010 | AutoItLibrary released for Robot Framework (Jukka Rahko) |
| Today | Used widely for desktop and file upload automation |

# Summary

|  |  |
| --- | --- |
| **AutoIt** | Automation tool for Windows GUI (independent software) |
| **AutoItLibrary** | Robot Framework library that connects to AutoIt |
| **Developer of AutoItLibrary** | Jukka Rahko |
| **Why developed** | To automate Windows desktop apps with Robot Framework |

AutoIt Library’s Core Dependency – AutoIt Tool

Only Windows automation tool.

AutoItLibrary does not support iOS or Linux.

ios and linux have different graphical interfaces and ui components that AutoIt is not designed to interact with the system API’s (Application Programming Interfaces) in these OS are very different from Windows.

Alternate for ios and Linux

For ios

* Appium (mobile automation tool)
* XCUITest (for IOS-specific testing)

For Linux

* sikuliX (visual automation)
* X11 (GUI automation)

### **Why Was AutoItLibrary Developed?**

**AutoItLibrary** was developed to fill a specific gap in Robot Framework:

1. **Windows Automation**:
   * While **Robot Framework** is excellent for automating **web applications** (e.g., using Selenium), it doesn't handle **desktop applications** and **system-level pop-ups** like **Windows file upload dialogs**.
2. **File Uploads & System Dialogs**:
   * Web applications often have **file upload dialogs** that are part of the Windows OS, which cannot be automated by web-based tools like Selenium.
   * **AutoItLibrary** solves this by using **AutoIt** to handle system-level windows and dialogs.
3. **Desktop Application Testing**:
   * It allows for **desktop testing automation** by interacting with Windows applications, clicking on buttons, entering text, reading data from windows, etc.

Example

\*\*\* Settings \*\*\*

Library AutoItLibrary

\*\*\* Test Cases \*\*\*

Automate Notepad

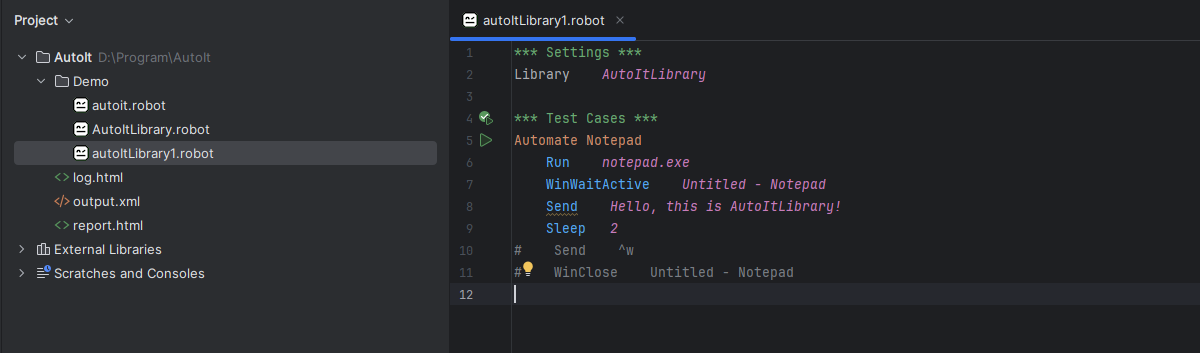
Run notepad.exe

WinWaitActive Untitled - Notepad

Send Hello, this is AutoItLibrary!

