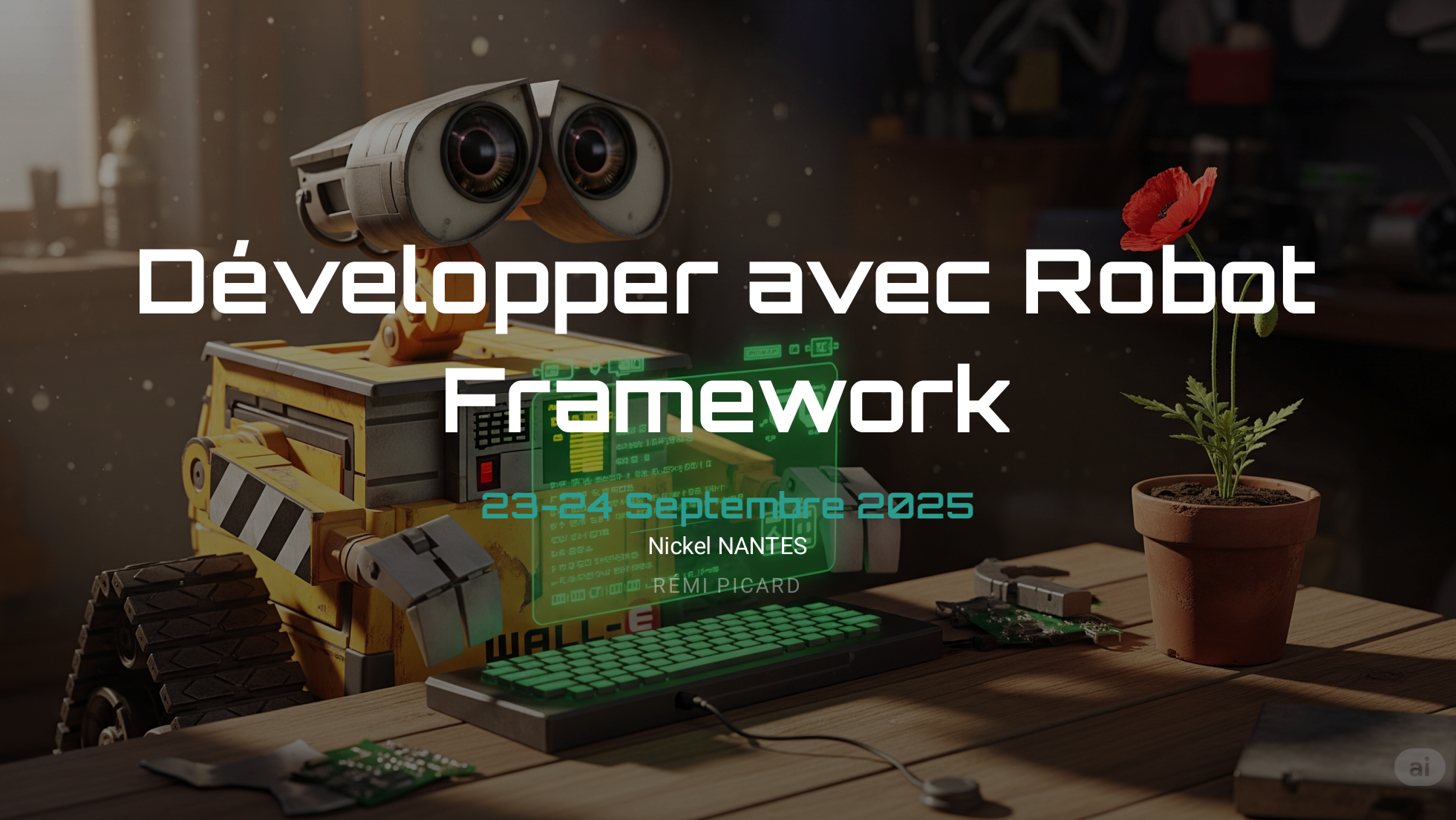


Développer avec Robot Framework

23-24 Septembre 2025

Nickel NANTES

RÉMI PICARD

Tour de table

- Expérience dev
- Expérience testeur
- Expérience avec Robot Framework
- Attente vis à vis de cette formation ?

Qui suis-je ?

- Rémi PICARD
- Dev Scala Cobalt
- 4 ans chez Nickel
- 4 ans d'expérience avec Robot Framework 🤖
- Passionné par les technos Web, Data et DevOps
- 13 ans d'expérience dans l'IT 👾
- Joueur d'échecs ♟

Jour 1

Découverte de Robot Framework

Robot Framework => "Robot"

RF / RBF / RBT

Plan

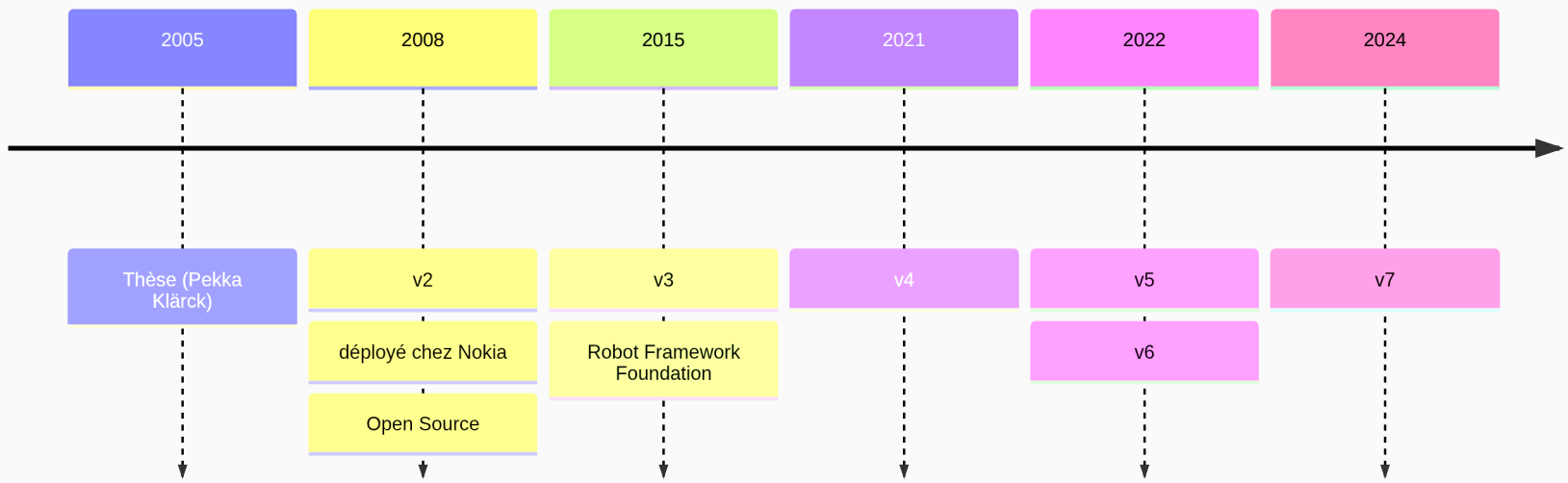
- Généralités
- Rappels Python
- Comprendre le fonctionnement de Robot
- Installer l'env de dev
- Apprendre le langage Robot
- Découvrir la ligne de commande `robot`
- Codelab Tests API
- Codelab Tests UI avec Playwright

Présentation

- Outil d’automatisation
- Langage
- Open Source codé en Python
- Fonctionnalités clefs en main (assertions, rapport de tests…)
- Extensible via des librairies Robot Framework (HTTP, JSON, SQL, Kafka …)
- Extensible via des librairies Python

Histoire

20 ans déjà !

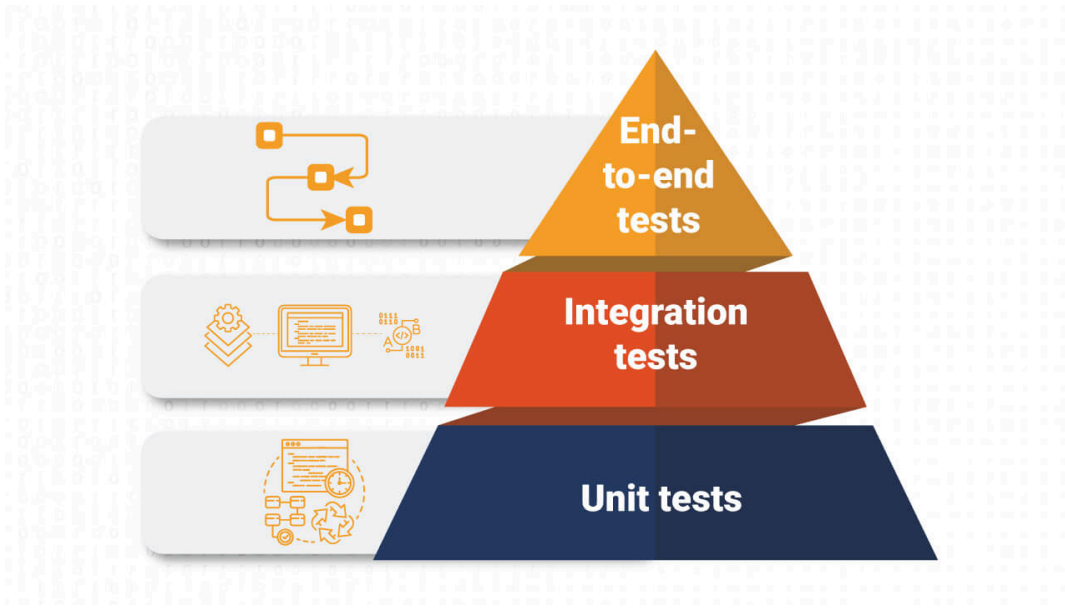


Communauté

- Slack
- RoboCon / RBCN : conférence annuelle à Helsinki 🇫🇮
- Documentation

Pyramide des tests

Tests End-To-End

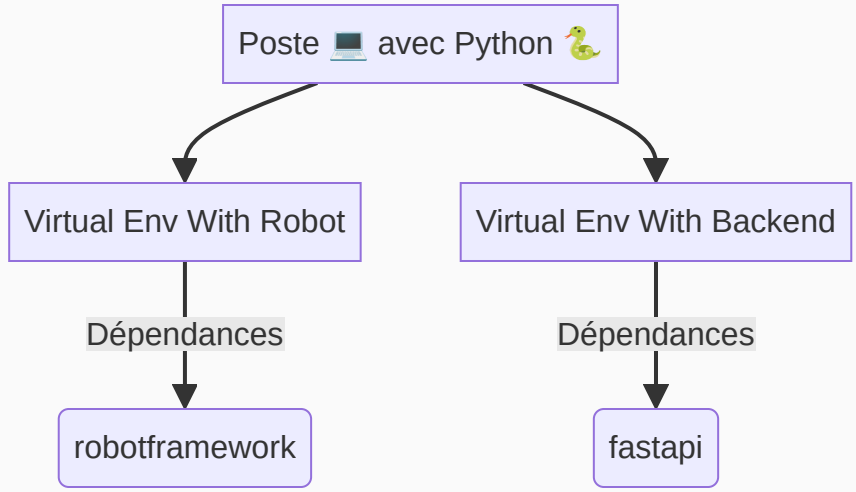


Source : https://blog.takima.fr/saffranchir-de-la-pyramide-des-tests/

Env de dev 💻

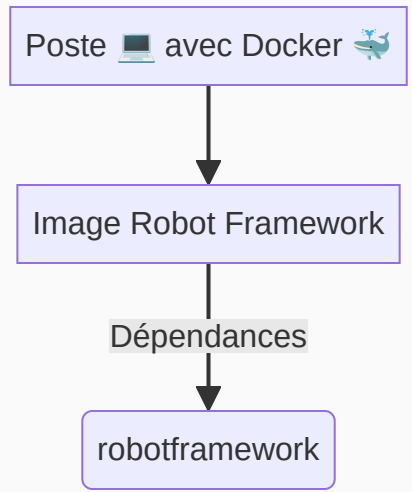
Installation

Python Virtual Environment



Python Virtual Environment

Installation
Docker



Installation
PyCharm / VSCode

Set up your IDE

Codelab 💻

- Installation env de dév
- Hello World
- Configuration IDE

Développer avec Python 🐍

Rappels

Types natifs

Type	Mot-clé	Exemple
Entier	`int`	`42`, `-17`, `0`
Flottant	`float`	`3.14`, `2.5e-3`
Booléen	`bool`	`True`, `False`
Chaîne	`str`	`"Hello"`, `'World'`
Tableau	`list`	`["Hello", "World"]`
Map	`dict`	`{"key": "value", "key2": "value2"}`
Aucun	`NoneType`	`None`

Documentation

Méthodes

```python
def ma_fonction(arg1: int, arg2: str, default_arg="default", *args, **kwargs) -> int:
    # ...
    return 42

ma_fonction(42, "Quarente-deux")
ma_fonction(42, "Quarente-deux", "override default value")
ma_fonction(42, "Quarente-deux", "default", 1, 2, 3)
ma_fonction(42, "Quarente-deux", "default", robot="Nono", john="Doe")
ma_fonction(42, "Quarente-deux", "default", 1, 2, 3, robot="Nono", john="Doe")
```

Développer avec Robot Framework 🤖

Tests, Variables, Keyword, Python, Structure ...

