Librairies

Installation

Base

```
pip install <librairie>
```

Environnement virtuel

Installer puis utiliser pipenv

Pycharm

- créer un environnement virtuel (proposé par Pycharm lors de la création du projet)
- A postériori aller dans « File / Settings... / Project / Python Interpreter » cliquer sur « + » puis taper le nom du module à rechercher puis en bas cliquer sur « Install Package »

Pipenv

Pipenv

Objet

Simplification dans l'utilisation des librairies et création automatique d'un environnement virtuel.

Utilisation

Installation Ubuntu

```
sudo apt install pipenv
```

Installation Redhat/CentOS

```
pip3 install pipenv
```

Installation Windows

- Installer pipenv

```
pip install pipenv
```

- Modifier la variable PATH pour ajouter les 2 chemins suivants :

```
C:\Users\<username>\AppData\Roaming\Python\Python38\Site-Packages
C:\Users\<username>\AppData\Roaming\Python\Python38\Scripts
```

Initialisation

Se positionner dans le répertoire du projet et lancer la commande :

```
pipenv install --python /usr/bin/python3.8
```

La commande va installer toutes les librairies nécessaires et créer les 2 fichiers suivants s'ils n'existent pas encore :

- Pipfile : contient les dépendances
- Pipfile.lock : contient les informations de version

Variables d'environnement

Possibilité de créer un fichier .env contenant les variables d'environnement pour l'exécution du programme. Par exemple pour positionner le niveau de log souhaité :

```
## LOG LEVEL ##
LOGURU_LEVEL=DEBUG
```

Exécution

```
pipenv run python <fichier.py>
```

Dépannage

Débordement de tempo

Si l'installation prend trop de temps elle peut terminer en échec notamment sur Raspberry PI ou bien en cas de connexion internet trop lente. Pour laisser plus de temps à l'installation il est possible de modifier 2 temporisation avant de lancer la commande PIPENV :

```
export PIPENV_INSTALL_TIMEOUT=9999
```

```
export PIPENV_TIMEOUT=999
```

Source

Tendo

Tendo

Objet

Ajout de fonctionnalités basiques.

Fonctions

SingleInstance

Pour s'assurer de n'avoir qu'une seule instance du programme.

```
from tendo import singleton

singleton.SingleInstance() # fera un sys.exit(-1) si une autre instance est en cours d'exécution
```

Loguru

Loguru

Objet

Librairies pour gérer simplement les logs du programme.

Utilisation

Fonctions

- debug : information détaillée uniquement pour diagnostiquer un problème
- info : confirmation du déroulement normal du programme
- warning : indication de quelque chose d'inattendu, mais le programme fonctionne normalement

- error : problème plus important, le programme n'a pas pu faire quelque chose
- critical : erreur sérieuse, indique que le programme risque de ne pas pouvoir poursuivre.

Niveau d'information

Chacune des fonctions correspond au niveau de log correspondant. Le niveau par défaut est positionné à `WARNING`, ce qui veut dire que seuls les messages à partir de ce niveau sont affichés (donc pas les niveau `DEBUG` et `INFO`).

Exemple

```
from loguru import logger

logger.setLevel(logging.WARNING)
logger.debug("Ce message ne s'affichera pas")
logger.info("Celui-ci non plus")
logger.warning("Premier message à s'afficher")
logger.error("Celui-ci aussi")
logger.critical("Ce dernier également")
```

Aiohttp

Aiohttp

Objet

Client/serveur http asynchrone.

Utilisation

Serveur

```
from aiohttp import web

async def handle(request):
    name = request.match_info.get('name', "Anonymous")
    text = "Hello, " + name
    return web.Response(text=text)

app = web.Application()
app.add_routes([web.get('/', handle),
                web.get('/{name}', handle)])
```

```
if __name__ == '__main__':
    web.run_app(app)
```

Client

```
import aiohttp
import asyncio

async def main():

    async with aiohttp.ClientSession() as session:
        async with session.get('http://python.org') as response:

            print("Status:", response.status)
            print("Content-type:", response.headers['content-type'])

            html = await response.text()
            print("Body:", html[:15], "...")

loop = asyncio.get_event_loop()
loop.run_until_complete(main())
```