Sleep 2

WinClose Untitled – Notepad



# 2. Installation

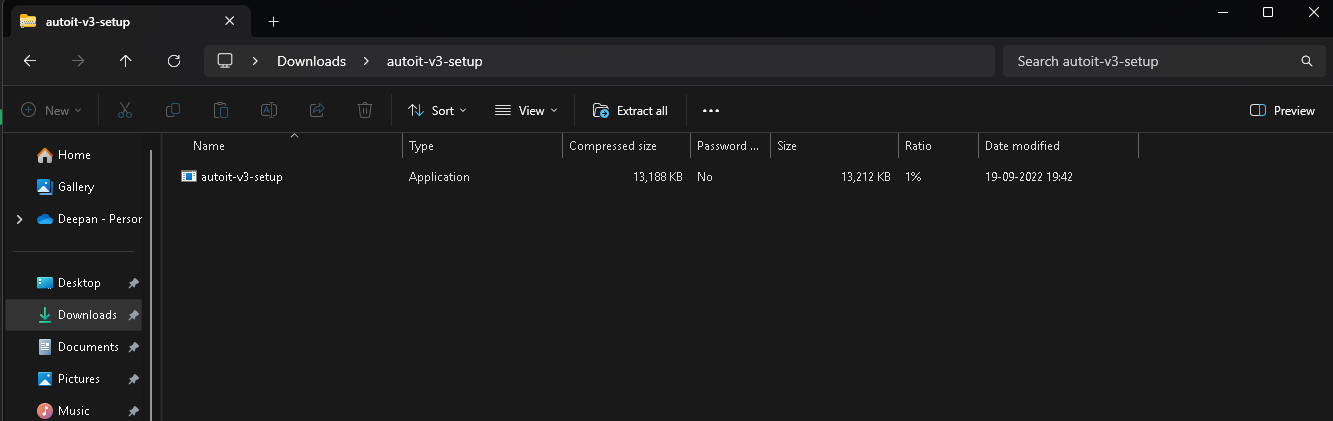
Install AutoIt

1. Download AutoIt from the official Website

<https://www.autoitscript.com/site/autoit/downloads/>

pip install robotframework-autoitlibrary

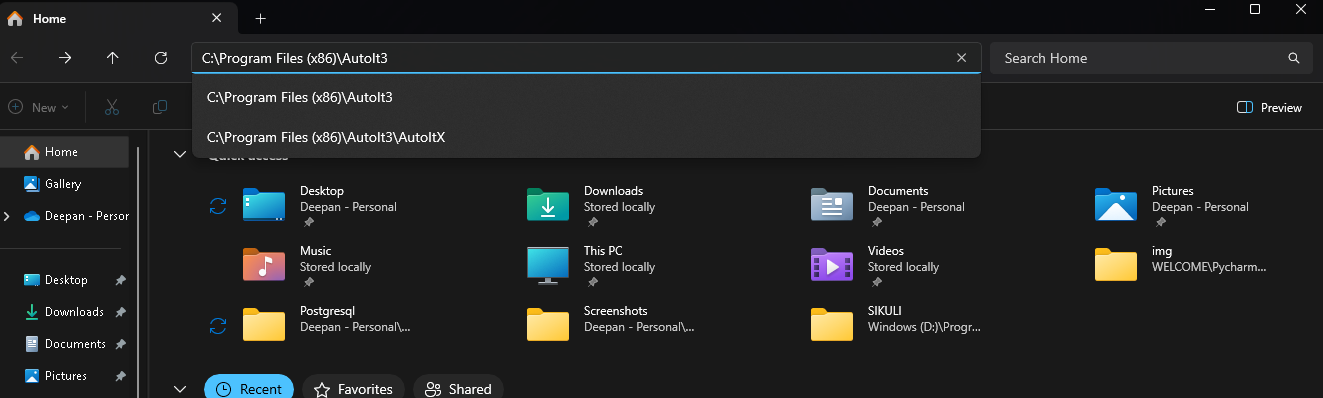
robot –version

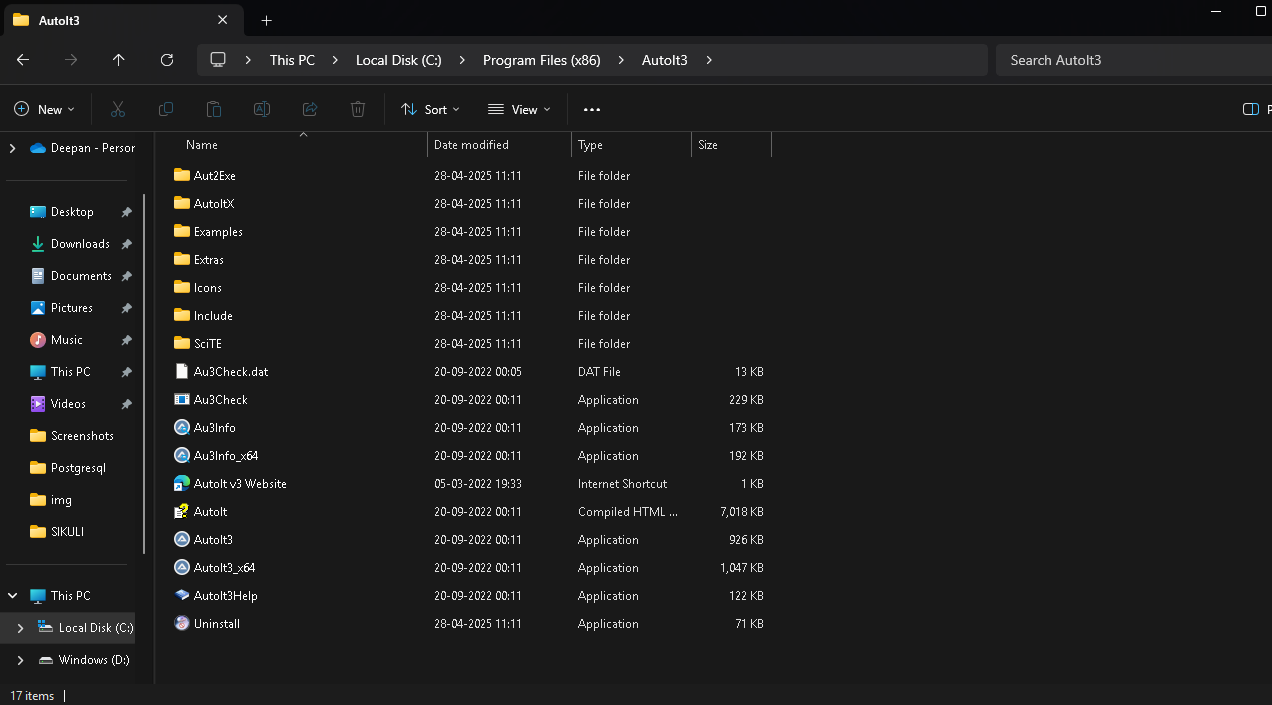


Install the application

Check the Code in file manager file is present or not

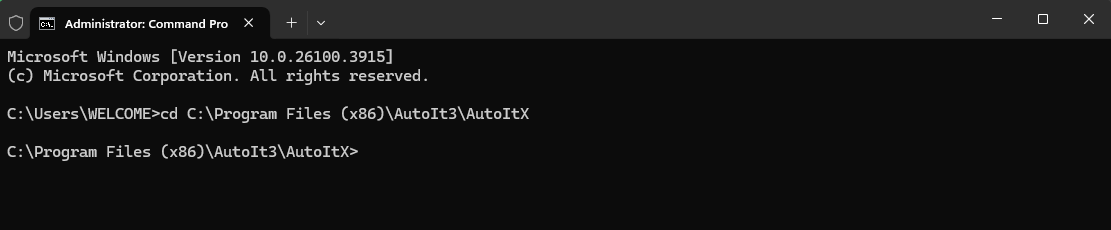
If you use tool open the folder SciTe to open the AutoIt Tool



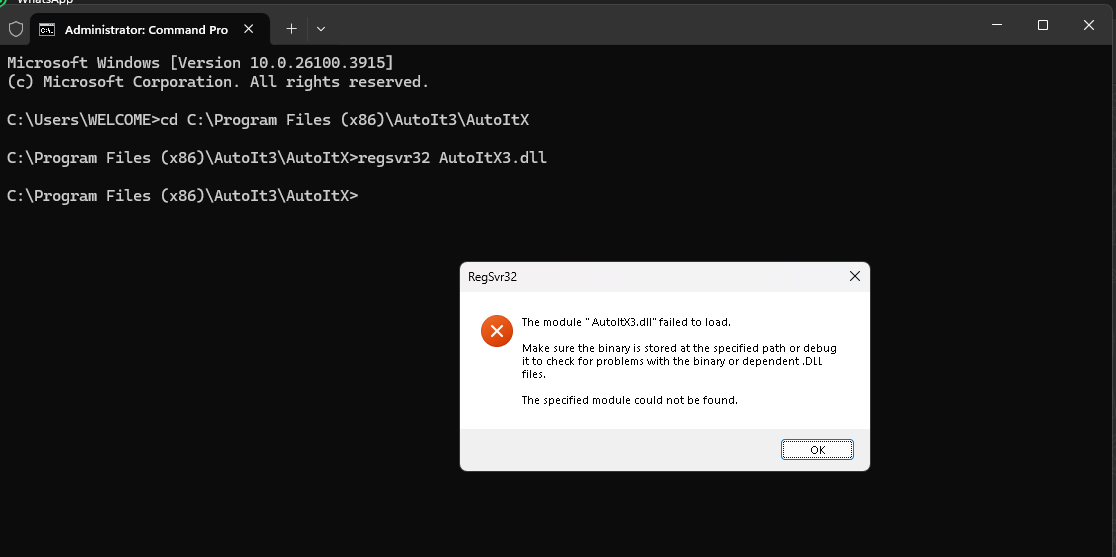


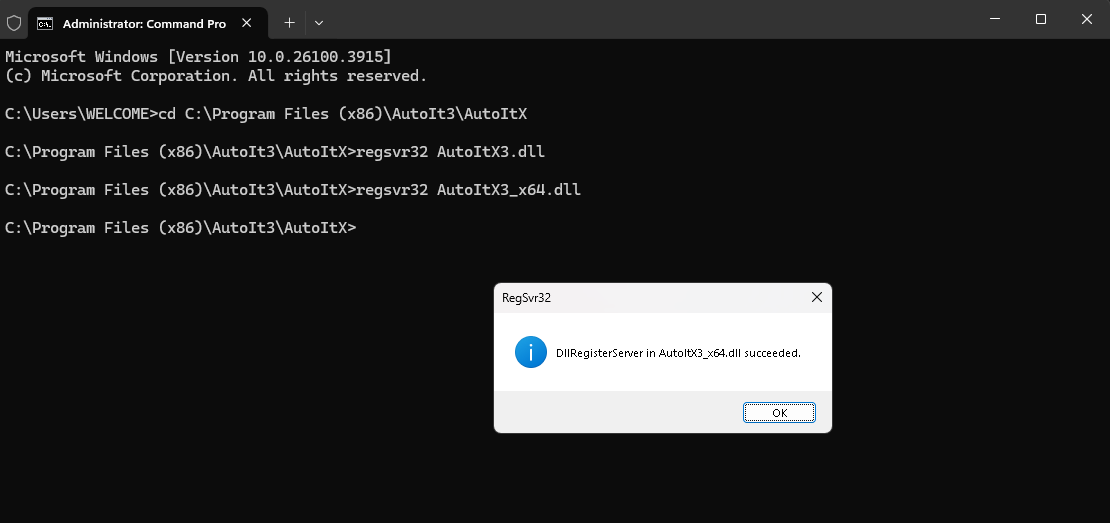
Open Cmd And Run on adminstrator

cd "C:\Program Files (x86)\AutoIt3\AutoItX"



regsvr32 AutoItX3.dll (Not Error enter OK)



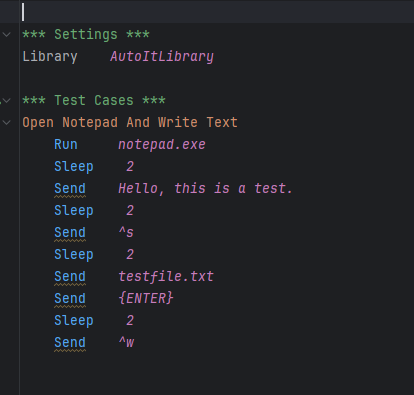
regsvr32 AutoItX3\_x64.dll

Open Pycharm Properly work or not

from AutoItLibrary import AutoItLibrary

lib = AutoItLibrary()

lib.run("notepad.exe")



# 3. AutoIt Function

1. Run ()

The Run () function is used to launch or open an external application (like a program or a file)

2. send()

Sends keystrokes (like typing text or simulating key presses) to the active window

3. Mouse Click

Simulates a mouse click at the specified position (x,y coordinates) or on a control.

4. Mouse Move

Moves the mouse to a specified position on the screen.

5. Wait for Window

Waits for a specific window to appear, making it useful for handling pop-ups or dialogs.

