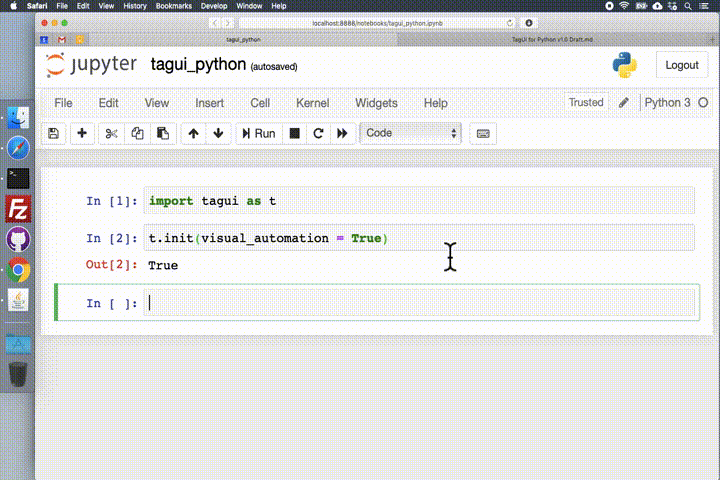
**RPA for Python 🐍**

[**v1.50**](https://github.com/tebelorg/RPA-Python/releases) • [**Use Cases**](https://github.com/tebelorg/RPA-Python#use-cases) • [**API Reference**](https://github.com/tebelorg/RPA-Python#api-reference) • [**About & Credits**](https://github.com/tebelorg/RPA-Python#about--credits) • [**Try on Cloud**](https://colab.research.google.com/drive/1or8DtXZP8ZxJYK52me0dA6O9A1dXKKOE?usp=sharing) • [**PyCon Video**](https://www.youtube.com/watch?v=F2aQKWx_EAE) • [**Telegram Chat**](https://t.me/pythonrpa) • [*中文*](https://github-com.translate.goog/tebelorg/RPA-Python?_x_tr_sl=en&_x_tr_tl=zh-CN&_x_tr_hl=en-US&_x_tr_pto=wapp) • [*हिन्दी*](https://github-com.translate.goog/tebelorg/RPA-Python?_x_tr_sl=en&_x_tr_tl=hi&_x_tr_hl=en-US&_x_tr_pto=wapp) • [*Español*](https://github-com.translate.goog/tebelorg/RPA-Python?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=en-US&_x_tr_pto=wapp) • [*Français*](https://github-com.translate.goog/tebelorg/RPA-Python?_x_tr_sl=en&_x_tr_tl=fr&_x_tr_hl=en-US&_x_tr_pto=wapp) • [*عربى*](https://github-com.translate.goog/tebelorg/RPA-Python?_x_tr_sl=en&_x_tr_tl=ar&_x_tr_hl=en-US&_x_tr_pto=wapp) • [*বাংলা*](https://github-com.translate.goog/tebelorg/RPA-Python?_x_tr_sl=en&_x_tr_tl=bn&_x_tr_hl=en-US&_x_tr_pto=wapp) • [*Русский*](https://github-com.translate.goog/tebelorg/RPA-Python?_x_tr_sl=en&_x_tr_tl=ru&_x_tr_hl=en-US&_x_tr_pto=wapp) • [*Português*](https://github-com.translate.goog/tebelorg/RPA-Python?_x_tr_sl=en&_x_tr_tl=pt&_x_tr_hl=en-US&_x_tr_pto=wapp) • [*Bahasa*](https://github-com.translate.goog/tebelorg/RPA-Python?_x_tr_sl=en&_x_tr_tl=id&_x_tr_hl=en-US&_x_tr_pto=wapp) • [*Deutsch*](https://github-com.translate.goog/tebelorg/RPA-Python?_x_tr_sl=en&_x_tr_tl=de&_x_tr_hl=en-US&_x_tr_pto=wapp) • [*More..*](https://github-com.translate.goog/tebelorg/RPA-Python?_x_tr_sl=en&_x_tr_tl=sr&_x_tr_hl=en-US&_x_tr_pto=wapp)

[](https://raw.githubusercontent.com/tebelorg/Tump/master/tagui_python.gif)

