\*\*\*\* REFERENCE: https://devopedia.org/robot-framework

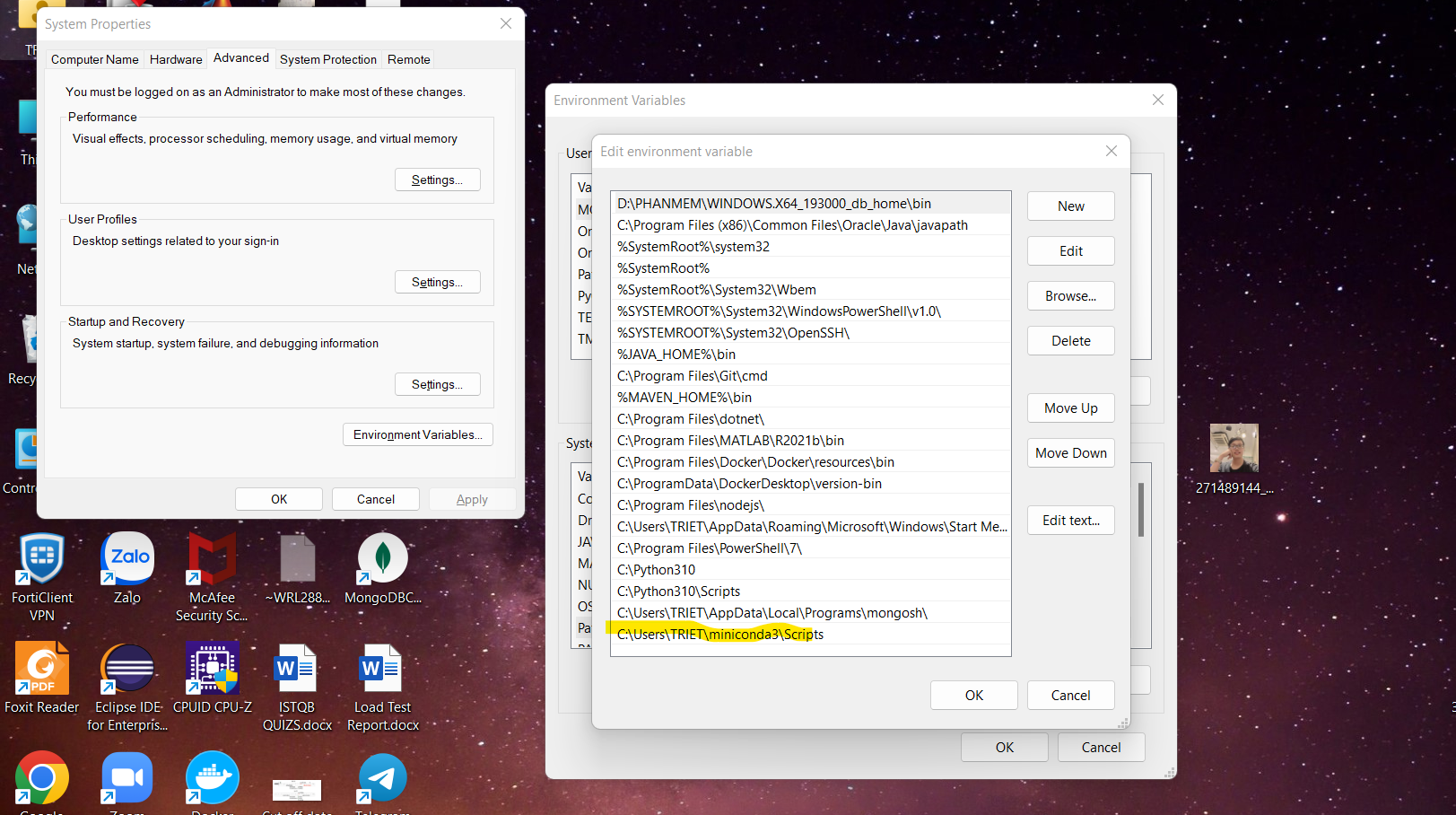
https://robotframework.org/robotframework/latest/RobotFrameworkUserGuide.html

1/ Issue fixed:

a/ If conda install stuck at Set up base environment (while installing) => try to download and install miniconda

b/ After installed conda (or miniconda), we need to set variable settings:

C:\Users\TRIET\miniconda3\Scripts



c/ If conda create --name py39 python=3.9

and see error: conda install packages error: Collecting package metadata (current\_repodata.json): failed

try this: https://github.com/conda/conda/issues/8273

Steps:

Go to C:\Users\TRIET\miniconda3\Library\bin

Copy 4 files

libcrypto-1\_1-x64.pdb

libcrypto-1\_1-x64.dll

libssl-1\_1-x64.pdb

libssl-1\_1-x64.dll

and paste them to C:\Users\TRIET\miniconda3\DLLs

REMEMBER TO RESTART COMMAND PROMT TO TAKE EFFECTS

Retry conda create --name py39 python=3.9

2/ chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://docs.conda.io/projects/conda/en/4.6.0/\_downloads/52a95608c49671267e40c689e0bc00ca/conda-cheatsheet.pdf

3/ Steps to install RobotFramework:

a/ Install conda (these steps can be skipped if we install python and pip directly into PC without conda)

+ Install conda/miniconda

+ Set up variable environment:"C:\Users\TRIET\miniconda3\Scripts"

Check if conda installed by: conda --version

+ Create a new environment named py35, install Python 3.5

conda create --name py35 python=3.5

+ Activate the new environment to use it

activate py35

b/ Install robotframework

+ pip install robotframework

c/ Install selenium via robotframework

pip install robotframework-seleniumlibrary

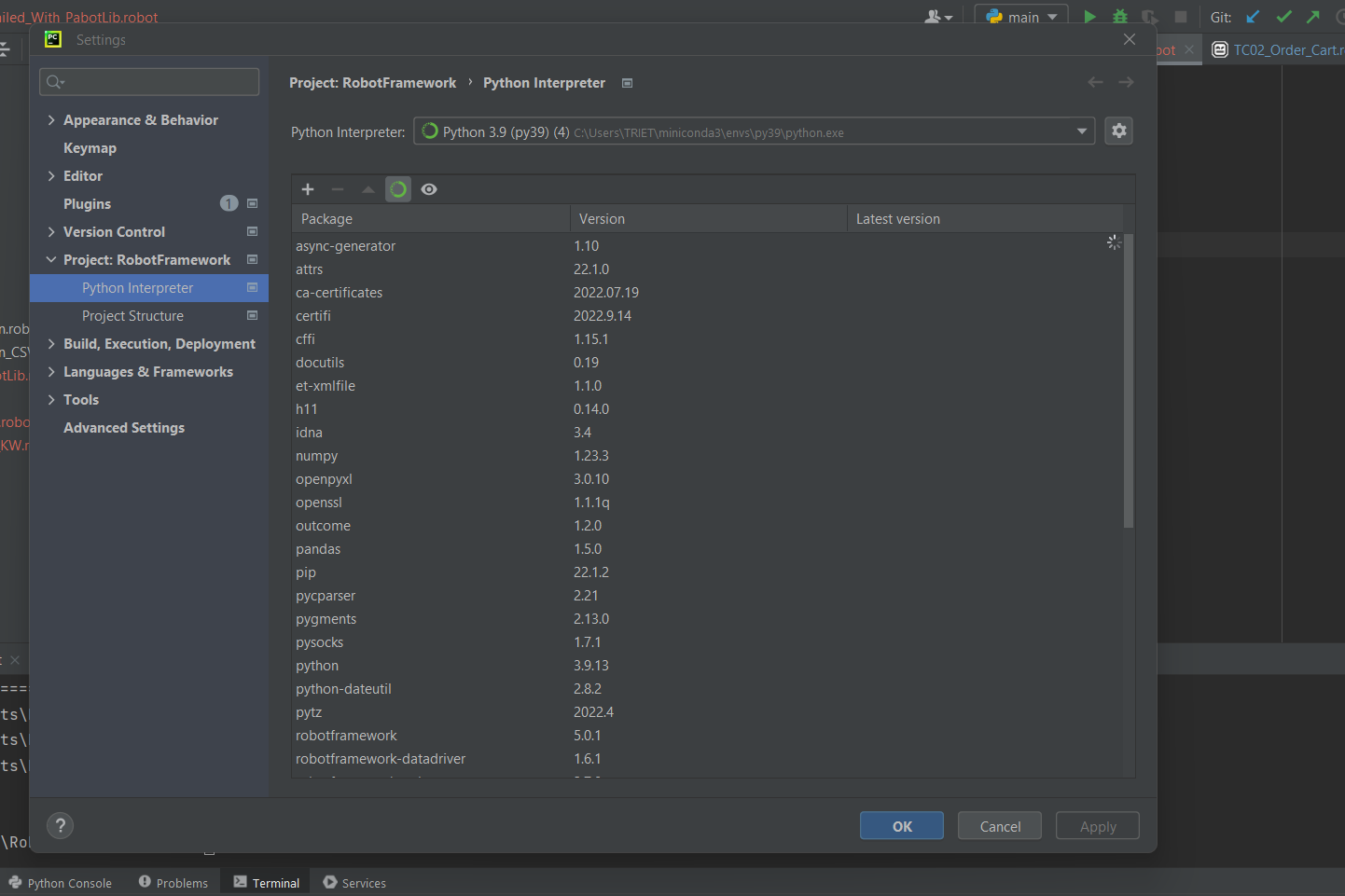
Reference: https://robotframework.org/SeleniumLibrary/SeleniumLibrary.html

