

RAPPORT DU PROJET D'ALGO RÉPARTIS « RPC »

NOM : MOHAMMEDI

PRENOM : AMIRA

MATRICULE : 181831061126

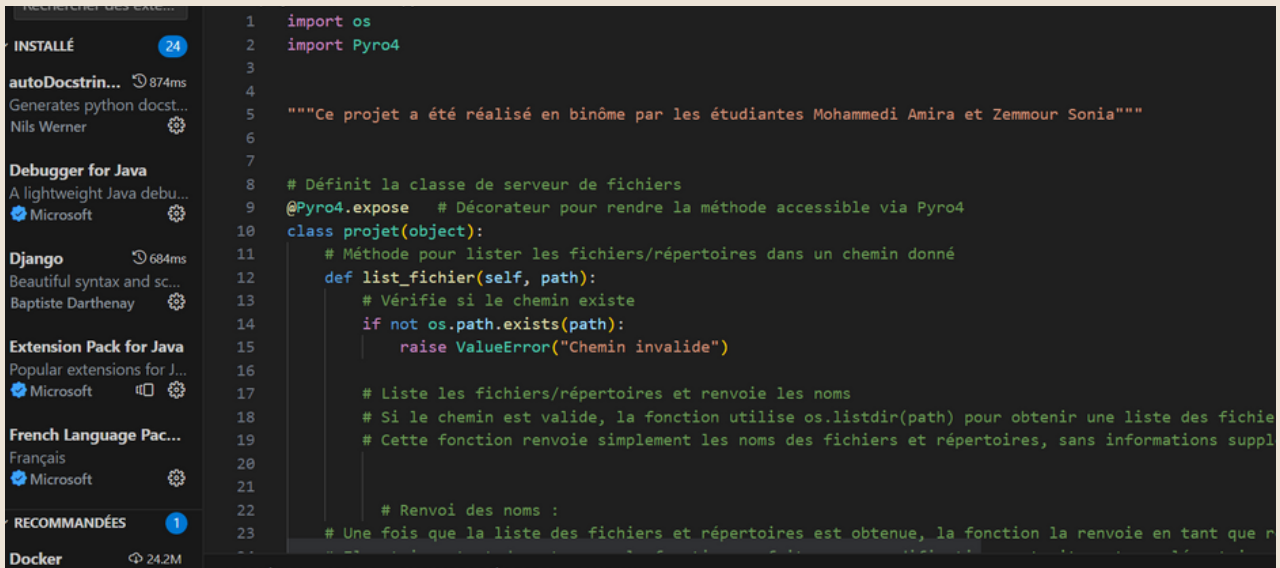
M1 HPC

NOM : ZEMMOUR

PRENOM : SONIA

MATRICULE : 181933031462

ANNÉE UNIVERSITAIRE : 2022/2023



```
1 import os
2 import Pyro4
3
4
5 """Ce projet a été réalisé en binôme par les étudiantes Mohammadi Amira et Zemmour Sonia"""
6
7
8 # Définit la classe de serveur de fichiers
9 @Pyro4.expose # Décorateur pour rendre la méthode accessible via Pyro4
10 class projet(object):
11     # Méthode pour lister les fichiers/répertoires dans un chemin donné
12     def list_fichier(self, path):
13         # Vérifie si le chemin existe
14         if not os.path.exists(path):
15             raise ValueError("Chemin invalide")
16
17         # Liste les fichiers/répertoires et renvoie les noms
18         # Si le chemin est valide, la fonction utilise os.listdir(path) pour obtenir une liste des fichiers
19         # Cette fonction renvoie simplement les noms des fichiers et répertoires, sans informations supplémentaires
20
21         # Renvoi des noms :
22         # Une fois que la liste des fichiers et répertoires est obtenue, la fonction la renvoie en tant que liste
```

code serveur.

Les modules `os` et `Pyro4` sont importés pour utiliser leurs fonctionnalités respectives.

