Robot-Framework

Table des matières

[ROBARDET-NICOLAS Gaelle 1](#_Toc159605961)

[Installations Python + PyCharm + RobotFramework 2](#_Toc159605962)

[Utilisation de fichier.robot 3](#_Toc159605963)

[Objectif = faire des Test Cases le plus clair et lisible possible 3](#_Toc159605964)

[\*\*\* Test Cases \*\*\*, Exécution file.robot, report.html, log.html 3](#_Toc159605965)

[Lancement robot (option -t ) 3](#_Toc159605966)

[Langage Robot – KEYWORD (tableau) 4](#_Toc159605967)

[Lib BuildIn ( Should … pour vérifier ) 4](#_Toc159605968)

[Lib OperatingSystem ( Gestion des fichiers .. ) 4](#_Toc159605969)

[[Setup] et [TearDown] par test ou Suite Setup TearDown en settings par defaut 5](#_Toc159605970)

[\*\*\* Keyword \*\*\* = création de ses propres keyword 5](#_Toc159605971)

[Retour d’un Keyword = Variable ${xx} en début 5](#_Toc159605972)

[\*\*\* Variables \*\*\* - Les Variables **${xx}** ou robot option **-v** 5](#_Toc159605973)

[Tags = étiquette - Lancement avec filtrage ou exclusion par tag **( option -i ou -e )** 6](#_Toc159605974)

[Nommage de Keyword avec des paramètres aussi utilisé à l’intérieur (très pratique !) 6](#_Toc159605975)

[TP avec fichier python ( Lib Process ) - hello.py 6](#_Toc159605976)

[Run process vs Start process 6](#_Toc159605977)

[Accès aux éléments d’un objet ${res.stdout} 6](#_Toc159605978)

[Visualisation des double espaces dans l’éditeur 7](#_Toc159605979)

[Check avec Integer ( ex ${0} ) 7](#_Toc159605980)

[TP avec test fichier division.py – test\_division.robot 7](#_Toc159605981)

[Paramètres des Keywords (entrée) avec [Arguments] et RETURN 7](#_Toc159605982)

[Tableau d’entrée de paramètres avec [Template] 7](#_Toc159605983)

[Ajout de Traces Log ou Log Many 7](#_Toc159605984)

[TP – Test04.robot - List ${} @{},FOR ,IF 7](#_Toc159605985)

[@{xx} utile pour List, syntaxe qui split une liste en plusieurs éléments 7](#_Toc159605986)

[&{xx} utile pour Dictionnaire Clé-Valeur 7](#_Toc159605987)

[Boucle FOR IN – FOR RANGE - IF 7](#_Toc159605988)

[extension .ressource = Fichier de keyword outils externalisé (sans TestCase !) 8](#_Toc159605989)

[TP – Test site Test SauceDemo - test05.robot 8](#_Toc159605990)

[ANNEXE – PyCharm Raccourcis 8](#_Toc159605991)

[ANNEXE – Aides html simplifié - Répertoire Robot doc 8](#_Toc159605992)

[ANNEXE -Python infos 8](#_Toc159605993)

[Pb de type float avec mauvais (tout langage) 8](#_Toc159605994)

## ROBARDET-NICOLAS Gaelle

<https://formationspython.com/>

# Installations Python + PyCharm + RobotFramework

<https://www.python.org/downloads/>

python --version

python 3.12.2 > D:\Python312

<https://www.jetbrains.com/pycharm/download/?section=windows> pyCharm

"C:\Program Files\JetBrains\PyCharm Community Edition 2023.3.3\bin\pycharm64.exe"

**Settings** / Python interpréter / Add « D:\Python312 »

**Plugins** / Add robot framework langage server + IntelliBot SeleniumLibrary Patched

Restart

Plugins / Robot Framework project / Language /

**Settings** / Languages & Framework / Robot Framework (project)

Language Server Python (mettre le chemin du python.exe sous venv…scripts)

= D:\OneDrive - ENI Ecole Informatique\\_ENI testeur\workspace-pycharm\formation\_robot\.venv\Scripts\python.exe

