



# Développer avec Robot Framework

22-23 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

# Découverte de Robot Framework

- Généralités
- Installer l'env de dev
- Rappels Python
- Comprendre le fonctionnement de Robot Framework
- Découvrir la ligne de commande robot
- Ecrire mes premiers tests (codelab)

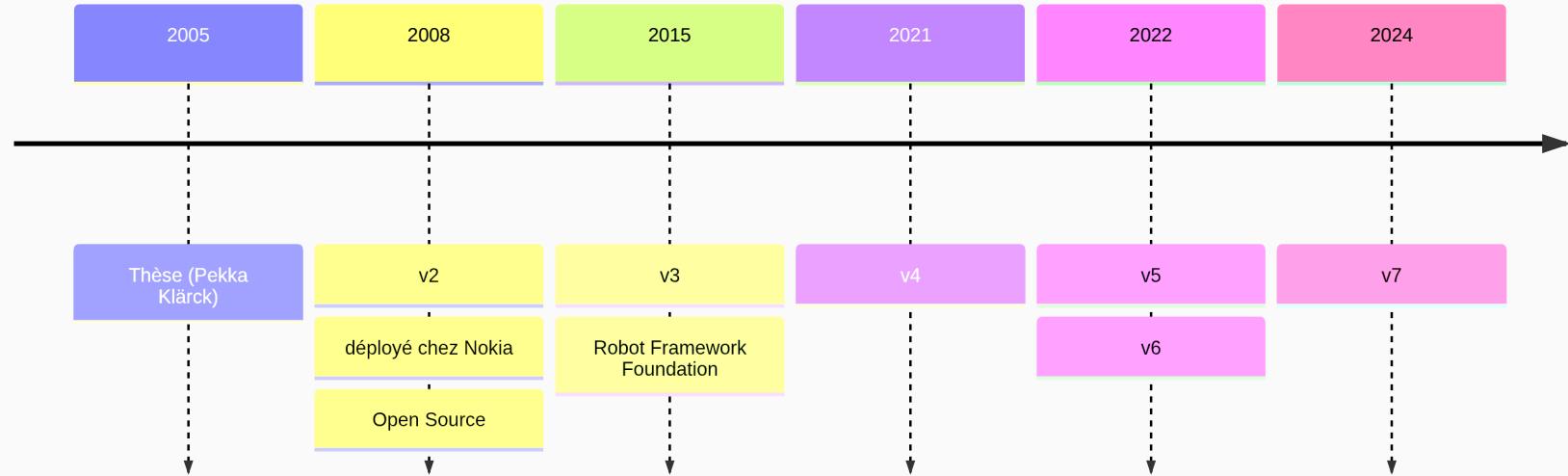
# Présentation

- 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

**Robot Framework => "Robot"**  
**RF / RBF / RBT**

# Histoire

## 20 ans déjà !

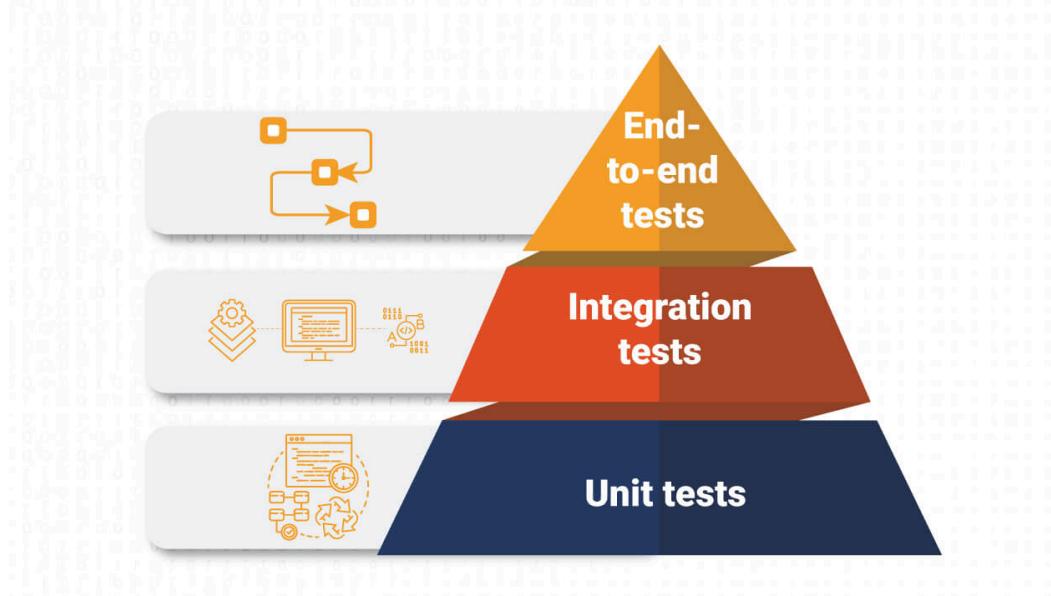


# Communauté

- Slack
- RoboCon / RBCN : conférence annuelle à Helsinki 
- [Documentation](#)