Tests

- fichier `*.robot`
- indentation comme en Python 🐍
- ensemble de phrases 🪄
- Section `*** Test Cases ***`
- Import librairies dans `*** Settings ***`

Tests

Déclaration

```
*** Settings ***
Library     String

*** Test Cases ***
Mon Premier Test
    ${chaine}=    Generate Random String    10
    Log    Hello ${chaine}
```

Variables

- Même types qu'en Python 🐍
- Syntaxe `${...}` (comme en Bash)
- Création dans un Test ou Keyword
- Création en global dans la section `*** Variables ***`
- Import des variables Python possible

Variables

Section Variables

```
*** Variables ***
${nombre}       42
${chaine}       Ma chaîne de caractères
@{tab}          1    2    3
&{map}          clef1=valeur1    clef2=valeur2

*** Test Cases ***
Teste Variables
    Log    nombre=${nombre}
    Log    chaine=${chaine}
    Log    tab=${tab}
    Log    map=${map}
```

Variables

Variables

Déclaration

```
*** Test Cases ***
Creation Variable
    ${ma_variable}    Set Variable    C3PO
    Log    ma_variable=${ma_variable}
```

Variables

Portée

- Local `Set Variable`
- Test `Set Test Variable`
- Suite `Set Suite Variable`
- Global `Set Global Variable`
- 💡Limiter au maximum la portée

Variables

Import YAML

- option `--variablefile` / `-V`
- `robot --variablefile conf/local.yaml tests/14-variablefile.robot`

Variables

Import Python

```python
# resources/mes_variables_python.py
variable_python = 42
```

```
*** Settings ***
Variables    resources/mes_variables_python.py

*** Test Cases ***
Utiliser Variable Python
    Log    variable_python=${variable_python}
```

Variables

ENV

- Syntaxe `%{VARIABLE_ENV=default_value}`

Injection Python 🐍

Evaluate

```
*** Test Cases ***
Teste Evaluate
    ${nb}=    Evaluate    41 + 1
    Log    nb=${nb}

Teste Evaluate Autre Syntaxe
    ${nb}=    Set Variable    ${{41 + 1}}
    Log    nb=${nb}
```

Keyword

Concept

- Ensemble de mots clés (séparés par 1 espace)
- Forme une phrase 🪄
- Représente une **action** 🕹️
- Déclaré dans la section `*** Keywords ***`
- Robot Framework traduit les phrases en appels Python 🐍

Keyword
Syntaxe Robot Framework

```
*** Keywords ***
Mon Premier Keyword
    Log    Hello World

Mon Premier Keyword Avec Argument
    [Arguments]    ${name}
    Log    Hello ${name}

Mon Premier Keyword Avec Argument Et Return
    [Arguments]    ${name}
    Log    Hello ${name}
    RETURN    42
```

Keyword
Arg dans Keyword

```
*** Keywords ***
Keyword Avec ${arg1} Intégré
    Log    Hello ${arg1}

Keyword Avec ${arg1} Intégré Et Arguments
    [Arguments]    ${name}
    Log    Hello ${arg1}, ${name}

*** Test Cases ***
Appel Keywords
    ${arg1}    Set Variable    Ma Variable
    Keyword Avec ${arg1} Intégré
    Keyword Avec ${arg1} Intégré Et Arguments    Arg2
```

Keyword

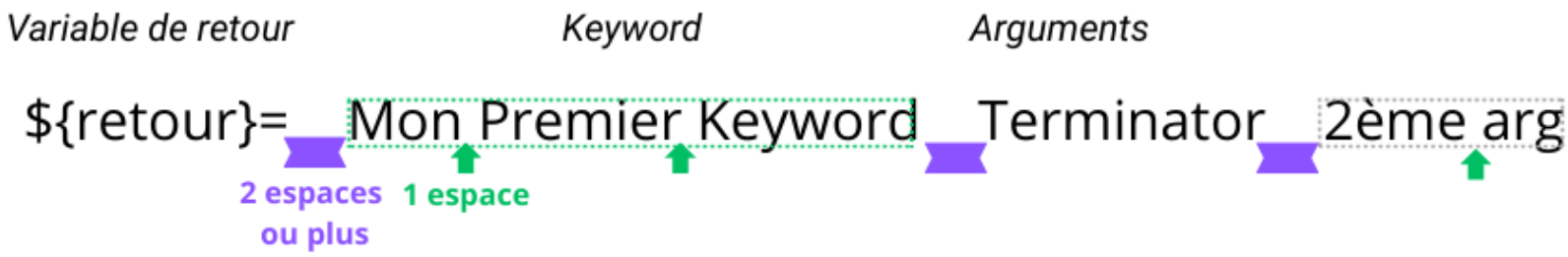
List (args) / Dict (kwargs)

```
*** Keywords ***
Keyword Avec Args
    [Arguments]    @{list}
    FOR    ${i}    IN    @{list}
        Log    i=${i}
    END

Keyword Avec Kwargs
    [Arguments]    &{map}
    FOR    ${k}    ${v}    IN    &{map}
        Log    key=${k}, value=${v}
    END

*** Test Cases ***
Appel Keywords
    ${list}    Create List    1    2    3
    Keyword Avec Args    ${list}

    ${map}    Create Dictionary    cle1=valeur1    cle2=valeur2
    Keyword Avec Kwargs    &{map}
```

Keyword

Gestion des espaces

- **1 espace** entre chaque mot
- **2 espaces ou +** (ou **tabulation**) entre chaque argument



Keyword

Gestion des chaînes

- inutile de mettre des `"` ou des `'` autour des chaînes de caractère

Keyword

Appel

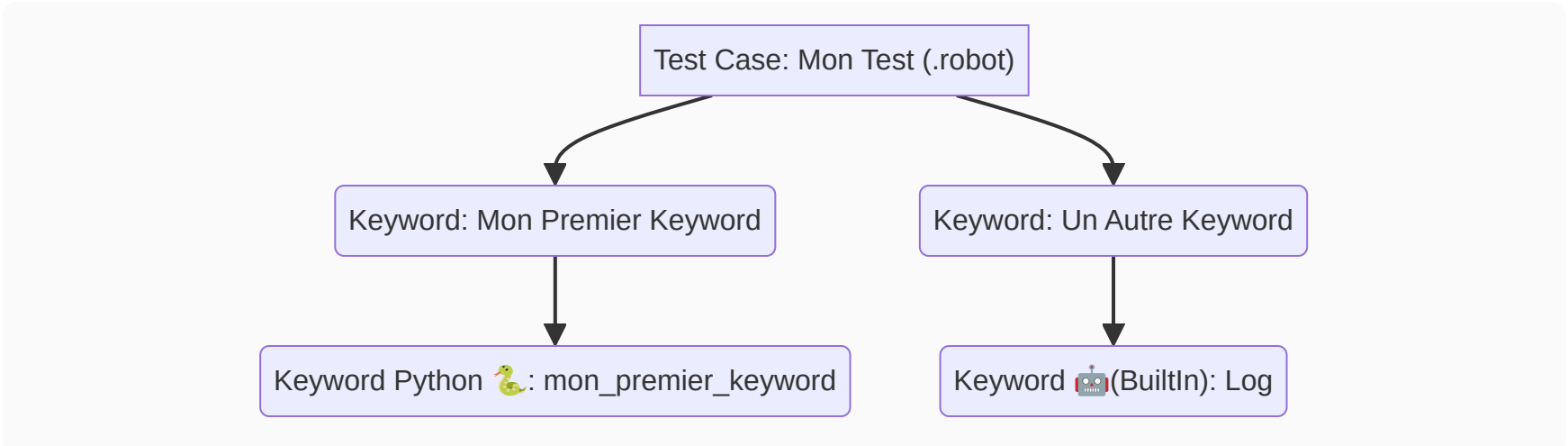
```
*** Test Cases ***
Mon Premier Test
    Mon Premier Keyword
    Mon Premier Keyword Avec Argument    Nono le petit robot
    ${retour}=    Mon Premier Keyword Avec Argument Et Return    Terminator

*** Keywords ***
Mon Deuxième Keyword
    Mon Premier Keyword
```

Keyword
Syntaxe Python

```python
# mon_keyword.py
# Mon Premier Keyword
def mon_premier_keyword():
    print("Hello World")

# ${retour}=    Mon Premier Keyword Avec Argument Et Return    ${name}
def mon_premier_keyword_avec_argument_et_return(name) -> int:
    print(name)
    return 42
```

```
*** Settings ***
Library    mon_keyword.py

*** Test Cases ***
Mon Test
    Mon Premier Keyword
    ${retour}=    Mon Premier Keyword Avec Argument Et Return    R2D2
```

Test -> Keyword -> Keyword



Sections

- `*** Settings ***` : imports librairies / ressources / variables
- `*** Test Cases ***` : déclarations Tests
- `*** Keywords ***` : déclarations Keywords
- `*** Variables ***` : déclarations Variables

Structures

- boucle `FOR`
- boucle `WHILE`
- condition `IF` / `ELSE`
- gestion erreur `TRY` / `EXCEPT`
- Control structures