To install this Python package for RPA (robotic process automation) -

pip install rpa

To use it in Jupyter notebook, Python script or interactive shell -

import rpa as r

Notes on operating systems and optional visual automation mode -

* 🏳️‍🌈 **Windows -** if visual automation is faulty, try setting your display zoom level to recommended % or 100%
* 🍎 **macOS -** due to tighter security, [install PHP manually](https://github.com/tebelorg/RPA-Python/issues/335#issuecomment-989470056) and see solutions for [PhantomJS](https://github.com/tebelorg/RPA-Python/issues/79) and [Java popups](https://github.com/tebelorg/RPA-Python/issues/78)
* 🐧 **Linux -** visual automation mode requires special setup on Linux, see how to [install OpenCV and Tesseract](https://sikulix-2014.readthedocs.io/en/latest/newslinux.html)
* 🍇 **Raspberry Pi -** [use this setup guide](https://www.techgence.com/d/29-install-rpa-python-on-raspberry-pi-updated-2022) to run the package on Raspberry Pies (low-cost automation servers)

**Use Cases**

RPA for Python's simple and powerful API makes robotic process automation fun! You can use it to quickly automate away repetitive time-consuming tasks on websites, desktop applications, or the command line.

As a bonus and token of my appreciation, any new bug reported will be appreciated with a US$200 gift card from your preferred merchant. Any feature suggestion accepted will be appreciated with a US$100 gift card.

**WEB AUTOMATION**

r.init()

r.url('https://duckduckgo.com')

r.type('//\*[@name="q"]', 'decentralisation[enter]')

r.wait() # ensure results are fully loaded

r.snap('page', 'results.png')

r.close()

**VISUAL AUTOMATION**

r.init(visual\_automation = True)

r.dclick('outlook\_icon.png')

r.click('new\_mail.png')

...

r.type('message\_box.png', 'Hi Gillian,[enter]This is ...')

r.click('send\_button.png')

r.close()

**OCR AUTOMATION**

r.init(visual\_automation = True, chrome\_browser = False)

print(r.read('pdf\_report\_window.png'))

print(r.read('image\_preview.png'))

r.hover('anchor\_element.png')

print(r.read(r.mouse\_x(), r.mouse\_y(), r.mouse\_x() + 400, r.mouse\_y() + 200))

r.close()

**KEYBOARD AUTOMATION**

r.init(visual\_automation = True, chrome\_browser = False)

r.keyboard('[cmd][space]')

r.keyboard('safari[enter]')

r.keyboard('[cmd]t')

r.keyboard('snatcher[enter]')

r.wait(2.5)

r.snap('page.png', 'results.png')

r.close()

**MOUSE AUTOMATION**

r.init(visual\_automation = True)

r.type(600, 300, 'neo kobe city')

r.click(900, 300)

r.snap('page.png', 'results.png')

r.hover('button\_to\_drag.png')

r.mouse('down')

r.hover(r.mouse\_x() + 300, r.mouse\_y())

r.mouse('up')

r.close()

**TELEGRAM NOTIFICATION**

*first, look up @rpapybot on your Telegram app to approve receiving messages*

r.telegram('1234567890', 'ID can be string or number, r.init() is not required')

r.telegram(1234567890, 'Hello World. Olá Mundo. नमस्ते दुनिया. 안녕하세요 세계. 世界,你好。')

r.telegram(1234567890, 'Use backslash n for new line\nThis is line 2 of the message')

**SECURE TEMPORARY STORAGE**

*securely share files up to 100 MB on PrivateBin, which will self-destruct after 1 week*

bin\_url = r.bin('secret\_agent\_report.pdf', 'optional password')

r.telegram(1234567890, 'Access confidential report at ' + bin\_url)

**API Reference**

[**Notes**](https://github.com/tebelorg/RPA-Python#general-notes) • [**Element Identifiers**](https://github.com/tebelorg/RPA-Python#element-identifiers) • [**Core Functions**](https://github.com/tebelorg/RPA-Python#core-functions) • [**Basic Functions**](https://github.com/tebelorg/RPA-Python#basic-functions) • [**Pro Functions**](https://github.com/tebelorg/RPA-Python#pro-functions) • [**Helper Functions**](https://github.com/tebelorg/RPA-Python#helper-functions)

**GENERAL NOTES**

See [sample Python script](https://github.com/tebelorg/RPA-Python/blob/master/sample.py), the [RPA Challenge solution](https://github.com/tebelorg/RPA-Python/issues/120#issuecomment-610518196), and [RedMart groceries example](https://github.com/tebelorg/RPA-Python/issues/24). To send a Telegram app notification, simply [look up @rpapybot](https://github.com/tebelorg/RPA-Python/issues/281#issue-942803794) to allow receiving messages. To automate Chrome browser invisibly, use [headless mode](https://github.com/tebelorg/RPA-Python/issues/240#issuecomment-839981773). To run 10X faster instead of normal human speed, use [turbo mode](https://github.com/tebelorg/RPA-Python/issues/297) (read the caveats!). Some CAPTCHAs can be solved using services like [2Captcha](https://2captcha.com/), [Capsolver](https://www.capsolver.com/) or directly by [replicating user actions](https://github.com/tebelorg/RPA-Python/issues/399#issuecomment-1163879428).