# Pyramide des tests

## Tests End-To-End

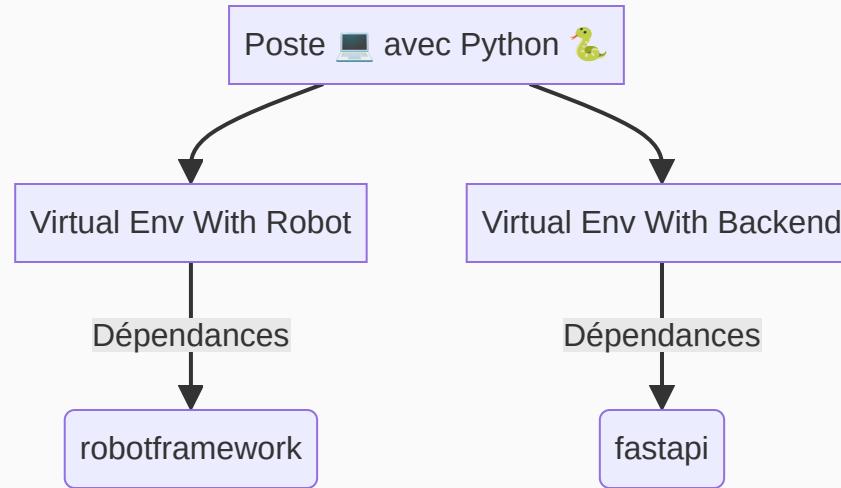


Source: <https://blog.takima.fr/saffranchir-de-la-pyramide-des-tests/>

**Env de dev** 

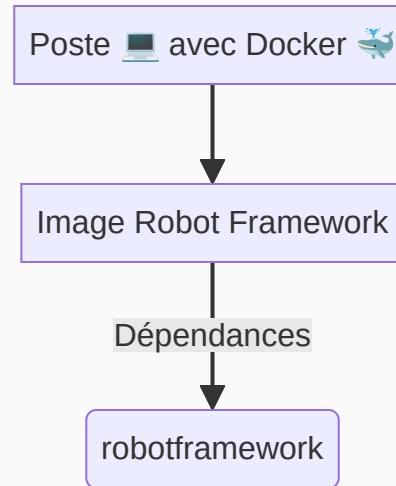
# Installation

## 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

```
1 def ma_fonction(arg1: int, arg2: str, default_arg="default", *args, **kwargs) -> int:
2     # ...
3     return 42
4
5 ma_fonction(42, "Quarante-deux")
6 ma_fonction(42, "Quarante-deux", "override default value")
7 ma_fonction(42, "Quarante-deux", "default", 1, 2, 3)
8 ma_fonction(42, "Quarante-deux", "default", robot="Nono", john="Doe")
9 ma_fonction(42, "Quarante-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

```
1 *** Settings ***
2 Library      String
3
4 *** Test Cases ***
5 Mon Premier Test
6     ${chaine}=  Generate Random String    10
7     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

## Déclaration

```
1 *** Test Cases ***
2 Creation Variable
3     ${ma_variable}    Set Variable    C3PO
4     Log    ma_variable=${ma_variable}
```

# Variables

## Import

```
1 # resources/mes_variables_python.py  
2  
3 variable_python = 42
```

```
1 *** Settings ***  
2 Variables      resources/mes_variables_python.py  
3  
4 *** Test Cases ***  
5 Utiliser Variable Python  
6     Log      variable_python=${variable_python}
```

# Variables

## Section Variables

```
1 *** Variables ***
2 ${nombre}      42
3 ${chaine}      Ma chaîne de caractères
4 @{tab}          1  2  3
5 &{map}          clef1=valeur1    clef2=valeur2
6
7 *** Test Cases ***
8 Teste Variables
9 Log    nombre=${nombre}
10 Log   chaine=${chaine}
11 Log   tab=${tab}
12 Log   map=${map}
```

# Injection Python

## Evaluate

```
1  *** Test Cases ***
2  Teste Evaluate
3      ${nb}=    Evaluate    41 + 1
4      Log      nb=${nb}
5
6  Teste Evaluate Autre Syntaxe
7      ${nb}=    Set Variable   ${${41 + 1}}
8      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

```
1 *** Keywords ***
2 Mon Premier Keyword
3     Log    Hello World
4
5 Mon Premier Keyword Avec Argument
6     [Arguments]    ${name}
7     Log    Hello ${name}
8
9 Mon Premier Keyword Avec Argument Et Return
10    [Arguments]   ${name}
11    Log    Hello ${name}
12    RETURN    42
```

# Keyword

## Arg dans Keyword

```
1  *** Keywords ***
2  Keyword Avec ${arg1} Intégré
3      Log    Hello ${arg1}
4
5  Keyword Avec ${arg1} Intégré Et Arguments
6      [Arguments]    ${name}
7      Log    Hello ${arg1}, ${name}
8
9  *** Test Cases ***
10 Appel Keywords
11     ${arg1}    Set Variable    Ma Variable
12     Keyword Avec ${arg1} Intégré
13     Keyword Avec ${arg1} Intégré Et Arguments    Arg2
```

# Keyword

## List (args) / Dict (kwargs)

```
1  *** Keywords ***
2  Keyword Avec Args
3      [Arguments]    @{list}
4      FOR    ${i}    IN    @{list}
5          Log    i=${i}
6      END
7
8  Keyword Avec Kwargs
9      [Arguments]    &{map}
10     FOR    ${k}    ${v}    IN    &{map}
11         Log    key=${k}, value=${v}
12     END
13
14 *** Test Cases ***
15 Appel Keywords
16     ${list}    Create List    1    2    3
17     Keyword Avec Args    ${list}
18
19     ${map}    Create Dictionary    cle1=valeur1    cle2=valeur2
20     Keyword Avec Kwargs    &{map}
```

# Keyword

## Gestion des espaces

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

*Variable de retour*

*Keyword*

*Arguments*

`${retour}= Mon Premier Keyword Terminator 2ème arg`



# Keyword

## Gestion des chaînes

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

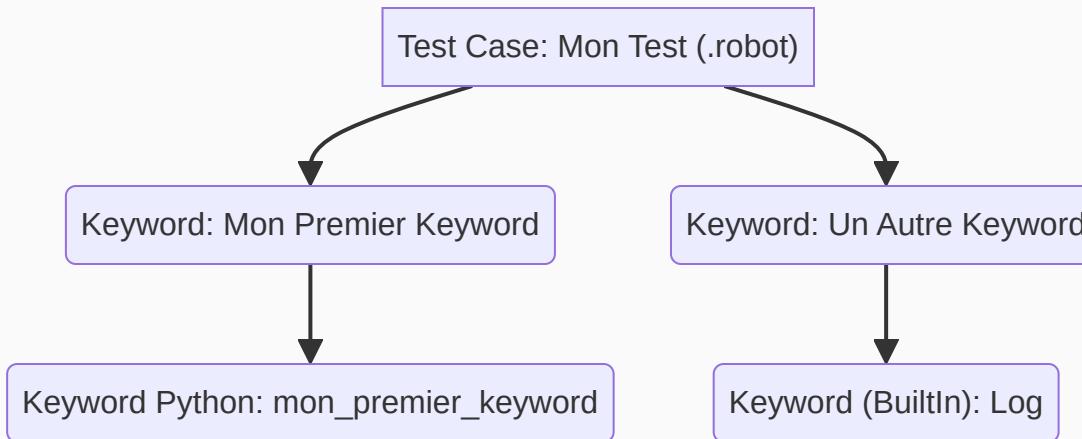
# Keyword Appel