Ligne de commande `robot`

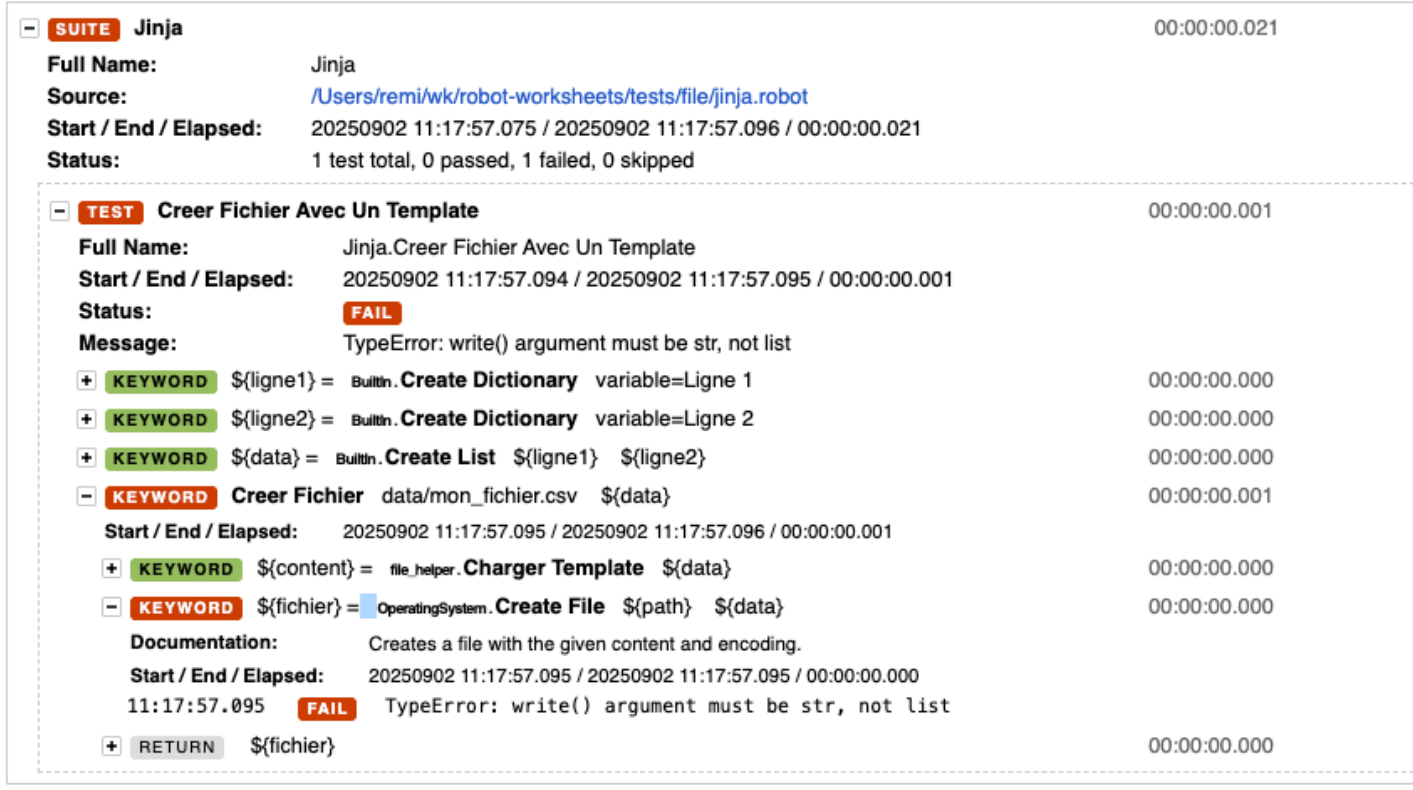
```
robot --help

# Lance tous les tests présents dans le dossier tests
robot tests

# Lance le test "Mon Test"
robot -t "Mon Test" tests
```

Résultats

- `log.html` : détails par tests et keywords
  - 🔴 ERROR => automatiquement affiché (avec focus sur le keyword en erreur)
  - 🟢 SUCCESS
- `output.xml` / `xunit.xml` : sortie technique pour intégration continue / outils

Résultats

log.html



Quizz

Lien ou passer à la slide suivante

Sign in to your Google Account

You must sign in to access this content

Sign in

TP Keyword 💻

Tester les syntaxes

- Keywords
- Tests
- Variables

Requests Library

- Tests d'API
- Wrapper de la lib Python requests
- Documentation
- Manipulation JSON simple

Requests Library

```
pip install robotframework-requests
```

Requests Library

```
*** Settings ***
Library     RequestsLibrary

*** Test Cases ***
Quick Get Request Test
    ${response}    GET    https://www.google.com

Quick Get Request With Parameters Test
    ${response}    GET    https://www.google.com/search    params=query=ciao    expected_status=200

Quick Get A JSON Body Test
    ${response}    GET    https://jsonplaceholder.typicode.com/posts/1
    Should Be Equal As Strings    1    ${response.json()}[id]

Create Booking
    ${booking_dates}    Create Dictionary    checkin=2022-12-31    checkout=2023-01-01
    ${body}    Create Dictionary
    ...    firstname=Hans
    ...    lastname=Gruber
    ...    totalprice=200
    ...    depositpaid=false
    ...    bookingdates=${booking_dates}
    ${response}    POST    url=https://restful-booker.herokuapp.com/booking    json=${body}
    ${id}    Set Variable    ${response.json()}[bookingid]
```

Codelab 💻

API Booker

Browser Library

- Tests UI
- Wrapper de Playwright
- Documentation
- Keywords
- Exemples / Comparaison

Browser Library

- Rapide 🚀
- Fiable ✅
- Contrôle du navigateur 👨‍✈️
- Remplaçant de Selenium Library

Browser Library

```
# Installer Node https://nodejs.org/en/download/

pip install robotframework-browser

# Télécharge le navigateur
rfbrowser init
```

Browser Library

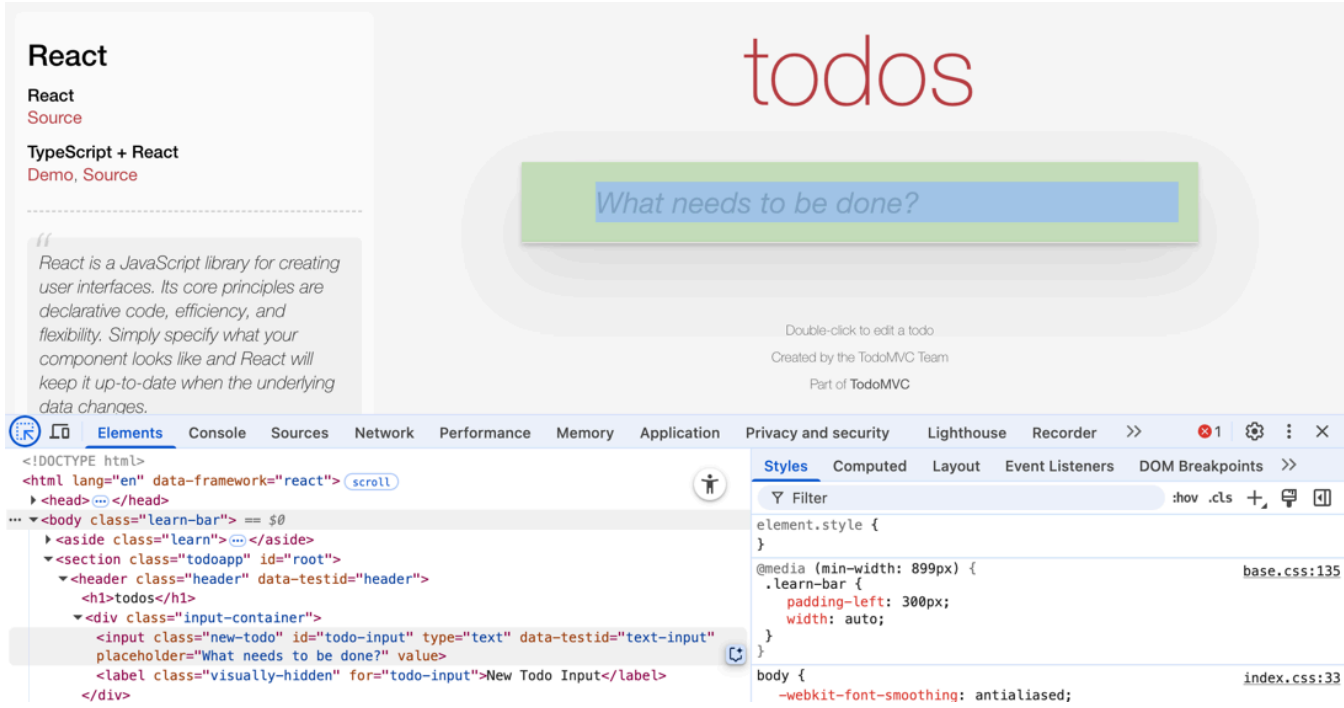
```
*** Settings ***
Library     Browser

*** Test Cases ***
Go To Playwright With Browser Library
    # Opens a new browser instance. Use this keyword for quick experiments or debugging sessions.
    Open Browser

    New Page    https://playwright.dev/
    Get Title    contains    Playwright
    Take Screenshot

    Click    a >> "Get started"
    Get Element States    h1 >> "Installation"    contains    visible
    Take Screenshot
```

Sélecteurs

- CSS (id, class)
- XPATH

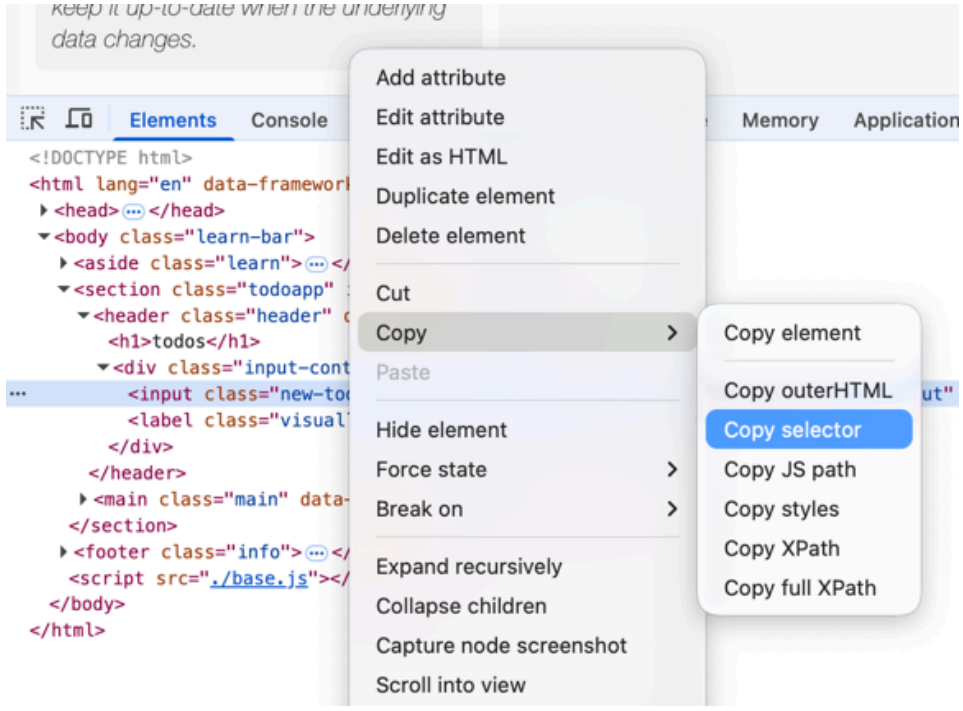
Selecteurs

DevTools



Selecteurs

DevTools



Codelab 💻

TODO MVC

Jour 2

Bonnes pratiques et Industrialisation

Plan

- Organiser les tests et les ressources
- Utiliser les librairies standards de Robot Framework
- Découvrir les syntaxes avancées
- Ecrire des tests robustes
- Découvrir les outils autour de Robot Framework
- Mettre en place une intégration continue

Organiser les tests et les ressources

Structure standard d'un projet

- `tests/` : fichiers de tests (`.robot`) et suite de tests
- `resources/` : keywords partagés, variables et librairies maison (`.resource` , `.py`)

```
mon-projet/
├── tests/
│   └── cas_de_test.robot
│   └── suite/
│       └── autre_cas_de_test.robot
├── resources/
│   └── keyword_communs.resource
│   └── ma_librairie.py
```

RobotFramework - Project Structure

Syntaxes avancées 

Test Template

```
*** Test Cases ***
Normal test case with embedded arguments
    The result of 1 + 1 should be 2
    The result of 1 + 2 should be 3

Template with embedded arguments
    [Template]    The result of ${calculation} should be ${expected}
    1 + 1    2
    1 + 2    3

*** Keywords ***
The result of ${calculation} should be ${expected}
    ${result} =    Evaluate    ${calculation}
    Should Be Equal As Strings    ${result}    ${expected}
```

- Test Templates

Behavior Driven Development (BDD)

Syntaxe Given-When-Then

```
*** Test Cases ***
Mon Test Bdd
    Given Pré Requis
    When Action
    Then Verifications

*** Keywords ***
Pré Requis
    Log    Pré Requis

Action
    Log    Action

Verifications
    Log    Verifications
```