[Securely share files](https://github.com/tebelorg/RPA-Python/issues/396#issuecomment-1169409452) up to 100 MB with built-in temporary online storage, on a dedicated [PrivateBin server](https://tebel.org/bin/). You can even run RPA on your phone browser [using this Colab notebook](https://colab.research.google.com/drive/1or8DtXZP8ZxJYK52me0dA6O9A1dXKKOE?usp=sharing) (eg datascraping with up to 5 Colab sessions). By design this package has [enterprise security](https://github.com/aisingapore/TagUI/blob/master/README.md#enterprise-security-by-design) and you can install, update and use it [without the internet](https://github.com/tebelorg/RPA-Python/issues/36#issuecomment-543670292).

Fully control error handling by [setting error(True)](https://github.com/tebelorg/RPA-Python/issues/299#issuecomment-1110361923) to raise Python exception on error, and manage with try-except. For fine-grained control on web browser file download location, use [download\_location()](https://github.com/tebelorg/RPA-Python/issues/279#issuecomment-877749880). For overriding default folder location to install and invoke TagUI (a [forked version](https://github.com/tebelorg/TagUI) optimised for rpa package), use [tagui\_location()](https://github.com/tebelorg/RPA-Python/issues/257#issuecomment-846602776).

If you are using non-English operating system and get "invalid continuation byte" error, you can set code page to support UTF-8 or change your Python script's encoding to your OS encoding. [See this example for Chinese](https://github.com/tebelorg/RPA-Python/issues/451#issuecomment-1556169481). Use focus() to make Windows/Mac application windows to be in focus (see here for [pywin32 alternative](https://github.com/tebelorg/RPA-Python/issues/478#issuecomment-1653117053)).

**ELEMENT IDENTIFIERS**

An element identifier helps to tell RPA for Python exactly which element on the user interface you want to interact with. For example, //\*[@id='email'] is an XPath pointing to the webpage element having the id attribute 'email'.

* 🌐 For web automation, the web element identifier can be XPath selector, CSS selector, or the following attributes - id, name, class, title, aria-label, text(), href, in decreasing order of priority. Recommend writing XPath manually or simply using attributes. There is automatic waiting for an element to appear before timeout happens, and error is returned that the element cannot be found. To change the default timeout of 10 seconds, use timeout(). PS - if you are using a Chrome extension to read XPaths, use [SelectorsHub](https://chrome.google.com/webstore/detail/selectorshub/ndgimibanhlabgdgjcpbbndiehljcpfh?hl=en).
* 📸 An element identifier can also be a .png or .bmp image snapshot representing the UI element (can be on desktop applications, terminal window or web browser). If the image file specified does not exist, OCR will be used to search for that text on the screen to act on the UI element containing the text, eg r.click('Submit Form.png'). Transparency (0% opacity) is supported in .png images. x, y coordinates of elements on the screen can be used as well. Notes for visually [automating 2 monitors](https://github.com/tebelorg/RPA-Python/issues/252#issuecomment-844277454), and macOS [Retina display issue](https://github.com/tebelorg/RPA-Python/issues/170#issuecomment-843168745).
* 📄 A further image identifier example is a png image of a window (PDF viewer, MS Word, textbox etc) with the center content of the image set as transparent. This allows using read() and snap() to perform OCR and save snapshots of application windows, containers, frames, textboxes with varying content. See this [image example](https://user-images.githubusercontent.com/10379601/124394598-b59cfd80-dd32-11eb-93bb-68504c91afb9.png) of a PDF frame with content removed to be transparent. For read() and snap(), x1, y1, x2, y2 coordinates pair can be used to define the region of interest on the screen to perform OCR or capture snapshot.

**CORE FUNCTIONS**

| **Function** | **Parameters** | **Purpose** |
| --- | --- | --- |
| init() | visual\_automation=False,chrome\_browser=True | start TagUI, auto-setup on first run |
| close() |  | close TagUI, Chrome browser, SikuliX |
| pack() |  | for deploying package without internet |
| update() |  | for updating package without internet |
| error() | True or False | set to True to raise exception on error |
| debug() | True or False or text\_to\_log | print & log debug info to rpa\_python.log |

*by default RPA for Python runs at normal human speed, to run 10X faster use init(turbo\_mode = True)*