La classe `projet` est définie et décorée avec `@Pyro4.expose`, ce qui permet de rendre les méthodes de la classe accessibles via `Pyro4`.

Plusieurs méthodes sont définies dans la classe `projet` pour effectuer des opérations telles que la liste des fichiers/répertoires, la création et la suppression de fichiers ou de répertoires, et l'ajout de données à un fichier existant.

Un objet `file_server` est créé en utilisant la classe `projet` pour représenter le serveur de fichiers.

Le démon `Pyro4` est configuré et démarré.

- Définit la classe de serveur de fichiers :

@Pyro4.expose Décorateur pour rendre la méthode accessible via Pyro4

Pyro4.expose

- Méthode pour lister les fichiers

Cette fonction renvoie simplement les noms des fichiers et répertoires

```
class projet(object):  
    # Méthode pour lister les fichiers/répertoire  
    def list_fichier(self, path):  
        # Vérifie si le chemin existe  
        if not os.path.exists(path):  
            raise ValueError("Chemin invalide")
```

- Méthode pour créer un fichier

Ouvre le fichier en mode écriture avec le context manager 'with' et le nomme 'file'

```
@Pyro4.expose  
def cree_fichier(self, path):  
    with open(path, 'w'):  
        pass
```

- Méthode pour supprimer un fichier

Utilise la fonction `os.remove` pour supprimer le fichier spécifié par 'path'

```
@Pyro4.expose
def supprimer_fichier(self, path):
    os.remove(path)
```

- Méthode pour supprimer un répertoire

La fonction `os.rmdir` supprime le répertoire spécifié par 'path'.

```
@Pyro4.expose
def supprimer_repertoire(self, path):
    os.rmdir(path)
```

- Méthode pour ajouter des données à un fichier existant

La méthode utilise l'instruction `with open` pour ouvrir le fichier spécifié par 'path' en mode d'ajout ('a').

```
@Pyro4.expose
def ajouter_des_donnee(self, path, data):
    with open(path, 'a') as f:
        f.write(data)
```

- Crée un objet de serveur de fichiers :

```
le_server = projet()
```

- serveur config :

Configuration du démon Pyro4 :

```
daemon = Pyro4.Daemon(host="127.0.10.5", port=5000)
```

- Démarre le démon Pyro4 et enregistre l'objet de serveur de fichiers:

```
uri = daemon.register(file_server)
```

- Afficher l'URI du serveur :

```
print("URI du serveur : ", uri)
```

- Démarrer le serveur Pyro4 :

```
daemon.requestLoop()
```

- exécution du code Serveur :

```
URI du serveur : PYRO:obj_6b84324d98f14782bf5be1e4ff4c39ce@127.0.10.5:5000
```

```
1 import Pyro4
2 import os
3
4 """Ce projet a été réalisé en binôme par les étudiantes Mohammedi Amira et Zemmour Sonia"""
5
6
7 # Récupère l'objet de serveur de fichiers à partir de son URI
8 uri = input("URI du serveur : ").strip()
9
10 #connecter au serveur
11 projet = Pyro4.Proxy(uri)
12
13 # Boucle principale du programme
14 while True:
15     print("Menu:")
16     print("1. Lister les fichiers/répertoires")
17     print("2. Créer un fichier")
18     print("3. Supprimer un fichier")
19     print("4. Créer un répertoire")
20     print("5. Supprimer un répertoire")
21     print("6. Ajouter des données à un fichier")
```

code client².

Le code importe les modules `os` et `Pyro4` pour leurs fonctionnalités. Ensuite, il définit une classe appelée `projet` et la décore avec `@Pyro4.expose` pour rendre ses méthodes accessibles via `Pyro4`. La classe contient plusieurs méthodes pour effectuer des opérations sur les fichiers et répertoires, telles que la liste, la création, la suppression et l'ajout de données à des fichiers.