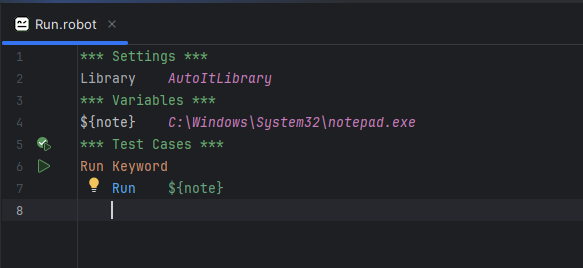
6. Win Exists

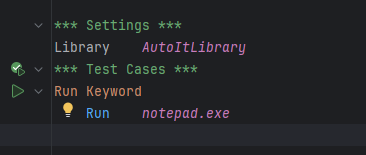
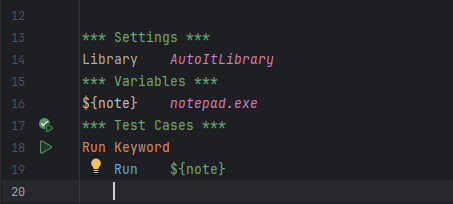
Checks if a window exists.

7. Win Close

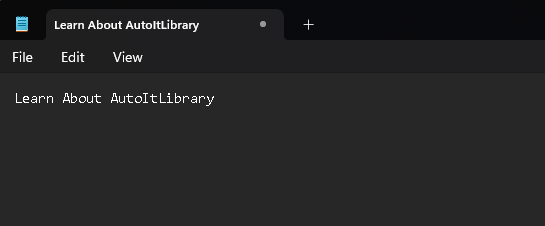
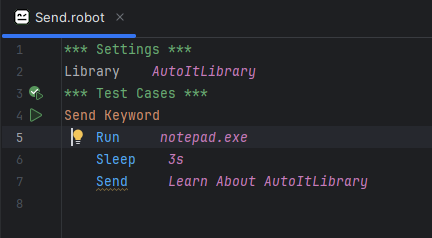
Closes a window using the window’s title

1. Run

 Not Working



2. Send



3. Mouse Click

MouseClick left 500 300

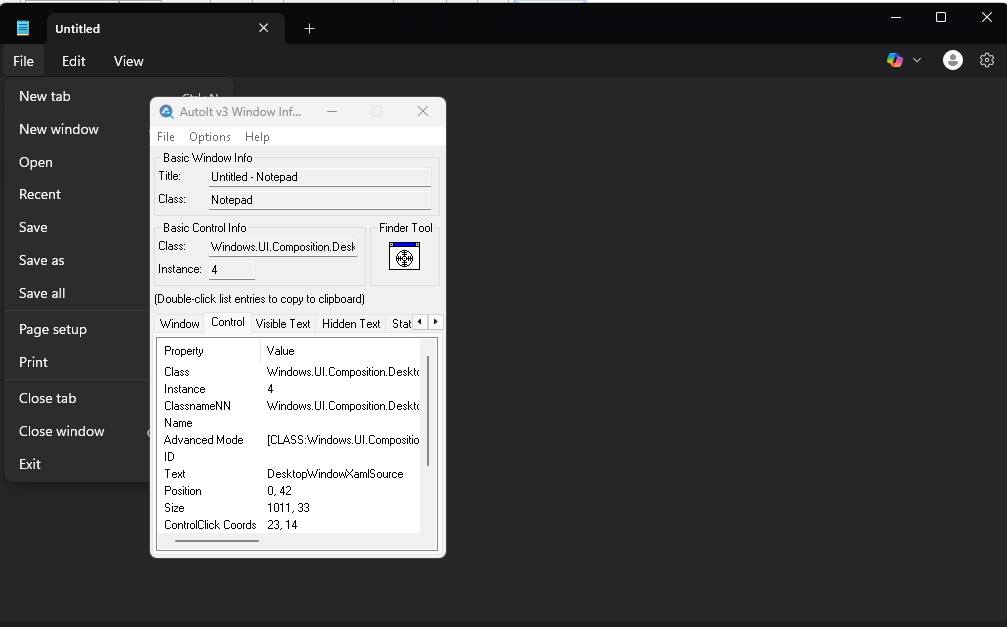
MouseClick right 600 400 2

MouseClick middle 250 250 1 5



4. ControlClick

ControlClick Untitled - Notepad "DesktopWindowXamlSource" [CLASS:Windows.UI.Composition.DesktopWindowContentBridge; INSTANCE:4]

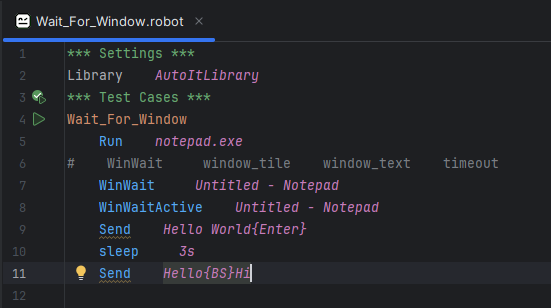


5. Mouse Move

MouseMove *30 40 50*

Mouse Move x y Speed

6. Wait For Window (Win Wait)



Sleep Command

Pauses the execution of the script for a specified amount of time (in millisecond)

Use cases

Use when you need to pause the execution for a set duration, regardless of window states or conditions.

Syntax

Sleep millisecond

Sleep 2s or 2000 (2s)

Win Wait

Waits for a window to appear before continuing with the script. You can set a timeout to stop waiting after a certain period.or Window specific title (or other matching criteria) to appear.

Use When you need to wait for a window to appear before performing further actions

WinWait window\_title timeout=30 (default is 30 second)

WinWaitAppear

Wais for the window with a specific title (or other matching criteria) to appear and become visible.

Syntax

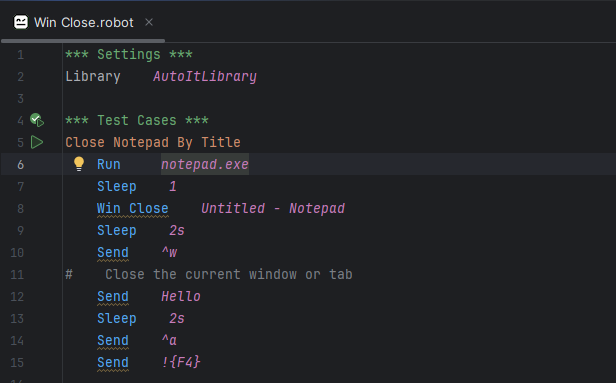
WinWaitAppear Window\_title timeout

9. Win Close

Syntax

Win Close window\_Title

Send ^w (close the current tab or window) or !{F4}



Send ^c # Simulates Ctrl+C to copy

Send ^v # Simulates Ctrl+V to paste

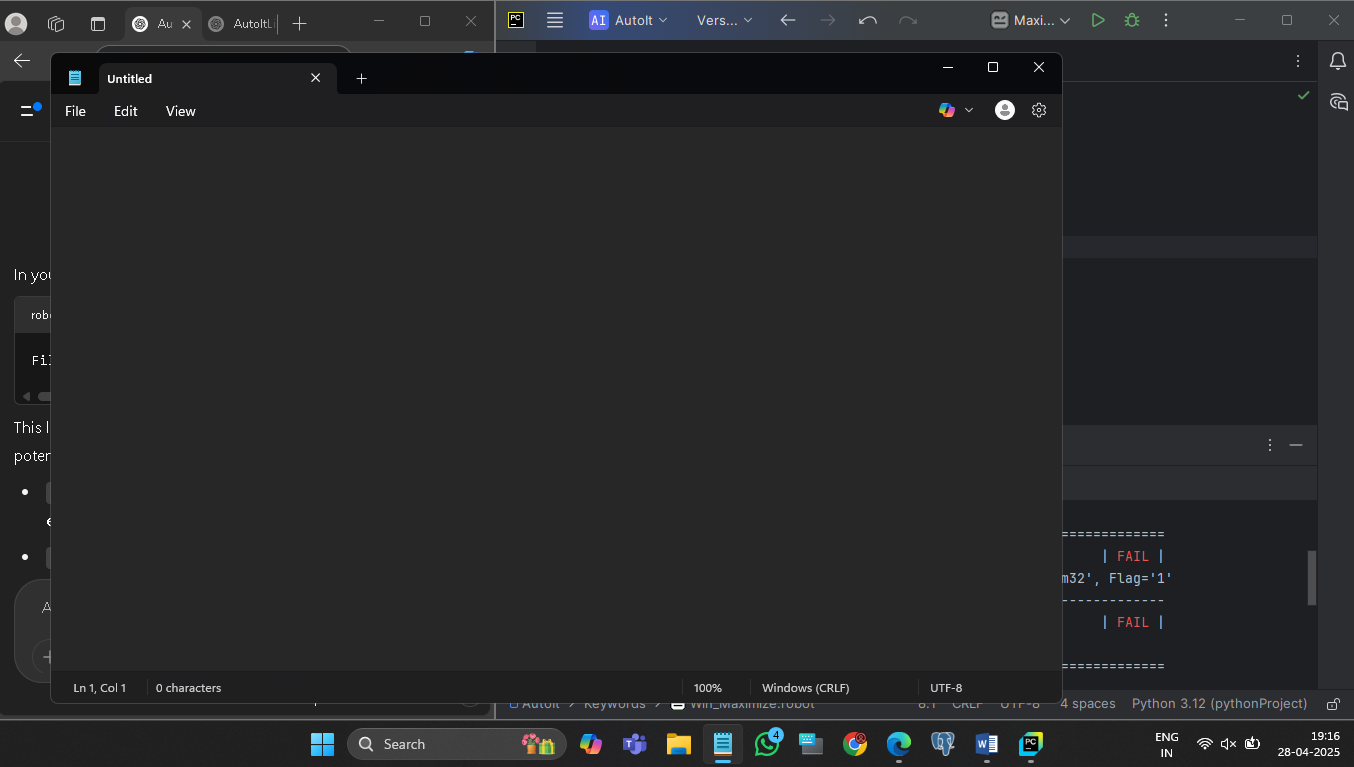
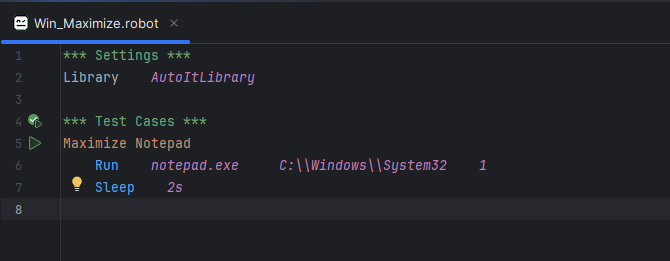
Send ^a # Simulates Ctrl+A to select all

