

UHA

# Documentation d'installation et d'utilisation

FLORIAN VISCA

# Sommaire

## Table des matières

Sommaire.....	1
I. Introduction.....	2
II. Installation du Serveur.....	3
1. Récupération du code Master.py : .....	3
2. Bibliothèques : .....	4
3. Préparation du serveur : .....	4
III. Installation Slave.....	6
1. Récupération du code Slave.py : .....	6
2. Bibliothèques et Compilateurs : .....	7
3. Préparation du Slave : .....	7
IV. Installation Client .....	11
1. Récupération du code Client.py : .....	11
2. Bibliothèques : .....	12
3. Préparation du Client : .....	12
V. Recommandation d'utilisation .....	14

## I. Introduction

Ce document vous permettra d'avoir une approche rigoureuse et méthodique pour l'installation des différents composants réalisés pour le projet SAE 3.02. Cela comprendra les composants suivants :

### **Le Serveur :**

- Élément central de mon architecture, le serveur coordonne les échanges entre les clients et les slaves.
- Son installation et sa configuration permettent d'assurer une communication fluide au sein du réseau.

### **Le Slave :**

- Ce composant permet de réaliser toutes les exécutions de programme en langage Python, Java, C et C++.
- Sa mise en place demande l'installation de plusieurs compilateurs pour réussir l'exécution des différents programmes.

### **Le Client :**

- Possède une interface graphique pour simplifier l'utilisation et la visualisation des résultats.
- Avec son installation, un utilisateur n'aura besoin que d'un seul langage de programmation qui est le Python.

Le guide décrit non seulement les étapes nécessaires à l'installation, mais également les paramètres importants à prendre en compte pour éviter des erreurs, comme des conflits de ports ou des problèmes de compatibilité entre les composants.

En suivant ce document, vous serez en mesure de configurer correctement chaque partie du système, tout en assurant le bon déroulement du projet SAE 302.

## II. Installation du Serveur

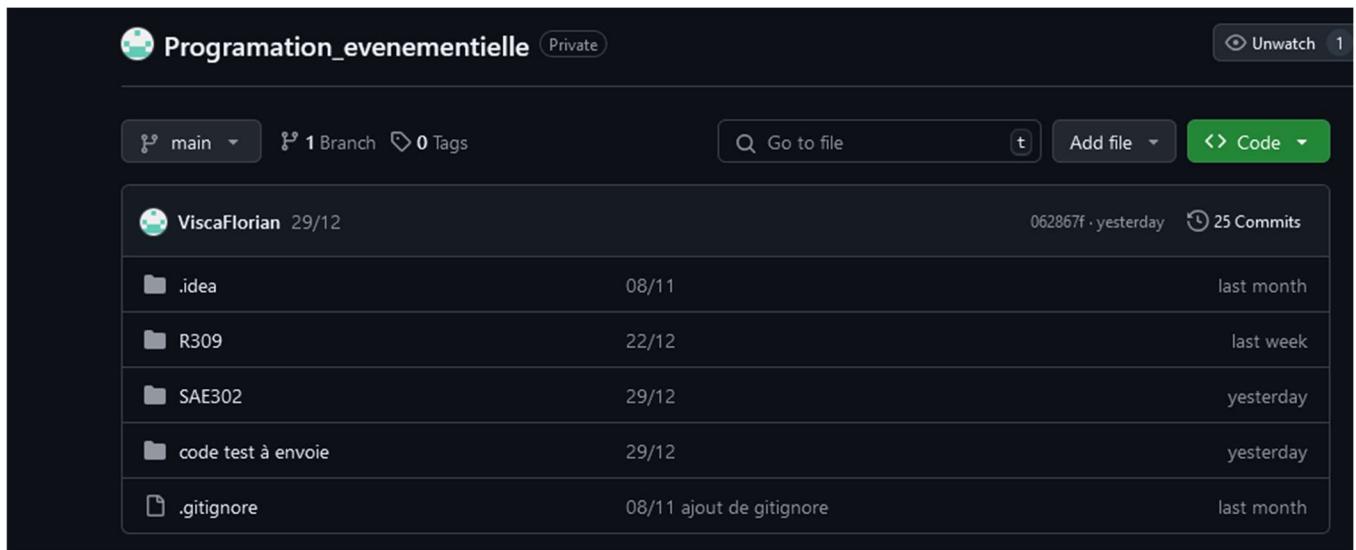
### 1. Récupération du code Master.py :

#### Comment récupérer le fichier master.py

Pour télécharger le fichier **master.py**, rendez-vous sur le dépôt GitHub correspondant à l'adresse suivante :

👉 [https://github.com/ViscaFlorian/Programation\\_evenementielle.git](https://github.com/ViscaFlorian/Programation_evenementielle.git)

Après avoir cliqué sur ce lien, vous serez redirigé vers la page principale du dépôt. Cette page contient tous les fichiers et dossiers nécessaires pour le projet, y compris le fichier **master.py** dans le répertoire approprié.



