Tests automatisés avec Playwright, Appium et Pytest

Pytest est une bibliothèque Python permettant de créer des tests automatisés pour tout genre d’application (sites web ou applications mobiles par exemple).

Pour ce tutoriel il est recommandé d'avoir une petite base de connaissance de la programmation avec python mais le code sera expliqué tout au long en considérant que vous débutez.

Installation de Pytest :

Ce tutoriel a été réalisé avec l’éditeur de code Visual Studio Code, c’est un éditeur gratuit très répandu. Téléchargement sur <https://code.visualstudio.com/download>

Une fois Visual Studio Code installé ouvrez le et créez un nouveau dossier :

Une image contenant capture d’écran, texte, Logiciel multimédia, Logiciel de graphisme

Le contenu généré par l’IA peut être incorrect.

Cliquez sur l’icône explorer et ouvrez un dossier

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Le contenu généré par l’IA peut être incorrect.

Tout au long nous allons coder avec le langage de programmation Python, c’est un des langages informatiques le plus utiliser car il facile à apprendre pour un débutant et l’on peut l’utiliser pour divers choses (applications de bureau, Data Science, IA, jeux vidéo…)

Le téléchargement est possible via le Microsoft Store : <https://apps.microsoft.com/detail/9PNRBTZXMB4Z?hl=en-us&gl=FR&ocid=pdpshare>

Ou sur le site web officiel : <https://www.python.org/downloads/>

Créez un nouveau fichier python en cliquant sur cette icône :

Une image contenant texte, capture d’écran, Police, nombre

Le contenu généré par l’IA peut être incorrect.

Appelez ce fichier « test\_playwright.py ».

Il faut maintenant configurer l’éditeur pour qu’il exécute python :

En bas à droite de la fenêtre cliquez sur le menu suivant :

Une image contenant texte, capture d’écran, Police

Le contenu généré par l’IA peut être incorrect.

Puis en haut, sélectionnez la version de python que vous aviez installé :

Une image contenant texte, Police, logiciel, capture d’écran

Le contenu généré par l’IA peut être incorrect.

Afin de tester si python fonctionne bien, codez cette ligne, la fonction print sert à afficher du texte, le texte que l'on veut écrire est mis entre les parenthèses :

Attention, en programmation, le texte (chaîne de caractère ou "string") est toujours entre guillemets ' ou double guillemets ".

***print("Hello World")***

Maintenant le programme via cette icône en haut à droite :

Une image contenant texte, capture d’écran, horloge, Police

Le contenu généré par l’IA peut être incorrect.

Le texte s'affiche correctement dans le terminal :

Une image contenant texte, capture d’écran, Police

Le contenu généré par l’IA peut être incorrect.



Avant d'aller plus loin, il va falloir créer ce que l’on appelle un environnement, ouvrez un nouveau terminal via le menu Terminal ou le raccourci Ctrl + Shift + ¨

Une image contenant texte, capture d’écran, Police

Le contenu généré par l’IA peut être incorrect.

Une image contenant texte, capture d’écran, logiciel, Logiciel multimédia

Le contenu généré par l’IA peut être incorrect.

Le terminal s’ouvre dans cette fenêtre :

Une image contenant texte, capture d’écran

Le contenu généré par l’IA peut être incorrect.

Entrez la commande :

***python -m venv venv***

Cette commande signifie que l'on veut créer un environnement virtuel (venv) et qu'on lui donne venv comme nom.

Pour utiliser la venv il suffit d'entrer cette commande :

***venv\Scripts\Activate***

Pour l'ensemble de nos tests on va utiliser la bibliothèque Pytest, installable via cette commande :

***pip install pytest***

Et voilà, Pytest est installé, dans la prochaine partie nous allons faire nos premiers tests web.

**Tests web automatisés :**

Installation de Playwright :

Avant de pouvoir utiliser Playwright, il faut l’installer et télécharger les navigateurs nécessaires. Utilisez les commandes :

***pip install playwright***

***playwright install***

***pip install pytest-playwright***

Playwright permet d’automatiser des sites web comme un vrai utilisateur : cliquer sur des boutons, remplir des champs, charger des images, etc…

Premier test :

Notre premier test de déroulera comme suivant :

1. Aller sur le site de Wikipédia.
2. Rechercher un mot (pour ce tuto on va rechercher "Python").
3. Vérifier que le page contient le on titre.