Send ^z # Simulates Ctrl+Z for undo action

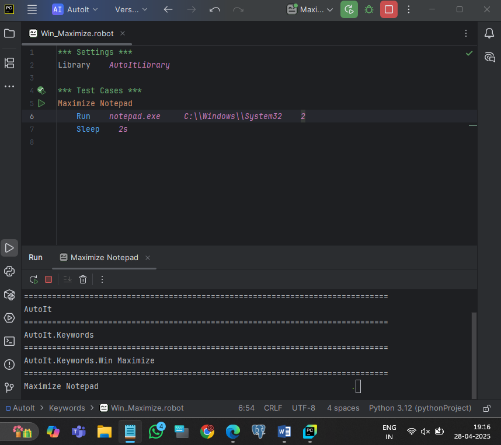
10. Maximize

Run Filename WorkingDir flag

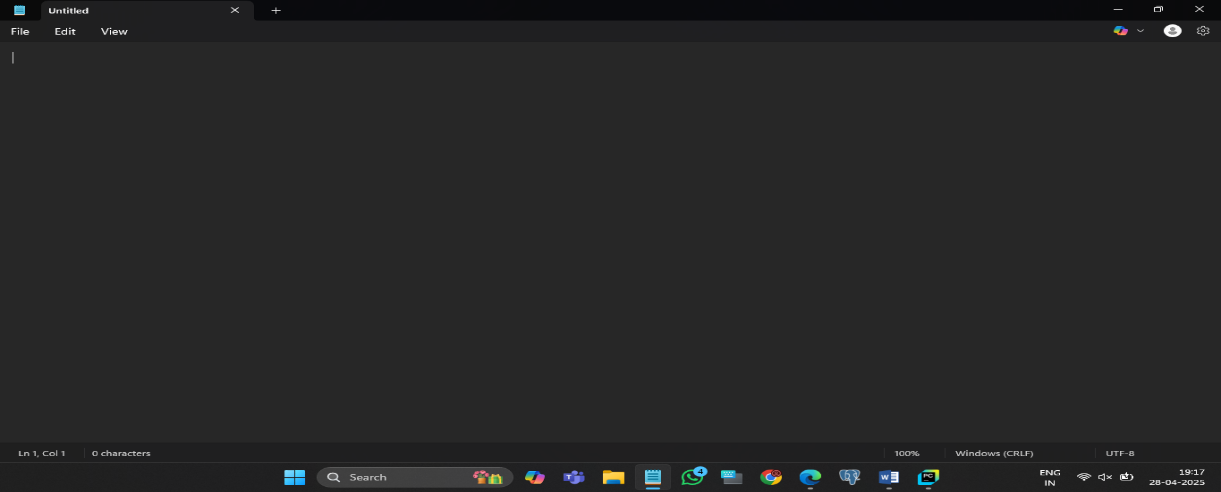
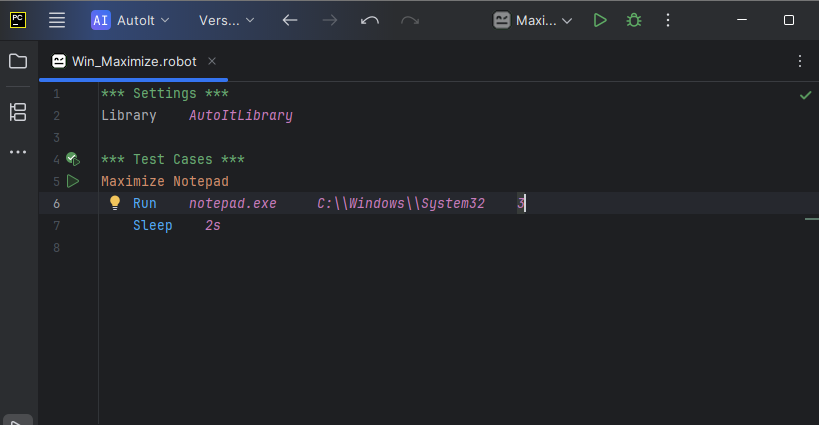
1. Normal Open



2. Only open not show



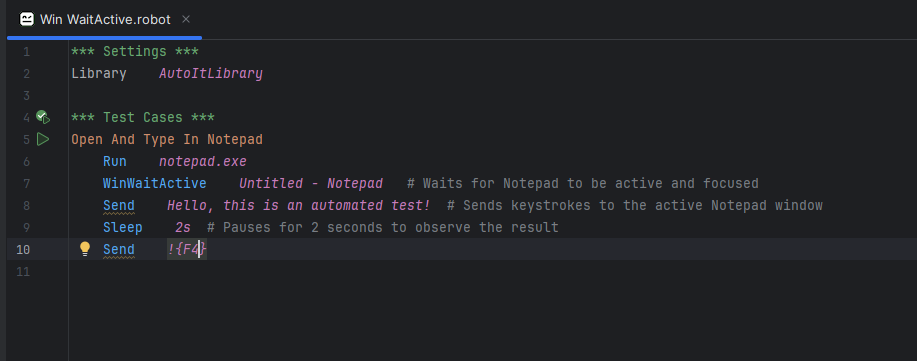
3. Open in Maximize Screen



11. WinWaitActive

Let's say you're automating the process of opening **Notepad** and typing some text in it. If you try to send keystrokes (text) to Notepad before it's fully opened or focused, the keystrokes might not go to the Notepad window, and you could encounter errors or unexpected behavior.

**WinWaitActive** ensures that the Notepad window is active and in the foreground before sending any keystrokes.

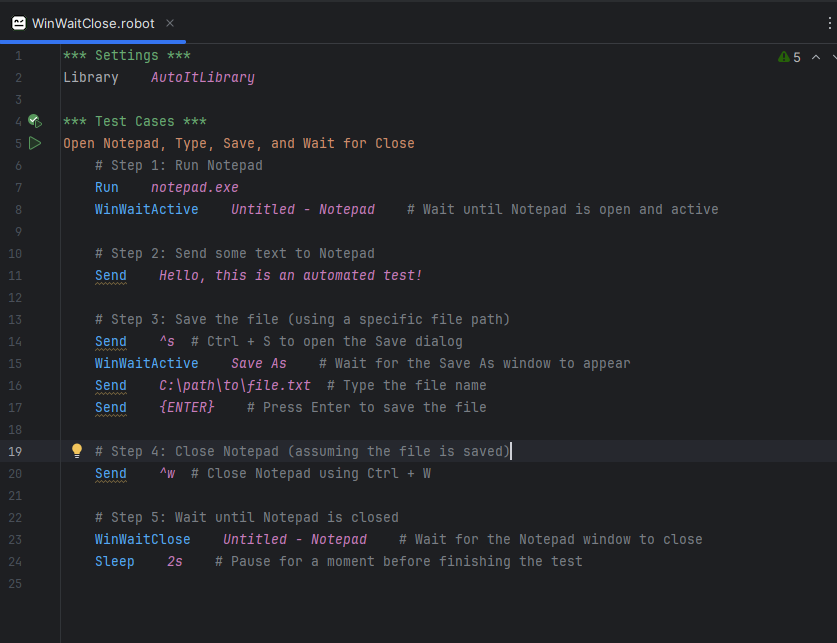


12. WinWaitClose

You would use **WinWaitClose** when your script is interacting with a window that could be closed as part of the task. For example, if you're automating the process of opening a program, performing some action, and then waiting for the program to close, you would use **WinWaitClose** to ensure the script waits until the window is fully closed before moving on to the next step.

