

Debrief séance 2

UE12 P24 - Python

JSON

- **JSON** : format de données standard pour représenter des objets
- Ressemble aux dictionnaires et listes Python
- Tous les langages ont une bibliothèque permettant de lire et écrire du JSON
- On s'en est servi dans un fichier, mais on aurait aussi pu l'échanger sur le réseau
 - C'est le format le plus utilisé pour échanger des données entre 2 programmes sur internet

JSON - Exemple

Un objet JSON = un dictionnaire Python

```
1 {  
2     "id": 1,  
3     "name": "My username"  
4 }
```

Une liste JSON = une liste Python

```
1 ["Alice", "Bob", "Charlie"]
```

On les combine comme on veut

```
1 [  
2     {"id": 18, "name": "Alice"},  
3     {"id": 5, "name": "Bob"}  
4 ]
```

JSON - Lecture et écriture

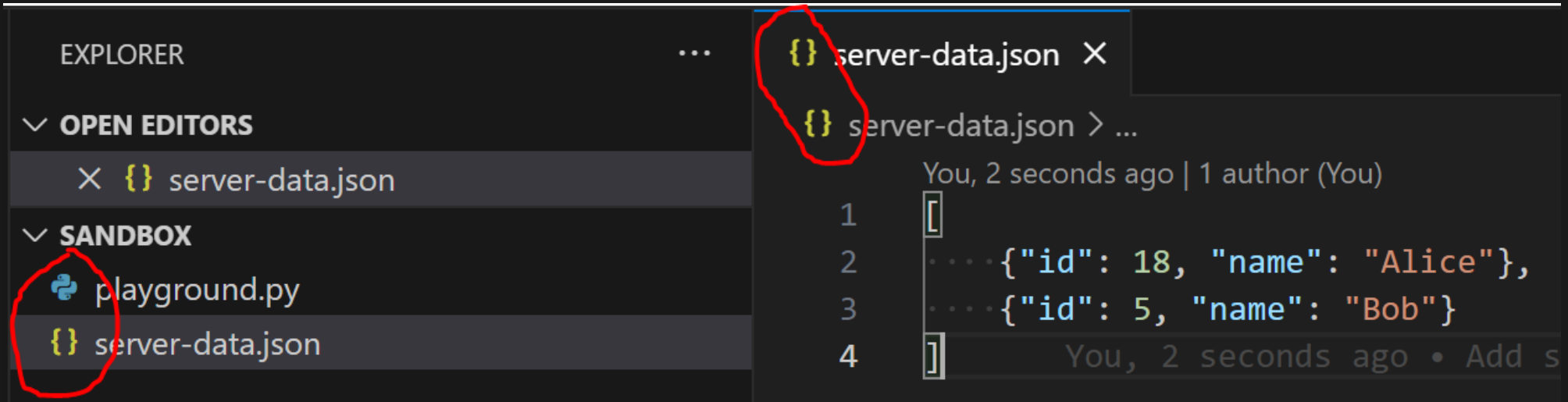
```
1 import json
2
3 JSON_FILE_NAME = 'data.json'
4
5 # Lecture
6 with open(JSON_FILE_NAME) as json_file:
7     data = json.load(json_file)
8
9 # Ecriture
10 with open(JSON_FILE_NAME, 'w') as json_file:
11     json.dump(data, json_file)
```

Warning

`json_file` n'est plus valide en dehors du bloc `with`

VS Code

VS Code reconnaît le format **JSON** :



et vous montre lorsque vous avez des erreurs :

EXPLORER


...

OPEN EDITORS

✕ {} server-data.json

1, U

SANDBOX

 playground.py

{} server-data.json

1, U

{} server-data.json 1, U ✕

{} server-data.json > ...

1 [

2{"id": 18, "name": "Alice"},

3{"id": 5, "name": "Bob"}, Trailing comma

4]

CLI (Command Line Interface)

Une commande peut prendre des arguments, sous forme courte (un tiret suivi d'une lettre) ou longue (2 tirets suivis d'un mot). En général, **-h** ou **--help** affiche de l'aide :

```
git --help  
git push --help
```

Git et GitHub

- Git : logiciel de versionnage de code, indépendant de GitHub
- GitHub : site d'hébergement de code
 - Permet de partager facilement votre code
 - Permet de collaborer avec d'autres développeurs
 - Utilise Git

Git et GitHub

- Git : je versionne mon code en local

```
1 git init
2 git add file.py
3 git commit -m "Add file.py"
```

- GitHub : je synchronise mon code local et le dépôt distant

```
1 git clone
2 git remote -v # Liste les dépôts distants connus
3 git pull
4 git push
```

Raccourcis utiles

Terminal

- **Ctrl + C** : interrompt le programme en cours
- **Ctrl + Maj + C** : copier
- **Ctrl + Maj + V** : coller

VS Code

- **F1** : ouvre le menu des actions

Python - Listes et itérations

```
1 users = [{'id': 7, 'name': 'Alice'}, {'id': 3, 'name': 'Bob'}]
```

Dans la mesure du possible, évitez les tournures avec **range** et **len** :

```
1 n = len(users)
2 for i in range(n):
3     print(users[i]['id'], users[i]['name'])
```

```
7 Alice
3 Bob
```

Préférer une itération directe :

```
1 for user in users:
2     print(user['id'], user['name'])
```

```
7 Alice
3 Bob
```

Python - enumerate

Si vous avez besoin de l'indice dans la liste, privilégiez **enumerate**:

```
1 for index, user in enumerate(users):  
2     print(index, user['id'], user['name'])
```

```
0 7 Alice  
1 3 Bob
```

Nommage de variables

Le nom d'une variable indique **ce que la variable représente**, pas **comment on le représente**

On écrit **for user in users** et non **for d in users**