```
1 *** Test Cases ***
2 Mon Premier Test
3     Mon Premier Keyword
4         Mon Premier Keyword Avec Argument      Nono le petit robot
5             ${retour}=    Mon Premier Keyword Avec Argument Et Return    Terminator
6
7 *** Keywords ***
8 Mon Deuxième Keyword
9     Mon Premier Keyword
```

# Keyword Syntaxe Python

```
1 # Mon Premier Keyword
2 def mon_premier_keyword():
3     print("Hello World")
4
5 # Mon Premier Keyword Avec Argument Et Return    ${name}
6 def mon_premier_keyword_avec_argument_et_return(name) -> int:
7     print(name)
8     return 42
```

# Test -> Keyword -> Keyword



# Section

- **\*\*\* 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

```
1  robot --help  
2  
3  # Lance tous les tests présents dans le dossier tests  
4  robot tests  
5  
6  # Lance le test "Mon Test"  
7  robot -t "Mon Test" tests
```

# Résultats

- `report.html` : synthèse
- `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

- <b>SUITE</b> <b>Jinja</b>	00:00:00.021
<b>Full Name:</b> Jinja	
<b>Source:</b> /Users/remi/wk/robot-worksheets/tests/file/jinja.robot	
<b>Start / End / Elapsed:</b> 20250902 11:17:57.075 / 20250902 11:17:57.096 / 00:00:00.021	
<b>Status:</b> 1 test total, 0 passed, 1 failed, 0 skipped	
- <b>TEST</b> <b>Creer Fichier Avec Un Template</b>	00:00:00.001
<b>Full Name:</b> Jinja.Creer Fichier Avec Un Template	
<b>Start / End / Elapsed:</b> 20250902 11:17:57.094 / 20250902 11:17:57.095 / 00:00:00.001	
<b>Status:</b> <b>FAIL</b>	
<b>Message:</b> TypeError: write() argument must be str, not list	
+ <b>KEYWORD</b> \${ligne1} = <b>Builtin.Create Dictionary</b> variable=Ligne 1	00:00:00.000
+ <b>KEYWORD</b> \${ligne2} = <b>Builtin.Create Dictionary</b> variable=Ligne 2	00:00:00.000
+ <b>KEYWORD</b> \${data} = <b>Builtin.Create List</b> \${ligne1} \${ligne2}	00:00:00.000
- <b>KEYWORD</b> <b>Creer Fichier</b> data/mon_fichier.csv \${data}	00:00:00.001
<b>Start / End / Elapsed:</b> 20250902 11:17:57.095 / 20250902 11:17:57.096 / 00:00:00.001	
+ <b>KEYWORD</b> \${content} = <b>file_helper.Charger Template</b> \${data}	00:00:00.000
- <b>KEYWORD</b> \${fichier} = <b>OperatingSystem.Create File</b> \${path} \${data}	00:00:00.000
<b>Documentation:</b> Creates a file with the given content and encoding.	
<b>Start / End / Elapsed:</b> 20250902 11:17:57.095 / 20250902 11:17:57.095 / 00:00:00.000	
11:17:57.095 <b>FAIL</b> TypeError: write() argument must be str, not list	
+ <b>RETURN</b> \${fichier}	00:00:00.000

Quizz

TODO Klaxoon ?



Tester les syntaxes

- Keyword
- Test
- Variable

# Requests Library

- Tests d'API
- Wrapper de la lib Python requests
- [Documentation](#)
- Manipulation JSON simple

# Requests Library

```
1 pip install robotframework-requests
```

# Requests Library

```
1 *** Settings ***
2 Library      RequestsLibrary
3
4 *** Test Cases ***
5 Quick Get Request Test
6     ${response}    GET    https://www.google.com
7
8 Quick Get Request With Parameters Test
9     ${response}    GET    https://www.google.com/search    params=query=ciao    expected_status=200
10
11 Quick Get A JSON Body Test
12     ${response}    GET    https://jsonplaceholder.typicode.com/posts/1
13     Should Be Equal As Strings    1    ${response.json()[id]}
14
15 Create Booking
16     ${booking_dates}    Create Dictionary    checkin=2022-12-31    checkout=2023-01-01
17     ${body}    Create Dictionary
18     ...    firstname=Hans
19     ...    lastname=Gruber
20     ...    totalprice=200
21     ...    depositpaid=false
22     ...    bookingdates=${booking_dates}
23     ${response}    POST    url=https://restful-booker.herokuapp.com/booking    json=${body}
24     ${id}    Set Variable    ${response.json()[bookingid]}
```



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

```
1 # Installer Node https://nodejs.org/en/download/
2 pip install robotframework-browser
3 rfbrowser init
```

# Browser Library

```
1 *** Settings ***
2 Library      Browser
3
4 *** Test Cases ***
5 Go To Playwright With Browser Library
6     New Page    https://playwright.dev/
7     Get Title   contains   Playwright
8     Take Screenshot
9
10    Click      a >> "Get started"
11    Get Element States  h1 >> "Installation"  contains  visible
12    Take Screenshot
```





TODO MVC

**Jour 2**

**Bonnes pratiques et Industrialisation**

# Bonnes pratiques Robot Framework

- 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`)

```
1 mon-projet/
2   └── tests/
3     ├── cas_de_test.robot
4     └── suite/
5       └── autre_cas_de_test.robot
6   └── resources/
7     └── keyword_commons.resource
8     └── ma_librarie.py
```

## RobotFramework - Project Structure

# Syntaxes avancées



# Test Template

```
1 *** Test Cases ***
2 Normal test case with embedded arguments
3     The result of 1 + 1 should be 2
4     The result of 1 + 2 should be 3
5
6 Template with embedded arguments
7 [Template]    The result of ${calculation} should be ${expected}
8     1 + 1    2
9     1 + 2    3
10
11 *** Keywords ***
12 The result of ${calculation} should be ${expected}
13     ${result} =    Evaluate    ${calculation}
14     Should Be Equal As Strings    ${result}    ${expected}
```

- Test Templates

# Behavior Driven Development (BDD)

## Given-When-Then

