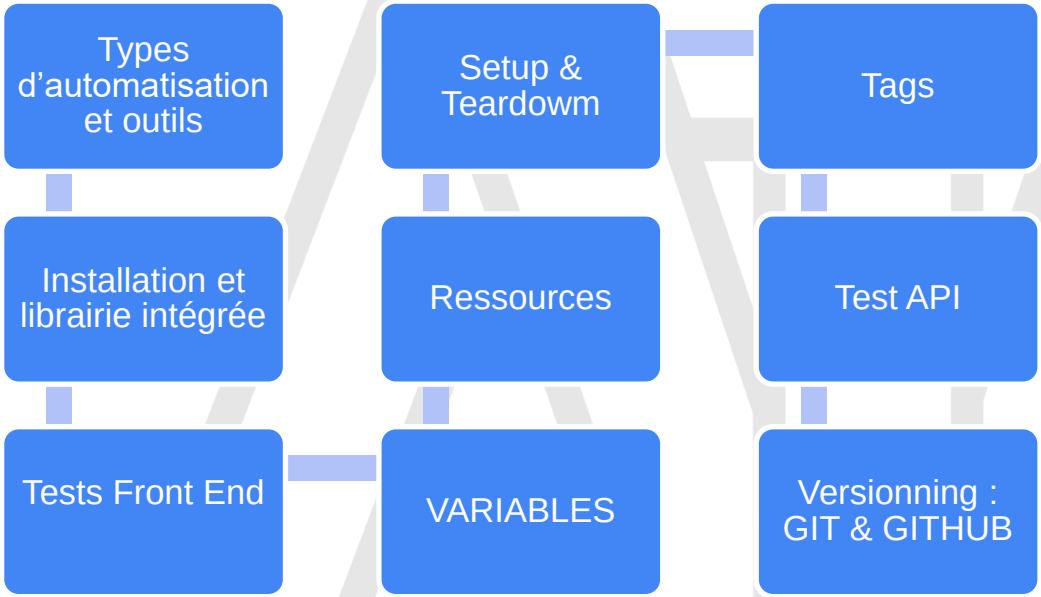
Automatisation des tests avec ROBOT FRAMEWORK