The screenshot shows the GitHub repository 'Programation\_evenementielle'. At the top, it says 'Private'. Below that, there are buttons for 'main' (selected), '1 Branch', '0 Tags', a search bar 'Go to file', and buttons for 'Add file' and 'Code'. The commit history is listed:

Author	Date	Commit Message	Time Ago
ViscaFlorian	29/12	062867f · yesterday	25 Commits
	08/11		last month
	22/12		last week
	29/12		yesterday
	29/12		yesterday
	08/11 ajout de gitignore		last month

#### Explication pour naviguer dans le dossier SAE302 et accéder au fichier master.py

Dans le dossier **SAE302**, vous trouverez trois sous-dossiers principaux : **Client**, **Serveur** et **Sujet**. Celui qui nous intéresse pour cette étape est le dossier **Serveur**.

En ouvrant ce dossier, vous découvrirez deux autres répertoires :

- **Srv\_master**
- **Srv\_slave**

Pour continuer, sélectionnez le dossier **Srv\_master**. A l'intérieur, vous trouverez le fichier nommé **master.py**

## 2. Bibliothèques :

### Bibliothèques nécessaires au bon fonctionnement du Serveur

Pour assurer le bon fonctionnement de ce projet, les bibliothèques suivantes sont nécessaires

- **sys**
- **socket**
- **threading**
- **time**
- **argparse**
- **PyQt6**

La plupart de ces bibliothèques sont natives à Python, ce qui signifie qu'elles sont disponibles par défaut sans installation supplémentaire. Cependant, la bibliothèque **PyQt6** n'étant pas native.

## 3. Préparation du serveur :

### Sur Windows :

#### 1) Vérification de l'installation de Python :

Tout d'abord, il est important de vérifier si **Python** est installé sur votre machine. Pour cela, ouvrez un terminal et exécutez la commande suivante :

```
python --version
```

Si un résultat s'affiche sous cette forme :

Python 3.12.1

Cela signifie que Python est correctement installé sur votre système.

#### 2) Installation de PyQt6 :

Ensuite, pour installer la bibliothèque PyQt6, exécutez la commande suivante dans votre terminal :

```
pip install PyQt6
```

Ci-dessous ce que vous devriez obtenir :

#### 3) Lancement du fichier master.py :

Une fois l'installation terminée, vous pouvez lancer le fichier **master.py**. Pour ce faire, ouvrez un terminal et exécutez la commande suivante :

```
py.exe <chemin_vers_master.py>\master.py --portC <port_client> --portS <port_slave>
```

Remplacez **<chemin\_vers\_master.py>** par le chemin d'accès au fichier **master.py** et définissez les ports d'écoute pour le socket client (**--portC**) et le socket slave (**--portS**).

## Sur Debian :

### 1) Vérification de l'installation de Python :

Tout d'abord, il est important de vérifier si **Python** est installé sur votre machine. Pour cela, ouvrez un terminal et exécutez la commande suivante :

```
python --version
```

Si un résultat s'affiche sous cette forme :

```
Python 3.12.1
```

Cela signifie que Python est correctement installé sur votre système.

### 2) Installation de PyQt6 :

#### a) Création d'un environnement virtuel (venv) :

Assurez-vous que python3-venv est installé sur votre système. Si ce n'est pas le cas, vous pouvez l'installer avec la commande suivante :

```
sudo apt update
```

```
sudo apt install python3-venv
```

Il est recommandé de créer un environnement virtuel pour gérer les dépendances du projet. Pour ce faire, exécutez les commandes suivantes dans votre terminal :

```
python -m venv venv
```

Cette commande crée un dossier nommé **venv** dans votre répertoire actuel, contenant un environnement isolé.

Activez l'environnement virtuel **venv** avec la commande suivante :

```
source venv/bin/activate
```

Une fois activé, le nom de l'environnement virtuel (par exemple, **venv**) devrait apparaître dans votre terminal.