A gauche, **Python Packages** / installer robotframework 7.0

<https://robotframework.org/>

Robot > Langage spécialisé (pas directement Python)

# Utilisation de fichier.robot

## Objectif = faire des Test Cases le plus clair et lisible possible

Pas de convention d’ordre \*\*\*

Mais par habitude

Settings

variables

Test Cases ( le plus lisible possible )

Keywords ( souvent lourd, parfois externalisé en library )

## \*\*\* Test Cases \*\*\*, Exécution file.robot, report.html, log.html

Fichier1 = test01.robot

\*\*\* Test Cases \*\*\*  
Mon premier test  
 Log Hello word  
 Should Be Equal a a

### Lancement robot (option -t )

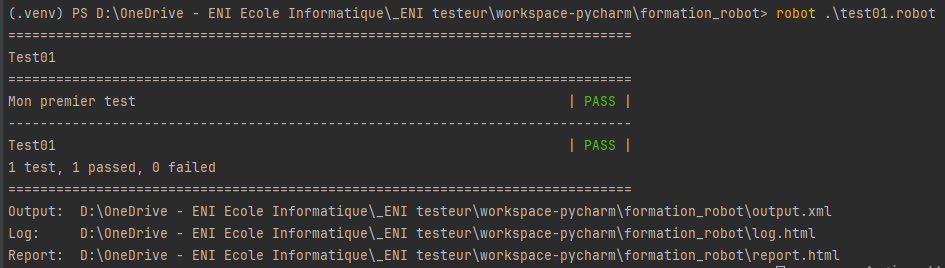
icône triangle vert dans le fichier (à coté des \*\*\*)

ou

Terminal > robot .\test01.robot (avec de la couleur)

Ou robot -t .\test01.robot (option -t title)

Ou robot -t .\test\*.robot



report.html (permet de voir le résultat global d’avancement )

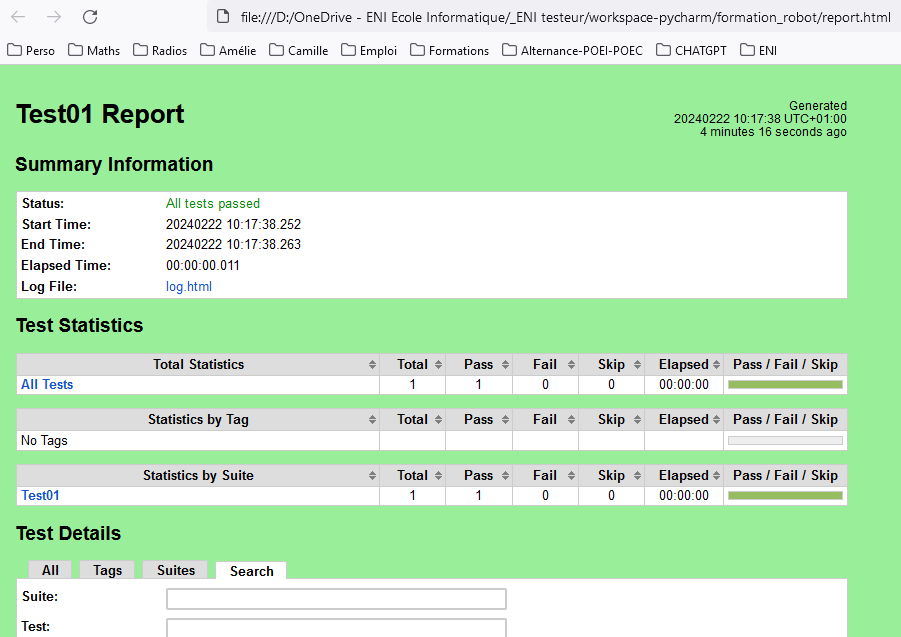
fond Vert si globalement PASS

fond Rouge si du FAILED ( si 1er Keyword est en échec, tout s’arrête et FAILED )

Si FAILED, un message est affiché



Log.html permet de voir les détails de chaque fichier de tests ( Log affiche dans fichier Log)