Automation Tester Academy

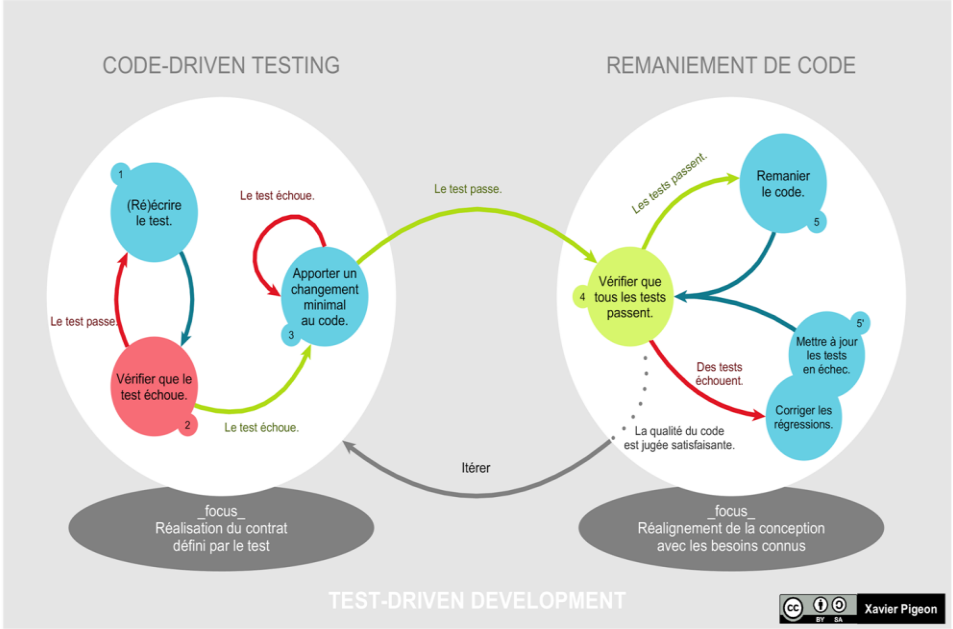
Sommaire



Types d'automatisation et outils

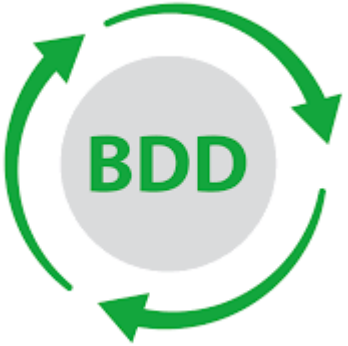
Automation
Tester Academy

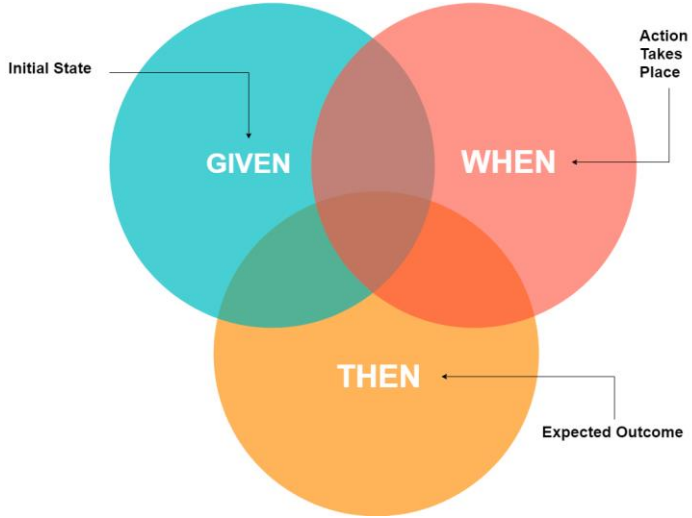
Test Driven Development



Automation
Tester Academy

Behaviour Driven Development



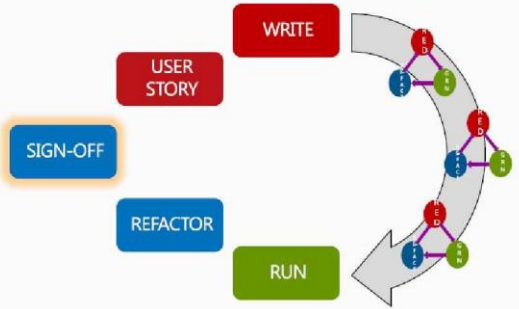


Automation
Tester Academy

Acceptance Tests Driven Development

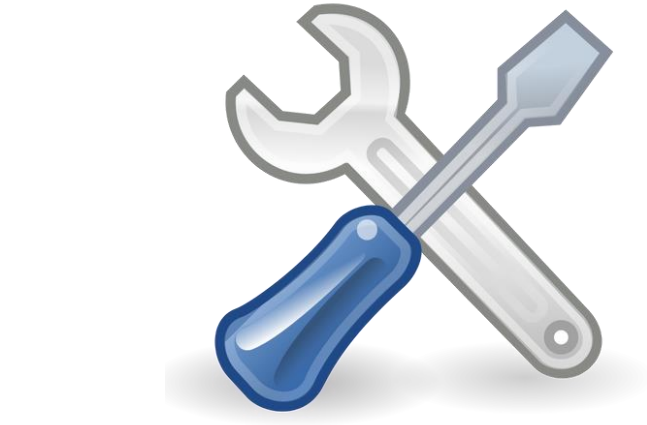
(Acceptance Test Driven Development)

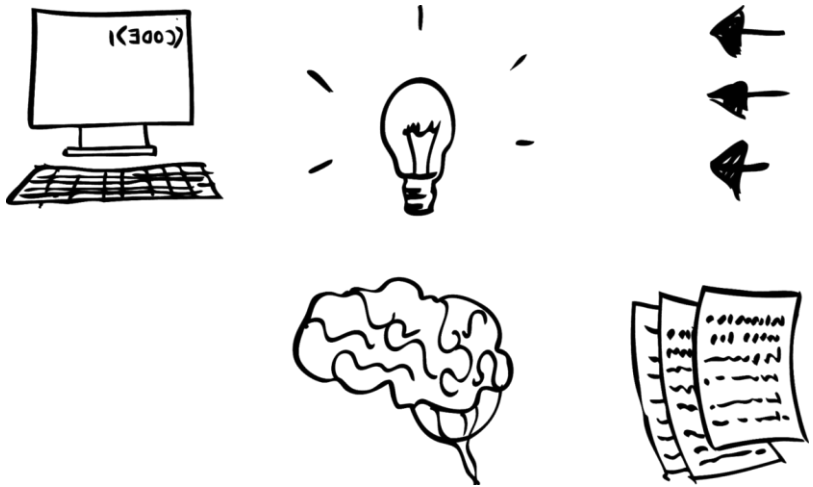
- Select User story
- Write Acceptance Test
- Implement User Story
- Run Acceptance Test
- (Refactor)
- Get Sign-Off

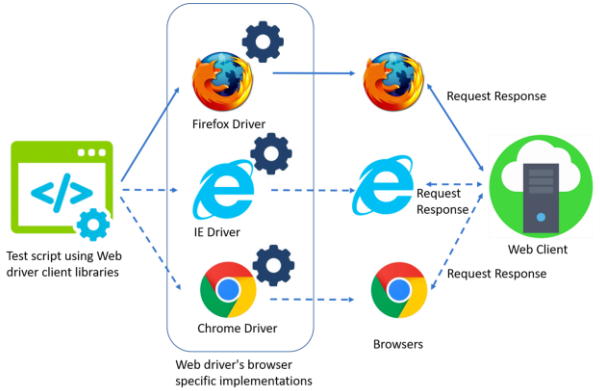


Automation
Tester Academy

Quel(s) outil(s) ?






Firefox Driver
IE Driver
Chrome Driver
Test script using Web driver client libraries
Web driver's browser specific implementations
Request Response
Request Response
Request Response
Web Client
Browsers

Qu’est-ce que l’automatisation ?

Pourquoi automatiser ?

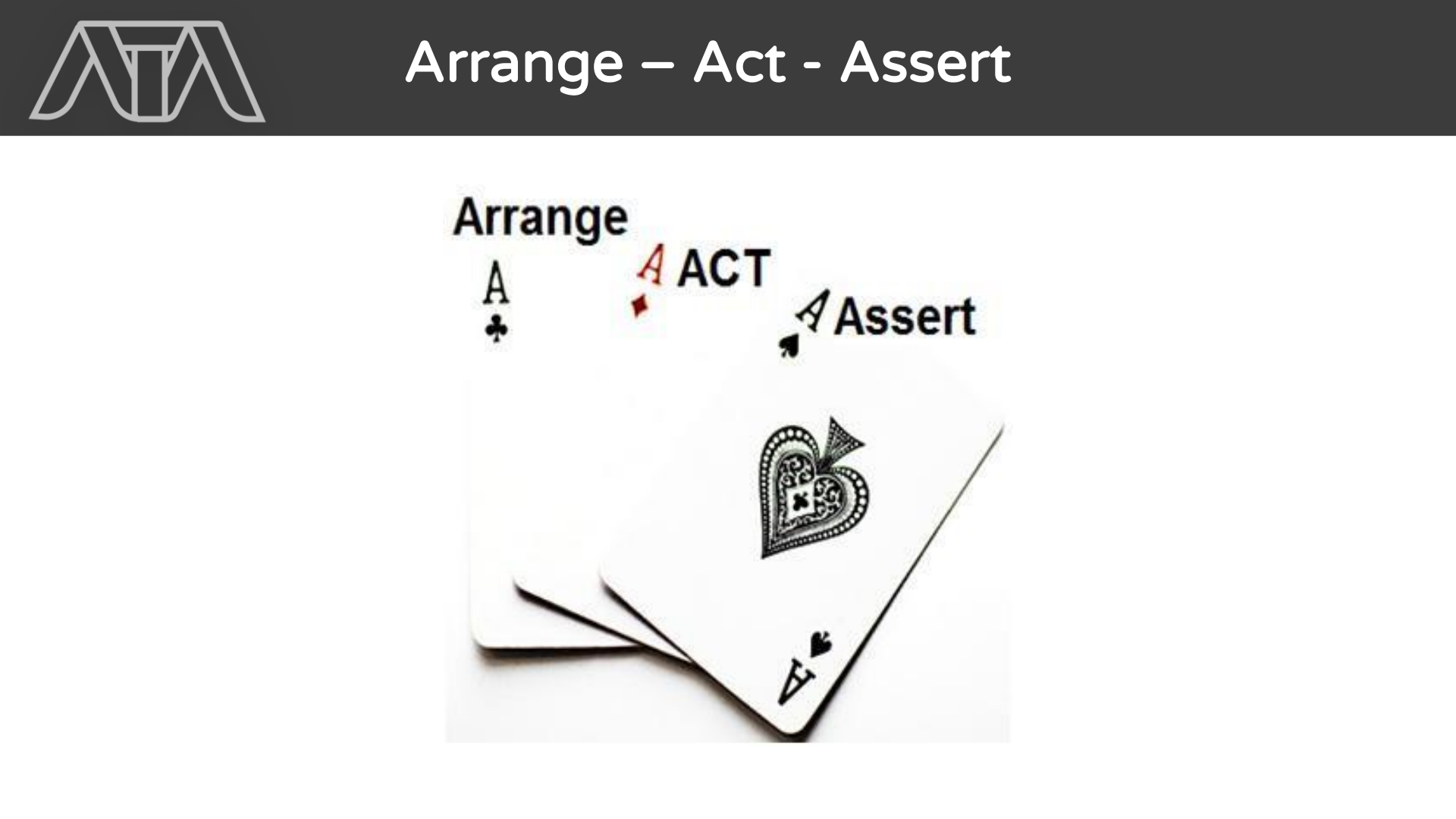
RobotFramework c’est quoi ?