#### b) Installation de PyQt6 :

Ensuite, pour installer la bibliothèque PyQt6, exécutez la commande suivante dans votre terminal :

```
pip install PyQt6
```

Ci-dessous ce que vous devriez obtenir :

```
(venv) root@debian:~# pip install PyQt6
Collecting PyQt6
  Downloading PyQt6-6.8.0-cp39-abi3-manylinux_2_35_x86_64.whl (8.3 MB)
    ━━━━━━━━━━━━━━━━ 8.3/8.3 MB 31.8 MB/s eta 0:00:00

Collecting PyQt6-sip<14,>=13.8
  Downloading PyQt6_sip-13.9.1-cp311-cp311-manylinux_2_5_x86_64.manylinux1_x86_64.whl (294 kB)
    ━━━━━━━━━━━━━━ 294.5/294.5 kB 22.3 MB/s eta 0:00:00

Collecting PyQt6-Qt6<6.9.0,>=6.8.0
  Downloading PyQt6_Qt6-6.8.1-py3-none-manylinux_2_35_x86_64.whl (72.6 MB)
    ━━━━━━━━━━━━━━ 72.6/72.6 MB 13.3 MB/s eta 0:00:00

Installing collected packages: PyQt6-Qt6, PyQt6-sip, PyQt6
Successfully installed PyQt6-6.8.0 PyQt6-Qt6-6.8.1 PyQt6-sip-13.9.1
```

### 3) Lancement du fichier master.py :

Une fois l'installation terminée, vous pouvez lancer le fichier **master.py**. Pour ce faire, ouvrez un terminal et exécutez la commande suivante :

```
py.exe <chemin_vers_master.py>\master.py --portC <port_client> --portS <port_slave>
```

Remplacez **<chemin\_vers\_master.py>** par le chemin d'accès au fichier **master.py** et définissez les ports d'écoute pour le socket client (**--portC**) et le socket slave (**--portS**).

## III. Installation Slave

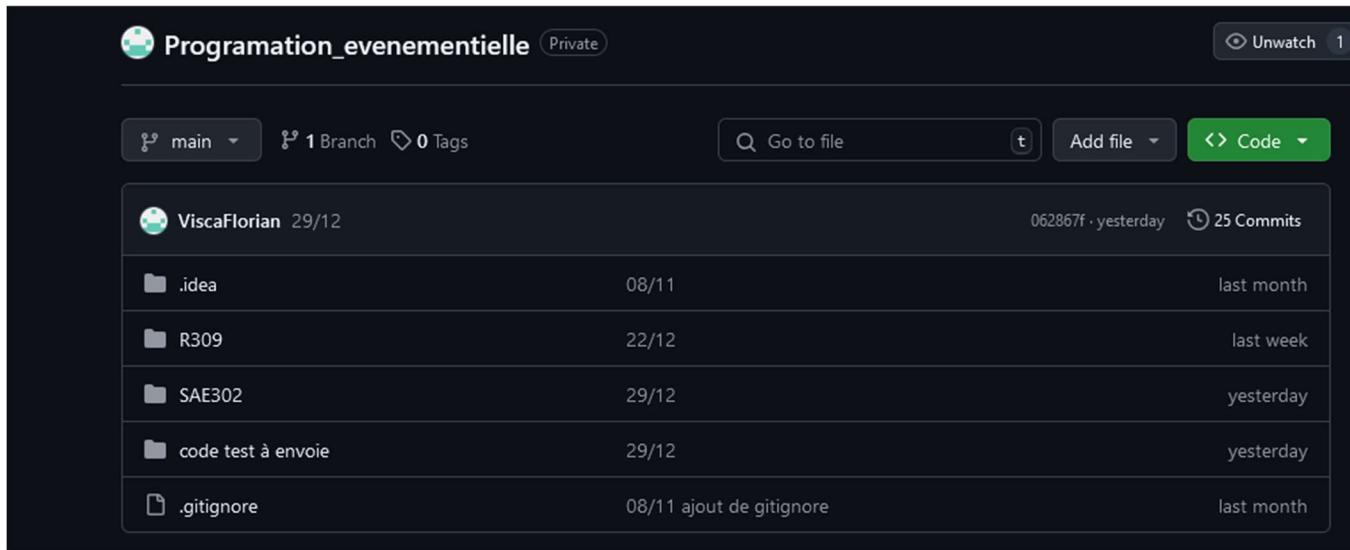
### 1. Récupération du code Slave.py :