```
1 *** Test Cases ***
2 Mon Test Bdd
3     Given Pré Requis
4     When Action
5     Then Verifications
6
7 *** Keywords ***
8 Pré Requis
9     Log    Pré Requis
10
11 Action
12     Log    Action
13
14 Verifications
15     Log    Verifications
```

BDD

## Test Execution Log

-	<b>SUITE</b>	09-Bdd
	<b>Full Name:</b>	09-Bdd
	<b>Source:</b>	/Users/remi/wk/robot-examples/tests/09-bdd.robot
	<b>Start / End / Elapsed:</b>	20250903 14:07:37.963 / 20250903 14:07:37.973 / 00:00:00.010
	<b>Status:</b>	1 test total, 1 passed, 0 failed, 0 skipped
-	<b>TEST</b>	Mon Test Bdd
	<b>Full Name:</b>	09-Bdd.Mon Test Bdd
	<b>Start / End / Elapsed:</b>	20250903 14:07:37.973 / 20250903 14:07:37.974 / 00:00:00.001
	<b>Status:</b>	PASS
-	<b>KEYWORD</b>	Given Pré Requis
	<b>Start / End / Elapsed:</b>	20250903 14:07:37.973 / 20250903 14:07:37.973 / 00:00:00.000
+	<b>KEYWORD</b>	Builtin.Log Pré Requis
-	<b>KEYWORD</b>	When Action
	<b>Start / End / Elapsed:</b>	20250903 14:07:37.973 / 20250903 14:07:37.973 / 00:00:00.000
+	<b>KEYWORD</b>	Builtin.Log Action
-	<b>KEYWORD</b>	Then Verifications
	<b>Start / End / Elapsed:</b>	20250903 14:07:37.973 / 20250903 14:07:37.973 / 00:00:00.000
+	<b>KEYWORD</b>	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

```
1 *** Settings ***
2 Library Collections
3 Library OperatingSystem
4 Library Process
5 Library String
```

# BuiltIn

```
1 *** Test Cases ***
2 Some Assertions
3     ${var}=    Set Variable    1.0
4     Should Not Be Equal    1    ${var}
5     Should Be Equal As Numbers    1    ${var}
6     Should Be Equal As Strings    1.0    ${var}
7     Should Be True    ${var} == 1.0
8     Should Not Be True    ${var} > 10
9
10    ${string}=   Set Variable   My String
11    Should Start With    ${string}    My
12    Length Should Be    ${string}    9
```

# Gestion des fichiers



# Fichier Template

- [Templates Ninja2](#)
- Création contenu de fichiers
- Couplé à `OperatingSystem`

# Fichier Template

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

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

- Utilisation dans Keyword Python

```
1  from jinja2 import Environment, FileSystemLoader
2
3  # Initialiser l'environnement avec un dossier de templates
4  env = Environment(loader=FileSystemLoader("templates"))
5  template = env.get_template("mon_template.csv.j2")
6
7
8  def charger_template(data):
9      contenu = template.render(data=data)
10     return contenu
```

# Fichier Template

- Utilisation dans Robot Framework

```
1  *** Settings ***
2  Library      OperatingSystem
3  Library      resources/file_helper.py
4
5  *** Test Cases ***
6  Creer Fichier Avec Un Template
7      ${ligne1}    Create Dictionary    variable=Ligne 1
8      ${ligne2}    Create Dictionary    variable=Ligne 2
9      ${data}      Create List        ${ligne1}    ${ligne2}
10     Creer Fichier   data/output/mon_fichier.csv   ${data}
11
12  *** Keywords ***
13  Creer Fichier
14      [Arguments]  ${path}    ${data}
15      ${contenu}   Charger Template   ${data}
16      ${fichier}   Create File     ${path}    ${contenu}
17      ${contenu}   Get File       ${path}
18      Log         ${contenu}
```

# Fichier Template

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

Codelab  Fichiers

## Autres Librairies / Outils

# Autres Librairies

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

## DataDriver

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

# PaBot

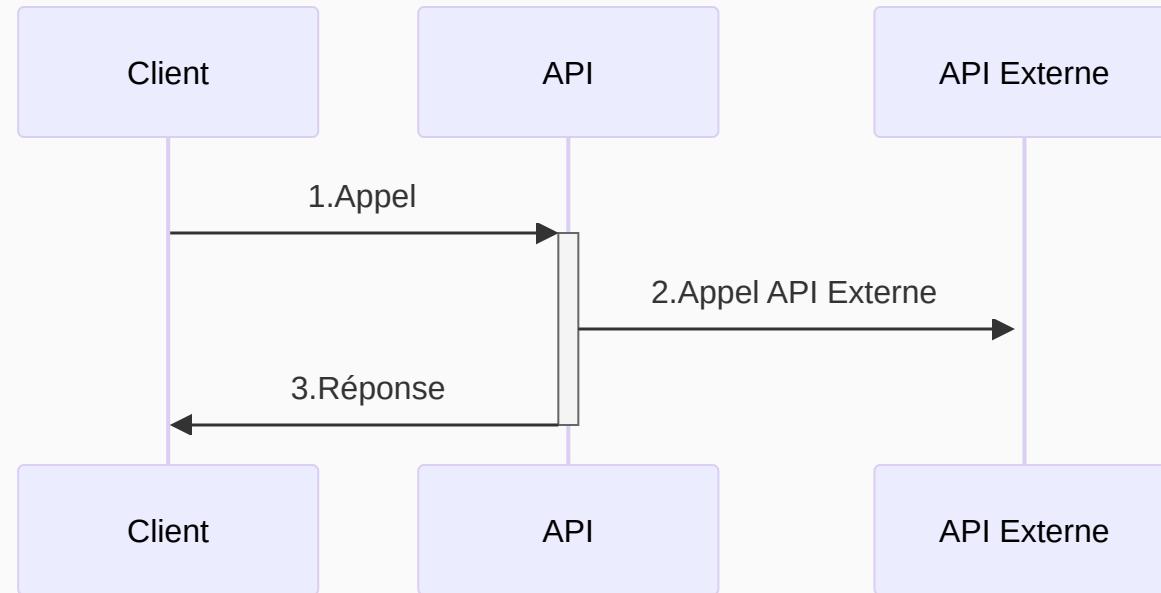
- Lancement des tests en parallèle 
-  Isolation des tests
- CLI `pabot` wrapper de `robot` (même options)
- `pip install robotframework-pabot`
- [PaBot](#)

## 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

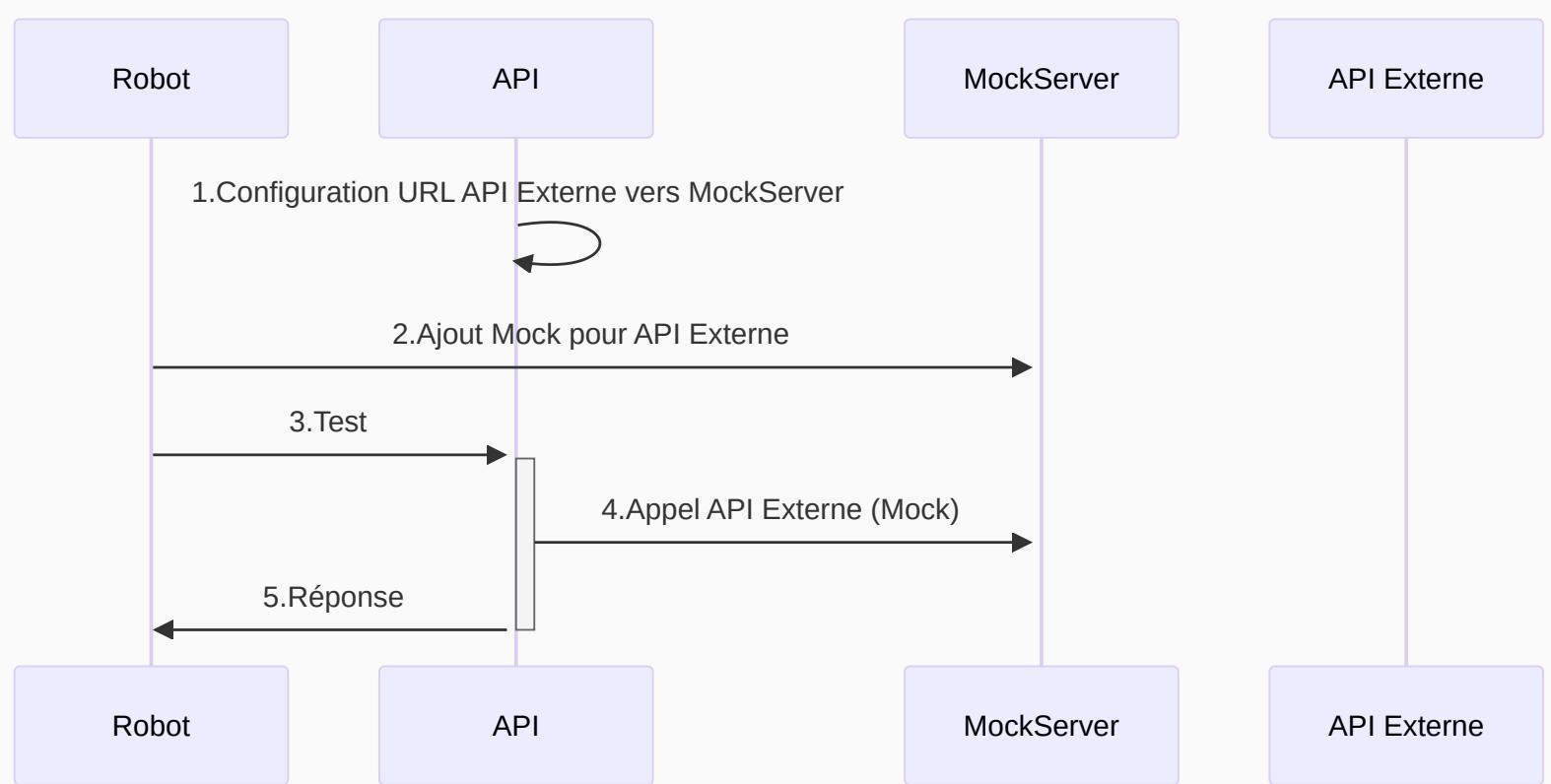
# MockServer

## API dépend de API Externe



# MockServer

## Flow de TEST



# Ligne de commande



# Ligne de commande robot