- Framework facile à configurer, utiliser pour : web, desktop, mobile, REST
- Syntaxe basée sur les mots-clés (vs fonctions d'autres frameworks)
- Extensible facilement via des bibliothèques en Python
- Création rapide de nouveaux mots-clés
- Plusieurs exemples et projets démo disponible sur le web

- Prend en charge le Data Driven Testing
- Possible d'écrire les tests en Gherkin (Given, When, Then)
- Approche de test basé sur le comportement
- Format tabulaire
- Les utilisateurs ont la possibilité de créer leurs mots-clés.
- Utilisation de différents types de variables
- Permet de catégoriser les cas de test à l'aide des tags (smoke, end to end, security, ou autres)
- Fournit des rapports et des logs sans aucune configuration de plus

Pattern d’automatisation

Pour écrire des bons tests



Arrange – Act - Assert

Arrange - Act - Assert (Arrange)

Préparer les prérequis et les cibles :

1. Préparer le cas de test (création de compte ou connexion ..)
2. Préparer les locators des éléments web
3. Préparer la BDD si besoin
4. Gérer toutes les opérations pré-test

Arrange - Act - Assert (Act)

Agir sur la cible :

1. Couvrir ce qu'on veut réellement tester
2. Interagir avec les éléments de la page (bouton, champs texte ..)
3. Faire les appels HTTP
4. Focus sur le comportement cible

Arrange - Act - Assert (Assert)

Vérifications sur le comportement attendu.
Act doit produire quelques réponses. ***Assert*** les vérifie

1. Vérifier la présence des éléments
2. Vérifier le contenu texte ou l'application d'un style
3. Vérifier des valeurs alphanumériques
4. Vérifier des réponses HTTP

Les assertions déterminent si le test passe ou échoue !

Installation et librairie intégrée

Installer un IDE

Pycharm (Community edition)
https://www.jetbrains.com/fr-fr/pycharm/

OU BIEN

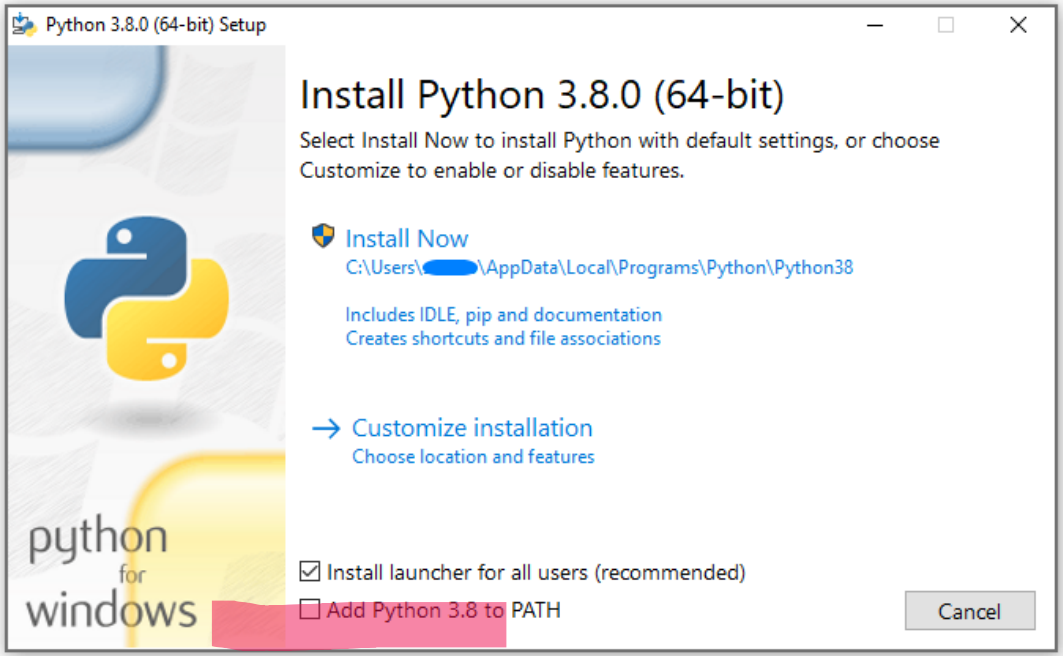
Visual Studio Code
https://code.visualstudio.com/

Et le plugin robot :
https://marketplace.visualstudio.com/items?itemName=robocorp.robotframework-lsp

Télécharger Python

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

N’oubliez pas d’ajouter Python au PATH



Vérifier l'installation de *PIP*

```
pip --version
```

PIP est un gestionnaire de paquets pour installer des modules pour python

Installer un Robot Framework

```
pip install robotframework
```

OOO

Vérifier l'installation

```
robot --version
```

Exécuter les scripts !

