**Robot Framework**

Robot Framework is an open-source automation framework for acceptance testing and robotic process automation (RPA). It is an application and platform independent project with a growing ecosystem of external tools and libraries. The source code for this framework can be accessed publicly through GitHub and the maintainers provide guidelines on how anyone can [contribute to it](https://github.com/robotframework/robotframework/blob/master/CONTRIBUTING.rst). Contributions to the framework can also be done in other ways too. Such as helping with the various tools and libraries displayed on [Robot Framework’s website](https://robotframework.org/), the curated [awesome-robotframework list](https://github.com/fkromer/awesome-robotframework), or the [Robot Framework Market Square](https://github.com/MarketSquare).

[**Robot Framework**](https://robotframework.org/) is a generic, Python-based, open-source automation framework. It can be used for test automation and robotic process automation (RPA).

Robot Framework is supported by [Robot Framework Foundation](https://robotframework.org/foundation/). Many industry-leading companies use the tool in their software development.

**Robot Framework Language Server**

The Robot Framework Language Server extension provides code completion, syntax validation and highlighting, code formatting, and other powerful robot development features.

**Robocorp Code**

With Robocorp Code extension, you can create a new robot, run it locally, and publish it to Robocorp Control Room right from your favorite editor.

**RPA Framework**

[**RPA Framework**](https://rpaframework.org/) is a collection of open-source libraries and tools for Robotic Process Automation (RPA), and it is designed to be used with both [Robot Framework](https://robotframework.org/) and [Python](https://www.python.org/). The goal is to offer well-documented and actively maintained core libraries for Software Robot Developers.

The project is:

* 100% open-source
* **Sponsored by**[**Robocorp**](https://robocorp.com/)
* Optimized for [Robocorp Control Room](https://robocorp.com/docs/control-room) and [developer tools](https://robocorp.com/download)
* Accepting external contributions

**Robocorp** provides tools to write, execute and orchestrate software robots that are powered by Robot Framework to be used in RPA, so understanding the basics is fundamental for any Software Robot Developer.

If you are completely new to Robot Framework and its use in RPA, we recommend taking our [Beginners' course](https://robocorp.com/docs/courses/beginners-course), where we build a robot from start to finish. This article will provide only a very quick introduction to the main concepts.

**Training course and Certifications** - <https://robocorp.com/docs/courses>

(중요: 실체를 이해하는 데 필요 과정)

Completing the 1) [Beginners' course](https://robocorp.com/docs/courses/beginners-course) will grant you the Robocorp Level I certificate! If you want more challenge, try the Level II certificate course, 2) [Build a robot](https://robocorp.com/docs/courses/build-a-robot)! Learn even more by taking the level III certificate course, 3) [Work data management](https://robocorp.com/docs/courses/work-data-management). View [all Robocorp courses](https://robocorp.com/docs/courses) to learn more.

1. **Setup**

**환경 준비**

<https://koreapy.tistory.com/1129> 를 참조, python3.9를 삭제하고, conda가 제공하는 가상 환경으로 구성

1. ~~Python 3.9 설치 (3.9.13, 위치: c:\python\python39)~~ 🡪 삭제했음
2. VS Code 설치 (April 2022 (version 1.67))
   * “Python extension for Visual Studio Code” extension도 설치해 주어야 함
3. Conda 설치 (4.12.0, 위치: c:\users\LnY\anaconda3)
   * 가상환경 ‘pyenv39’ 생성하였음 (기본 ‘base’도 사용 가능한 듯)
   * $> conda activate pyenv39 혹은 base 사용 시 $> conda activate
   * $> conda deactivate
   * $> conda env list
   * $> conda create -n *venvName* python=3.9
   * $> conda update –all
   * $> conda env remove -n *venvName*
   * Ctrl+Shift+P 하여 “Python: select interpreter” 선택 🡪 pyenv39 또는 base 선택
4. VS Code에 Robocorp extension 설치
   * Robocorp code,
   * Robot Framework Language Server

**Default Structure**

* conda.yaml
  + controlling environment where the robots run.
  + 포함되는 package 정보
* robot.yaml
  + 실행 command 정의 및 위치
  + 2 mandatory parts
  + At least one task defined, and that must have either command, shell or robotTaskName defined.
  + Artifacts output path must be defined.
* tasks.robot 또는 task.py
  + 실행 action