```
1 # Vérifie la syntaxe de tous les tests
2 robot --dryrun tests
3
4 # Ajoute le répertoire courant dans le PYTHON PATH
5 robot --pythonpath . tests
6
7 robot --loglevel TRACE . tests
```

# PYTHON\_PATH

Sans --pythonpath :

```
1 *** Settings ***
2 Resource    ./resources/ma_lib.resource
3
4 *** Test Cases ***
5 Tester Ma Lib
6     Mon Keyword
```

Avec --pythonpath . ou -P . :

```
1 *** Settings ***
2 Resource    resources/ma_lib.resource
3
4 *** Test Cases ***
5 Tester Ma Lib
6     Mon Keyword
```

# API Python

- réflexion Tests / Keywords (métadonnées, hooks)
- parsing des résultats en Python

# Listener

- `--listener`
- API Python 

Bonnes pratiques



# Composition de Keywords

■

## 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

# Variable d'env

- configuration YAML
- `.env` avec `dotenv`

# Lire la documentation

- Keyword

# Formatteurs de code

- Robotidy : Robot
- Black : Python
- Pre Commit (avant chaque commit Git)

# Terminal interactif

- alias
- fzf
- Listener

# Gérer l'asynchronisme

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

```
1  *** Test Cases ***
2  Attendre Creation Fichier
3      Démarrer Batch
4          Wait Until Keyword Succeeds    2x    1s    File Should Exist    path=output.txt
5          ${fichier}    Get File    path=output.txt
6          # ... Assertion sur le fichier
7
8  *** Keywords ***
9  Démarrer Batch
10     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
- Le dossier exécuté est une suite
- Tous les dossiers enfants sont des suites
- Stats par suite

## Tag

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

# Log

- Ne pas TROP logger

# loglevel

- `--loglevel TRACE` est intéressant en local

## rebot

- Filtre les rapports `output.xml`
- Avec arguments `--flattenkeywords`, `--removekeywords`

# Structure GROUP

```
1 *** Test Cases ***
2 Mon Test Avec Groupes
3     GROUP      Groupement de 2 Keywords
4         Premier Keyword
5         Deuxieme Keyword
6     END
7     Troisieme Keyword
8     Quatrieme Keyword
9
10    *** Keywords ***
11    Premier Keyword
12        Log      Keyword1
13
14    Deuxieme Keyword
15        Log      Keyword2
16
17    Troisieme Keyword
18        Log      Keyword3
19
20    Quatrieme Keyword
21        GROUP      Groupe dans un Keyword
22            Log      Keyword4
23            Log      Keyword4bis
24        END
```