BDD

Test Execution Log

SUITE **09-Bdd**

Full Name:	09-Bdd
Source:	/Users/remi/wk/robot-examples/tests/09-bdd.robot
Start / End / Elapsed:	20250903 14:07:37.963 / 20250903 14:07:37.973 / 00:00:00.010
Status:	1 test total, 1 passed, 0 failed, 0 skipped

TEST **Mon Test Bdd**

Full Name:	09-Bdd.Mon Test Bdd
Start / End / Elapsed:	20250903 14:07:37.973 / 20250903 14:07:37.974 / 00:00:00.001
Status:	PASS

KEYWORD **Given Pré Requis**

Start / End / Elapsed: 20250903 14:07:37.973 / 20250903 14:07:37.973 / 00:00:00.000

KEYWORD BuiltIn . **Log** Pré Requis

KEYWORD **When Action**

Start / End / Elapsed: 20250903 14:07:37.973 / 20250903 14:07:37.973 / 00:00:00.000

KEYWORD BuiltIn . **Log** Action

KEYWORD **Then Verifications**

Start / End / Elapsed: 20250903 14:07:37.973 / 20250903 14:07:37.973 / 00:00:00.000

KEYWORD BuiltIn . **Log** Verifications

Utiliser les librairies standards 🪛

Standard Library

- Inclus avec Robot Framework Core
- BuiltIn (importé automatiquement)
- String
- Collections
- DateTime
- OperatingSystem
- Screenshot
- Process
- XML
- Standard Library

Standard Library

Import

```
*** Settings ***
Library  Collections
Library  OperatingSystem
Library  Process
Library  String
```

BuiltIn

```
*** Test Cases ***
Some Assertions
    ${var}=    Set Variable    1.0
    Should Not Be Equal    1    ${var}
    Should Be Equal As Numbers    1    ${var}
    Should Be Equal As Strings    1.0    ${var}
    Should Be True    ${var} == 1.0
    Should Not Be True    ${var} > 10

    ${string}=    Set Variable    My String
    Should Start With    ${string}    My
    Length Should Be    ${string}    9
```

Gestion des fichiers 🗂️

Fichier Template

- Templates Jinja2
- Création contenu de fichiers
- Couplé à `OperatingSystem`

Fichier Template

- Template Jinja2 `templates/mon_template.csv.j2`

```
entete1;entete2
{%- for d in data %}
CONSTANTE;{{ d.variable }}
{%- endfor %}
```

- Utilisation dans Keyword Python

```python
from jinja2 import Environment, FileSystemLoader

# Initialiser l'environnement avec un dossier de templates
env = Environment(loader=FileSystemLoader("templates"))
template = env.get_template("mon_template.csv.j2")


def charger_template(data):
    contenu = template.render(data=data)
    return contenu
```

Fichier Template

- Utilisation dans Robot Framework

```
*** Settings ***
Library     OperatingSystem
Library     resources/file_helper.py

*** Test Cases ***
Creer Fichier Avec Un Template
    ${ligne1}    Create Dictionary    variable=Ligne 1
    ${ligne2}    Create Dictionary    variable=Ligne 2
    ${data}    Create List    ${ligne1}    ${ligne2}
    Creer Fichier    data/output/mon_fichier.csv    ${data}

*** Keywords ***
Creer Fichier
    [Arguments]    ${path}    ${data}
    ${contenu}    Charger Template    ${data}
    ${fichier}    Create File    ${path}    ${contenu}
    ${contenu}    Get File    ${path}
    Log    ${contenu}
```

Fichier Template

```
1  entete1;entete2
2  CONSTANTE;Ligne 1
3  CONSTANTE;Ligne 2
```

TP Fichiers 💻

- Créer et remplir un fichier
- Récupérer le fichier
- Déplacer le fichier
- Supprimer le fichier

Autres Librairies / Outils 🛠️

Autres Librairies

- Requests Library
- Browser Library
- Database
- JSON
- Kafka
- DataDriver

DataDriver

- JSON, CSV ou Excel => Données de tests
- `pip install robotframework-datadriver`

PaBot

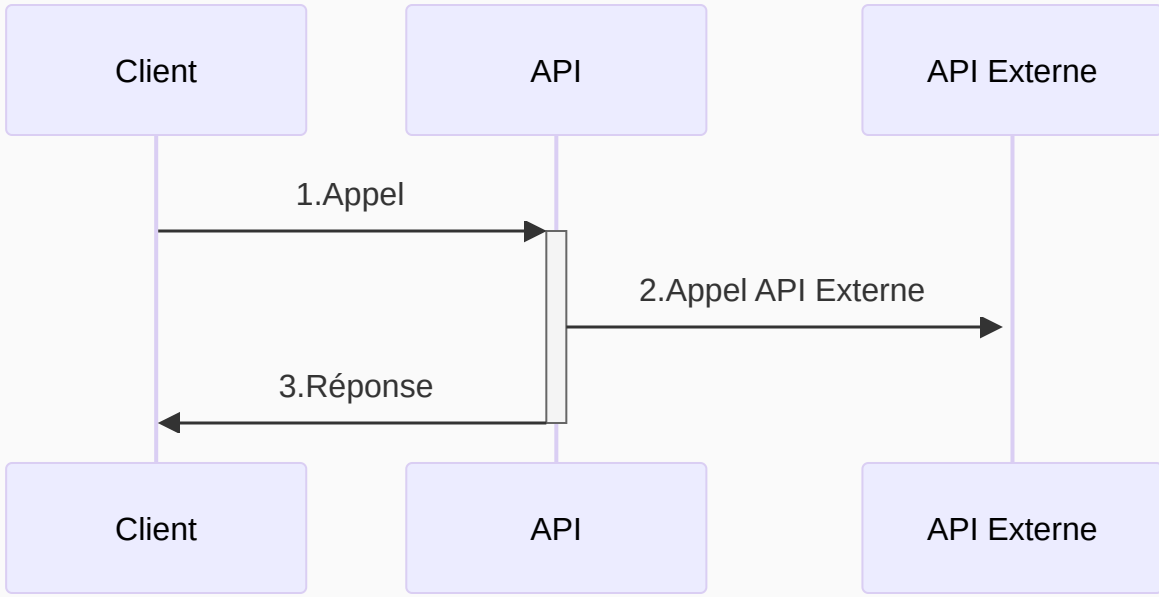
- Lancement des tests en parallèle 🚀
- ⚠️ Isolation des tests
- CLI `pabot` wrapper de `robot` (même options)
- `pip install robotframework-pabot`
- PaBot

MockServer

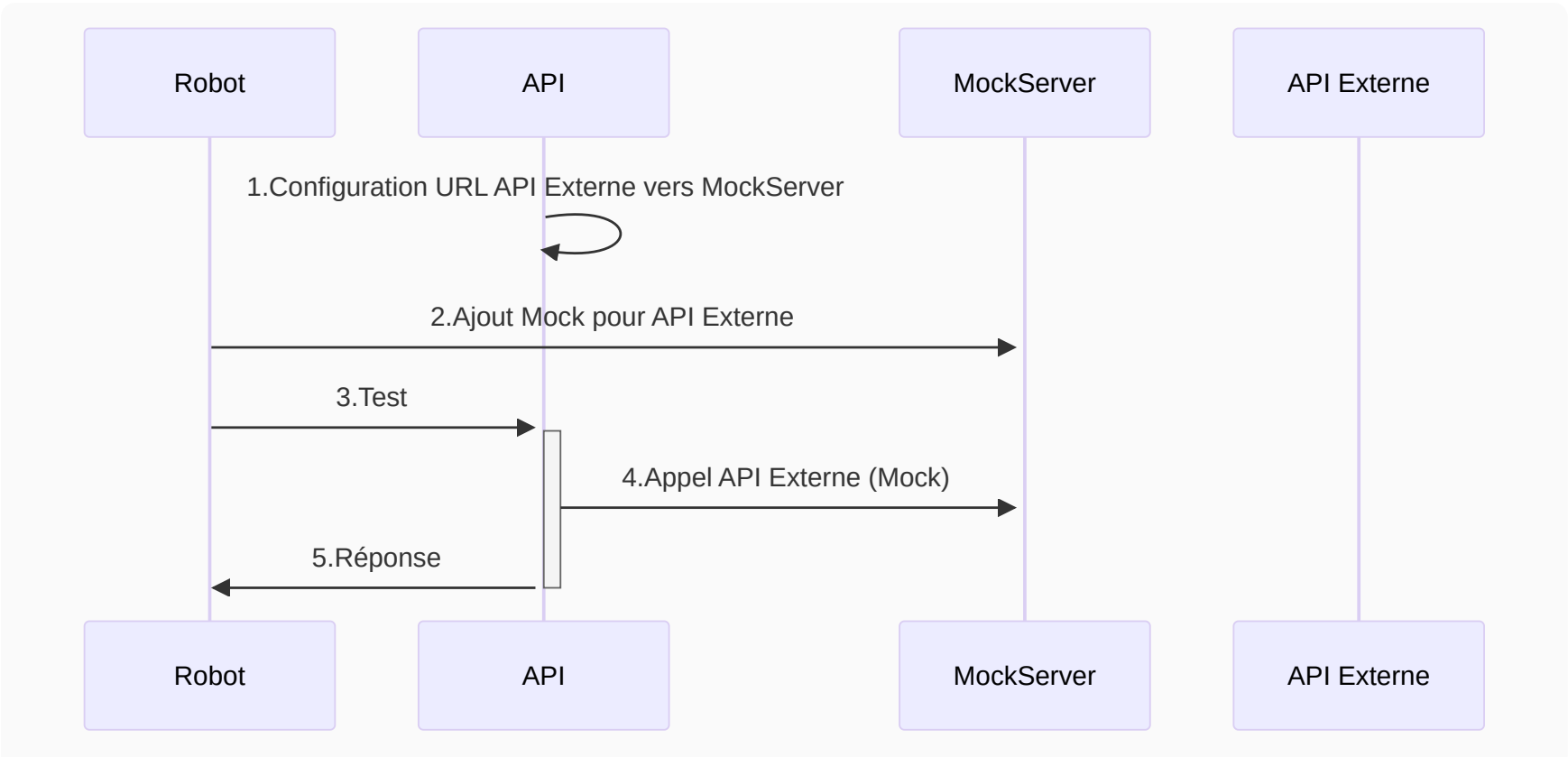
- L’API à tester a besoin d’un service externe
- Le service externe n’est pas disponible sur l’env de test
- MockServer remplace le service externe
- MockServer configurable par API

MockServer

API dépend de API Externe



MockServer
Flow de TEST



Ligne de commande 

Ligne de commande `robot`

```
# Vérifie la syntaxe de tous les tests
robot --dryrun tests

# Ajoute le répertoire courant dans le PYTHON PATH
robot --pythonpath . tests
```

PYTHON_PATH

Sans `--pythonpath` :

```
*** Settings ***
Resource    ../resources/ma_lib.resource

*** Test Cases ***
Tester Ma Lib
    Mon Keyword
```

Avec `--pythonpath .` ou `-P .` :

```
*** Settings ***
Resource    resources/ma_lib.resource

*** Test Cases ***
Tester Ma Lib
    Mon Keyword
```

API Python

- Lancement `robot`
- Réflexion Tests / Keywords (métadonnées, hooks)
- Parsing des résultats en Python

API Python

```
1  import robot
2
3  def run_tests():
4    robot.run("robot", outputdir="data")
```

Pattern Visitor

```python
from robot.running import TestSuiteBuilder
from robot.model import SuiteVisitor

class TagsOnTestCasesFinder(SuiteVisitor):
    def __init__(self):
        self.tests = list()

    def visit_test(self, test):
        self.tests.append(test.name)

def list_tests():
    builder = TestSuiteBuilder()
    testsuite = builder.build("tests/")
    finder = TagsOnTestCasesFinder()
    testsuite.visit(finder)
    for test in sorted(finder.tests):
        print(test)

if __name__ == "__main__":
    list_tests()
```