Cloner ce projet d'exemple
https://github.com/VanitaCW/Robot-Framework-concepts-examples/tree/main/ExampleRobotTests

Mots-clés intégrés

Sleep

script.robot

```
*** Test Cases ***

Doit suspendre l'exécution du test pendant 3 secondes
            Sleep       3
```

Log

script.robot

```
*** Test Cases ***

Doit écrire un message de log
            Log         Ceci est un log d'info              level=info
            Log         Ceci est un log de warning          level=warn
            Log         Ceci est un log d'erreur            level=error
```

Log To Console

script.robot

```
*** Test Cases ***

Doit écrire un message de log
    Log To Console    Je suis passé par là
```

Should Be Equal

script.robot

```
*** Variables ***

${premiere variable}              Ali
${deuxieme variable}              Ali


*** Test Cases ***

Doit être égale
            Should Be Equal       ${premiere variable}   ${deuxieme variable}
```

Lancer un script de tests

```
robot tests.robot
```

Commande simple pour lancer le script **tests.robot**

Lancer tous les scripts robot d’un répertoire

```
robot .
```

Should Not Be Equal

script.robot

```
*** Variables ***

${premiere variable}              Ali
${deuxieme variable}              Jean


*** Test Cases ***

Ne doit pas être égale
            Should Not Be Equal    ${premiere variable}    ${deuxieme variable}
```

Should Contain

script.robot

```
*** Variables ***

${premiere variable}              Ceci est une chaine de caractère
${deuxieme variable}              chaine de caractère


*** Test Cases ***

Doit contenir
            Should Contain        ${premiere variable}    ${deuxieme variable}
```

Should Not Contain

script.robot

```
*** Variables ***

${premiere variable}              Robot Framework
${deuxieme variable}              Python


*** Test Cases ***

Ne doit pas contenir
            Should Not Contain    ${premiere variable}    ${deuxieme variable}
```

Should Start With

script.robot

```
*** Variables ***

${premiere variable}              Ceci est une chaine de caractère
${deuxieme variable}              Ceci est une


*** Test Cases ***

Doit commencer par
           Should Start With      ${premiere variable}   ${deuxieme variable}
```

Should Not Start With

script.robot

```
*** Variables ***

${premiere variable}              Ceci est une chaine de caractère
${deuxieme variable}              est une


*** Test Cases ***

Ne doit pas commencer par
          Should Not Start With              ${premiere variable}   ${deuxieme variable}
```

Should End With

script.robot

```
*** Variables ***

${premiere variable}          Ceci est une chaine de caractère
${deuxieme variable}          caractère


*** Test Cases ***

Doit finir par
            Should End With                 ${premiere variable}    ${deuxieme variable}
```

Should Not End With

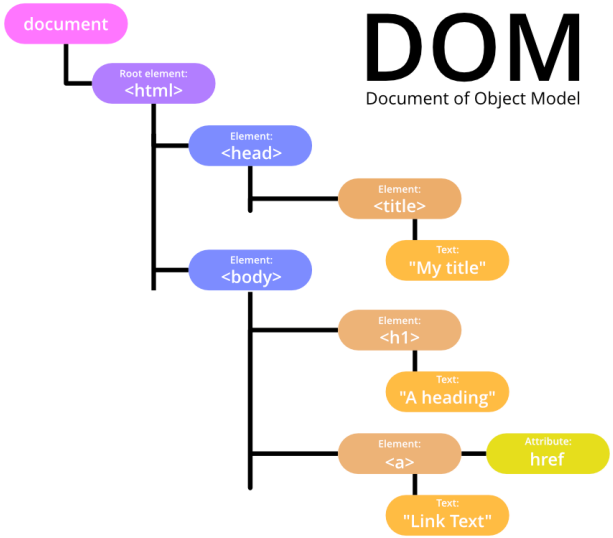
script.robot

```
*** Variables ***

${premiere variable}              Ceci est une chaine de caractère
${deuxieme variable}              Ceci


*** Test Cases ***

Ne doit pas finir par
            Should Not End With                ${premiere variable}    ${deuxieme variable}
```

Tests Front End

- Locator
- Selenium library


PASSPORT
United States of America
ATLANTIC OCEAN
EUROPE
RUSSIA
ASIA



LOCALISER + AGIR

Strategy	Match based on	Example
id	Element id.	id:example
name	name attribute.	name:example
identifier	Either id or name.	identifier:example
class	Element class.	class:example
tag	Tag name.	tag:div
xpath	XPath expression.	xpath://div[@id="example"]
css	CSS selector.	css:div#example
dom	DOM expression.	dom:document.images[5]
link	Exact text a link has.	link:The example
partial link	Partial link text.	partial link:he ex
sizzle	Sizzle selector deprecated.	sizzle:div.example
data	Element data-* attribute	data:id:my_id
jquery	jQuery expression.	jquery:div.example
default	Keyword specific default behavior.	default:example


SCRIPT


string sInput;
int iLength, iN;
double dblTemp;
bool again = true;
while (again) {
iN = -1;
again = false;
getline(cin, sInput);
system("cls");
stringstream(sInput) >> dblTemp;
iLength = sInput.length();
if (iLength < 4) {
again = true;
continue;
} else if (sInput[iLength - 3] != '.') {
again = true;
continue;
} while (++iN < iLength) {
if (isdigit(sInput[iN])) {
continue;
} else if (iN == (iLength - 3)) {

```
*** Settings ***
Documentation     A test suite with a single test for valid login.
...
...               This test has a workflow that is created using keywords in
...               the imported resource file.
Resource          login.resource

*** Test Cases ***
Valid Login
    Open Browser To Login Page
    Input Username    demo
    Input Password    mode
    Submit Credentials
    Welcome Page Should Be Open
    [Teardown]    Close Browser
```

Mots-clés librairie Selenium

Installer la librairie Selenium

```
pip install robotframework-seleniumlibrary
```