d/ Select intepreter in pycharm/visualstudio to py35 folder (conda) which contains all robotframework libraries

from step 3

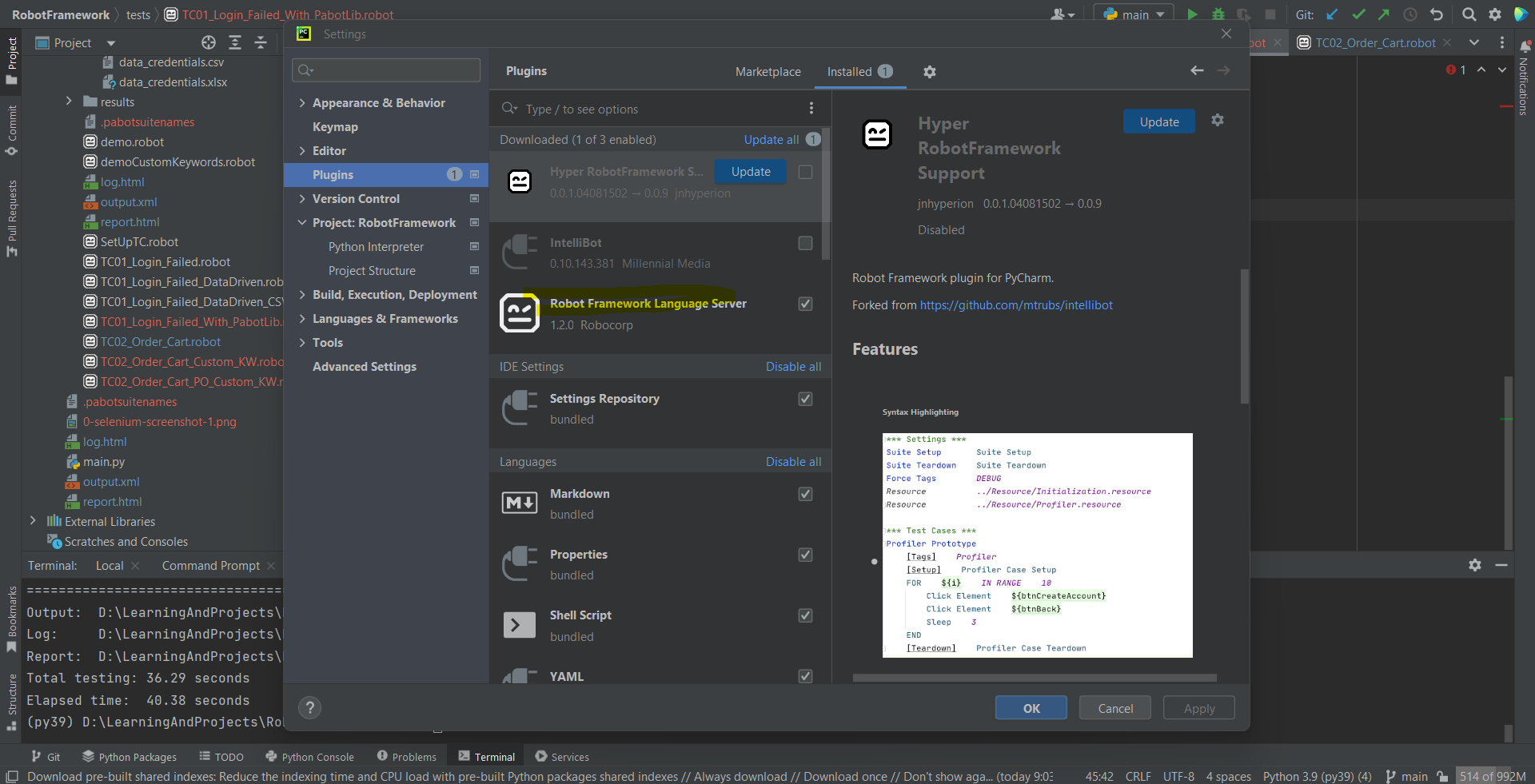
C:\Users\TRIET\miniconda3\envs\py39\pthon.exe