**BASIC FUNCTIONS**

| **Function** | **Parameters** | **Purpose** |
| --- | --- | --- |
| url() | webpage\_url (no parameter to return current URL) | go to web URL |
| click() | element\_identifier (or x, y using visual automation) | left-click on element |
| rclick() | element\_identifier (or x, y using visual automation) | right-click on element |
| dclick() | element\_identifier (or x, y using visual automation) | double-click on element |
| hover() | element\_identifier (or x, y using visual automation) | move mouse to element |
| type() | element\_identifier (or x, y), text ('[enter]'/'[clear]') | enter text at element |
| select() | element\_identifier (or x, y), value or text (or x, y) | choose dropdown option |
| read() | element\_identifier ('page' is web page) (or x1, y1, x2, y2) | return element text |
| snap() | element\_identifier ('page' is web page), filename\_to\_save | save screenshot to file |
| load() | filename\_to\_load | return file content |
| dump() | text\_to\_dump, filename\_to\_save | save text to file |
| write() | text\_to\_write, filename\_to\_save | append text to file |
| ask() | text\_to\_prompt | ask & return user input |

*to wait for an element to appear until timeout() value, use hover(). to drag-and-drop,*[*do it this way*](https://github.com/tebelorg/RPA-Python/issues/58#issuecomment-570778431)

**PRO FUNCTIONS**

| **Function** | **Parameters** | **Purpose** |
| --- | --- | --- |
| telegram() | telegram\_id, text\_to\_send (first look up @rpapybot) | send Telegram message |
| keyboard() | keys\_and\_modifiers (using visual automation) | send keystrokes to screen |
| mouse() | 'down' or 'up' (using visual automation) | send mouse event to screen |
| focus() | app\_to\_focus (full name of app) | make application in focus |
| wait() | delay\_in\_seconds (default 5 seconds) | explicitly wait for some time |
| table() | table number or XPath, filename\_to\_save | save webpage table to CSV |
| bin() | file\_to\_bin, password (optional but recommended) | secure temporary storage |
| upload() | element\_identifier (CSS), filename\_to\_upload | upload file to web element |
| download() | download\_url, filename\_to\_save (optional) | download from URL to file |
| unzip() | file\_to\_unzip, unzip\_location (optional) | unzip zip file to specified location |
| frame() | main\_frame id or name, sub\_frame (optional) | set web frame, frame() to reset |
| popup() | string\_in\_url (no parameter to reset to main page, especially important when used to control another browser tab) | set context to web popup tab |
| run() | command\_to\_run (use ; between commands) | run OS command & return output |
| dom() | statement\_to\_run (JS code to run in browser) | run code in DOM & return output |
| vision() | command\_to\_run (Python code for SikuliX) | run custom SikuliX commands |
| timeout() | timeout\_in\_seconds (blank returns current timeout) | change wait timeout (default 10s) |

keyboard() modifiers and special keys -

*[shift] [ctrl] [alt] [win] [cmd] [clear] [space] [enter] [backspace] [tab] [esc] [up] [down] [left] [right] [pageup] [pagedown] [delete] [home] [end] [insert] [f1] .. [f15] [printscreen] [scrolllock] [pause] [capslock] [numlock]*

**HELPER FUNCTIONS**

| **Function** | **Parameters** | **Purpose** |
| --- | --- | --- |
| exist() | element\_identifier | True or False if element shows before timeout |
| present() | element\_identifier | return True or False if element is present now |
| count() | element\_identifier | return number of web elements as integer |
| clipboard() | text\_to\_put or no parameter | put text or return clipboard text as string |
| get\_text() | source\_text,left,right,count=1 | return text between left & right markers |
| del\_chars() | source\_text,characters | return text after deleting given characters |
| mouse\_xy() |  | return '(x,y)' coordinates of mouse as string |
| mouse\_x() |  | return x coordinate of mouse as integer |
| mouse\_y() |  | return y coordinate of mouse as integer |
| title() |  | return page title of current web page as string |
| text() |  | return text content of current web page as string |
| timer() |  | return time elapsed in sec between calls as float |

*to type a large amount of text quickly, use clipboard() and keyboard() to paste instead of type()*

**About & Credits**

TagUI is a leading open-source RPA software 🤖 with tens of thousands of users. It was created in 2016-2017 when I left DBS Bank as a test automation engineer, for a one-year sabbatical to Eastern Europe. Most of its code base was written in Novi Sad Serbia. In 2018, I joined AI Singapore to continue development of TagUI.

Over a few months in 2019, I took on a daddy role full-time, taking care of my newborn baby girl and wife 🤠🤱. In between nannying, I used my time pockets to create this Python package built on TagUI. I hope pip install rpa would make life easier for Python users from different walks of life.