Dans votre fichier \_test.py :

Entrez ou copiez le code suivant :

***from playwright.sync\_api import Page, expect***

***def test\_recherche\_wikipedia(page: Page):***

***page.goto("https://fr.wikipedia.org")***

***expect(page.get\_by\_role("searchbox", name="Rechercher sur Wikipédia")).to\_be\_visible()***

***expect(page.get\_by\_role("button", name="Rechercher")).to\_be\_visible()***

***page.get\_by\_role("searchbox", name="Rechercher sur Wikipédia").click()***

***page.get\_by\_role("searchbox", name="Rechercher sur Wikipédia").fill("Python")***

***page.get\_by\_role("button", name="Rechercher").click()***

***expect(page.locator("#firstHeading")).to\_contain\_text("Python")***

Explication du code :

***from playwright.sync\_api import Page, expect***

Cette ligne permet d'importer (***import***) les commandes ***Page*** et ***expect*** depuis (***from***) la bibliothèque playwright (***playwright.sync\_api***).

***def test\_recherche\_wikipedia(page: Page):***

Cette ligne définit (***def***) une fonction de test nommée test\_recherche\_wikipedia, qui prend en paramètre ***page*** (de type ***Page***), c’est-à-dire une instance de navigateur ouverte automatiquement par Playwright pour faire le test.

***page.goto("https://fr.wikipedia.org")***

Cette ligne permet d’ouvrir la page (***goto***) du site Wikipédia en français dans le navigateur.

***expect(page.get\_by\_role("searchbox", name="Rechercher sur Wikipédia")).to\_be\_visible()***

Cette ligne attend (***expect***) que le champ de recherche (repéré par rôle ***(.get\_by\_role***), du nom ***name="Rechercher sur Wikipédia"***) soit visible sur la page ***(.to\_be\_visible()***) avant de continuer. S'il ne trouve pas l'élément, une erreur se produira, faisant échouer le test.

***expect(page.get\_by\_role("button", name="Rechercher")).to\_be\_visible()***

Cette ligne attend (***expect***) que le bouton (repéré par rôle ***(.get\_by\_role***), du nom ***name="Rechercher"***) soit visible sur la page ***(.to\_be\_visible()***) avant de continuer. S'il ne trouve pas l'élément, une erreur se produira, faisant échouer le test.

***page.get\_by\_role("searchbox", name="Rechercher sur Wikipédia").click()***

Ici, on clique (***.click()***) dans la barre de recherche. Cela place le curseur dans le champ, comme si on cliquait avec la souris.

***page.get\_by\_role("searchbox", name="Rechercher sur Wikipédia").fill("Python")***

Cette ligne remplit (***fill***) la barre de recherche avec le mot Python.

***page.get\_by\_role("button", name="Rechercher").click()***

Cette ligne simule un clic sur le bouton Rechercher, ce qui lance la recherche de l’article Python.

***expect(page.locator("#firstHeading")).to\_contain\_text("Python")***

Cette ligne vérifie (***expect***) que le titre principal de la page (repéré par l’id (=identifiant) ***#firstHeading***) contient bien le texte ***(.to\_contain\_text***) "Python". Cela permet de confirmer que Wikipédia a bien affiché la bonne page.

**Attention, l'indentation en Python est très importante, elle sert à indiquer dans quel "morceau" du code on se situe, l'indentation est souvent faite avec une tabulation (touche Tab du clavier) mais des espaces fonctionnent aussi, la règle est que tout le code indenté ensemble doit être indenté au même niveau.**

**Une fois le "morceau" de code terminé, retirez l'indentation.**

Une image contenant texte, capture d’écran, Police, logiciel

Le contenu généré par l’IA peut être incorrect.



Exemple : tout le code situé dans la fonction a une tabulation, tout le code est aligné au même niveau

**Dans visual studio code, l'indentation est automatique mais il faut quand même -faire attention, souvenez-vous qu'après une ligne se terminant par un double point " :" il y a toujours une indentation après.**

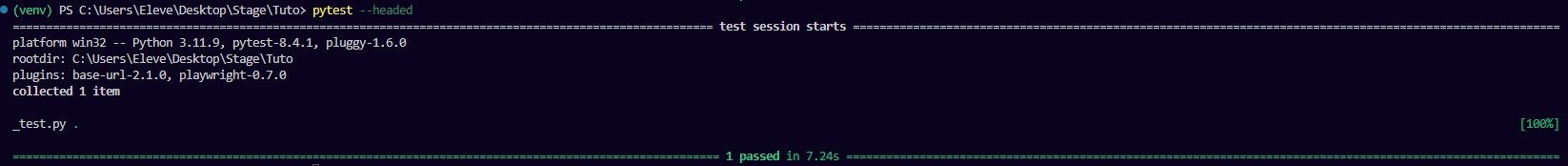
Pour lancer le test, utilisez la commande :

***pytest***

Et pour lancer le test avec l'interface :

***pytest –headed***

Si le texte a fonctionné, vous devriez voir dans la console quelque chose comme ceci :



Il existe bien d'autres commandes pour créer des tests et il n'est aussi pas super simple de rechercher chaque élément de la page pour écrire le code, surtout si le test comprend beaucoup d'éléments.