#### Comment récupérer le fichier slave.py

Pour télécharger le fichier **slave.py**, rendez-vous sur le dépôt GitHub correspondant à l'adresse suivante :

👉 [https://github.com/ViscaFlorian/Programation\\_evenementielle.git](https://github.com/ViscaFlorian/Programation_evenementielle.git)

Après avoir cliqué sur ce lien, vous serez redirigé vers la page principale du dépôt. Cette page contient tous les fichiers et dossiers nécessaires pour le projet, y compris le fichier **Slave.py** dans le répertoire approprié.



#### Explication pour naviguer dans le dossier SAE302 et accéder au fichier slave.py

Dans le dossier **SAE302**, vous trouverez trois sous-dossiers principaux : **Client**, **Serveur** et **Sujet**. Celui qui nous intéresse pour cette étape est le dossier **Serveur**.

En ouvrant ce dossier, vous découvrirez deux autres répertoires :

- **Srv\_master**
- **Srv\_slave**

Pour continuer, sélectionnez le dossier **Srv\_slave**. A l'intérieur, vous trouverez le fichier nommé **slave.py**

## 2. Bibliothèques et Compilateurs :

### **Bibliothèques nécessaires au bon fonctionnement du Slave**

Pour assurer le bon fonctionnement de ce projet, les bibliothèques suivantes sont nécessaires

- **Socker**
- **os**
- **subprocess**
- **argparse**
- **re**

Comparé au serveur, le slave utiliser uniquement des bibliothèques natives à python, Il n'y aura pas besoin d'installer manuellement de bibliothèques.

### **Compilateurs nécessaires au bon fonctionnement du Slave**

Le fichier **slave.py** doit être capable d'exécuter des codes dans plusieurs langages de programmation, tels que **Python**, **Java**, **C** et **C++**. Pour cela, le **slave** doit être configuré pour compiler et exécuter ces langages, en installant les compilateurs et environnements nécessaires.

## 3. Préparation du Slave :

### **Sur Windows :**

#### 1) Vérification de l'installation de Python :

Tout d'abord, il est important de vérifier si **Python** est installé sur votre machine. Pour cela, ouvrez un terminal et exécutez la commande suivante :

```
python --version
```

Si un résultat s'affiche sous cette forme :

```
Python 3.12.1
```

Cela signifie que Python est correctement installé sur votre système.

#### 1) Compilateur Java :

Pour exécuter des programmes **Java**, vous devez installer le **JDK** (Java Development Kit).

Télécharger et installer le JDK depuis le site officiel [Oracle](#) ou via un gestionnaire comme AdoptOpenJDK

Une fois installé, vérifiez l'installation avec la commande suivante exécuter votre terminal :

```
java --version
```

Cela devrait afficher la version d'**OpenJDK** installée, confirmant ainsi que le compilateur Java fonctionne correctement.

## 2) Compilateur C et C++ :

Pour compiler des programmes en C, vous devez installer GCC (GNU Compiler Collection).

### a) Téléchargement de TDM-GCC :

Aller sur le site officiel de [TDM-GCC](#)

### b) Installation de TDM-GCC :

Exécutez l'installateur téléchargé et suivez les instructions à l'écran pour installer TDM-GCC. Assurez-vous de cocher l'option pour ajouter TDM-GCC au PATH lors de l'installation. Cela permettra au compilateur de fonctionner directement à partir du terminal.