(need to check if this py39 has robotframework installed first by going to C:\Users\TRIET\miniconda3\Lib\site-packages)



e/ Settings -> Plugin to install Robot Framework Language Server' (robot framework language plugin)

Restart Pycharm if need and set path to python.exe in conda (the same interpreter)



4/

If we want to call two teardown keywords in test case but must not create new keyword for that.

The same as Documentation, there is a such syntax for keywords as for documentation or loops for example:

[Documentation] line1

... line2

... line3

Solution:

Use the "Run Keywords" keyword.

From doc "This keyword is mainly useful in setups and teardowns when they need to take care of multiple actions

and creating a new higher level user keyword would be an overkill"

Test Case

[Teardown] Run Keywords Teardown 1 Teardown 2

or also

Test Case

[Teardown] Run Keywords Teardown 1

... Teardown 2

and with arguments

Test Case

[Teardown] Run Keywords Teardown 1 arg1 arg2

... AND Teardown 2 arg1

\*\*\* Use with Run Keyword If

Run Keyword If condition Run Keywords Keyword1 AND Keyword2

... AND Keyword3

Reference:

https://stackoverflow.com/questions/22691941/how-to-make-multi-lines-test-setup-or-teardown-in-robotframework-without-creatin

https://stackoverflow.com/questions/44780327/handle-multiple-statements-in-run-keyword-if-robot-framework

5/ Tags in robot framework

https://testersdock.com/robot-framework-tags/

6/ Sample code to make Python Custom Keywords with Selenium and BuiltIn

from robot.api.deco import library, keyword

from robot.libraries.BuiltIn import BuiltIn

@library

class KeywordsWithCart:

xpath\_list\_product\_to\_cart = "//h4[@class ='card-title']/a"

def \_\_init\_\_(self):

self.selLib = BuiltIn().get\_library\_instance("SeleniumLibrary")

@keyword

def add\_items\_to\_cart(self, list\_products:list):

self.selLib.wait\_until\_element\_is\_visible(self.xpath\_list\_product\_to\_cart, 10, error="Element is not visible")

list\_product\_webelements = self.selLib.get\_webelements(self.xpath\_list\_product\_to\_cart)

for product\_web\_element in list\_product\_webelements:

product = self.selLib.get\_text(product\_web\_element)

if product in list\_products:

index = self.selLib.get\_index\_from\_list(list\_product\_webelements,product\_web\_element)

index += 1

self.selLib.click\_button("//div[@class='card-footer']/button)[" + index + "]")

7/ Continue from 6

If we import Python Custom Library to robotframework

And it show this error:

"Unresolved library: Browser. Error generating libspec: Initializing library 'Browser' with no arguments failed: robot running error...."

=> In Visual Code

Try solution in this link: https://stackoverflow.com/questions/71931370/robotframework-some-libraries-are-not-imported/72324894#72324894

And NOTE: Use python 3.9

Because some how the Robot Framework Language Server extension only works with Python 3.9

Example: C:\Users\TRIET\miniconda3\envs\py39\python.exe

If using Pycharm, change interpreter to py39 (python 3.9)

and use python 3.9 as command line in terminal (activate py39)

8/

|  |  |
| --- | --- |
| Command Line Scenarios | Commands |
| Execute All Tests in the folder | robot . |
| Execute Specific Test | Robot <testfilename>.robot |
| Execute Test Cases by test name | Robot --t <testname> |
| Execute Test cases by Tags in any file | Robot --include <tagname> |
| Execute Test Cases by Multiple Tags | Robot --include <tagname1>AND<tagname1>  Robot --include <tagname1> OR<tagname1> |
| Exclude the tests by tag | Robot --exclude <tagname> |
| Execute Tests by suite name | Robot --suite <foldername> |
| Execute only Failed Tests | Robot --rerunfailed output.xml |

Example: robot –t –include REGRESSION TC01.robot

* Select TC01.robot and in TC01 only run TC that has REGRESSION tag

9/ Pabot

Pabot helps to run test in parallel.

Install: pip install -U robotframework-pabot

+ To run all test suites in parallel: pabot .

+ To run test cases in a specific suite in parallel: pabot –testlevelsplit filename.robot