Le code permet à l'utilisateur d'interagir avec un serveur de fichiers en utilisant une interface en ligne de commande. Il affiche un menu et attend les choix de l'utilisateur. En fonction du choix, des appels sont faits aux méthodes correspondantes de la classe `projet` pour effectuer les opérations demandées. L'utilisateur peut quitter le programme en saisissant "0" comme choix

- Récupère l'objet de serveur de fichiers à partir de son URI

```
uri = input("URI du serveur : ").strip()
```

- connecter au serveur

```
projet = Pyro4.Proxy(uri)
```

- Boucle principale du programme

```
while True:
    print("Menu:")
    print("1. Lister les fichiers/répertoires")
    print("2. Créer un fichier")
    print("3. Supprimer un fichier")
    print("4. Créer un répertoire")
    print("5. Supprimer un répertoire")
    print("6. Ajouter des données à un fichier")
    print("0. Quitter")

    choix = input("Entrez votre choix : ")
```

- Appel à la méthode list_fichier du serveur: Vérifie si le chemin existe et Affiche la liste des fichiers et répertoires

```
if choix == "1":
    path = input("Entrez un chemin : ")
    file_list = projet.list_fichier(path)

    if os.path.exists(path):
        print(f"Le chemin {path} existe.")
    else:
        print(f"Le chemin {path} n'existe pas.")

    for file_name in file_list:
        print(file_name)
```

- créer un fichier

```
elif choix == "2":
    path = input("Chemin pour créer un fichier : ")
    if not os.path.exists(path):
        projet.cree_fichier(path)
    else:
        print(f"Le répertoire {path} existe déjà.")
```

- supprimer fichier

```
elif choix == "3":
    path = input("Chemin pour supprimer un fichier : ")
    if not os.path.exists(path):
        print(f"Le fichier {path} n'existe pas.")
    else:
        projet.supprimer_fichier(path)
```

- créer un répertoire

```
elif choix == "4":
    path = input("Chemin pour créer un répertoire : ")
    # Vérifie si le chemin n'existe pas
    if not os.path.exists(path):
        projet.cree_repertoire(path)
    else:
        print(f"Le répertoire {path} existe déjà.")
```

- supprimer un répertoire

```
elif choix == "5":
    path = input("Chemin pour supprimer un répertoire : ")
    # Vérifie si le répertoire existe
    if not os.path.exists(path):
        print(f"Le répertoire {path} n'existe pas.")
    else:
        # Appel à la méthode supprimer_repertoire du serveur
        projet.supprimer_repertoire(path)
```


- Ajouter des données du serveur

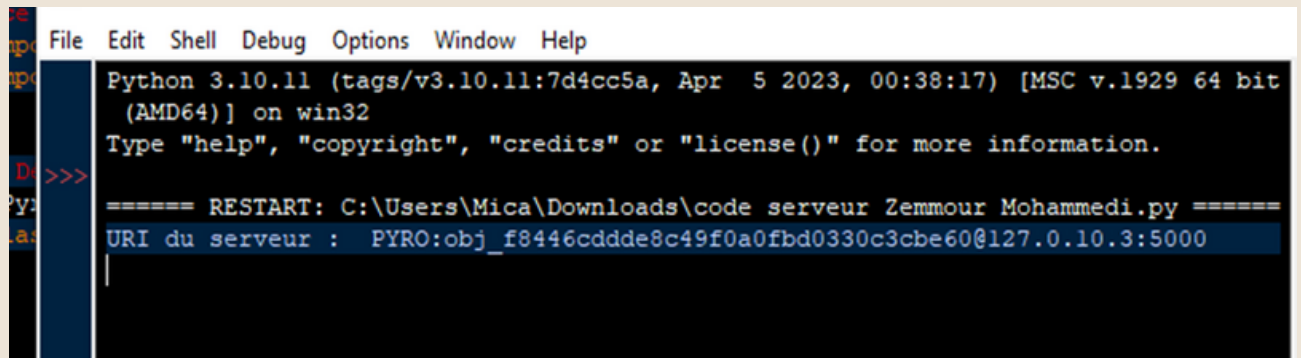
```
elif choix == "6":  
    path = input("Chemin du fichier pour les données : ")  
  
    # Vérifie si le fichier existe  
    if not os.path.exists(path):  
        print(f"Le fichier {path} n'existe pas.")  
    else:  
        data = input("Données à ajouter : ")  
        # Appel à la méthode ajouter_des_donnee du serveur  
        projet.ajouter_des_donnee(path, data)
```