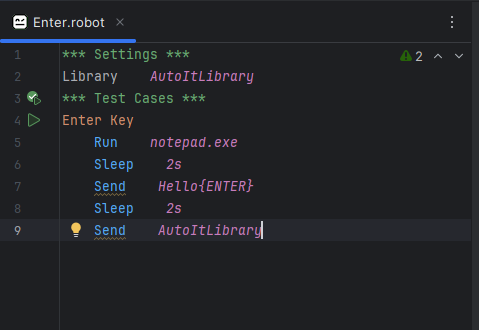
### Real-Life Example:

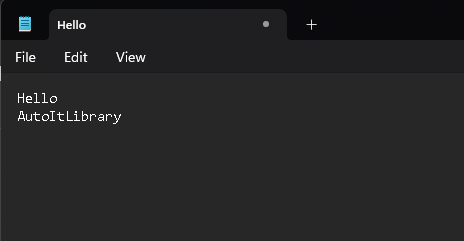
Let’s consider an example where you want to automate opening **Notepad**, typing some text, saving the file, and then waiting for **Notepad** to close before the script ends.



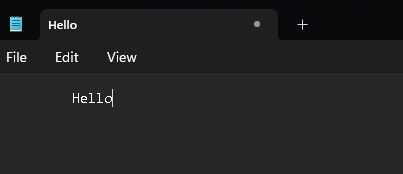
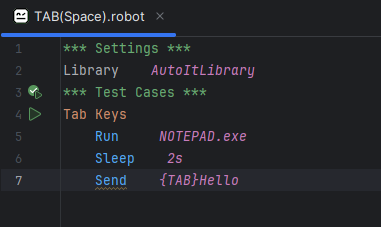
8. Special Keywords

1. Enter {ENTER}

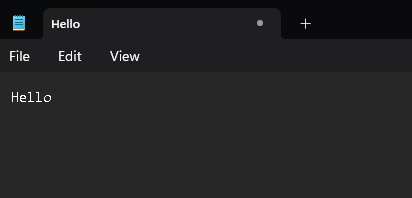
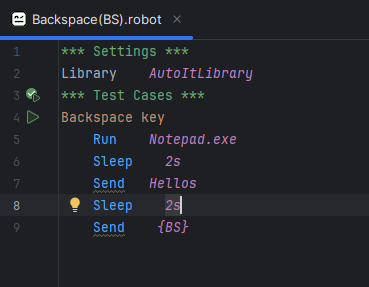




2. Tab {TAB} space

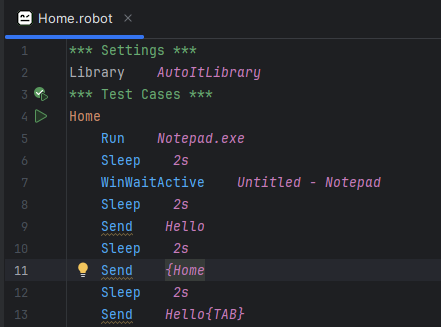


3. Backspace {BS}



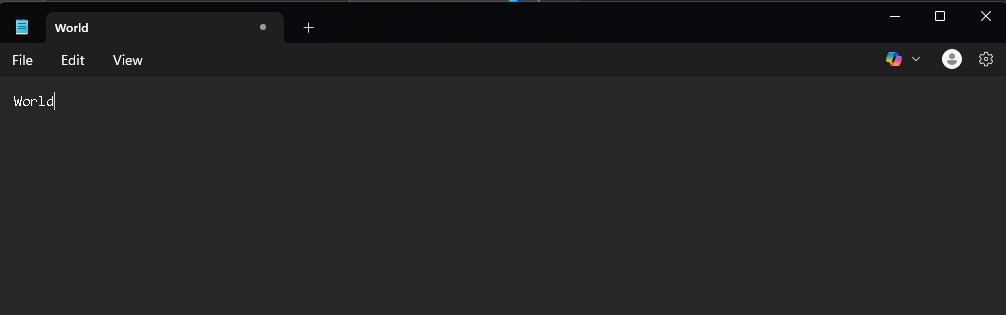
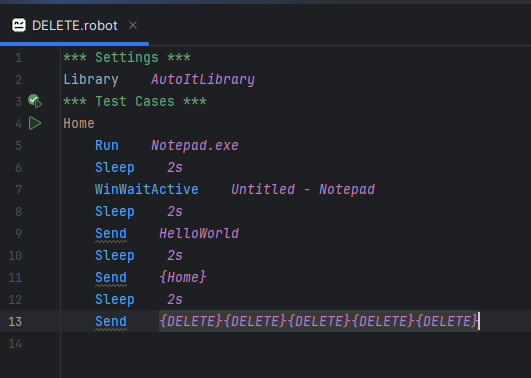
4. Home {Home}

Home Keyword move to start of line.



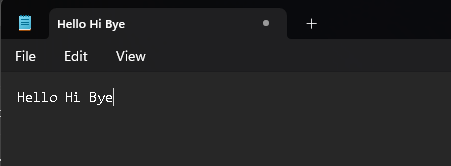
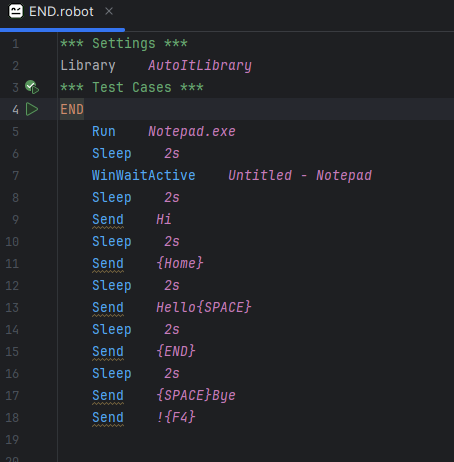
5. Delete {DELETE}

Delete Right

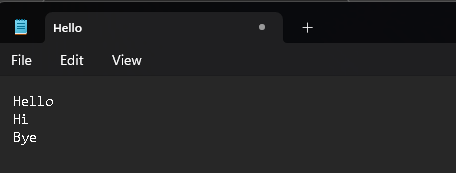
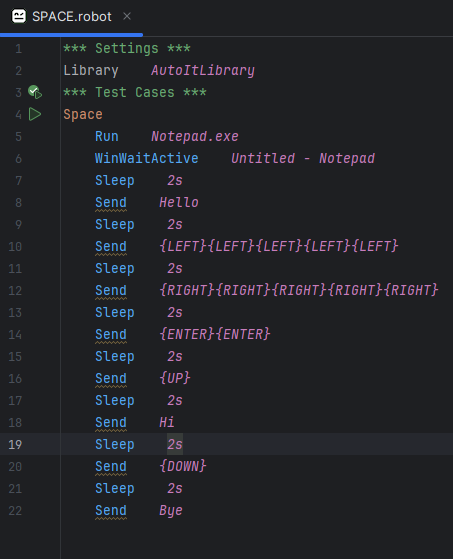


6. End {END}

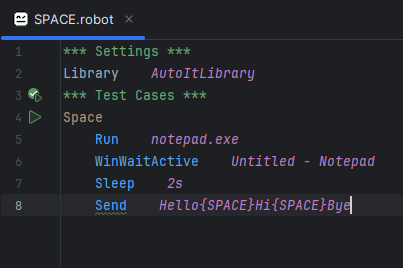
End (Move to end of line)