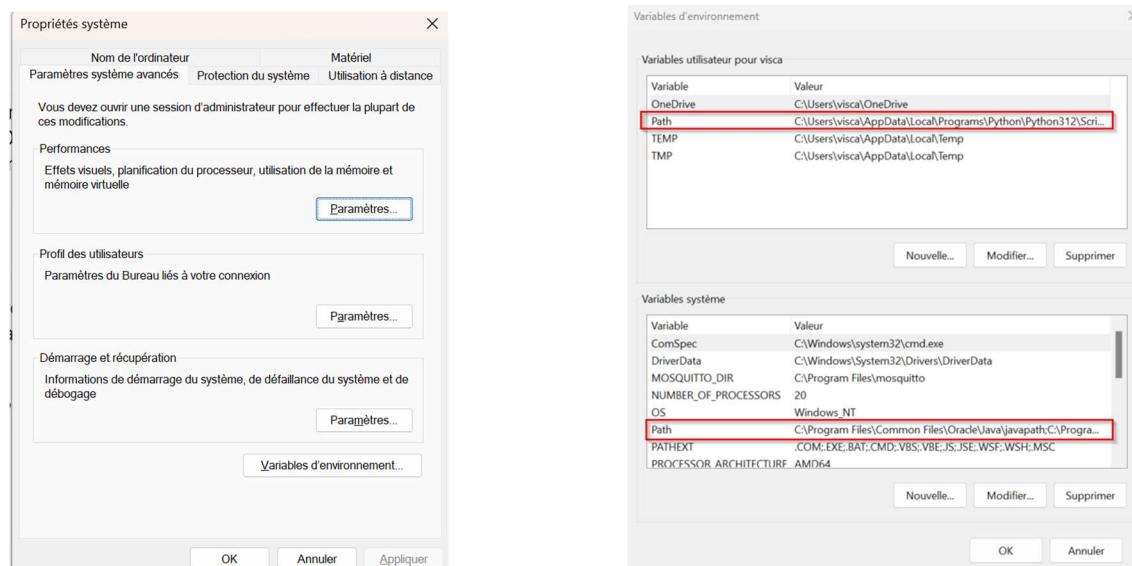
### c) Vérification de l'installation :

Une fois l'installation terminée, ouvrez un terminal et vérifiez que TDM-GCC est correctement installé en exécutant la commande suivante :

```
gcc --version
```

Si cela ne fonctionne toujours pas, vous devrez vérifier les variables d'environnement. Pour ce faire, suivez ces étapes :

- Ouvrez les **Paramètres système avancés** et cliquez sur **Variables d'environnement**.
- Dans les sections **Variables utilisateur** et **Variables système**, recherchez la variable **Path**.
- Vérifiez si le chemin « **C:\TDM-GCC-64\bin** » y est présent. Si ce n'est pas le cas, ajoutez-le dans les deux répertoires.



### 3) Lancement du fichier **slave.py** :

Une fois l'installation terminée, vous pouvez lancer le fichier **slave.py**. Pour ce faire, ouvrez un terminal et exécutez la commande suivante :

```
py.exe <chemin_vers_slave.py>\slave.py <IP du server> <port slave>
```

Remplacez les éléments suivants :

- <chemin\_vers\_slave.py> : le chemin d'accès complet au fichier **slave.py**.
- <IP\_du\_server> : l'adresse IP du serveur auquel le **slave** doit se connecter.
- <port\_slave> : le port utilisé pour la communication **slave** sur le socket du serveur.

Assurez-vous que l'adresse IP et le port correspondent à ceux configurés pour le serveur ainsi que pour le socket slave.

### **Sur Debian :**

#### 2) Vérification de l'installation de **Python** :

Tout d'abord, il est important de vérifier si **Python** est installé sur votre machine. Pour cela, ouvrez un terminal et exécutez la commande suivante :

```
python --version
```

Si un résultat s'affiche sous cette forme :

```
Python 3.12.1
```

Cela signifie que Python est correctement installé sur votre système.

#### 1) Compilateur **Java** :

Pour installer **OpenJDK 17** (le compilateur Java), ouvrez un terminal et exécutez les commandes suivantes :

```
sudo apt update
```

```
sudo apt install openjdk-17-jdk
```

Une fois l'installation terminée, vous pouvez vérifier que **Java** est correctement installé en exécutant la commande suivante :

```
java --version
```

Cela devrait afficher la version d'**OpenJDK** installée, confirmant ainsi que le compilateur Java fonctionne correctement.

## 2) Compilateur C et C++ :

Pour installer les compilateurs **GCC** (pour **C**) et **G++** (pour **C++**), exécutez les commandes suivantes dans un terminal :

```
sudo apt install gcc  
sudo apt install g++
```

Une fois l'installation terminée, vous pouvez vérifier que les compilateurs sont correctement installés en exécutant les commandes suivantes :

Pour GCC (C) :

```
Gcc --version
```

Pour G++ (C++) :

```
g++ --version
```

Cela devrait afficher les versions de **GCC** et **G++**, confirmant ainsi que les compilateurs sont installés et fonctionnels.