- Vérifie si l'utilisateur a saisi "0" comme choix, ce qui indique qu'il souhaite quitter le programme.

```
elif choix == "0":  
    break  
  
else:  
    print("Choix invalide. Veuillez réessayer.")
```

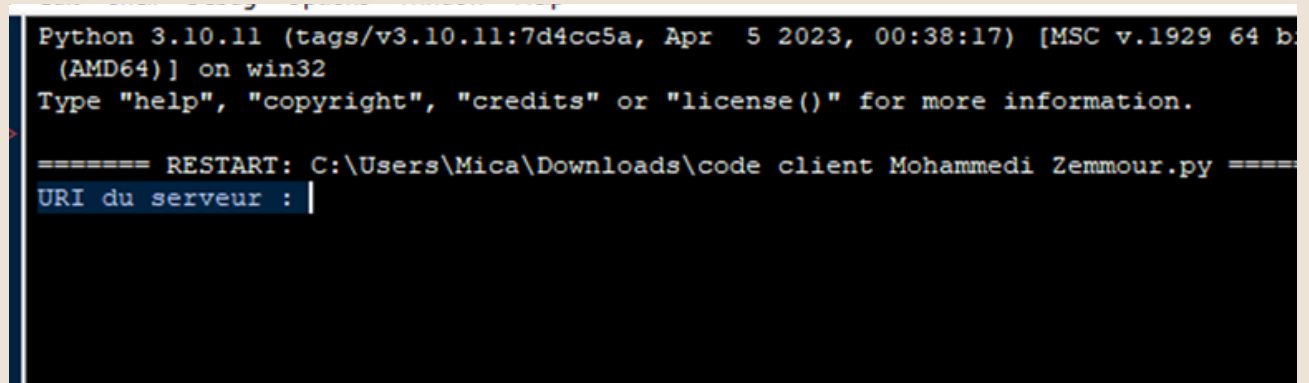
Mainienant, nous allons exécuter le programme

1) exécution du code Serveur :



```
File Edit Shell Debug Options Window Help
Python 3.10.11 (tags/v3.10.11:7d4cc5a, Apr 5 2023, 00:38:17) [MSC v.1929 64 bit
(AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\Mica\Downloads\code serveur Zemmour Mohammedi.py =====
URI du serveur : PYRO:obj_f8446cddde8c49f0a0fbd0330c3cbe60@127.0.10.3:5000
```

2) exécution du code client:



```
Python 3.10.11 (tags/v3.10.11:7d4cc5a, Apr 5 2023, 00:38:17) [MSC v.1929 64 b
(AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
===== RESTART: C:\Users\Mica\Downloads\code client Mohammedi Zemmour.py =====
URI du serveur :
```

3) copier l'URI du serveur dans le code client :

```
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\Mica\Downloads\code client Mohammedi Zemmour.py =====
URI du serveur : PYRO:obj_f8446cddde8c49f0a0fbd0330c3cbe60@127.0.10.3:5000
Menu:
1. Lister les fichiers/répertoires
2. Créer un fichier
3. Supprimer un fichier
4. Créer un répertoire
5. Supprimer un répertoire
6. Ajouter des données à un fichier
0. Quitter
Entrez votre choix : |
```

4) choisir un choix :