# GROUP

- <b>SUITE</b> <b>Group</b>	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
- <b>TEST</b> <b>Mon Test Avec Groupes</b>	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:	<b>PASS</b>
- <b>GROUP</b> <b>Groupement de 2 Keywords</b>	00:00:00.001
Start / End / Elapsed:	20250902 16:46:13.235 / 20250902 16:46:13.236 / 00:00:00.001
+ <b>KEYWORD</b> <b>Premier Keyword</b>	00:00:00.000
+ <b>KEYWORD</b> <b>Deuxieme Keyword</b>	00:00:00.000
+ <b>KEYWORD</b> <b>Troisieme Keyword</b>	00:00:00.000
- <b>KEYWORD</b> <b>Quatrieme Keyword</b>	00:00:00.000
Start / End / Elapsed:	20250902 16:46:13.236 / 20250902 16:46:13.236 / 00:00:00.000
- <b>GROUP</b> <b>Groupe dans un Keyword</b>	00:00:00.000
Start / End / Elapsed:	20250902 16:46:13.236 / 20250902 16:46:13.236 / 00:00:00.000
+ <b>KEYWORD</b> <b>Builtin.Log Keyword4</b>	00:00:00.000
+ <b>KEYWORD</b> <b>Builtin.Log Keyword4bis</b>	00:00:00.000

Intégration continue 🐛

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

# Notifications

## GChat / Slack

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



**cobalt-robot-news** APPLI 8 h 30  
10 tests en erreur en + depuis hier !  
23 erreurs sur 99 au total  
[Robot Diff](#) | [Last Report](#)



3 réponses Dernière réponse il y a 3 jours



RBF Application Hier 08:30  
131 tests en erreur en + depuis hier !  
153 erreurs sur 175 au total  
≡ [Robot Diff](#)  
≡ [Last Report](#)  
 1 2

# Intégration continue

- A chaque push
- Ne pas lancer les tests
- Exécuter dry-run

# Tests quotidiens

- Chaque nuit
- Publier les rapports HTML
- Effectuer un diff avec jours passés
- Notifier équipe

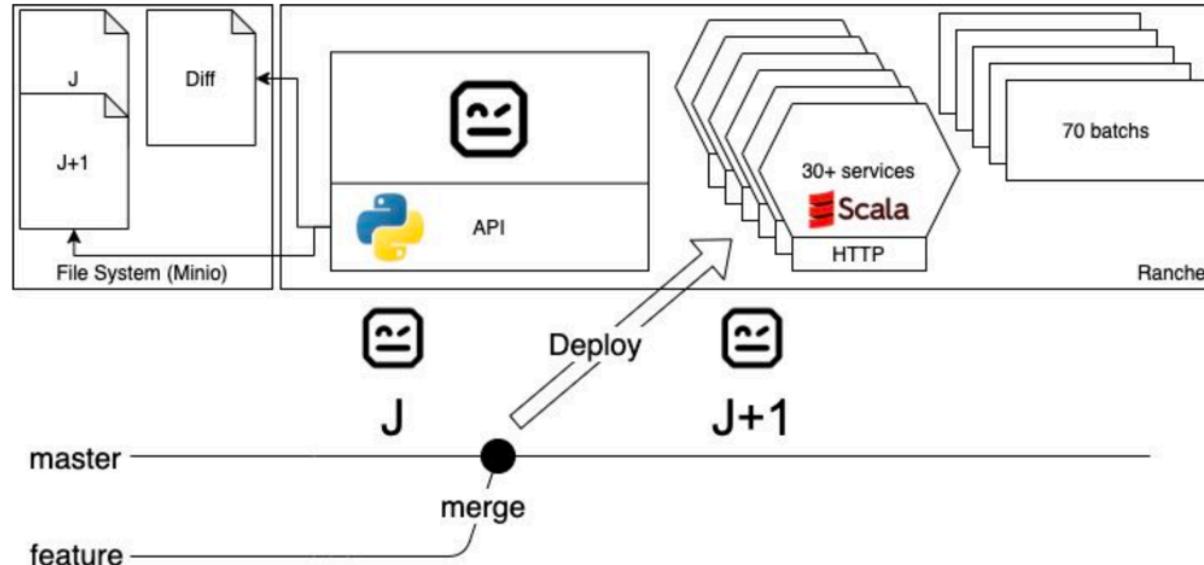
# Lancement Tests

## CI

-  Standard
-  Ouvertures de flux (https, bases, sftp...)
-  Publier les reports
-  https