Télécharger les webdrivers

https://chromedriver.chromium.org/downloads

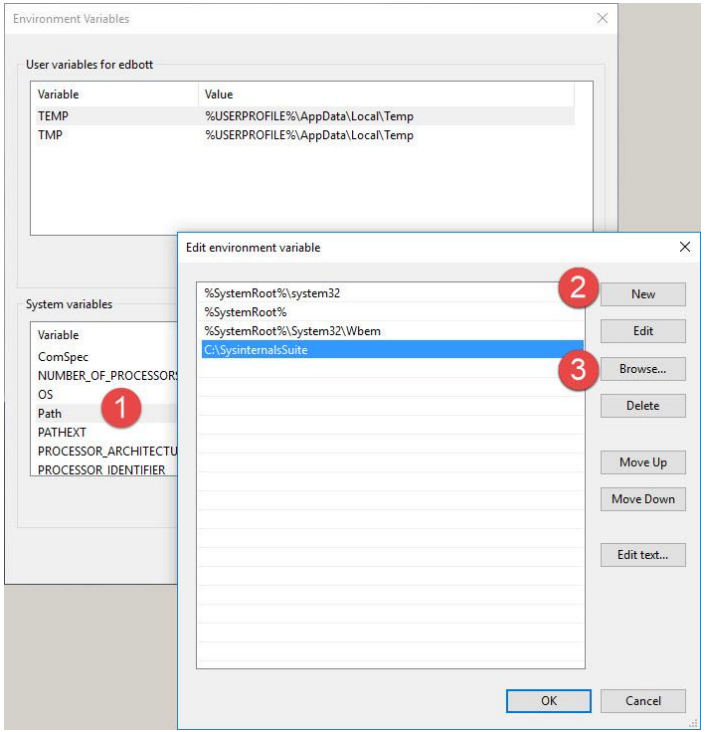
https://github.com/mozilla/geckodriver/releases

Ajouter les binaires dans le path !!

Ajouter les webdrivers dans le path !!

Windows

Ajouter le **chemin** du dossier contenant le webdriver comme nouvelle entrée à la variable Path



Mac OS

1- Editer le fichier **.zshrc**

```
$ cd
$ nano .zshrc
```

2- Ajouter le chemin des webdrivers au PATH

```
PATH="CHEMIN_DES_WEBDRIVERS:${PATH} »
exprt PATH
```

3- Appliquez les modifications

```
$ source .zshrc
```

Premier script : ouvrir **google** sur Chrome

script.robot

```
*** Settings ***
Library                 SeleniumLibrary


*** Test Cases ***
Ouvrir google.com
            Open Browser            https://www.google.com              chrome
```

Ouvrir **google.com** sur Firefox

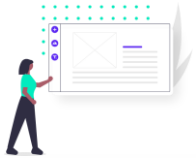
script.robot

```
*** Settings ***
Library                 SeleniumLibrary


*** Test Cases ***
Ouvrir google.com
            Open Browser            https://www.google.com              firefox
```

Cliquer sur un bouton

script.robot

```
*** Settings ***

Library                 SeleniumLibrary


*** Test Cases ***

Cliquer sur un bouton

            Open Browser            https://www.saucedemo.com           chrome
            Click Button            id:login-button
```

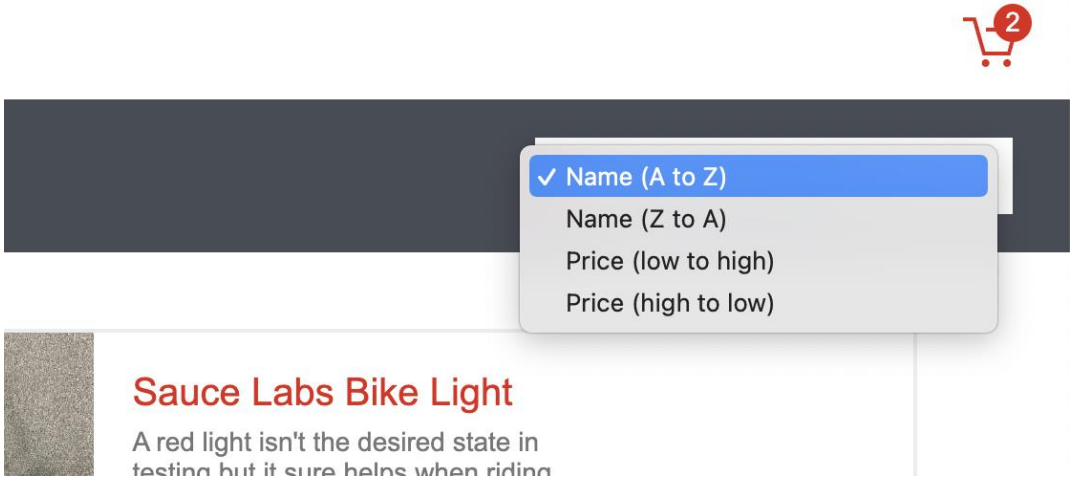


Remplir un champs texte

script.robot

```
*** Settings ***

Library                 SeleniumLibrary


*** Test Cases ***

Remplir un champs texte

            Open Browser            https://www.saucedemo.com           chrome
            Input Text              id:user-name                        standard_user
```

Sur le site **saucedemo**, on peut sélectionner de 3 manières différentes sur la liste déroulante de tri :



1. Sélectionner par label

script.robot

```
*** Settings ***

Library                 SeleniumLibrary


*** Test Cases ***

Sélectionner dans une liste déroulante

            Select From List By Label    css:[data-test="product_sort_container"]    Name (Z to A)
```

Avec utilisation de la stratégie de localisation : **CSS**

2. Sélectionner par index

script.robot

```
*** Settings ***

Library                 SeleniumLibrary


*** Test Cases ***

Sélectionner dans une liste déroulante

            Select From List By Index    class:product_sort_container     2
```

Avec utilisation de la stratégie de localisation : **class**

3. Sélectionner par value

script.robot

```
*** Settings ***

Library                 SeleniumLibrary


*** Test Cases ***

Sélectionner dans une liste déroulante

            Select From List By Value    data:test:product_sort_container          za
```