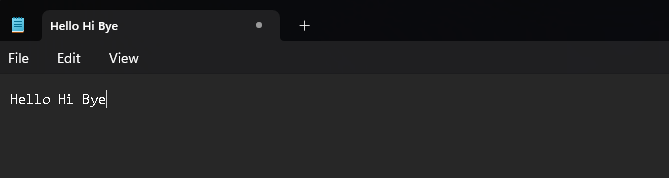


7. Arrow {UP}{DOWN}{RIGHT}{LEFT}

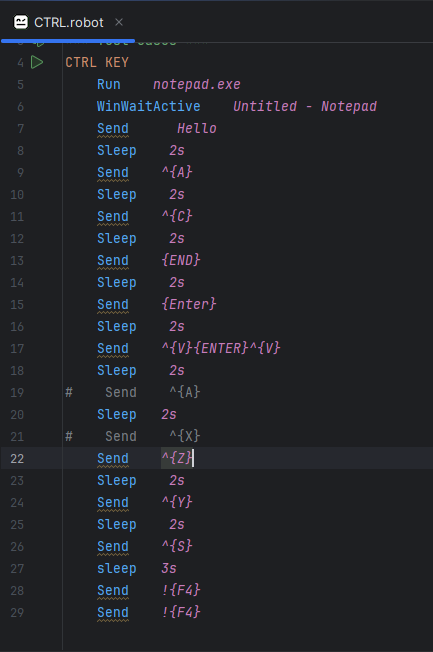


8. Space {SPACE}





9. Crtl



Keyboard Keys (Send Keyword)

When automating **Notepad** (or any Windows app) in Robot Framework using **AutoItLibrary**, you can simulate almost **any keyboard key** by using:

robot

Copy code

Send key\_sequence

👉 Commonly Used Keys in Notepad Automation:

| **Key Purpose** | **Command to Send** |
| --- | --- |
| Enter (new line) | {ENTER} |
| Tab (move focus) | {TAB} |
| Backspace (delete left) | {BACKSPACE} |
| Delete (delete right) | {DELETE} |
| Arrow Keys (navigation) | {UP}, {DOWN}, {LEFT}, {RIGHT} |
| Home (move to start of line) | {HOME} |
| End (move to end of line) | {END} |
| Ctrl + A (Select All) | ^{A} |
| Ctrl + C (Copy) | ^{C} |
| Ctrl + V (Paste) | ^{V} |
| Ctrl + X (Cut) | ^{X} |
| Ctrl + S (Save) | ^{S} |
| Ctrl + Z (Undo) | ^{Z} |
| Ctrl + Y (Redo) | ^{Y} |
| Alt + F4 (Close Window) | !{F4} |
| Ctrl + P (Print) | ^{P} |
| Shift + Arrow (Select text) | +{LEFT}, +{RIGHT} etc. |

Important Special Keys (Send)

1. {F1} to {F2} – Function keys

2. {ESC} – Escape

3. {PGUP}/{PGDN} – Page Up / Page Down

4. {NUMLOCK}/{SCROOLLOCK}/{CAPSLOCK} – Lock keys

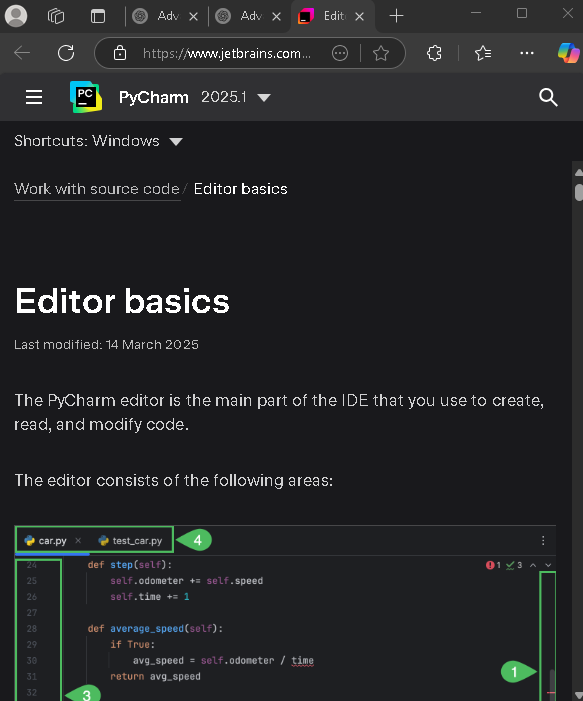
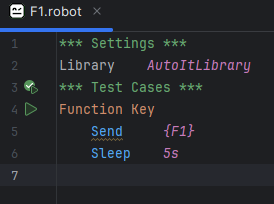
5. {PRINTSCREEN} – Screenshot key

6. Some typing special Character



1. F1 To F12

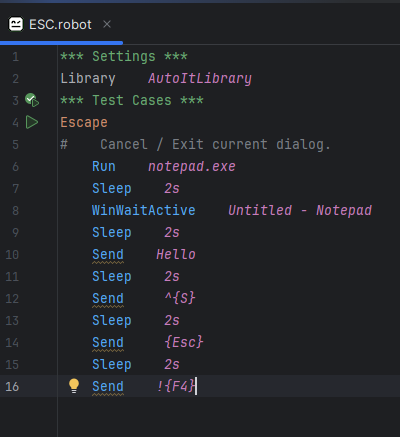
* F1 --- help



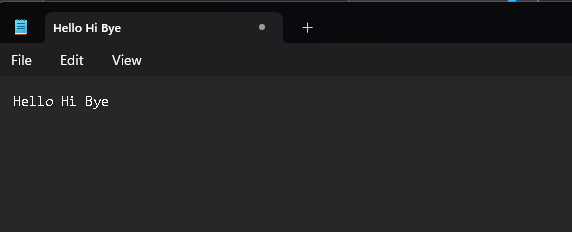
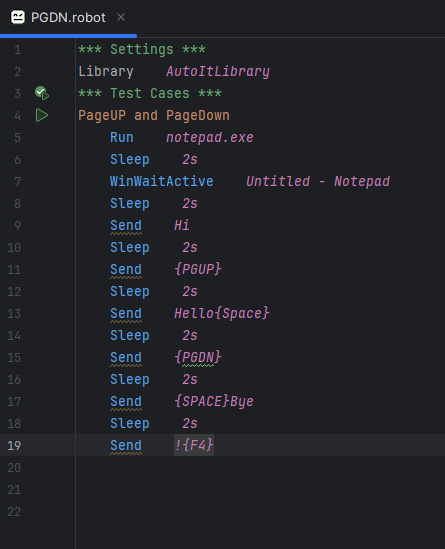
{F2}{F3}…….{F12}

2. ESC

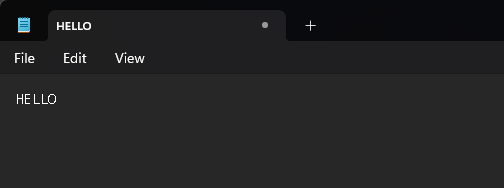
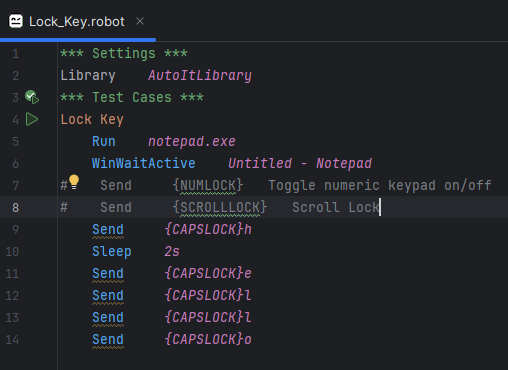
Closes popups like Save a file, Find, Replace or Save Confirmation



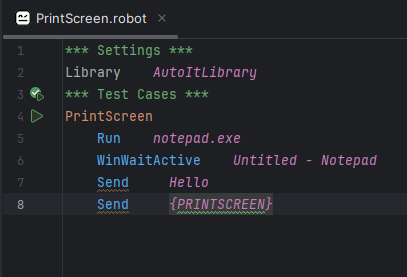
3. PageUp and PageDown



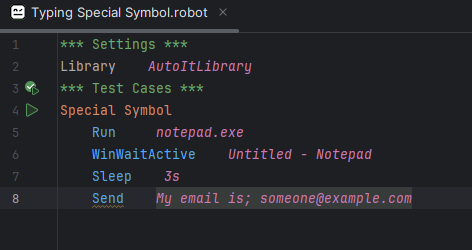
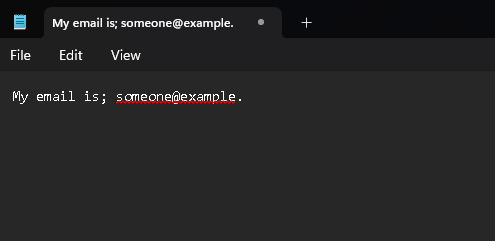
4. {NUMLOCK} / {SCROLLLOCK} / {CAPLOCK}