Listener

- Mécanisme d'envoi de notifications
- Inspection / Modification de données
- Lors de l'exécution d'un test
- Lors de l'exécution d'un keyword
- Lors de l'écriture des rapports
- `--listener`
- Documentation Listener

Listener

Exemple

```python
"""Listener that stops execution if a test fails."""

ROBOT_LISTENER_API_VERSION = 2

def end_test(name, attrs):
    if attrs['status'] == 'FAIL':
        print(f"Test '{name}'" failed: {attrs['message']}")
        input("Press enter to continue.")
```

```
robot --listener path/to/PauseExecution.py tests.robot
```

Bonnes pratiques 🥷

Composition de Keywords

- Passer en argument un Keyword à un Keyword
- Equivalent à passer une lambda ou un callback à une méthode Python 🐍
- `Run Keyword`
- `Run Keyword If`
- `Wait Until Keyword Succeeds`

Logs

Niveaux

- 🟢 TRACE : niveau le plus fin (ex : détails requête HTTP)
- 🟢 DEBUG : trace de debug
- 🟢 INFO : par défaut
- 🟠 WARN : avertissement erreur non bloquante
- 🔴 ERROR : erreur bloquante (le test s’arrête)

Logs
Définir niveau

```
1   # Augmente le niveau de logs à TRACE
2   robot --loglevel TRACE . tests
3
4   # Augmente le niveau à TRACE mais affiche par défaut en INFO
5   robot -L TRACE:INFO . tests
```

Logs
Keyword Python

```python
from robot.api import logger

def log_from_python():
    logger.info("Hello From Python Keyword")
```

Correlation ID

- Remplir Header `X-Request-ID`
- Pousser logs Robot dans Grafana
- Suivre logs des tests et applicatifs

Nommage / Découpage

- Rester générique
- Factoriser
- Donner des noms métier si possible

Documenter

- Bon nommage peut être suffisant
- Documenter les passages importants (non triviaux)

Structures avancées

- Tuple
- class
- dataclass (immutable)
- list-comprehension

Tuple

```
*** Test Cases ***
Teste Les Tuples
    ${robots}=    Get Robots
    Log    ${robots}
    Log    ${robots[0]}
    Log    ${robots[1]}

*** Keywords ***
Get Robots
    RETURN    R2D2    C3PO
```

Dataclass

```python
@dataclass
class ImmutableRobot:
    name: str
    color: str

# ${r2d2}=    Build Immutable Robot  R2D2  Bleue
# Log  ${r2d2.color}
def build_immutable_robot(name: str, color: str):
    return ImmutableRobot(name, color)
```

List Comprehension

```
*** Settings ***
Library     resources/robot_helper.py


*** Test Cases ***
Teste List Comprehension
    ${c3po}=    Build Immutable Robot    C3PO    Jaune
    ${r2d2}=    Build Immutable Robot    R2D2    Bleue

    ${robots}=    Create List    ${c3po}    ${r2d2}

    # Utilisez $robots (pas ${robots}) dans les expressions Python
    ${robots_jaunes}=    Evaluate    [r for r in $robots if r.color == "Jaune"]
```

Lire la documentation

- Guide
- Keyword

Catenate

Arguments

`* items`

Documentation

Catenates the given items together and returns the resulted string.

By default, items are catenated with spaces, but if the first item contains the string `SEPARATOR=<sep>`, the separator `<sep>` is used instead. Items are converted into strings when necessary.

Examples:

${str1} =	Catenate	Hello	world	
${str2} =	Catenate	SEPARATOR=---	Hello	world
${str3} =	Catenate	SEPARATOR=	Hello	world

=>

```
${str1} = 'Hello world'
${str2} = 'Hello---world'
${str3} = 'Helloworld'
```

Formatteurs de code

- Robotidy OU **Robocop** : Robot 🤖
- Black : Python 🐍
- Pre Commit (avant chaque commit Git)
- Vérification sur l'intégration continue

Terminal interactif

- fzf (fuzzy finder)
- Pattern Visitor

Terminal interactif

Gérer l'asynchronisme

- Pas de `Sleep` (attente fixe fragile et qui ralentit les tests)
- Utiliser boucle d'attente `Wait Until Keyword Succeeds`
- Utiliser `[Timeout]` sur Keyword / Test
- Documentation Timeout
- Utiliser les bons outils (`Playwright` attend par design)

```
*** Test Cases ***
Attendre Creation Fichier
    Démarrer Batch
    Wait Until Keyword Succeeds    2x    1s    File Should Exist    path=output.txt
    ${fichier}    Get File    path=output.txt
    # ... Assertion sur le fichier

*** Keywords ***
Démarrer Batch
    Log    Start Batch
```

SUITE **12-Wait Until**

Full Name: 12-Wait Until
Source: /Users/remi/wk/robot-examples/tests/12-wait_until.robot
Start / End / Elapsed: 20250903 15:22:56.685 / 20250903 15:22:57.701 / 00:00:01.016
Status: 1 test total, 0 passed, 1 failed, 0 skipped

TEST **Attendre Creation Fichier**

Full Name: 12-Wait Until.Attendre Creation Fichier
Start / End / Elapsed: 20250903 15:22:56.694 / 20250903 15:22:57.700 / 00:00:01.006
Status: FAIL
Message: Keyword 'File Should Exist' failed after retrying 2 times. The last error was: No keyword with name 'File Should Exist' found.

KEYWORD **Démarrer Batch**

KEYWORD BuiltIn . **Wait Until Keyword Succeeds** 2x 1s File Should Exist path=output.txt

Documentation: Runs the specified keyword and retries if it fails.
Start / End / Elapsed: 20250903 15:22:56.695 / 20250903 15:22:57.700 / 00:00:01.005

KEYWORD **File Should Exist** path=output.txt

Start / End / Elapsed: 20250903 15:22:56.695 / 20250903 15:22:56.695 / 00:00:00.000
15:22:56.695 FAIL No keyword with name 'File Should Exist' found.

KEYWORD **File Should Exist** path=output.txt

Start / End / Elapsed: 20250903 15:22:57.698 / 20250903 15:22:57.698 / 00:00:00.000
15:22:57.699 FAIL No keyword with name 'File Should Exist' found.

15:22:57.699 FAIL Keyword 'File Should Exist' failed after retrying 2 times. The last error was: No keyword with name 'File Should Exist' found.

KEYWORD ${fichier} = **Get File** path=output.txt

Gérer les erreurs

- Tenter des retry avec `Wait Until Keyword Succeeds`
- Gérer les erreurs avec TRY/EXCEPT
- `Fail` un test si nécessaire
- `Log`

Rendre plus lisible le rapport HTML

Suite

- Chaque fichier robot est une suite
- Les dossiers contenant les fichiers robot sont des suites
- Stats par suite

Tag

- Les tests peuvent être taggués
- Options CLI `--include` / `--exclude` pour filtrer
- Stats par tag

Log

- Ne pas TROP logguer

Remove Keywords

```
# WUKS = Wait Until Keyword Succeeds
robot --removekeywords PASSED --removekeywords WUKS tests/17-remove_keywords.robot
```

Remove Keywords

SUITE **17-Remove Keywords**

Full Name: 17-Remove Keywords
Source: /Users/remi/wk/robot-examples/tests/17-remove_keywords.robot
Start / End / Elapsed: 20250904 14:23:46.452 / 20250904 14:23:47.466 / 00:00:01.014
Status: 2 tests total, 1 passed, 1 failed, 0 skipped

TEST **Test Ok**

Full Name: 17-Remove Keywords.Test Ok
Start / End / Elapsed: 20250904 14:23:46.461 / 20250904 14:23:46.462 / 00:00:00.001
Status: PASS

KEYWORD **All Is Ok**

Start / End / Elapsed: 20250904 14:23:46.461 / 20250904 14:23:46.461 / 00:00:00.000
Message: Content removed using the --remove-keywords option.

TEST **Remove Wait Until**

Full Name: 17-Remove Keywords.Remove Wait Until
Start / End / Elapsed: 20250904 14:23:46.461 / 20250904 14:23:47.466 / 00:00:01.005
Status: FAIL
Message: Keyword 'File Should Exist' failed after retrying 2 times. The last error was: File '/Users/remi/wk/robot-examples/output.txt' does not exist.

KEYWORD BuiltIn . **Wait Until Keyword Succeeds** 2x 1s File Should Exist path=output.txt

Documentation: Runs the specified keyword and retries if it fails.
Start / End / Elapsed: 20250904 14:23:46.462 / 20250904 14:23:47.466 / 00:00:01.004
Message: 1 failing item removed using the --remove-keywords option.

KEYWORD OperatingSystem . **File Should Exist** path=output.txt

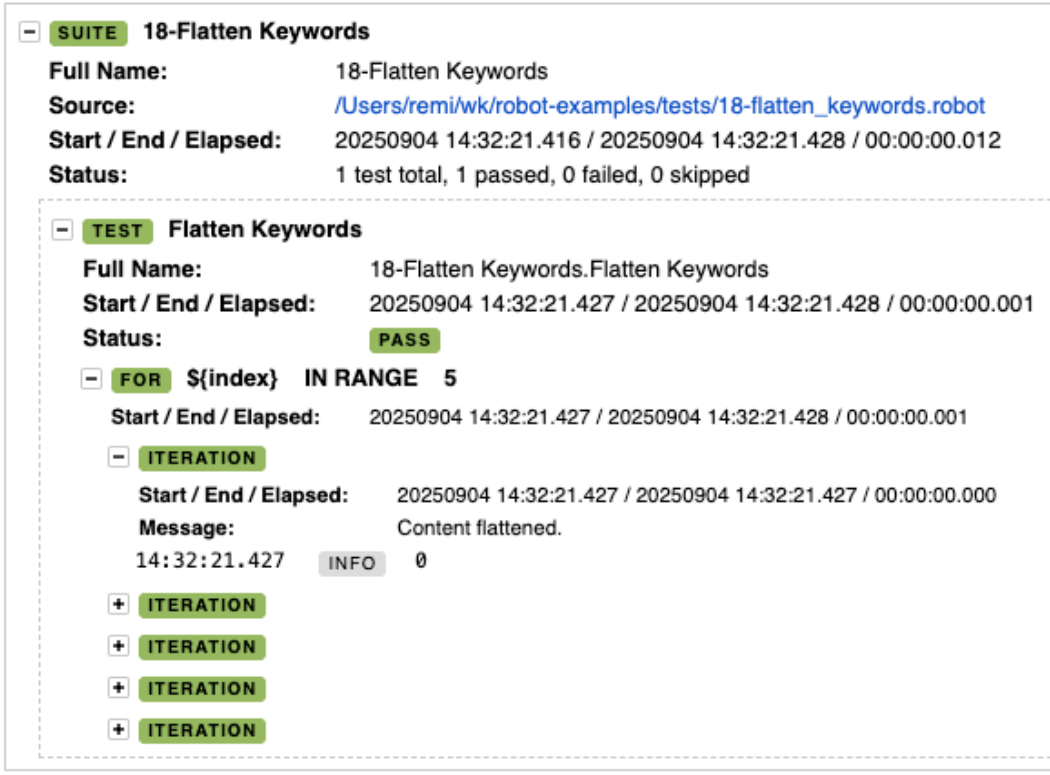
Documentation: Fails unless the given `path` points to an existing file.
Start / End / Elapsed: 20250904 14:23:47.464 / 20250904 14:23:47.465 / 00:00:00.001

```
14:23:47.465 FAIL File '/Users/remi/wk/robot-examples/output.txt' does not exist.
14:23:47.465 FAIL Keyword 'File Should Exist' failed after retrying 2 times. The last error was: File '/Users/remi/wk/robot-examples/output.txt' does not exist.
```