### Langage Robot – KEYWORD (tableau)

Proche python avec indentation

Pas de ponctuation

Approche par tableau selon item (Test Cases …)

Une tabulation correspond à une colonne ( format TSV tabulation separator)

**Attention 2 espaces crée une colonne !!**

NB sous pyCharm, tabulation = 4 espaces associé à Python

NB insensible à la casse pour les commandes

NB pour boolean, true=yes et false=no

NB commentaire avec #

NB keyword Fail fait échoué le test

Un test s’arrête dès qu’il y a une erreur

<https://robotframework.org/robotframework/#standard-libraries>

<https://robotframework.org/robotframework/latest/libraries/BuiltIn.html>

Décrit les Keyword ( ex Shoud be equal détails )

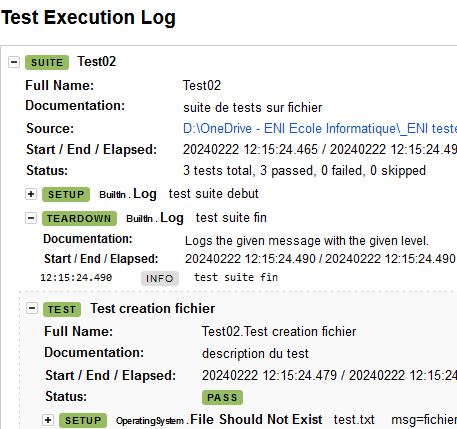
### Lib BuildIn ( Should … pour vérifier )

### Lib OperatingSystem ( Gestion des fichiers .. )

<https://robotframework.org/robotframework/latest/libraries/OperatingSystem.html>

NB import des libraries automatique à partir de clé + ctrl espace

### [Setup] et [TearDown] par test ou Suite Setup TearDown en settings par defaut

paramètres de test (meta command)

(début fin test) exécution indépendante du résultat du test !

Attention 1 seul TearDown par Test !!

Test Setup Log debut test #set up par defaut qui peut etre surchargé par test  
Suite Setup Log test suite debut   
Suite TearDown Log test suite fin

Les Setup et TearDown sont écrits au début

mais l’exécution ne sa fait pas au même moment !!

## \*\*\* Keyword \*\*\* = création de ses propres keyword

Les keyword de base n’acceptent d’une action, donc création de keyword perso plus grande.

### Retour d’un Keyword = Variable ${xx} en début

${contenu} Get File test.txt

${retour}

${retour} = (accepté aussi mais non utile)

NB il peut y avoir **plusieurs retours possible** à une fonction Keyword

## \*\*\* Variables \*\*\* - Les Variables **${xx}** ou robot option **-v**

Elles sont locales à un test par defaut.

${texte} Set Variable contenu du fichier  
Create File test.txt ${texte} # creation fichier  
File Should Exist test.txt msg=fichier aurait du etre créé  
${contenu} Get File test.txt  
Should Be Equal ${contenu} ${texte}

NB en python, les constantes n’existent pas !

VAR ${texte} contenu du fichier

${texte} Set Variable contenu du fichier

\*\*\* Variables \*\*\*  
${NOM\_FICHIER} test.txt

robot **-v "NOM\_FICHIER:truc.txt"** ./test02.robot

cela surcharge la variable du code.

### Tags = étiquette - Lancement avec filtrage ou exclusion par tag **( option -i ou -e )**

On peut alors filtrer par Tag.

Il y a des statistiques par Tag proposées.

Tag = feature\_US1 (ex **par fonctionnalité** )

Tag **identification** (ex auteur\_PGS )

[Tags] bidule,toto

robot **-i PGS** ./test02.robot ( -i inclusion du tag )

Tag bug\_found pour éliminer les tests en échec

robot -e bug\_found ./test02.robot ( -e exclusion du tag )

robot -e « \*PGS\* » ./test02.robot ( -e exclusion du tag PGS)

## Nommage de Keyword avec des paramètres aussi utilisé à l’intérieur (très pratique !)

Division ${numerateur} par {diviseur} égal à {Resultat}

# TP avec fichier python ( Lib Process ) - hello.py

<https://robotframework.org/robotframework/latest/libraries/Process.html>

Lancement du fichier python

**python hello.py**

test03.robot tester le programme fichier python

## Run process vs Start process

Run process normal

Start si process avec attente (service, navigateur …)

NB Run process, bien espacer les valeurs paramètres !!