## 3) Lancement du fichier slave.py :

Une fois l'installation terminée, vous pouvez lancer le fichier **slave.py**. Pour ce faire, ouvrez un terminal et exécutez la commande suivante :

```
py.exe <chemin_vers_slave.py>\slave.py <IP du server> <port slave>
```

Remplacez les éléments suivants :

- <chemin\_vers\_slave.py> : le chemin d'accès complet au fichier **slave.py**.
- <IP\_du\_server> : l'adresse IP du serveur auquel le **slave** doit se connecter.
- <port\_slave> : le port utilisé pour la communication **slave** sur le socket du serveur.

Assurez-vous que l'adresse IP et le port correspondent à ceux configurés pour le serveur et le socket du **slave**.

## IV. Installation Client

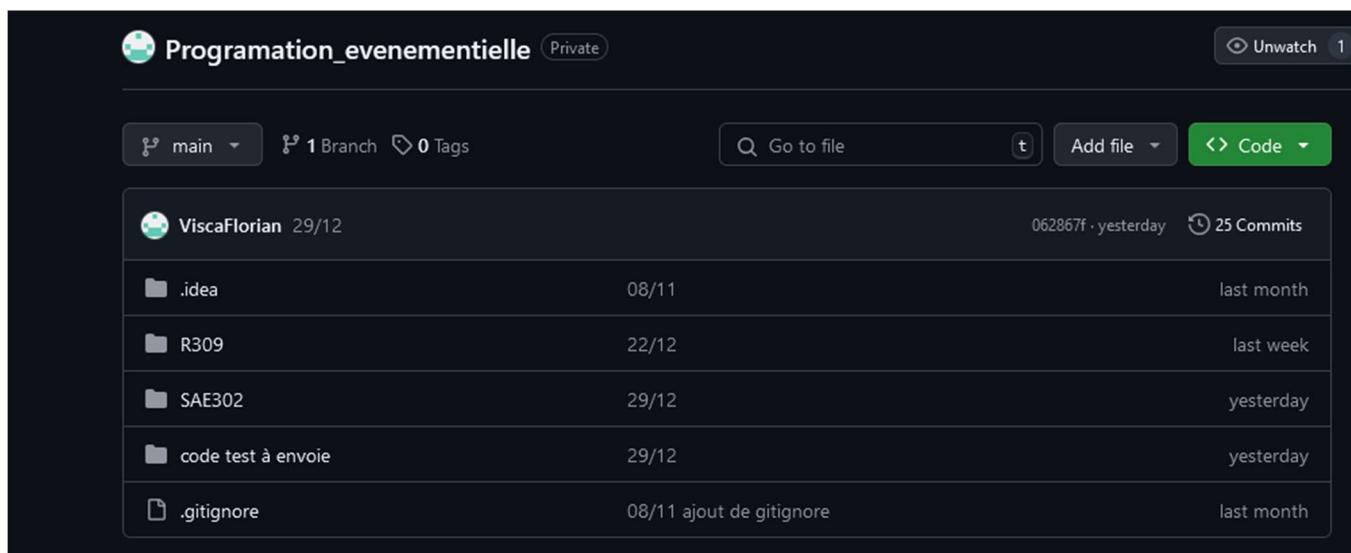
### 1. Récupération du code Client.py :

#### Comment récupérer le fichier client.py

Pour télécharger le fichier **client.py**, rendez-vous sur le dépôt GitHub correspondant à l'adresse suivante :

👉 [https://github.com/ViscaFlorian/Programation\\_evenementielle.git](https://github.com/ViscaFlorian/Programation_evenementielle.git)

Après avoir cliqué sur ce lien, vous serez redirigé vers la page principale du dépôt. Cette page contient tous les fichiers et dossiers nécessaires pour le projet, y compris le fichier **client.py** dans le répertoire approprié.



File/Folder	Last Commit	Author
.idea	08/11	last month
R309	22/12	last week
SAE302	29/12	yesterday
code test à envoie	29/12	yesterday
.gitignore	08/11 ajout de gitignore	last month

#### Explication pour naviguer dans le dossier SAE302 et accéder au fichier client.py

Dans le dossier **SAE302**, vous trouverez trois sous-dossiers principaux : **Client**, **Serveur** et **Sujet**. Celui qui nous intéresse pour cette étape est le dossier **Client.py**

## 2. Bibliothèques :

### Bibliothèques nécessaires au bon fonctionnement du Serveur

Pour assurer le bon fonctionnement de ce projet, les bibliothèques suivantes sont nécessaires

- **sys**
- **socket**
- **re**
- **PyQt6**

La plupart de ces bibliothèques sont natives à Python, ce qui signifie qu'elles sont disponibles par défaut sans installation supplémentaire. Cependant, la bibliothèque **PyQt6** n'étant pas native.