# Lancement Tests

## K8S / API



# Lancement Tests

## K8S / API

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

# Autres Usages



# 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...)

# Usage Développeur

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

Collaborer



# Toolkit

## Partager une partie du code

- Keywords
- Helpers

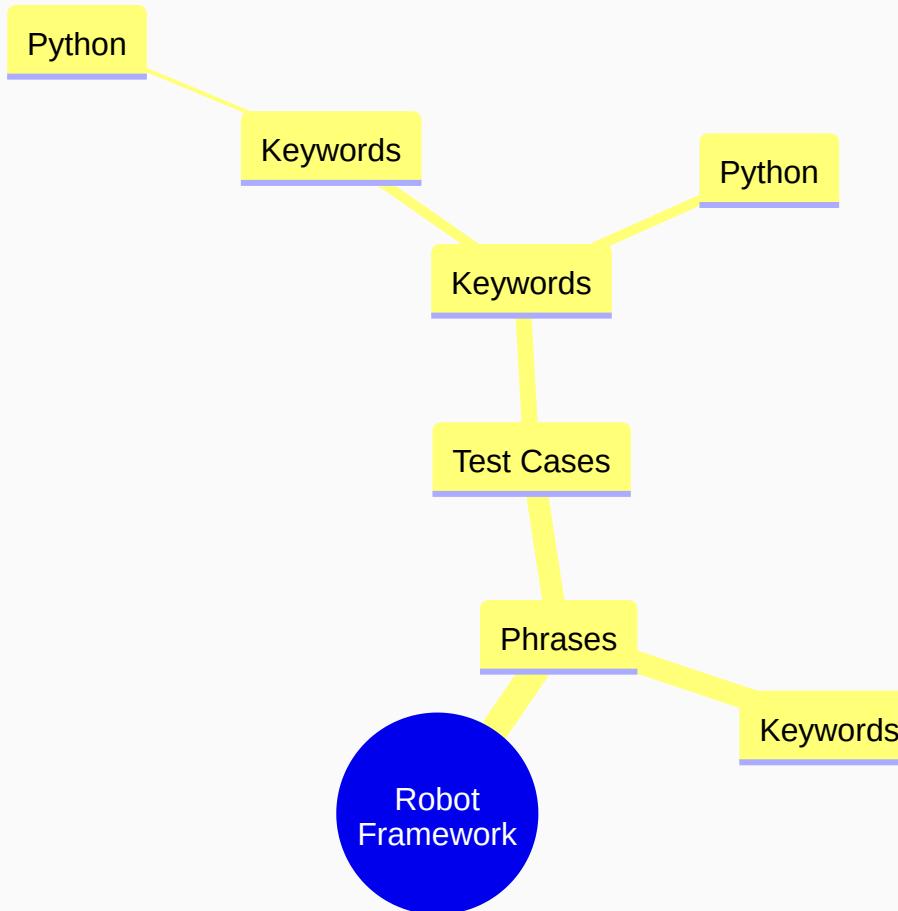
# Portail Documentaire

- libtoc
- libdoc

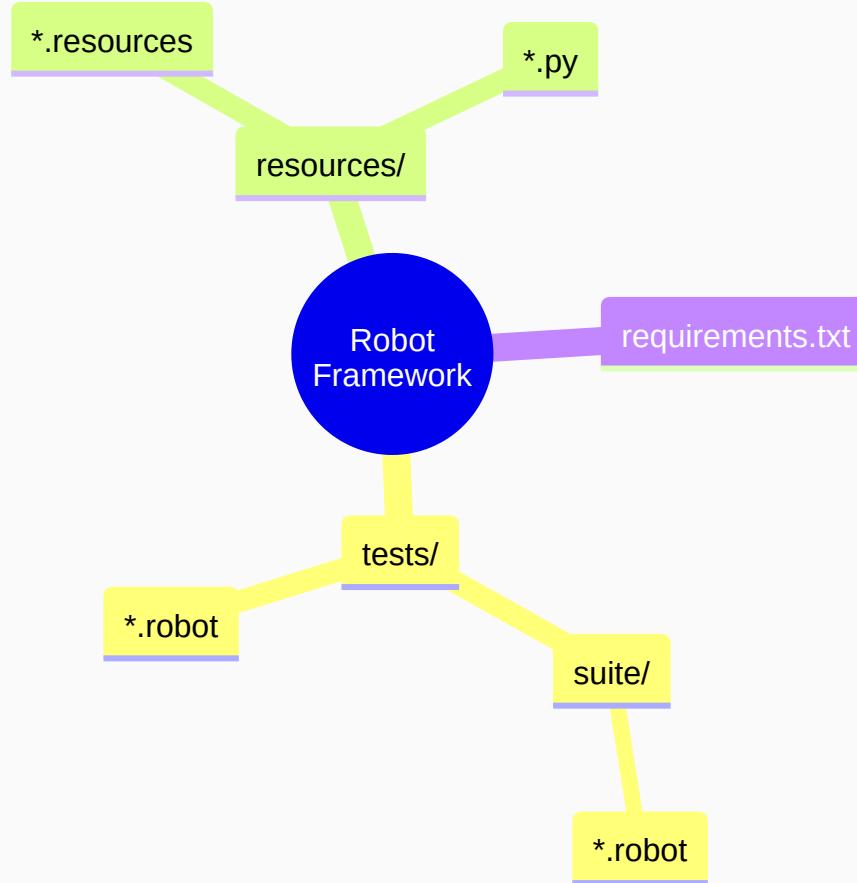