CHOIX 1 : LISTER LES FICHIERS ET LES RÉPERTOIRES

```
e chemin C:\Users\Amira\Desktop\tp_algo_repartie existe.
apture d'écran 2023-03-12 222306.png
lient-rpc.py
lient-rpc1.py
lient.py
lient2.py
lientdevoir.py
linet-rpc2.py
ours-rpc.pdf
evoir.py
EVOIRE
.py
ssai
rojet-algo-repartie
rojet_algor-repartie.txt
EP
eprtoir
pcgen(1) RPC protocol compiler - Linux man page.htm
erie1AlgoRepa.pdf
erver.py
erver2.py
erveur-rpc-3client.py
ocket
es
HREAD.py
P 2 RPC.pdf
p algo repartie
p-groupe de thread warker
p-multethread
P-rpc.py
P1 SOCKET.pdf
p2
P5 - Systèmes Distribués - RPC.htm
P6RPC;PY
ptravail.py
```

CHOIX 2 : CRÉÉ UN FICHIER

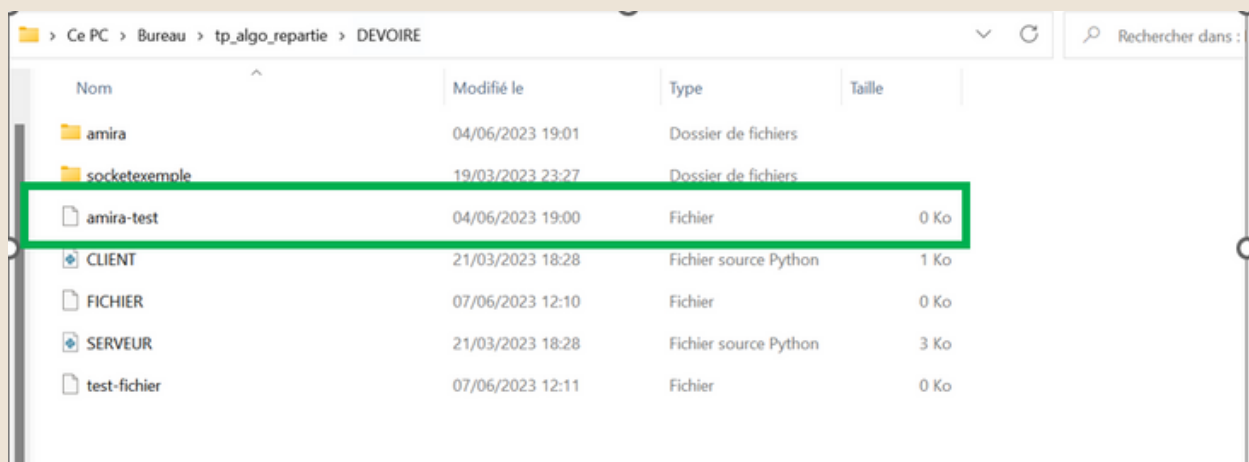
Menu:

1. Lister les fichiers/répertoires
2. Créer un fichier
3. Supprimer un fichier
4. Créer un répertoire
5. Supprimer un répertoire
6. Ajouter des données à un fichier
0. Quitter