⚠ Avec utilisation de la stratégie de localisation : **data**

Cocher une à case à cocher

script.robot

```
*** Settings ***

Library                 SeleniumLibrary


*** Test Cases ***

Cocher une case à cocher

            Select Checkbox         id:first-choice
```

Le cœur du test : les vérifications

Element Should Be Visible

script.robot

```
*** Settings ***
Library                 SeleniumLibrary


*** Test Cases ***
Le message d'erreur doit etre visible
            Should Be Visible       id:error-message
```

Element Should Not Be Visible

script.robot

```
*** Settings ***
Library                 SeleniumLibrary


*** Test Cases ***
La popup doit disparaitre
            Should Not Be Visible               id:cart-popup
```

Page Should Contain

script.robot

```
*** Settings ***
Library                 SeleniumLibrary


*** Test Cases ***
La page doit contenir du texte
            Page Should Contain             LOGIN
```

Page Should Not Contain

script.robot

```
*** Settings ***

Library                 SeleniumLibrary


*** Test Cases ***

La page ne doit pas contenir du texte

            Page Should Not Contain                     S'inscrire
```

Element Should Be Enabled

script.robot

```
*** Settings ***
Library                 SeleniumLibrary


*** Test Cases ***
L'élément doit être activé
            Element Should Be Enabled          id:login-button
```

Element Should Contain

script.robot

```
*** Settings ***
Library                 SeleniumLibrary


*** Test Cases ***
L'élément doit contenir un texte
            Element Should Contain          Se connecter
```

Element Text Should Be

script.robot

```
*** Settings ***

Library                 SeleniumLibrary


*** Test Cases ***

Le message d'erreur doit être nom d'utilisateur vide

            Element Text Should Be          id:error-message        nom d'utilisateur vide
```

List Should Have No Selections

script.robot

```
*** Settings ***

Library                 SeleniumLibrary


*** Test Cases ***

La liste ne doit pas avoir d'élément selectionné

            List Should Have No Selections                  id:sort-list
```

Location Should Be

script.robot

```
*** Settings ***

Library                 SeleniumLibrary


*** Test Cases ***

L'url doit être égale

            Location Should Be                  https://www.saucedemo.com/inventory.html
```

Location Should Contain

script.robot

```
*** Settings ***
Library                 SeleniumLibrary


*** Test Cases ***
L'url doit contenir /inventory.html
            Location Should Contain             /inventory.html
```

Get Location

script.robot

```
*** Settings ***

Library                 SeleniumLibrary


*** Test Cases ***

Récuperer l'url dans une valeur

            ${currentUrl}           Get Location
            Should Contain          ${currentUrl}           /inventory.html
```

Execute Javascript

script.robot

```
*** Settings ***

Library                 SeleniumLibrary


*** Test Cases ***

Doit scroller au pixel 1000 dans une page

            Execute Javascript                  window.scrollBy(0, 1000);
```

Fermer le navigateur

script.robot

```
*** Settings ***

Library                 SeleniumLibrary


*** Test Cases ***

Ouvrir et fermer le navigateur

            Close Browser
```

Automatiser les scénarios de test pour le site : www.saucedemo.com/

1. Authentification avec succès
2. Authentification en utilisant un **nom d'utilisateur vide** et un mot de passe
3. Authentification avec un nom d'utilisateur et un **mot de passe vide**
4. Authentification avec un utilisateur bloqué **locked_out_user**
5. Test de bout en bout avec achat de produit

Il est impératif d'ajouter des vérifications pour s'assurer que l'utilisateur a été connecté
Ou que le message d'erreur adéquat apparaisse en cas de connexion échouée !

VARIABLES

Types des variables

- ✓ Insensible à la casse
- ✓ En majuscules pour les variables globales
- ✓ En minuscules pur les variables locales
- ✓ Les 4 types de variables sont :

Type	Contenu	Déclaration
Scalaire	Valeur unique	${nom}
Liste	Liste des valeurs	@{nom}
Dictionnaire	Liste de clé / valeur	&{nom}
Variable d'environnement	Nom de la variable	%{nom}

Variables

script.robot

```
*** Settings ***
Library     Collections

*** Variables ***
${var}     Testvar
@{list}     a     b     c     2
&{dict1}    a=1     b=2

*** Test Cases ***
Sc_CreationListetDict
  Ct_CreationListetDict

*** Keyword ***
Ct_CreationListetDict
  ${Var1}  set variable  2
  ${var2}  Evaluate  2023

  @{list} = Create List  a  b  c  d
  ${tmp1} = Get From List ${list}  0

  &{dict} = Create Dictionary  a  1  b  2
  ${tmp2} = Get From Dictionary  ${dict}  b
```

Ressources

Fichier de ressources peut contenir :

- ✓ Settings
- ✓ Variables
- ✓ Keywords
- ✓ Données
- ✓ Librairie
- ✓ …

```
*** Settings ***
Library                 ExcelLibrary

*** Variables ***
${JDD1}                 ../ressources/JDD1.xlsx
${JDD2}                 ../ressources/JDD2.xlsx

*** Test Cases ***
Recup JDD    ${JDD1}    Connexion  2    1

*** Keyword ***
Recup JDD
  [Arguments]   ${JDD}    feuille    ligne    colonne
  Open Excel Document    ${JDD}  docname1
  ${data}=    Read Excel Cell    ligne    colonne    feuille
  [Return]    ${data}
```

Fichier de ressources

script.robot

```
Fichier Ressources : Res.TXT or Res.robot ...
*** Settings ***
Library                 SeleniumLibrary

*** Variables ***
${URL}                  Https://www.google.fr

*** Keyword ***
Login
  Open Browser  ${URL}

Test cases
*** Settings ***
Resource    Res.robot

*** Testcases ***
Login
```

Setup & Teardowm

- **Setup**
  - *s’exécute avant le cas de test*
  - *Exemple : Connexion*

- **Teardown**
  - *s’exécute après le cas de test*
  - *Exemple : Déconnexion*

script.robot

```
*** Settings ***
Library                 SeleniumLibrary
Test Setup    Connexion
Test Teardown    Deconnexion

*** Variables ***
${URL}                  Https://www.google.fr


*** Test Cases ***
Cas passant
  Connexion
  Saisie
  Verification
  Deconnexion

Cas Erreur
  Connexion
  Saisie
  Verification
  Deconnexion
```

```
*** Test Cases ***
Cas passant
  Saisie
  Verification


Cas Erreur
  Saisie
  Verification
```