## 3. Préparation du Client :

### Sur Windows :

#### 1) Vérification de l'installation de Python :

Tout d'abord, il est important de vérifier si **Python** est installé sur votre machine. Pour cela, ouvrez un terminal et exécutez la commande suivante :

```
python --version
```

Si un résultat s'affiche sous cette forme :

```
Python 3.12.1
```

Cela signifie que Python est correctement installé sur votre système.

#### 2) Installation de PyQt6 :

Ensuite, pour installer la bibliothèque PyQt6, exécutez la commande suivante dans votre terminal :

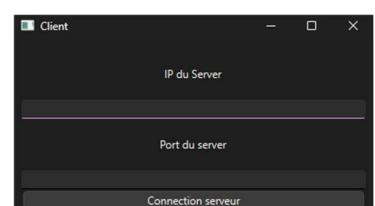
```
pip install PyQt6
```

Ci-contre ce que vous devriez obtenir :

#### 3) Lancement du fichier client.py :

Une fois l'installation terminée, vous pouvez lancer le fichier **client.py**. Pour ce faire, ouvrez un terminal et exécutez la commande suivante :

```
py.exe <chemin_vers_client.py>\client.py
```



Les informations sur le serveur seront à rentrer dans l'interface ci-contre :

## Sur Debian :

### 1) Vérification de l'installation de Python :

Tout d'abord, il est important de vérifier si **Python** est installé sur votre machine. Pour cela, ouvrez un terminal et exécutez la commande suivante :

```
python --version
```

Si un résultat s'affiche sous cette forme :

```
Python 3.12.1
```

Cela signifie que Python est correctement installé sur votre système.

### 2) Installation de PyQt6 :

#### a) Création d'un environnement virtuel (venv) :

Assurez-vous que python3-venv est installé sur votre système. Si ce n'est pas le cas, vous pouvez l'installer avec la commande suivante :

```
sudo apt update
```

```
sudo apt install python3-venv
```

Il est recommandé de créer un environnement virtuel pour gérer les dépendances du projet. Pour ce faire, exécutez les commandes suivantes dans votre terminal :

```
python -m venv venv
```

Cette commande crée un dossier nommé **venv** dans votre répertoire actuel, contenant un environnement isolé.

Activez l'environnement virtuel **venv** avec la commande suivante :

```
source venv/bin/activate
```

Une fois activé, le nom de l'environnement virtuel (par exemple, **venv**) devrait apparaître dans votre terminal.

#### b) Installation de PyQt6 :

Ensuite, pour installer la bibliothèque PyQt6, exécutez la commande suivante dans votre terminal :

```
pip install PyQt6
```

```
(venv) root@debian:~# pip install PyQt6
Collecting PyQt6
  Downloading PyQt6-6.8.0-cp39-abi3-manylinux_2_35_x86_64.whl (8.3 MB)
    8.3/8.3 MB 31.8 MB/s eta 0:00:00

Collecting PyQt6-sip<14,>=13.8
  Downloading PyQt6_sip-13.9.1-cp311-cp311-manylinux_2_5_x86_64.manylinux1_x86_64.whl (294 kB)
    294.5/294.5 kB 22.3 MB/s eta 0:00:00

Collecting PyQt6-Qt6<6.9.0,>=6.8.0
  Downloading PyQt6_Qt6-6.8.1-py3-none-manylinux_2_35_x86_64.whl (72.6 MB)
    72.6/72.6 MB 13.3 MB/s eta 0:00:00

Installing collected packages: PyQt6-Qt6, PyQt6-sip, PyQt6
Successfully installed PyQt6-6.8.0 PyQt6-Qt6-6.8.1 PyQt6-sip-13.9.1
```

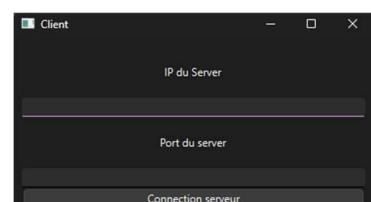
Ci-contre ce que vous devriez obtenir :