Une façon plus simple pour réaliser le code des tests est d'utiliser le ***recorder*** intégré è Playwright.

Pour l'utiliser, tapez cette commande dans le terminal :

***playwright codegen "URL\_du\_site"***

Exemple : ***playwright codegen*** [***https://fr.wikipedia.org***](https://fr.wikipedia.org)

Deux fenêtres s'ouvrent, la première est le navigateur avec le site indiqué dans la commande. En haut de cette fenêtre se trouve une barre d'outils.

Une image contenant texte, logiciel, Police, capture d’écran

Le contenu généré par l’IA peut être incorrect.

 Permet d'activer ou de désactiver l'enregistrement.

 Permet d'obtenir ce qui permet d'identifier un élément.

 Permet de vérifier la visibilité d'un élément

 Permet de vérifier la présence d'un texte précis dans un élément.

 Permet de vérifier la présence d'une valeur précise dans un élément.

Lorsque vous cliquez sur des éléments ou que vous rentrez des valeurs, tout est enregistré automatiquement.

La deuxième fenêtre est le générateur de code, toutes les actions que vous faites dans le navigateur y est transformé en code, dans de nombreux langages.

Une image contenant texte, Appareils électroniques, capture d’écran, logiciel

Le contenu généré par l’IA peut être incorrect.

Sélectionner Pytest dans le menu indiqué pour obtenir le code fonctionnant avec ce que nous avions mis en place.

Il suffit maintenant de copier le code et lorsque vous l'exécuterez, toutes les actions que vous aviez faites auparavant sur la page sera reproduit.

**Tests mobiles automatisés :**

Installation d'Appium :

Appium permet de contrôler un téléphone (Android ou IOS), combiné avec la bibliothèque de tests python Pytest que l'on a utilisé précédemment on peut faire des tests automatisés sur mobile.

Pour ce tutoriel, on va utiliser un téléphone physique utilisant Android.

Avant d'installer Appium, il faut avoir :

* Python (déjà installé plus tôt)
* Java JDK (installation sur <https://www.oracle.com/java/technologies/downloads/>)
* Node.js (installation sur <https://nodejs.org/fr/download>)

Maintenant installons Appium, ouvrez un nouveau dossier sur visual studio code et créez un fichier test\_appium.py. Ensuite créez un environnement virtuel et tapez ou copiez cette commande dans votre terminal :

***npm install -g appium***

Puis vérifions l'installation avec :

***appium -v***

Et installons les drivers Appium :

***appium driver install uiautomator2***

Vérifiez avec :

***appium driver list --installed***

Installons aussi un plugin (= extension) qui nous permettra d'afficher l'écran sur une page web et d'inspecter les éléments afin de récupérer de quoi les identifier (un peu à la manière de Playwright inspector) :

***appium plugin install --source=npm appium-inspector-plugin***

Appium a également besoin d'autres outils pour fonctionner.

On pourrait utiliser Android Studio pour aussi faire des tests via émulateur mais dans ce tuto on va s'en passer).

Il faut installer le SDK pour Android, plus précisément les "platform-tools" :

Téléchargez sur :

<https://developer.android.com/tools/releases/platform-tools?hl=fr>

Décompressez l'archive et placez le dossier "platform-tools" dans un dossier suivant cette adresse en créant les dossiers nécessaires : "***C:\Program Files\Android\cmdline-tools\latest\platform-tools***"

Il faut maintenant que nous créons les variables d'environnement, lancer PowerShell **en mode Administrateur** via le menu démarrer et entrez les commandes suivantes :

**Java JDK :**

***[System.Environment]::SetEnvironmentVariable("JAVA\_HOME", "C:\Program Files\Java\jdk-24", "Machine")***

**Si le chemin des programmes sur votre machine est différent, adaptez la commande.**

**Android :**

***[System.Environment]::SetEnvironmentVariable("ANDROID\_HOME", "C:\Program files\Android", "Machine")***

**PATH Android :**

***$oldPath = [System.Environment]::GetEnvironmentVariable("Path", "Machine")***

***$newPath = ";C:\Program Files\Android\cmdline-tools\latest\platform-tools"***

***[System.Environment]::SetEnvironmentVariable("Path", "$oldPath$newPath", "Machine")***

***[System.Environment]::SetEnvironmentVariable("Path", "$oldPath$newPath", "Machine")***

**Vérification :**

***echo $env:JAVA\_HOME***

***echo $env:ANDROID\_HOME***

***echo $env:Path***

Redémarez votre machine ou forcez la prise en compte immédiate avec :

***Stop-Process -Name explorer -Force***

***Start-Process explorer***