1. [**Robocorp extension**](https://marketplace.visualstudio.com/items?itemName=robocorp.robocorp-code)**for**[**Visual Studio Code**](https://code.visualstudio.com/)

<https://robocorp.com/docs/developer-tools/visual-studio-code/overview>

With the [**Robocorp extension**](https://marketplace.visualstudio.com/items?itemName=robocorp.robocorp-code)**for**[**Visual Studio Code**](https://code.visualstudio.com/), you can create, run, and debug robots, publish to Control Room, read secrets from a Control Room vault, use work items, locate UI elements in browsers and desktop applications using a visual inspector, and enjoy other powerful features.

The [**Robot Framework Language Server**](https://marketplace.visualstudio.com/items?itemName=robocorp.robotframework-lsp)**extension** provides code completion, syntax validation and highlighting, code formatting, and other powerful robot development features.

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

* Documentation, possibly some libraries, references to other files

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

Documentation Template robot main suite

\*\*\* Tasks \*\*\* -- suite

Minimal task -- task

Log Done -- keyword + argument

Open the intranet site

\*\*\* Keywords \*\*\*

Open the intranet site -- New keyword

**\*\*\* Task \*\*\***

* Define tasks
* “Minimal task” is the name of the task.

**Keyword** (=Function)

* Provide the robot with specific skills
* Input Text *locator* *text*

**Library**

* Teach the robot new skills by making new keyword available to them
* added in the \*\*\* Settings \*\*\* section of your .robot file.

**Template**

* Extended
* Python
* Standard

1. **Course I: Beginner’s course**

**Robot Framework**

<https://robocorp.com/docs/courses/beginners-course/collecting-the-results>

다양한 종류의 locator(예: id, name, xpath 등등)가 있는데, 이를 사용하는 방법

[How to find user interface elements using locators in web applications](https://robocorp.com/docs/development-guide/browser/how-to-find-user-interface-elements-using-locators-in-web-applications)

투비소프트 XPlatform 테스트 시 꼼꼼히 검토할 필요 있음

(참조 자료: <http://docs.tobesoft.com/admin_guide_xplatform_ko#1977446c60481709>)

대상 object를 right-click 🡪 검사 🡪 (DevTools 표시되고, object 부분 하이라이트) 🡪 right click 🡪 copy 🡪 Copy element, Copy selector, Copy XPath, Copy full XPath 등을 확인

Css. Div 등의 tag를 이용하여 정보 수집

We have a div element. It has class and role attributes. Since there are no id or name attributes to work with, we decide to use the [CSS](https://developer.mozilla.org/en-US/docs/Web/CSS) classes to target it. The sales-summary class seems like a good candidate because then it will be clear in our script what we are taking a screenshot of. Because we use CSS to locate the element, our locator will start with the css: prefix. There are multiple types of locators: id, name, xpath, depending on the [strategy](https://robotframework.org/SeleniumLibrary/SeleniumLibrary.html#Explicit%20locator%20strategy) used.

The css:div.sales-summary locator means: Using the CSS strategy, find me a div element that has the sales-summary CSS class.

<**div** class="alert alert-dark **sales-summary**" role="alert">

<div><span>Active sales people:</span>...</div>

</div>

${sales\_results\_html}= Get Element Attribute id:sales-results outerHTML

We create a variable (${sales\_results\_html}=). We store into it what we get out of the [Get Element Attribute](https://robocorp.com/docs/libraries/rpa-framework/rpa-browser-selenium/keywords#get-element-attribute) keyword. We pass two arguments to that keyword: the first is the locator for the element (id:sales-results); the second is the name of the attribute of the element we want to get.

**Running the robot in Control Room**

Control Room is very powerful and can support very complicated workflows.

To set this all up, we will have to go through these steps:

* Logging into Control Room (<https://id.robocorp.com/login>)
* Creating a new Organization for RobotSpareBin Industries
  + [Organizations](https://robocorp.com/docs/control-room/administration/organizations-and-workspaces) allow you to group users, workspaces, processes, assistants, and robots.
* Creating a new Workspace
* Creating a new Robot
* Uploading the robot code
* Creating a Process (the "assembly line", remember?)
* Adding the Step to the Process and setting it up
* Executing and scheduling the process.