### 3) Lancement du fichier client.py :

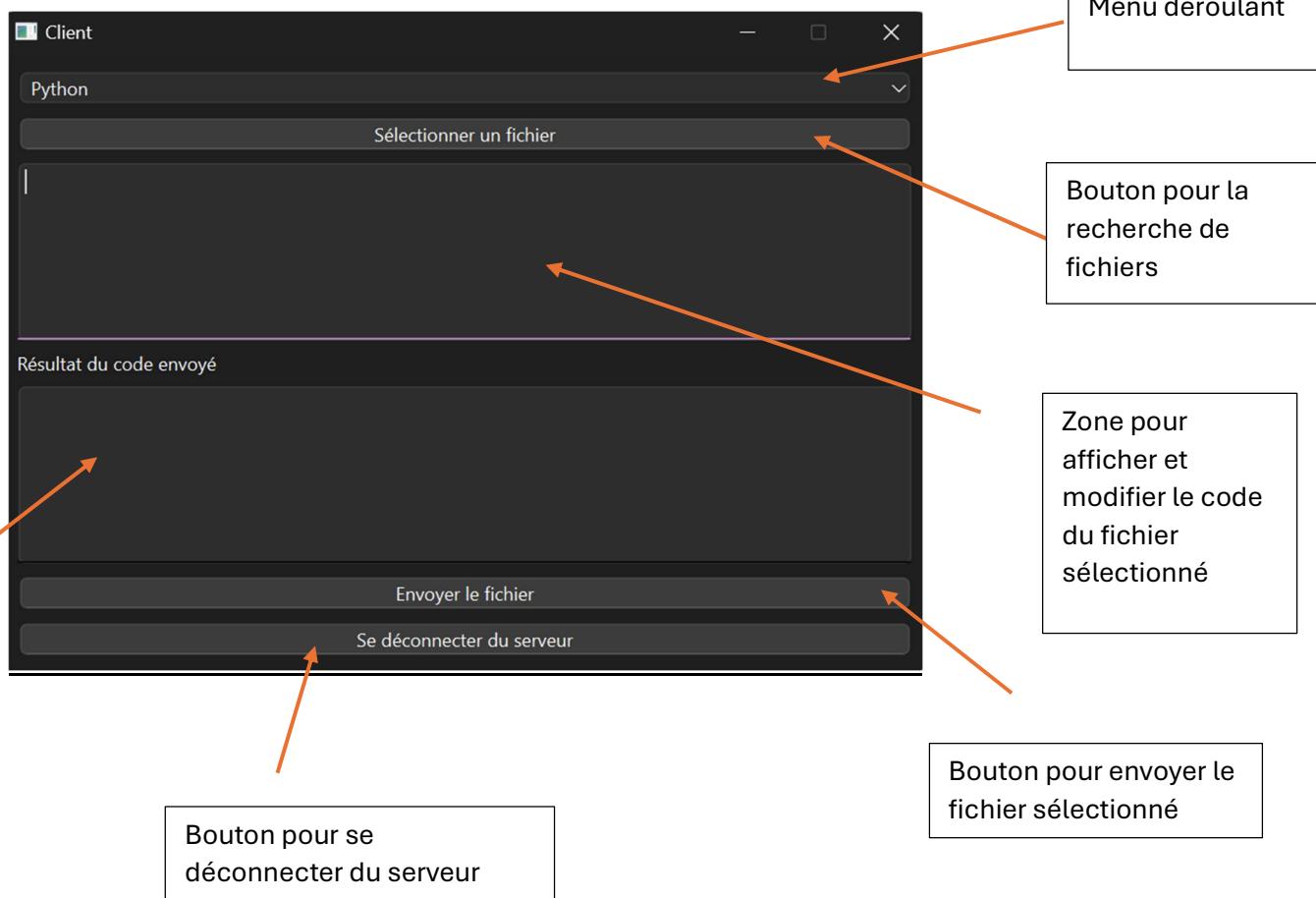
Une fois l'installation terminée, vous pouvez lancer le fichier **client.py**. Pour ce faire, ouvrez un terminal et exécutez la commande suivante :

```
py.exe <chemin_vers_client.py>\client.py
```

Les informations sur le serveur seront à rentrer dans l'interface ci-contre :



#### 4) Utilisation de l'interface graphique :



## V. Recommandation d'utilisation

Lorsque vous utilisez mon projet, je vous conseille de commencer par démarrer le **Serveur** en premier, puis de lancer tous les **Slaves** dont vous avez besoin. Une fois cela fait, vous pouvez démarrer les **Clients**.