Tags

➢ **[Tags]**

Une option du test cases qui permet de déclarer des tags , afin de servir lors de l'exécution

❑ *Exemple :*

- *Inclure un tag => $robot –i [tag] testcase.robot*
- *Exclure le tag => $robot –e [tag] testcase.robot*

script.robot

```
*** Settings ***
Library                 SeleniumLibrary
Force Tags        GlobalTest
Default Tags      Nominal

*** Variables ***


*** Test Cases ***
Cas passant
  Connexion
  Saisie
  Verification
  Deconnexion

Cas Erreur
  [Tags]  Erreur
  Connexion
  Saisie
  Verification
  Deconnexion
```

En complément :

- un ensemble de test cases constitue une test suite
- un dossier contenant des test cases forme une test suite
- un dossier test suite peut contenir des sous-dossiers eux-mêmes test suites
- Les fichiers/dossiers commençant par « . » ou « _ » seront ignorés ;
- les dossiers contenant « CVS » dans leur nom seront ignorés ;
- les fichiers de tests doivent avoir une extension reconnue (.txt recommandée) ;
- par défaut, RFW traite toutes les variables comme du texte.

Test API

Automation Tester Academy

Pré-requis

Pour test les APIs nous avons besoin des librairies suivantes :

- ✓ **SeleniumLibrary** installée auparavant
  Sinon "pip install robotframework-seleniumLibrary"
- ✓ **RequestsLibrary** – à installer avec la commande "*pip install robotframework-requests*"
- ✓ **JSONLibrary** – à installer "*pip install robotframework-jsonlibrary*"

- ✓ **Collections** – elle fait partie du package de base , il suffit de l'importer

- ✓ **BuiltIn** – Rien à faire, c'est la librairie standard de RFW ,
  pas besoin de l'importer non plus

Configuration

```
*** Settings ***    #import des librairies Requests et Json
Library  RequestsLibrary   # pip install robotframework-requests
Library  Collections
Library  JSONLibrary        # pip install robotframework-jsonLibrary



*** Variables ***
# l'URL principale de l'API (la partie fixe)
${url}        https://jsonplaceholder.typicode.com
# La partie relative de l'URL
${irl}        posts


*** Test Cases ***
Test FakeAPI
    # Créer une session HTTP vers le serveur , la nommer ex: "mine"
    Create Session    mine    ${url}
```

Automation Tester Academy

GET : Partie 1

```
# Envoyer une requête "GET" au serveur sur la session précédemment créée
    ${reponse}=    GET On Session    mine    /${irl}/10
    # La réponse sous format JSON s'affiche dans le rapport des logs , c'est un
dictionnaire "clé=valeur"
    Log  ${reponse.json()}
    # Pour accéder aux différents attributs de la réponse
    Log  ${reponse.json()}[id]
    Log  ${reponse.json()}[userId]
    Log  ${reponse.json()}[title]
    Log  ${reponse.json()}[body]
    # Chaque requête dispose d'un code, contenu (content) et infos d'entete
    Log    ${reponse.status_code}
    Log    ${reponse.content}
    Log    ${reponse.headers}
    # Convertir le contenu de la réponse en chaine de caractére
    ${body}=    convert to string    ${reponse.content}
```

Automation Tester Academy

GET : Partie 2

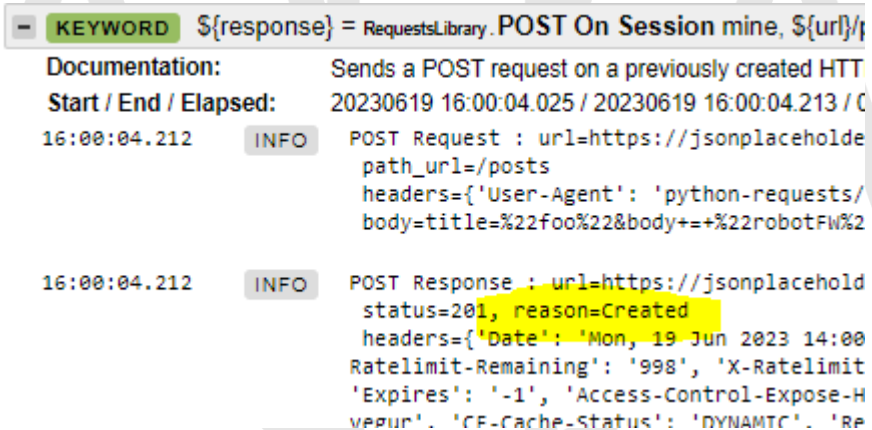
```
# Récupérer la valeur associée à la clé "Content-Type" du dictionnaire "headers" de la réponse
    ${contenu}    Get From Dictionary    ${reponse.headers}    Content-Type
    Log    ${contenu}
    # une autre façon de récupérer un élément du dictionnaire
    Log    ${reponse.headers}[Content-Type]
    # Vérifier la conformité des données de la réponse : code / contenu / headers
    # les vérifications se font par les keywords commençant par "Should"
    Should Be Equal As Integers    ${reponse.status_code}    200
    Should Contain    ${body}    "optio molestias id quia eum"
    Should Contain    ${reponse.headers}[Content-Type]    application/json
```

Automation Tester Academy

Post : Partie 1

```
#Créer les dictionnaires "body" (Content)  et "header" ( l'entete de la notre requete)
${body}=    Create dictionary    title="foo"    body = "robotFW"    userId = "25  id = "101"
${header}=    Create dictionary    Content-Type="application/json"

# Envoyer une requête POST au serveur avec le contenu et entête créés précédemment
${response}=    POST On Session    mine    ${url}/posts    data=${body}    headers=${header}
```

L'attribut « reason » et le « code-status » font partie de la réponse de la methode HTTP POST