**Other options for running the robot**

Running the robot on **1) your local machine** with Visual Studio Code, or **2)** [**RCC**](https://github.com/robocorp/rcc), or **3) in Control Room** are not the only options.

* + **RCC** (Robot Creation Command) is a set of tooling that allows you to create, manage, and distribute Python-based self-contained automation packages - or 'robots' as we call them.

If your robot needs to be run on a physical (or a virtual) machine to access, for example, desktop applications, you can use our applications to that:

* + Use **4)** [**Robocorp Workforce Agent**](https://robocorp.com/docs/control-room/configuring-workforce/overview)**for unattended** cases where the robot can work in the background without human intervention.
  + Use **5)** [**Robot Assistants**](https://robocorp.com/docs/control-room/operating-assistants)**for attended** cases where the robot can work together with humans to complete automated tasks!

**The same robot in Pure Python and with Playwright**

(<https://robocorp.com/docs/courses/beginners-course/python-robot>)

Pure Python

* + In conda.yaml, define dependencies and the initialization command:

node.js, rpaframework and robotframework-browser

* + robot.yaml:

robot.yaml configuration will handle executing the robot (ex. task.py):

* + python program:

may use the Playwright-based Browser library or Selenium library for the browser automation

Playwright

* + MicroSoft의 새로운 오픈 소스 자동화 테스트 도구

1. **Course II:** [**Build a robot**](https://robocorp.com/docs/courses/build-a-robot)

<https://robocorp.com/docs/courses/build-a-robot>

1. **Course III: Work Data Management**
2. **Project 방법론**
3. Software robot delivery process

* The initial idea of automating something 💡
* Evaluating whether the automation is technically feasible 🤔
* Calculating the return on investment (cost vs. benefits) 💰
* Documenting the process ✏️
* Implementing & testing the robot ⌨️
* Launching the robot to production 🚀
* Maintaining the robot ⚙️

1. Tips for finding potential processes to be automated

* What is most of the people in the organization working on? More people -> more work -> more tasks to automate.
* Ideas for automation can come from the person completing the process or someone else involved in the process.
* You can try [process mining](https://en.wikipedia.org/wiki/Process_mining) to help with analyzing and identifying potential processes with automation potential.
* Ideation workshops can be a valuable source of great automation ideas. Gather people together an let your imagination run wild!
* You might have an idea for a process that would improve efficiency but can not be completed even manually at the moment. Sometimes automation enables you to start creating entirely new processes instead of just automating existing ones.
* See Figure 12. "Distribution of use cases by complexity of implementation and benefit realized across functions and sectors" in the excellent [RPA use cases research report by Capgemini Research Institute](https://www.capgemini.com/wp-content/uploads/2018/10/Automation-Use-Cases_Digital1.pdf) (PDF file).
* Collect all the possible automation ideas in a backlog. Then decide which process makes sense to start with: depending on your use case, it could be the most straightforward process to automate or the one you think will have the most significant ROI. Do not just pick the first one that you find: having good first experiences is essential, both for people and for organizations! 🙂

1. Write Process Definition Document

* What is the process used for, and what is its end goal?
* What are the steps involved in the process? Are there any decision points?
* Who is in charge of the process?
* By whom and how often is the process executed?
* What are the systems involved in the process?
* Does the user need special authorization or roles to run the process?
* What are the possible problems and exceptions that can happen during the procedure? How does the user handle those?
* What is process flow?

1. **Keymap**

* Open the Command Palette by pressing Shift-Command-P (macOS) or Ctrl+Shift+P (Windows).
* (Optional): Type run robot. This will find the command you need.
* Press Enter to run the Robocorp: Run Robot command.
* Toggle (hide/show) the left panel: Command-B (macOS) or Ctrl+B (Windows)
* Toggle (hide/show) the bottom panel: Command-J (macOS) or Ctrl+J (Windows)
* Zoom out: Command and - (macOS) or Ctrl and - (Windows)
* Zoom in: Command and + (macOS) or Ctrl and + (Windows)