Flatten Keywords

```
1   robot --flattenkeywords ITERATION tests/18-flatten_keywords.robot
```

Flatten Keywords



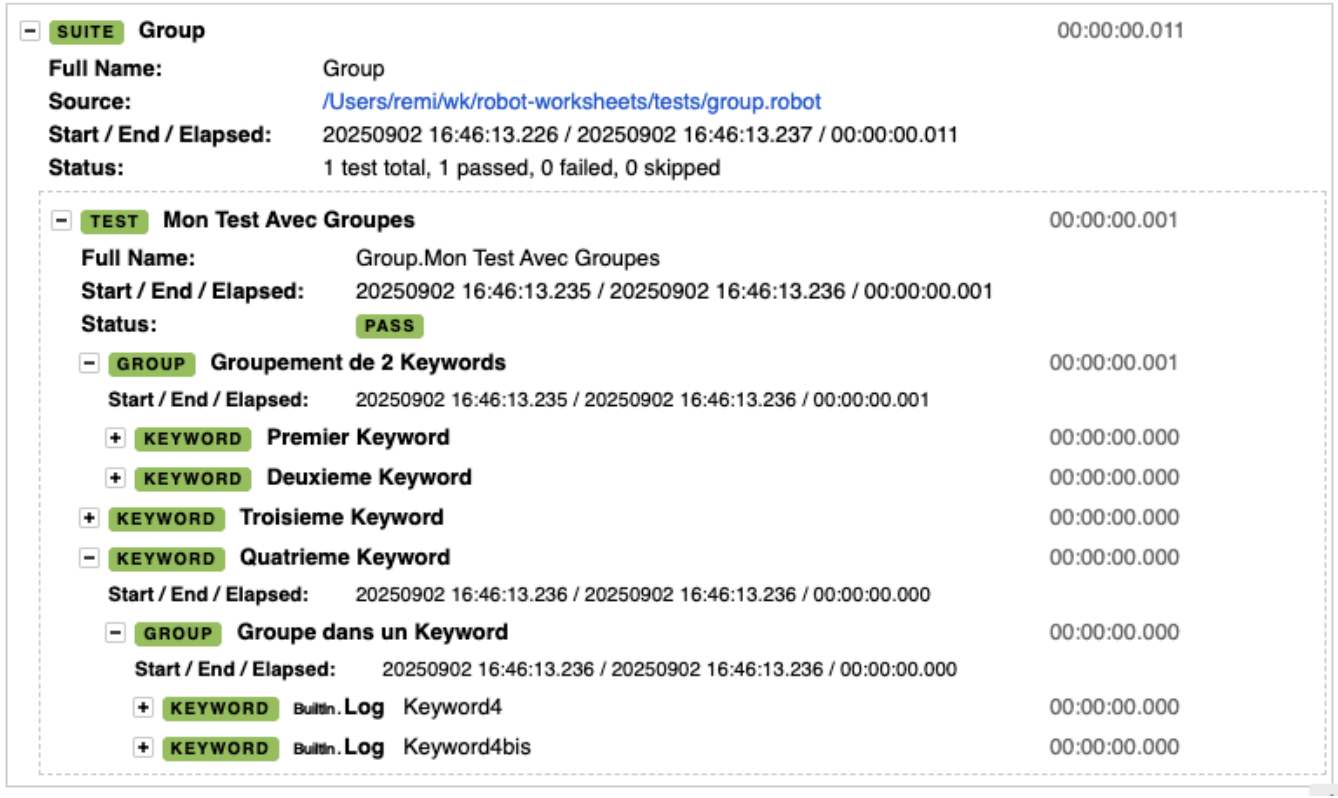
rebot

- Filtre les rapports de sortie

Structure GROUP

```
*** Test Cases ***
Mon Test Avec Groupes
    GROUP    Groupement de 2 Keywords
        Premier Keyword
        Deuxieme Keyword
    END
    Troisieme Keyword
    Quatrieme Keyword

*** Keywords ***
Premier Keyword
    Log    Keyword1

Deuxieme Keyword
    Log    Keyword2

Troisieme Keyword
    Log    Keyword3

Quatrieme Keyword
    GROUP    Groupe dans un Keyword
        Log    Keyword4
        Log    Keyword4bis
    END
```

GROUP

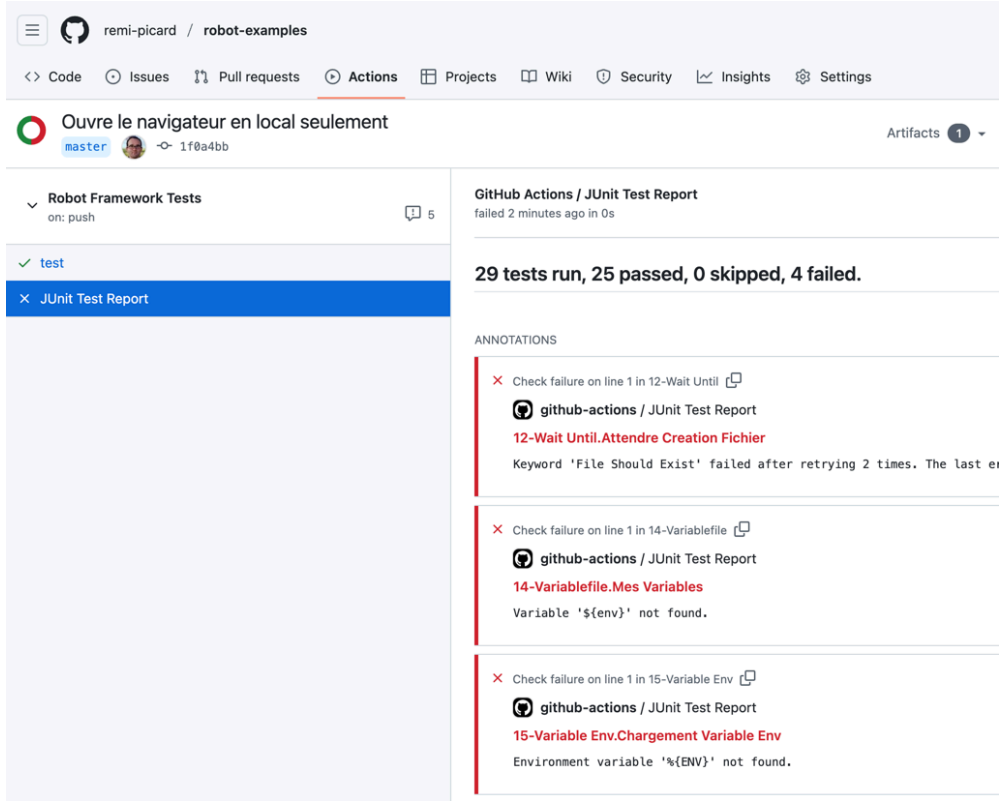

SUITE Group 00:00:00.011
Full Name: Group
Source: /Users/remi/wk/robot-worksheets/tests/group.robot
Start / End / Elapsed: 20250902 16:46:13.226 / 20250902 16:46:13.237 / 00:00:00.011
Status: 1 test total, 1 passed, 0 failed, 0 skipped
TEST Mon Test Avec Groupes 00:00:00.001
Full Name: Group.Mon Test Avec Groupes
Start / End / Elapsed: 20250902 16:46:13.235 / 20250902 16:46:13.236 / 00:00:00.001
Status: PASS
GROUP Groupement de 2 Keywords 00:00:00.001
Start / End / Elapsed: 20250902 16:46:13.235 / 20250902 16:46:13.236 / 00:00:00.001
KEYWORD Premier Keyword 00:00:00.000
KEYWORD Deuxieme Keyword 00:00:00.000
KEYWORD Troisieme Keyword 00:00:00.000
KEYWORD Quatrieme Keyword 00:00:00.000
Start / End / Elapsed: 20250902 16:46:13.236 / 20250902 16:46:13.236 / 00:00:00.000
GROUP Groupe dans un Keyword 00:00:00.000
Start / End / Elapsed: 20250902 16:46:13.236 / 20250902 16:46:13.236 / 00:00:00.000
KEYWORD BuiltIn.Log Keyword4 00:00:00.000
KEYWORD BuiltIn.Log Keyword4bis 00:00:00.000

Intégration continue 👾

Docker

- Utiliser ou Builder une image Docker
- Docker Images for Robot Framework

Github Actions



CI Systems / GitHub Actions

Intégration continue

- A chaque `push`
- Installer env python
- OU charger image Docker (évite de réinstaller les dépendances)
- Exécuter dry-run
- 💡 Ne pas lancer les tests

Tests quotidiens

- Chaque nuit
- Installer env python
- OU charger image Docker (évite de réinstaller les dépendances)
- Lancer les tests
- Publier les rapports HTML
- Effectuer un diff avec jours passés
- Notifier équipe

Tests quotidiens

Lancement via CI

- 🟢 Standard
- 🟠 Ouvertures de flux (https, bases, sftp…)
- 🟠 Publier les reports
- 🟠 https

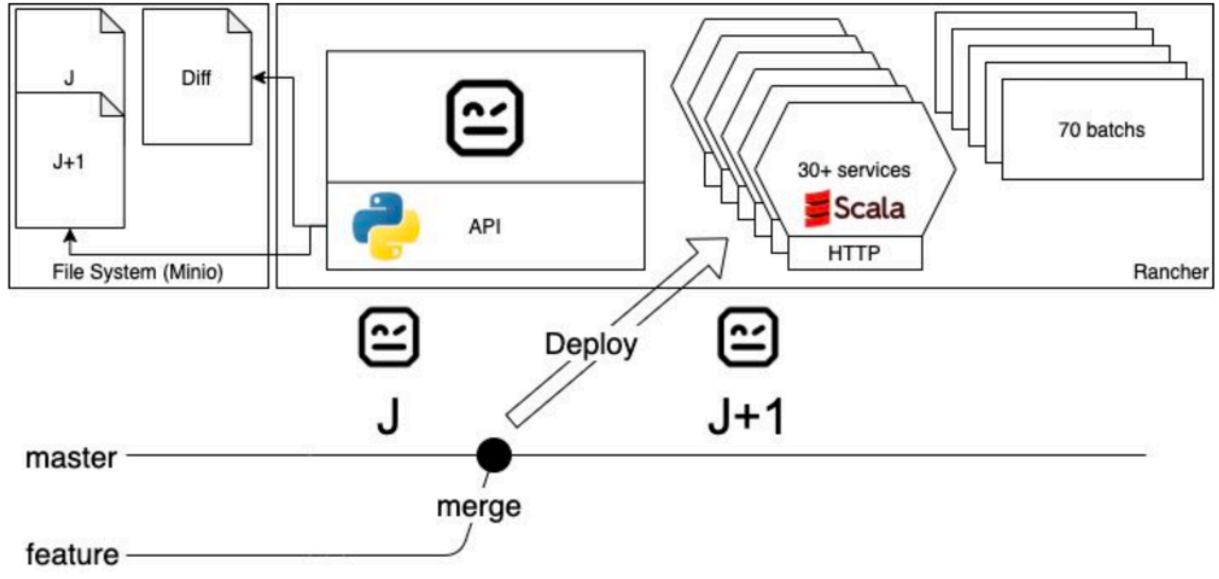
Tests quotidiens

Outils de Reporting

- Allure
- Report Portal
- Grafana
- Robot Framework Metrics
- Robot Framework Dashboard

Tests quotidiens

Lancement via K8S / API



Tests quotidiens

Lancement via K8S / API

- 🟢 File System Commun (input / output des apps, + report)
- 🟢 Lancement unitaire
- 🟢 http