5. PRINTSCREEN



6. Some typing special Character



# Extra Advanced

| **Task** | **How to Do** |
| --- | --- |
| Simulate Ctrl+S (save) | Send ^{S} |
| Press Alt+File Menu (Alt+F) | Send !{F} |
| Press Esc key to dismiss | Send {ESC} |
| Shift + Tab (move backward) | Send +{TAB} |

# 4.Mouse Keyword

These keywords are used to si mulate mouse interactions like clicking, dragging, or moving the cursor useful for automating GUI elements that don’t have standard controls.

| **Keyword** | **Description** |
| --- | --- |

|  |  |
| --- | --- |
| MouseClick | Clicks at a screen coordinate |

|  |  |
| --- | --- |
| MouseClickDrag | Click and drag from point A to point B |

|  |  |
| --- | --- |
| MouseMove | Moves the mouse cursor to a coordinate |

|  |  |
| --- | --- |
| MouseDown | Simulates pressing a mouse button down |

|  |  |
| --- | --- |
| MouseUp | Simulates releasing a mouse button |

|  |  |
| --- | --- |
| MouseGetPos | Returns the current position of the cursor |

1. MouseClick

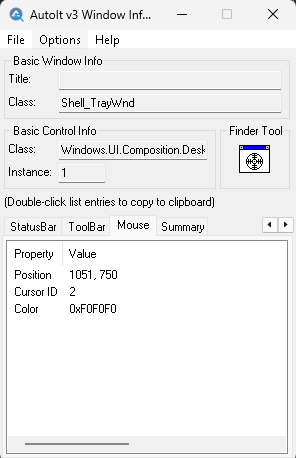
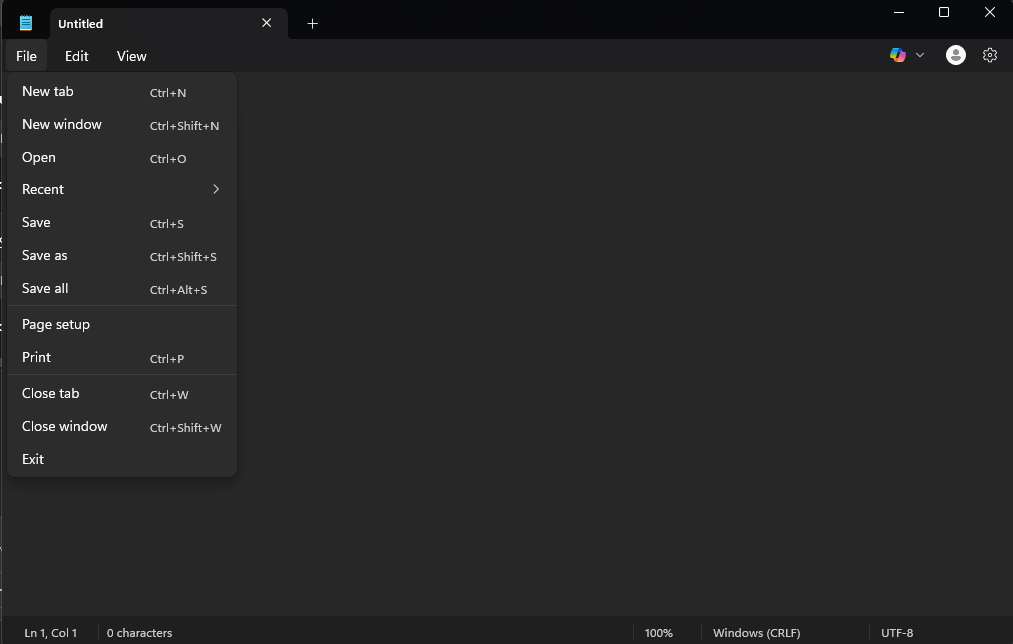
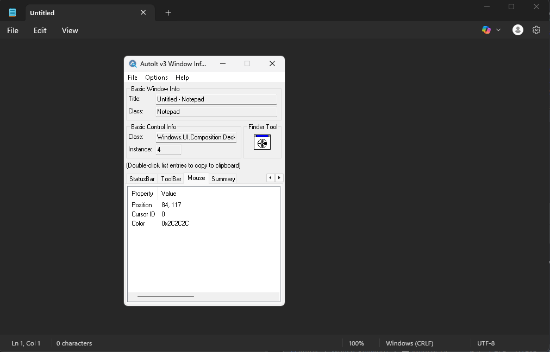
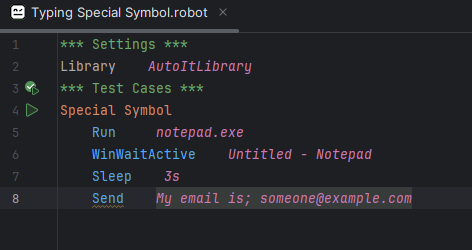
Syntax

MouseCLick <Button> <x> <y> <clicks=1> <speed=10>

<button> -- left, right, middle

<x>,<y> -- Coorinates

<clicks> -- how many times to click (default is 1)

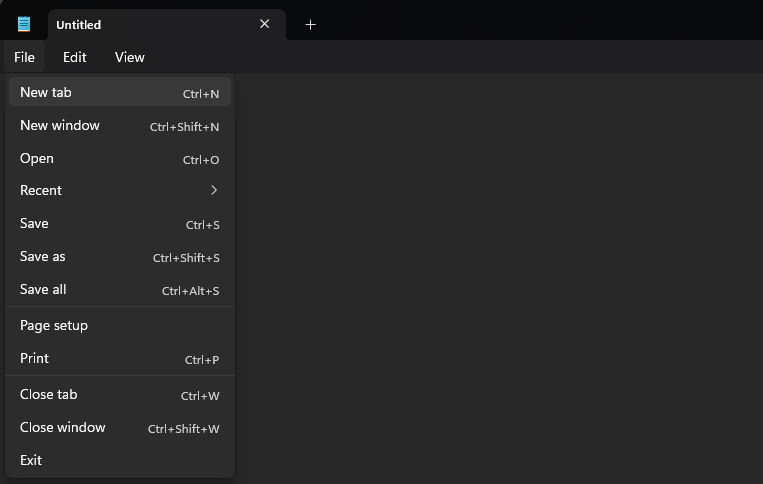
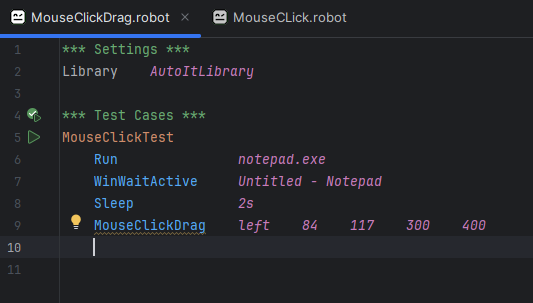
<speed> -- speed of mouse movement (-1 default)

2. MouseClickDrag

Clicks and drags from one point to another.

Syntax

MouseClickDrag <button> <x1> <y1> <x2> <y2> <speed=10>

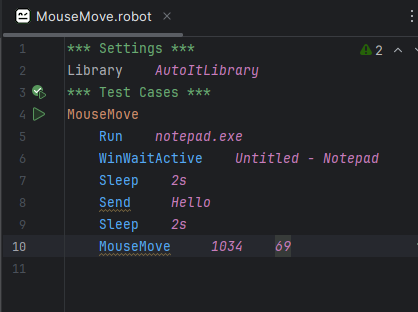
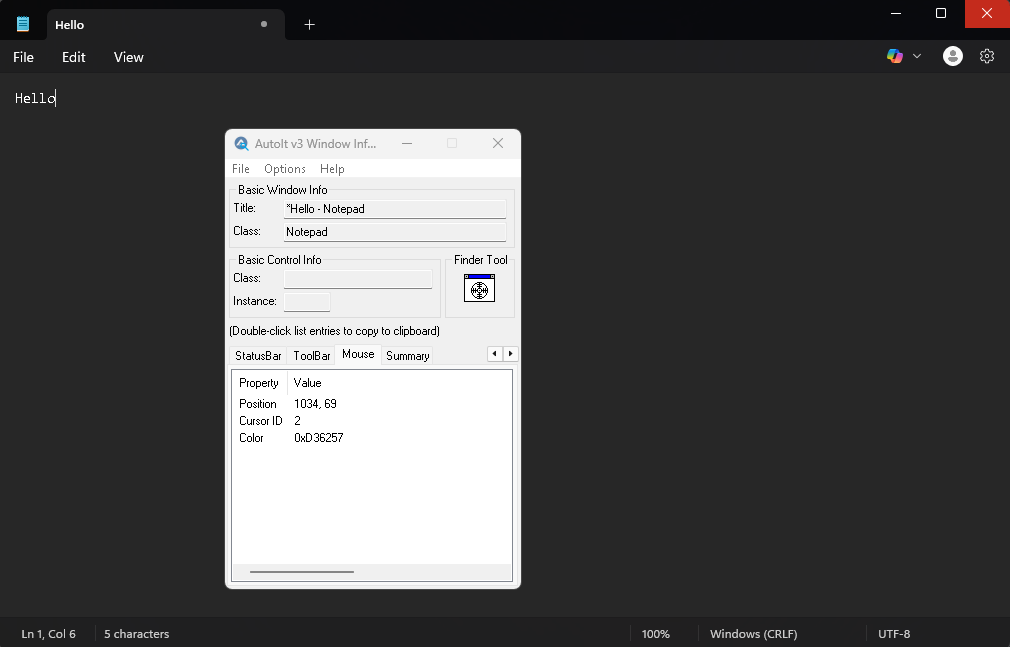