Entrez votre choix : 2

Chemin pour créer un fichier : C:\Users\Amira\Desktop\tp_algo_repartie\DEVOIR

Et « test-fichier » est le nom du fichier créé

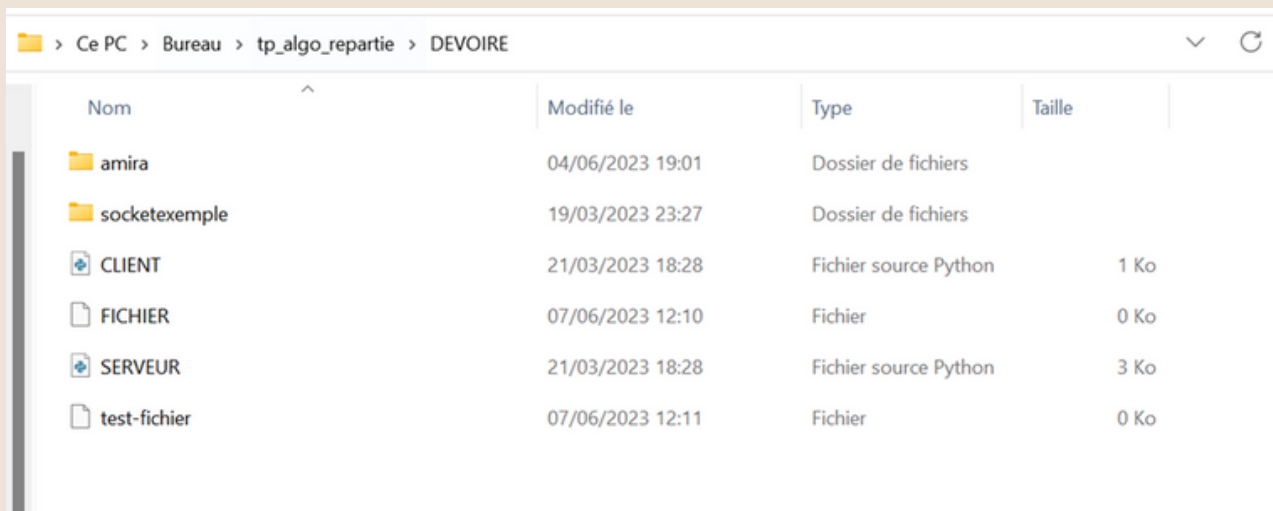


CHOIX 3 : SUPPRIMER UN FICHIER EXISTANT :

On va supprimer le fichier « amira-test »

```
Entrez votre choix : 3  
Chemin pour supprimer un fichier : C:\Users\Amira\Desktop\tp_algo_repartie\DEVOIRE\amira-test
```

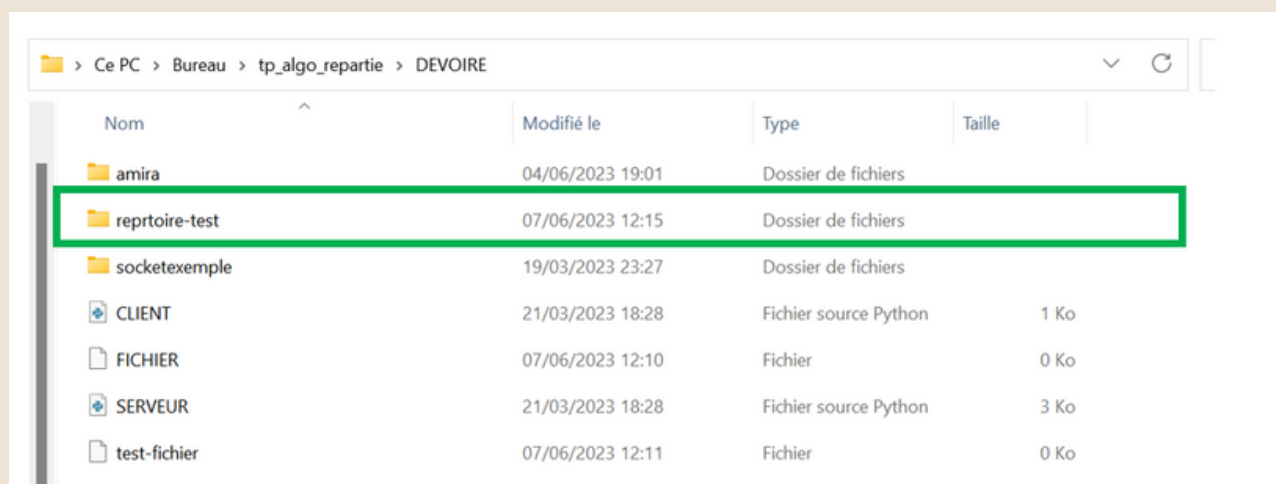
Résultat : Le fichier « amira-test » a été supprimé