Tests quotidiens

Robotdiff

Suite / Test	01/02	02/02	03/02
	2	1	2
	1 min	3 min	2 min
Account / Onboard A New Client	PASS	PASS	PASS
Error / Always Failed Test	FAIL	FAIL	FAIL
Error / Unstable Test	FAIL	PASS	FAIL
Transfer / Transfer Amount Cannot Be Negative	PASS	PASS	PASS
Transfer / Transfer Amount Cannot Be Zero	PASS	PASS	PASS
Transfer / Transfer Money Between Two Clients	PASS	PASS	PASS
Transfer / Transfer Money From Unprovisioned Account	PASS	PASS	PASS
Transfer / Transfer Money To Unknown Account	PASS	PASS	PASS

Robotdiff

Tests quotidiens

Notifications (GChat / Slack)

- Notifier **SI erreurs en +** par rapport à la veille

Autres Usages 

Usage Développeur

- Création de jeu données
- Tests semi-manuels
- Démo

Collaborer 

Toolkit

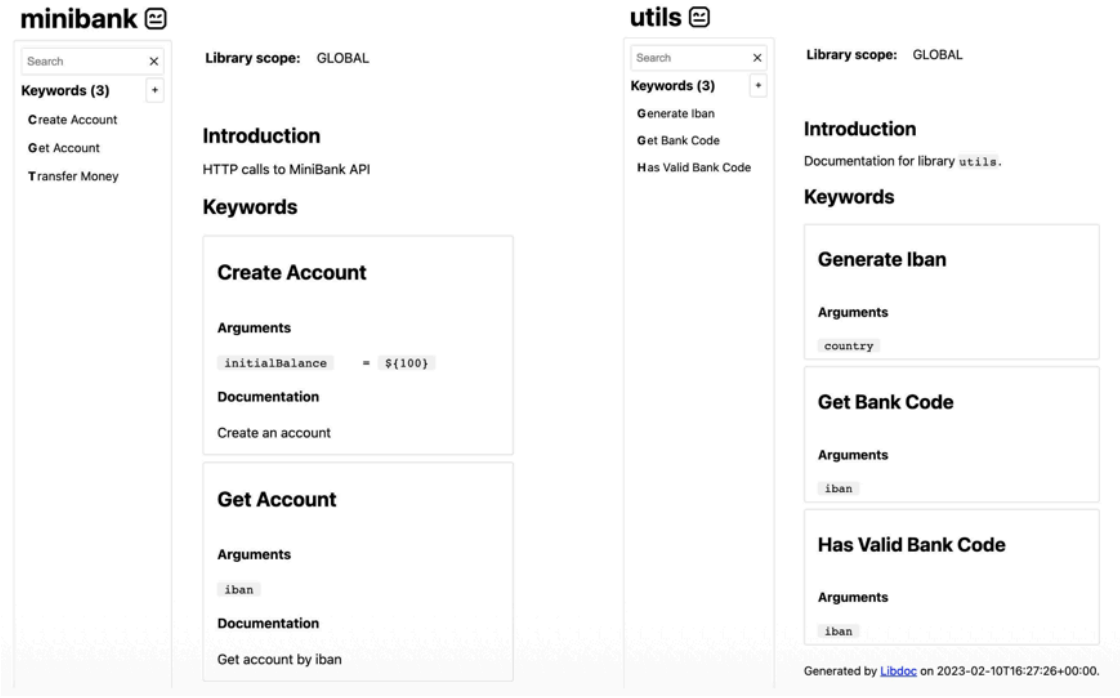
Partager une partie du code

- Keywords
- Helpers

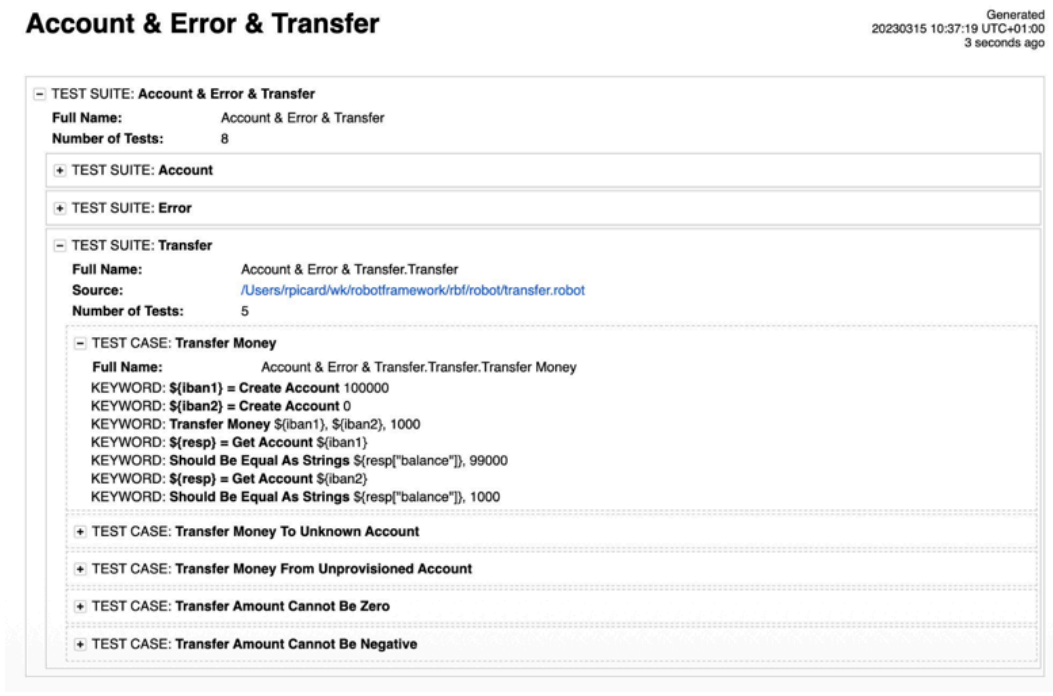
Portail Documentaire

- `libdoc` : documentation Keywords
- `libtoc` : Sidebar Keywords
- `testdoc` : documentation tests

libdoc + libtoc



testdoc



Account & Error & Transfer

Generated
20230315 10:37:19 UTC+01:00
3 seconds ago

TEST SUITE: **Account & Error & Transfer**

Full Name: Account & Error & Transfer
Number of Tests: 8

TEST SUITE: **Account**

TEST SUITE: **Error**

TEST SUITE: **Transfer**

Full Name: Account & Error & Transfer.Transfer
Source: /Users/rpicard/wk/robotframework/rbf/robot/transfer.robot
Number of Tests: 5

TEST CASE: **Transfer Money**

Full Name: Account & Error & Transfer.Transfer.Transfer Money

KEYWORD: **${iban1} = Create Account** 100000
KEYWORD: **${iban2} = Create Account** 0
KEYWORD: **Transfer Money** ${iban1}, ${iban2}, 1000
KEYWORD: **${resp} = Get Account** ${iban1}
KEYWORD: **Should Be Equal As Strings** ${resp["balance"]}, 99000
KEYWORD: **${resp} = Get Account** ${iban2}
KEYWORD: **Should Be Equal As Strings** ${resp["balance"]}, 1000

TEST CASE: **Transfer Money To Unknown Account**

TEST CASE: **Transfer Money From Unprovisioned Account**

TEST CASE: **Transfer Amount Cannot Be Zero**

TEST CASE: **Transfer Amount Cannot Be Negative**

Robotic Process Automation (RPA)

Robotic Process Automation (RPA) **is similar to test automation on the technical level**, but the mentality is different on the business and results side. In RPA, it is pretty standard that you are not running on a machine you control entirely, so your robot needs to be "self-sufficient" and isolated. Also, instead of finding and documenting places where robot execution fails or succeeds, **the aim is always to succeed and get the result of the process**.

- `*** Test Cases ***` => `*** Tasks ***`
- Librairies RPA (Desktop, Cloud, Scrapping…)

Codelab 💻

Jeu RPA


IA

IA

Avantages

- Apprendre
- Prototyper
- Résoudre problématique spécifique

IA

Avertissements

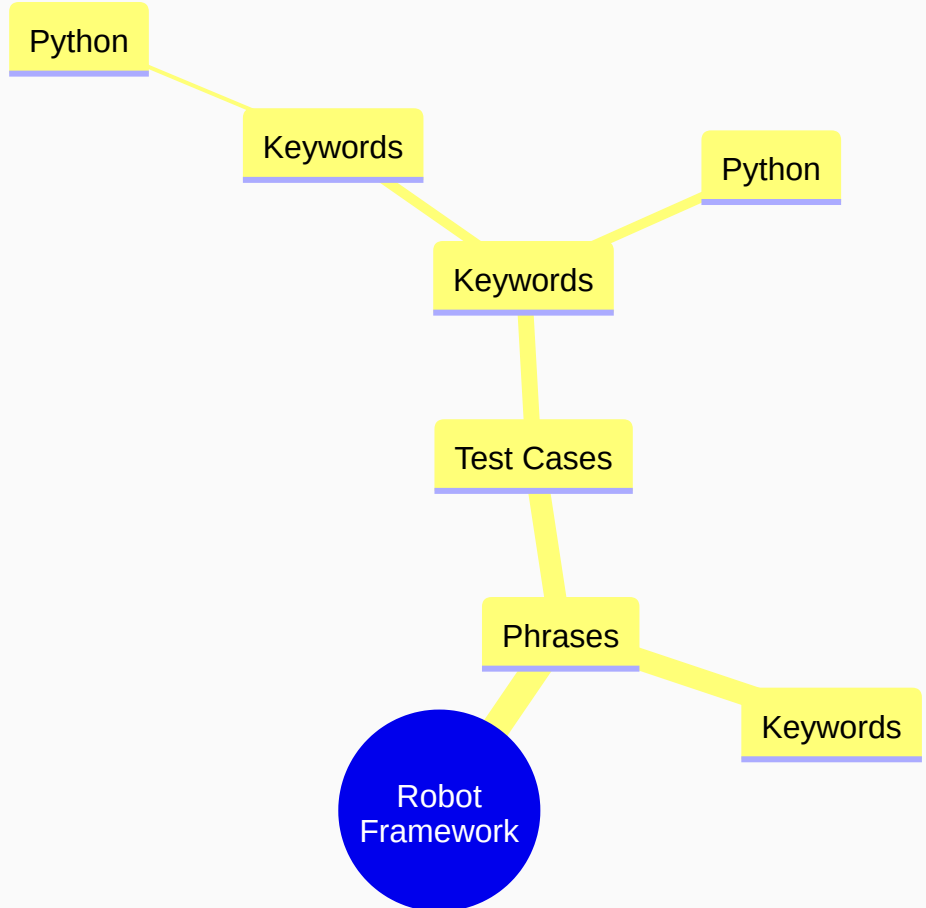
- Prendre du recul
- Perte de contrôle
- Empreinte écologique