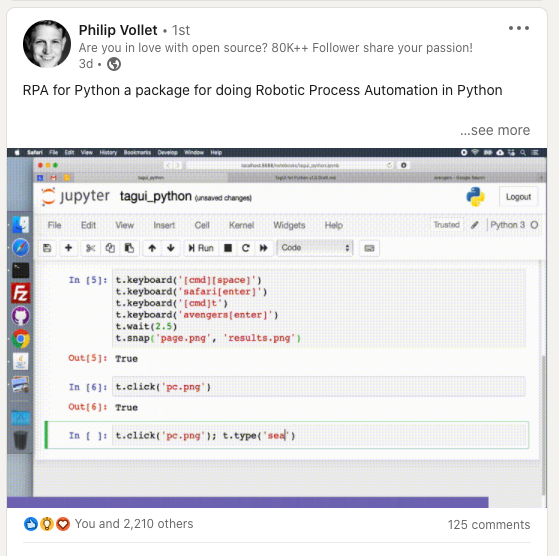
I had been maintaining the package (and a [forked version of TagUI](https://github.com/tebelorg/TagUI) optimised for it) in my personal time. But [Marcelo Cecin](https://www.linkedin.com/in/marcelocecin/), [Luis Alejandro](https://www.linkedin.com/in/luis-alejandro/), [Jozsef Fulop](https://www.linkedin.com/in/jozseffulop86/), [Tolani Jaiye-Tikolo](https://www.linkedin.com/in/tolani-jaiye-tikolo/), [Shyan Chua](https://www.linkedin.com/in/shyanchua/), [Laurence Liew](https://www.linkedin.com/in/laurenceliew/), [myself](https://www.linkedin.com/in/kensoh/), will be the new team maintaining this package. We're happy that tens of thousands of people use it 🐍

For technical info, see its intuitive architecture below and ample comments in this [single-file package](https://github.com/tebelorg/RPA-Python/blob/master/tagui.py).

[RPA for Python architecture](https://raw.githubusercontent.com/tebelorg/Tump/master/TagUI-Python/architecture.png)

I would like to credit and express my appreciation to these amazing open-source contributors below ❤️

* [TagUI](https://github.com/aisingapore/TagUI) - AI Singapore from Singapore / [@aisingapore](https://www.aisingapore.org/)
* [SikuliX](https://github.com/RaiMan/SikuliX1) - Raimund Hocke from Germany / [@RaiMan](https://github.com/RaiMan)
* [CasperJS](https://github.com/casperjs/casperjs) - Nicolas Perriault from France / [@n1k0](https://github.com/n1k0)
* [PhantomJS](https://github.com/ariya/phantomjs) - Ariya Hidayat from Indonesia / [@ariya](https://github.com/ariya)
* [SlimerJS](https://github.com/laurentj/slimerjs) - Laurent Jouanneau from France / [@laurentj](https://github.com/laurentj)
* [Philip Vollet](https://www.linkedin.com/in/philipvollet) from Germany, for spreading the word. Philip is a veteran in NLP and open-source. His sharing of RPA for Python helps spread the word to the vast and lovely open-source community about [pip install rpa](https://www.linkedin.com/posts/philipvollet_datascience-deeplearning-machinelearning-activity-6884853626183938048-Eqg3).

[](https://raw.githubusercontent.com/tebelorg/Tump/master/philip_vollet.png)

**License**

RPA for Python is open-source software released under Apache 2.0 license

**One Last Thing.. Mindly**

I rarely make product recommendations, other than the [amazing OpenRPA software](https://github.com/open-rpa/openrpa), and the open-source RPA tools I personally worked on. I'd like to recommend [Mindly mindmapping app](https://www.mindlyapp.com/) available on phone and macOS.

A mindmap is an intuitive way to store, organise and retrieve info, as it mimics how the mind works - relationships between different concepts and memories. It's perfect to make productive use of time pockets on the go.

Below image is a Mindly example on benefits of coffee. I personally use it to map out my life for the next 13 years, reflect how to be a better husband, keep a list of traditional British foods, store supermarket member barcodes, as well as note-taking on the go. There's even a mindmap for my 3YO daughter to play with, she just enjoys dragging the nodes into the bin. So I created a dummy mindmap on standby that she can destroy.

Best of all, the free version should meet the needs of most users. I have not exceeded the free limit of 100-node per mindmap, but I purchased it quite early on after using it, to support the work of the team behind this app.

PS - I don't know Mindly's team, just recommending the app here because it rocks

[](https://raw.githubusercontent.com/tebelorg/Tump/master/mindly_app.png)