Retour = **Result Object** ( code retour, stdout, stderr …. )

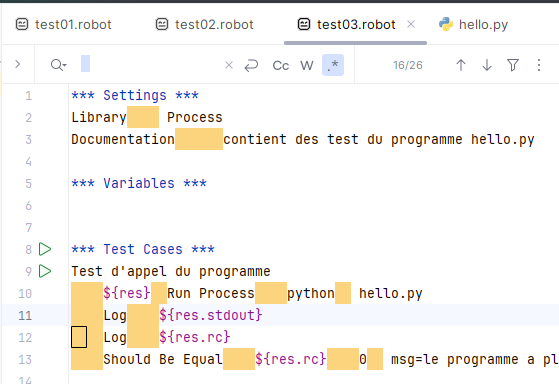
Rappel flux:

stdin > PROCESS > stdout + stderr + code retour

Code\_retour=0 si OK sinon erreur à définir (+infos stderr )

### Accès aux éléments d’un objet ${res.stdout}

Log ${res.stdout}  
Log ${res.rc}



### Visualisation des double espaces dans l’éditeur

Combinaison touche ctrl F (recherche) + champ .\* + double espace

Les doubles espace seront colorés

### Check avec Integer ( ex ${0} )

${zero} **Convert To Integer** 0  
Should Be Equal ${res.rc} ${zero} msg=le programme a planté

Ou Should Be Equal As Integers

Ou ${0}

NB: ${xx} permet d’utiliser le mode de python

Ex ${2+2}

{EMPTY} pour contenu vide

Ou Should Be Empty

# TP avec test fichier division.py – test\_division.robot

python division.py > manque argument

programme de division entire ou décimale selon si –reste

## Paramètres des Keywords (entrée) avec [Arguments] et RETURN

Eviter de passer des paramètres optionnels qui peuvent être vide !! (complexité > risque test)

## Tableau d’entrée de paramètres avec [Template]

Test tableau de données Template avec les paramètre d'une Keyword  
#limite 1 seul test !!  
 [Template] Division\_decimale  
 #numerateur diviseur resultat  
 5 2 2.50  
 10 2 5.00

\*\*\* Keywords \*\*\*  
Division\_decimale  
 [Documentation] *division sans reste* [Arguments] ${numerateur} ${diviseur} ${resultat}  
 ${resultat} Run Process python division.py ${numerateur} ${diviseur}

## Ajout de Traces Log ou Log Many

Log test setup 1

Log Many x1 x2 x3 #affiche 3 Log

# TP – Test04.robot - List ${} @{},FOR ,IF

## @{xx} utile pour List, syntaxe qui split une liste en plusieurs éléments

## &{xx} utile pour Dictionnaire Clé-Valeur

## Boucle FOR IN – FOR RANGE - IF

## extension .ressource = Fichier de keyword outils externalisé (sans TestCase !)

# TP – Test site Test SauceDemo - test05.robot

<https://www.saucedemo.com/> site de tests

selon le user, le comportement est different

Lib Selenium

<https://robotframework.org/SeleniumLibrary/SeleniumLibrary.html>

-utilisation de dictionnaire

-création de keyword avec python

# ANNEXE – PyCharm Raccourcis

Shift + Shift = aides PyCharm

Alt sur plusieurs points permet de grouper des déplacements ensuite

Import en faisant TAB devant le keyword

# ANNEXE – Aides html simplifié - Répertoire Robot doc

Voir le répertoire robot\_doc

# ANNEXE -Python infos

Pas de constante

Pas de visibilité private > tout est public !!

## Pb de type float avec mauvais (tout langage)

10/9 =1.11112 faux

11 % 2.2 = 2.1999993

Solution type Decimal à utiliser !

D(‘11’) % D(‘2.2’) = D(‘0.0’)