Automation Tester Academy

Post : Partie 2

```
    # Recuperer l'attribut "Reason" de la réponse puis le concrtir
en string pour faciliter verification
    ${strContent}=    Convert To String    ${response.reason}
    # Vérifier les elements de la réponse , code et raison
    Should Contain    ${strContent}    Created
    Should Be Equal As Integers    ${response.status_code}    201
    Log    ${response.status_code}
    Log    ${response.content}
    Log    ${response.headers}
```

Dans notre exemple il n'y a pas de Màj des données au niveau du serveur mais en réalité on peut faire un GET sur ce qu'on a créé par le POST

Exemple fichier JSON

Début du fichier


```
{
    "name": "United States of America",
    "topLevelDomain": [
        ".us"
    ],
    "alpha2Code": "US",
    "alpha3Code": "USA",
    "callingCodes": [
        "1"
    ],
    "capital": "Washington, D.C.",
    "altSpellings": [
        "US",
        "USA",
        "United States of America"
    ],
    "region": "Americas",
    "subregion": "Northern America",
    "population": 323947000,
    "latlng": [
        38,
        -97
    ],
    "demonym": "American",
    "area":  9629091,
    "gini":  48,
    "timezones": [
        "UTC-12:00",
        "UTC-11:00",
],
    "borders": [
        "CAN",
        "MEX"
    ],
 "nativeName": "United States",
    "numericCode":  "840",
    "currencies": [
        {
            "code": "USD",
            "name": "United States dollar",
            "symbol": "$"
        }
    ],
    "languages": [
        {
            "iso639_1": "en",
            "iso639_2": "eng",
            "name": "English",
            "nativeName": "English"
        }
    ],
    "translations": {
        "br": "Estados Unidos",
        "de": "Vereinigte Staaten von Amerika",
        "es": "Estados Unidos",
        "fa": "\u0627\u06cc\u0627\u0644\u0627\u062a \u0645\u062a\u062d\u062f\u0647",
        "fr": "\u00c9tats-Unis",
        "hr": "Sjedinjene Ameri\u010dkeDr\u017eave",
        "it": "Stati Uniti D\u0027America",
        "ja": "\u30a2\u30e1\u30ea\u30ab\u5408\u8846\u56fd",
        "nl": "Verenigde Staten",
        "pt": "Estados Unidos"
    },
    "flag": "https:\/\/api.countrylayer.com\/v2\/",
    "regionalBlocs": [
        {
            "acronym": "NAFTA",
            "name": "North American Free Trade Agreement",
            "otherNames": [
                "Tratado de Libre Comercio de Am\u00e9rica del Norte",
                "Accord de Libre-u00e9change Nord-Amu00e9ricain"
            ]
        }
    ],
    "cioc":  "USA"
}
```


Fin du fichier

Automation
Tester Academy

Manipulation d'un fichier JSON

```
# Récupérer un fichier JSON au bon format ( sans erreur)
${json_obj}=    Load Json From File    C:\\Users\\...\\testfichier.json
# Récupérer les données via la notation Point / crochets
${json_data1}=    Get Value From Json    ${json_obj}    $.name
Log    ${json_data1}
${json_data2}=    Get Value From Json    ${json_obj}    $.translations
Log    ${json_data2}
```

L'idée est de vérifier les données du fichier via les Keywords « Should … »
ou bien de comparer les données du fichier avec la réponse de la methode GET d'une requête HTTP .

Liens utiles

Pour bien comprendre la structure JSON et manipuler les données

- ✓ ***https://developer.mozilla.d/Objects/JSON***

Pour vérifier le bon format JSON d'un fichier

- ✓ ***https://jsonr.copathfindem/***

Pour exploiter les données via la notation point / crochets

- ✓ ***https://jsonpath.com/***

Versionning : GIT & GITHUB

Git et **Github** sont deux outils indispensables dans la gestion du travail collaboratif sur un projet de développement logiciel.

Git est un logiciel de **versioning décentralisé**

Installation :

https://gitforwindows.org/

vérification : git --version

Configuration :

Dans le git bash / terminal :

```
git config --global user.name "Votre nom"
git config --global user.email votre@email.com
```

Les commandes principales :

git init : Transforme le dossier courant en dépôt git
ls –a : affiche le contenu du dossier y compris les éléments cachés
git add : ajoute le fichier à l'index pour le commiter
git restore –staged : désindexer un fichier
git status : l'etat des lieux des fichiers à commiter
git commit -m ''message''
git log : afficher la liste des commits
git log –help
git checkout –b nom_de_la_branche : créer une branche
git branch –m : renommer une branche
git branch –D : supprimer une branche
git merge nom_de_la_branche_a_fusionner : il faut se positionner sur la branche de destionation

Github est un service web d'hébergement de code

Créer un compte sur ***https://github.com/***

Etapes :
Créer un nouveau repository

Générer une clé SSH sur son ordinateur
`ssh-keygen -t ed25519 -C emailGithub@gmail.com`

Copier le contenu du fichier de la clé
`Clip < ~/.ssh/id_ed_25519.pub`

Dans l'interface github > settings > add ssh key
`Coller le contenu et valider`

Vérifier la connexion
`Ssh -T  git@github.com`

Commandes principales

Ajouter le repository distant comme origine (référence)

```
git remote add origin  git@github.com:usernamegithub/nomrepository.git
```

Renommer la branche master en main

```
git branch –M main
```

Pousser la branche locale sur la branche main de github

```
Git push –u origin main
```

Copier un projet en local à partir de Github

```
git clone <le lien copié> ( lien copié à partir de github code puis SSH )
```

Liens Utiles

C’est en anglais mais il suffit de refaire les exemples :

- ✓ https://youtu.be/zcT8hSipe2A

Pour mieux comprendre les boucles avant de voir l’application dans RFW

- ✓ https://openclassrooms.com/fr/courses/19980-apprenez-a-programmer-en-c/14722-repetez-des-instructions-grace-aux-boucles

Pour se former en HTML , CSS et JS

- ✓ https://www.w3schools.com/html/
- ✓ https://www.w3schools.com/css/
- ✓ https://www.w3schools.com/js/

Besoin d’un accompagnement pour se former !!!

- ✓ https://www.automationtesteracademy.com/

Pour toute question

Anissformation23@gmail.com

GOOD LUCK