Ce PC > Bureau > tp_algo_repartie > DEVOIRE				
Nom	Modifié le	Type	Taille	
amira	04/06/2023 19:01	Dossier de fichiers		
socketexemple	19/03/2023 23:27	Dossier de fichiers		
CLIENT	21/03/2023 18:28	Fichier source Python	1 Ko	
FICHIER	07/06/2023 12:10	Fichier	0 Ko	
SERVEUR	21/03/2023 18:28	Fichier source Python	3 Ko	
test-fichier	07/06/2023 12:11	Fichier	0 Ko	

CHOIX 4 : CRÉÉ UN RÉPERTOIRE

```
Entrez votre choix : 4
Chemin pour créer un répertoire : C:\Users\Amira\Desktop\tp_algo_repartie\DEVOIRE\reprtoire-test
Menu:
```

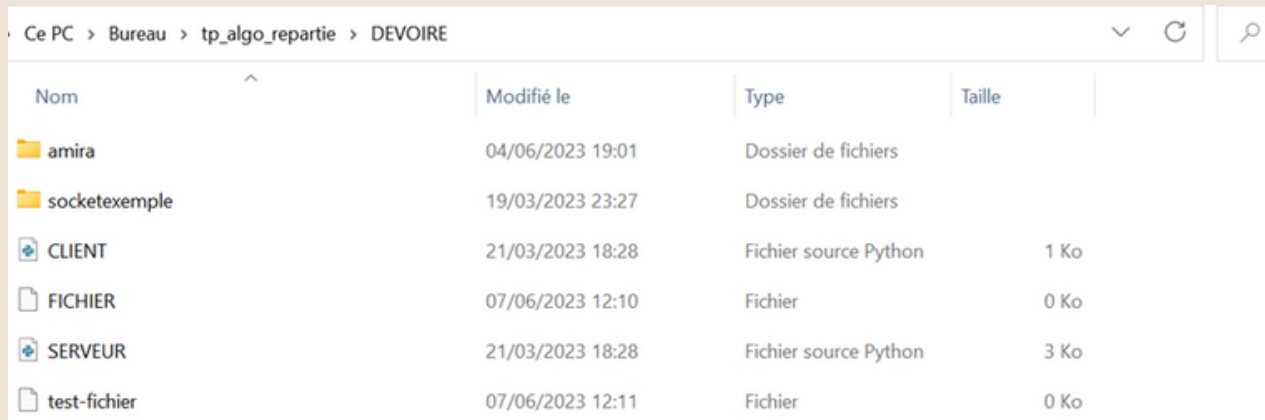


CHOIX 5= SUPPRIMER UN RÉPERTOIRE

On va choisit de supprimer le répertoire « reprtoire-test »

```
Entrez votre choix : 5
Chemin pour supprimer un répertoire : C:\Users\Amira\Desktop\tp_algo_repartie\DEVOIRE\reprtoire-test
Menu:
```

Résulta :



Ce PC > Bureau > tp_algo_repartie > DEVOIRE

Nom	Modifié le	Type	Taille
amira	04/06/2023 19:01	Dossier de fichiers	
socketexemple	19/03/2023 23:27	Dossier de fichiers	
CLIENT	21/03/2023 18:28	Fichier source Python	1 Ko
FICHER	07/06/2023 12:10	Fichier	0 Ko
SERVEUR	21/03/2023 18:28	Fichier source Python	3 Ko
test-fichier	07/06/2023 12:11	Fichier	0 Ko

CHOIX 6 : AJOUTER DES DONNER A UN FICHER

```
Entrez votre choix : 6
Chemin du fichier pour les données : C:\Users\Amira\Desktop\tp_algo_repartie\DEVOIRE\test-fichier
Données à ajouter : Le programme rpcgen(an RPC Protocol compiler) : C'est un outil qui permet aux programmeur d'écrire facilement des applications distribuées basées sur le mécanisme RPC en langage C. En gros, il permet de générer les éléments de l'architecture RPC
```

Résulta :