Récap



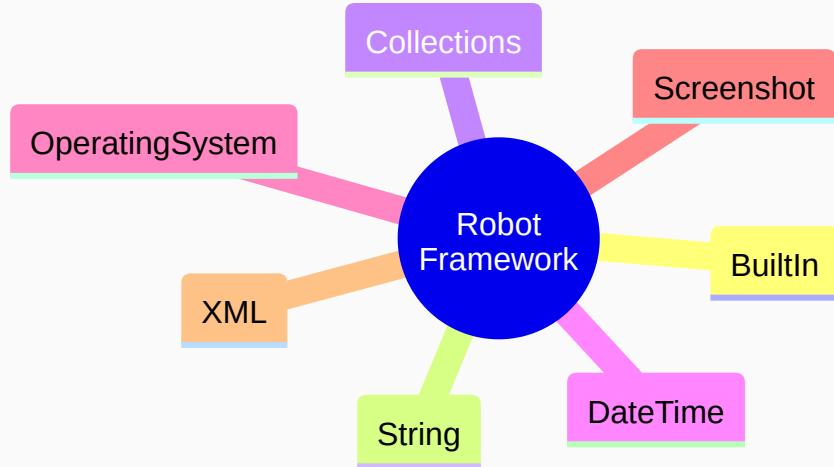
# Keywords / Phrases



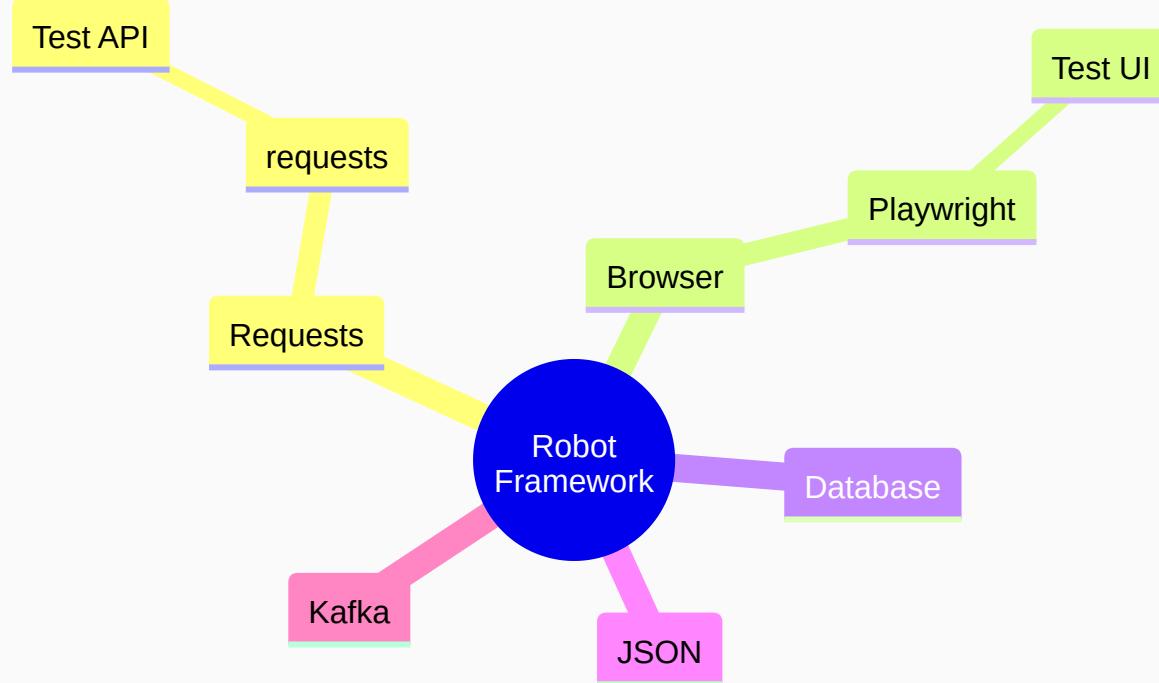
# Structure projet



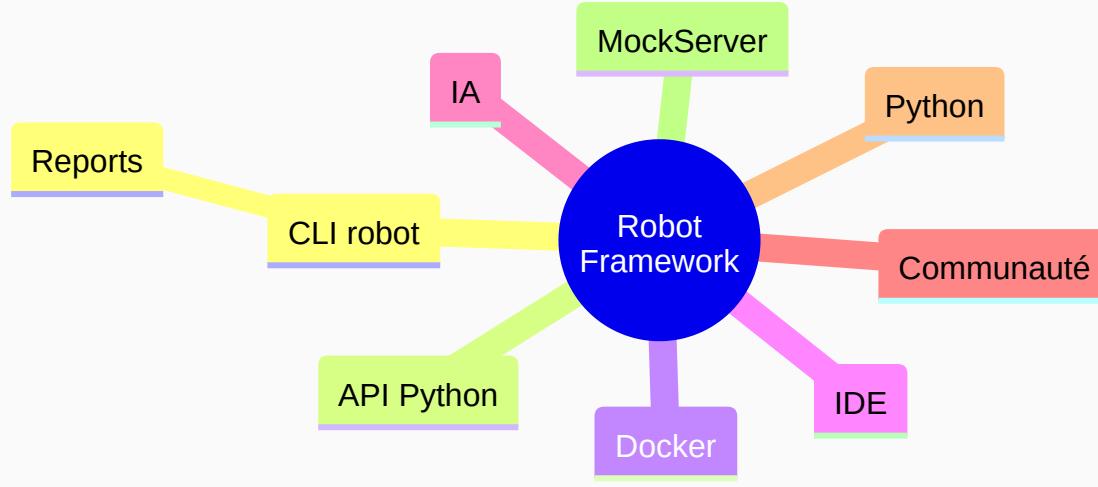
# Librairies Core



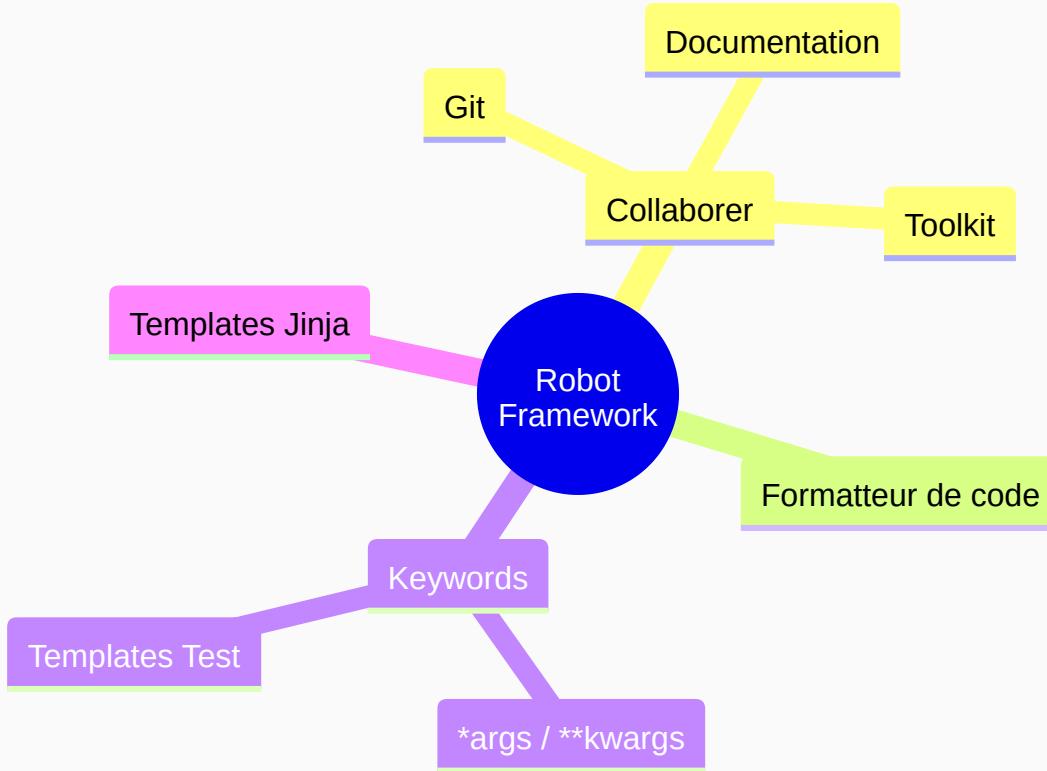
# Librairies



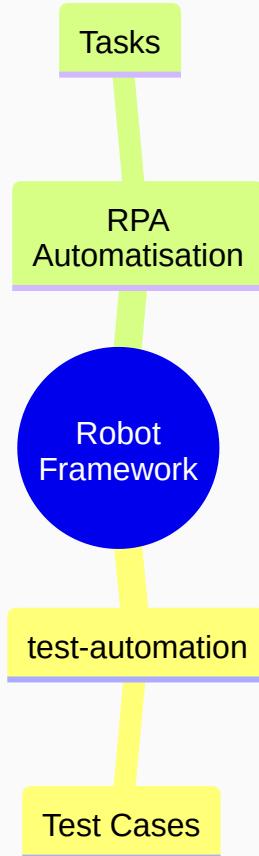
# Ecosystème



# Bonnes pratiques



# test-automation VS RPA



Rétro



TODO Klaxoon ?

## Questions & Réponses

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