3. MouseMove

Moves the cursor to the specified position

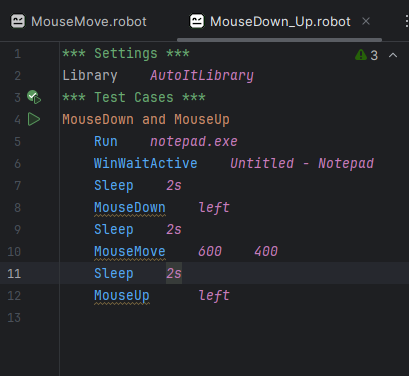
Syntax

MouseMove <x> <y> <speed=-1>



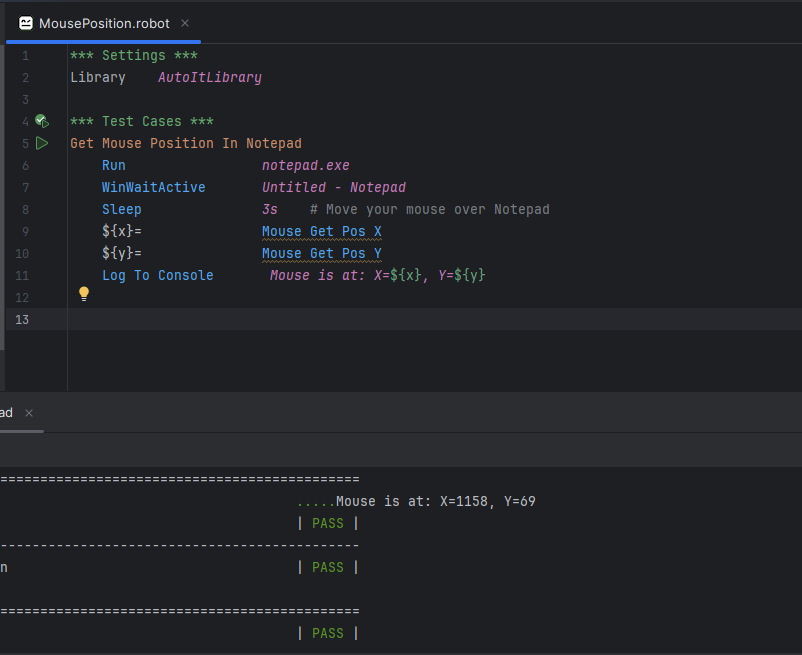
4. MouseDown and MouseUp

Simulates pressing and releasing mouse buttons (useful for drag/drop)



5. MouseGetPos

It retrives the current x,y screen coordinates of the mouse pointer and returns them as a list.



Window Management Actions

Control Actions

Mouse Actions

Keyboard action

Utility & Waits

# 5. Control Action

A Control Action is a command used to interact directly with GUI element (Controls) like button, textboxes, list boxes, checkboxes, etc. without using the mouse or keyboard.

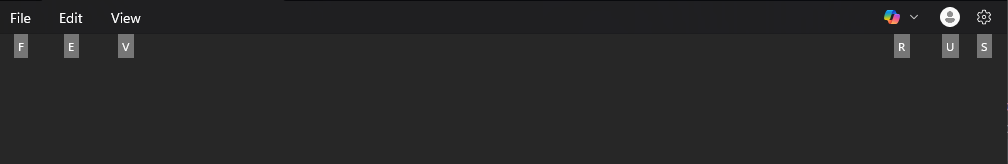
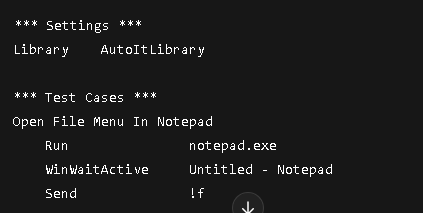
Why Use Control Actions?

| **Advantage** |  | **Description** |
| --- | --- | --- |
| 🎯 **Precision** |  | Works by referencing control IDs instead of screen coordinates |
| 🎯 **Background Operation** |  | Can work even when the window is not visible or active |
| 🎯 **No Resolution Dependency** |  | Works regardless of screen resolution or position |
| 🎯 **Faster & Reliable** |  | Faster than simulating keystrokes or mouse movements |

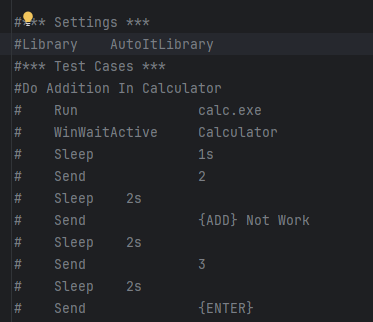
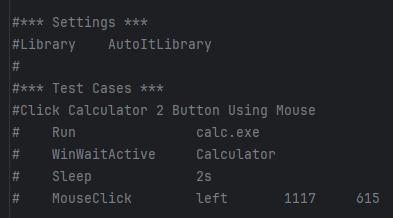
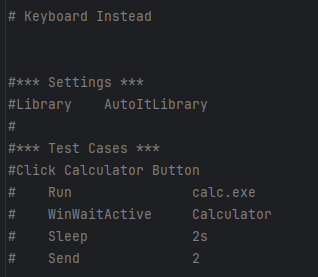
# Some Shortcuts

But File4 is not a valid ControlID for the”File” menu. The Notepad menu items (like File, Edit, etc.) are part of a Menu Bar, and AutoIt Control Click does not work on menu items directly.

We Use Send Alt+F4 or !f



Normal Method This



## When Should You Use Control Actions?

✅ Best used for **classic desktop apps** (Win32):

* Notepad
* WordPad
* Registry Editor
* Any custom apps with standard Windows controls

❌ Not ideal for **UWP apps** (e.g., modern Calculator) or **web apps** — use Send, MouseClick, or other libraries for them.

| **Control Type** | **Control Name (ClassNameNN)** | **Example** |
| --- | --- | --- |

|  |  |  |
| --- | --- | --- |
| **Edit Box** | Edit1, Edit2 | Text inputs |

|  |  |  |
| --- | --- | --- |
| **Button** | Button1, Button2 | Buttons |

|  |  |  |
| --- | --- | --- |
| **Combo Box** | ComboBox1 | Dropdowns |

|  |  |  |
| --- | --- | --- |
| **List View** | SysListView321 | File lists |

|  |  |  |
| --- | --- | --- |
| **Static Text** | Static1, Static2 | Labels |

|  |  |  |
| --- | --- | --- |
| **Checkbox** | Button3 | Checkboxes |

|  |  |  |
| --- | --- | --- |
| **Radio Button** | Button4 | Radio buttons |

|  |  |  |
| --- | --- | --- |
| **Tab Control** | SysTabControl321 | Tabs |

UMP

UMP (Univrsal Windows Platform) version of notepad (like on Windows 11), then

COntrolSetText, ControlClick and Similar AutoItLibrary method do not work reliable on UMP apps.

Why?

UMP app like the modern Windows11 Notepad, Calculator, etc.,

Use a new rendering engine like XAML, DirectX or D3D

Their UI controls aren’t traditional Win32 controls

AutoIt connot see or manipulates these controls commands

Use Optional

1. Send

2. Use UIAutomation or pywinauto for UWP

### How to Check If an App Is UWP:

1. Launch the app (e.g., Notepad).
2. Open AutoIt Window Info Tool.
3. Drag the Finder to the window.
4. If it shows:
   * Class: **ApplicationFrameWindow**
   * OR Intermediate D3D Window → it's a UWP app.

# 4. Advance AutoItLibrary automation techniques