Démo Génération de code

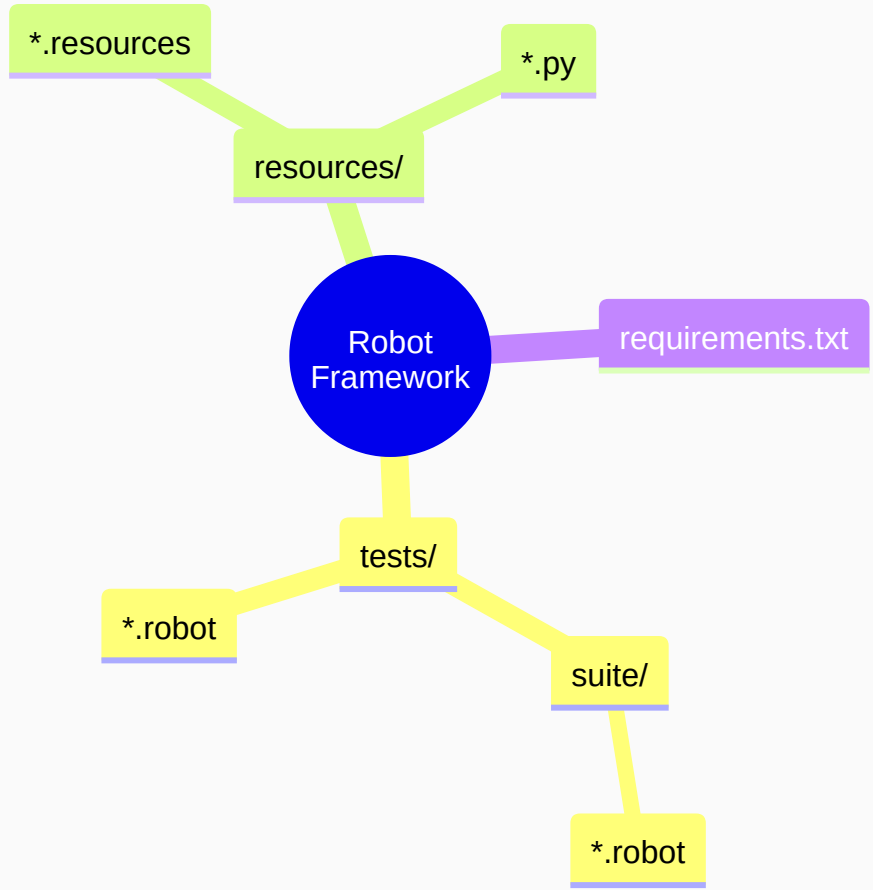
- Claude AI (Web)
- Gemini (CLI)

Récap 🤯

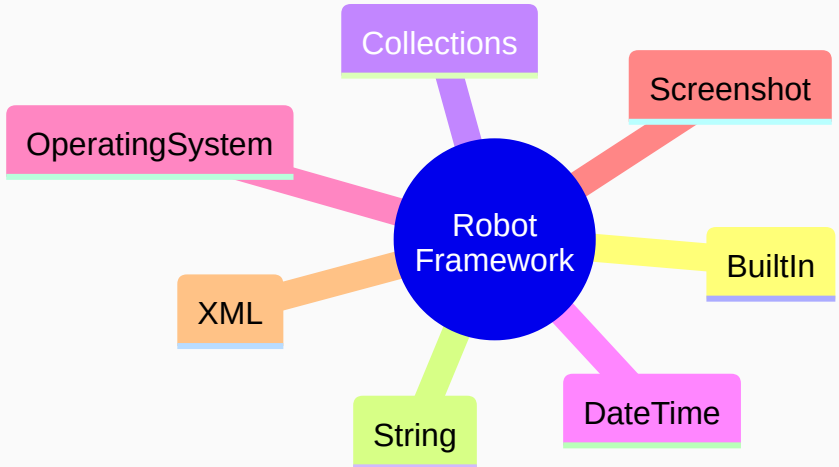
Keywords / Phrases



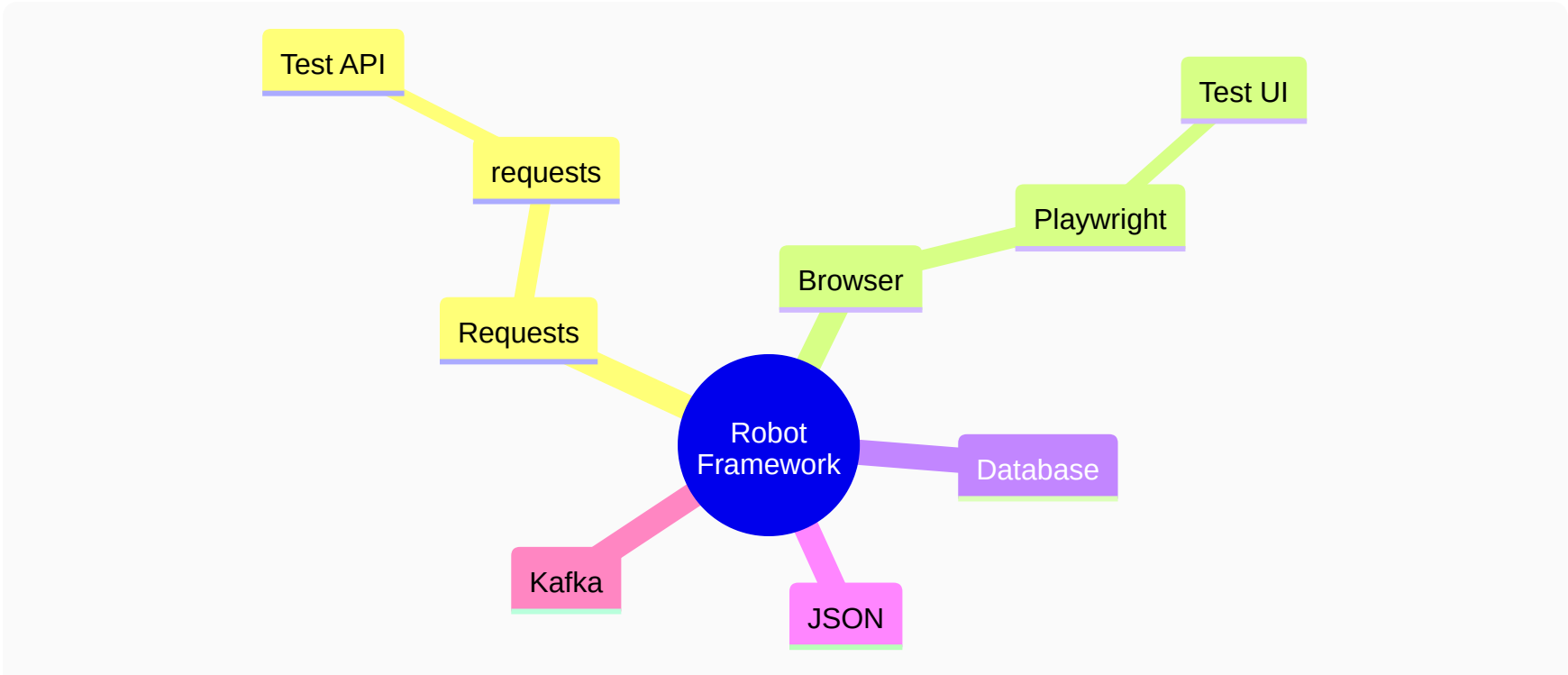
Structure projet



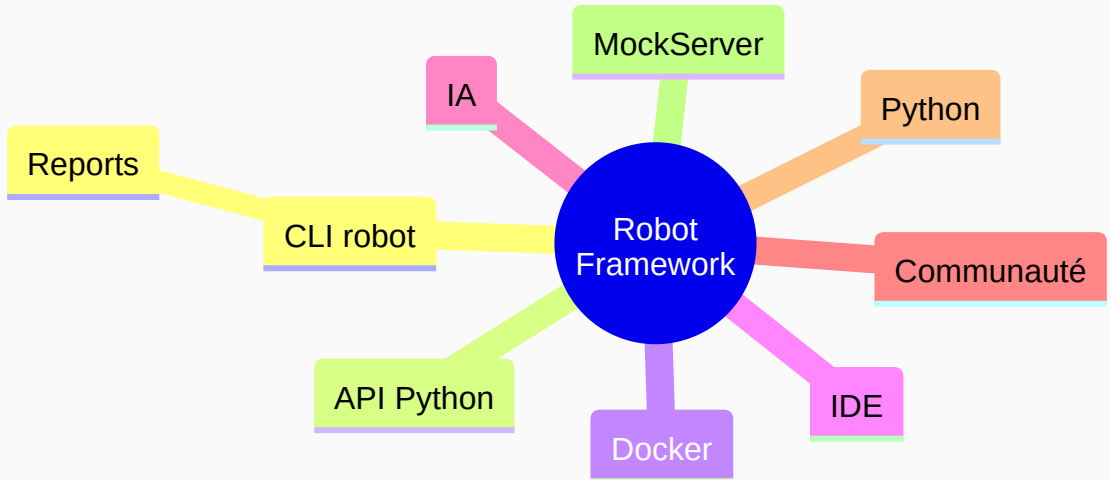
Librairies Core



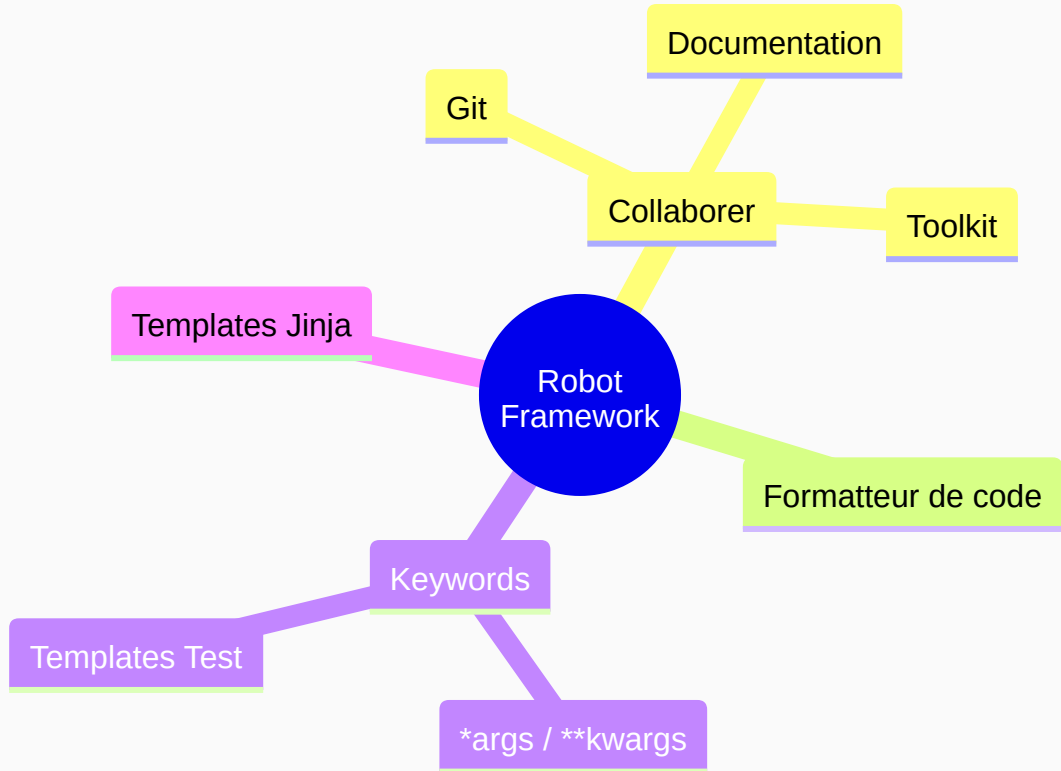
Librairies


Test API
requests
Requests
Test UI
Playwright
Browser
Robot Framework
Database
Kafka
JSON

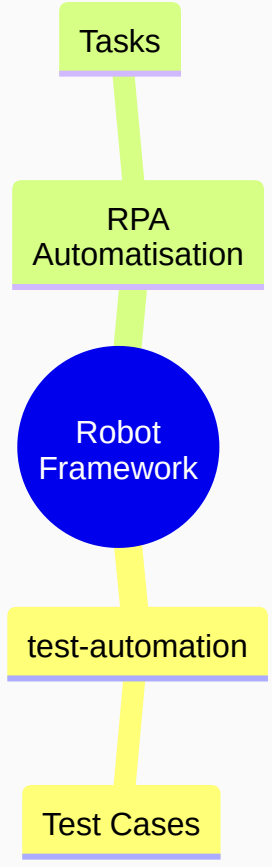
Ecosystème


MockServer
IA
Python
Reports
CLI robot
Robot Framework
Communauté
API Python
Docker
IDE

Bonnes pratiques



test-automation VS RPA



Rétro 

Questions & Réponses

- Merci pour votre attention. 🙏
- Bonne chance pour l’automatisation de vos tests !
- Amusez-vous bien !