2N6 Programmation 2

avec python™

Les fonctions et modules

Modules

Nous avons vus comment utiliser des modules.

Maintenant nous allons voir comment en créer.

Utilisation de modules

```
> # Création de répertoires
> import os

> cours='420-2N6R'
> groupe='1070'
> os.makedirs(cours+'/'+groupe)
>
```

- Ici, on importe le module os.
- os.makedirs utilise la fonction makedirs() du module os

Utilisation de modules

```
# Fake Store REST API
import requests
import json

BASE_URL = 'https://fakestoreapi.com'


# faire une demande de tous les produits (GET)
response = requests.get(f"{BASE_URL}/products")
donne_req = respsonse.json()
print( json.dumps( donne_req, indent=4 ))
```

- Ici, on importe les modules requests et json
- requests.get utilise la fonction get() du module request
- Puis on utilise la méthode .json() de l'objet response du module requests
- Json.dumps utilise la fonction dumps() du module json

Modules

- Fichiers de code Python réutilisable dans différents programmes.
- Regroupe fonctions, variables et autres éléments ayant une fonctionnalité similaire.
- Code mieux organisé et plus facile à maintenir.
- Différentes parties du code séparées en modules distincts.
- Python possède la librairie standard qui regroupe de nombreux modules prédéfinis et prêts à l'emploi. Donc le code est déjà écrit et testé.

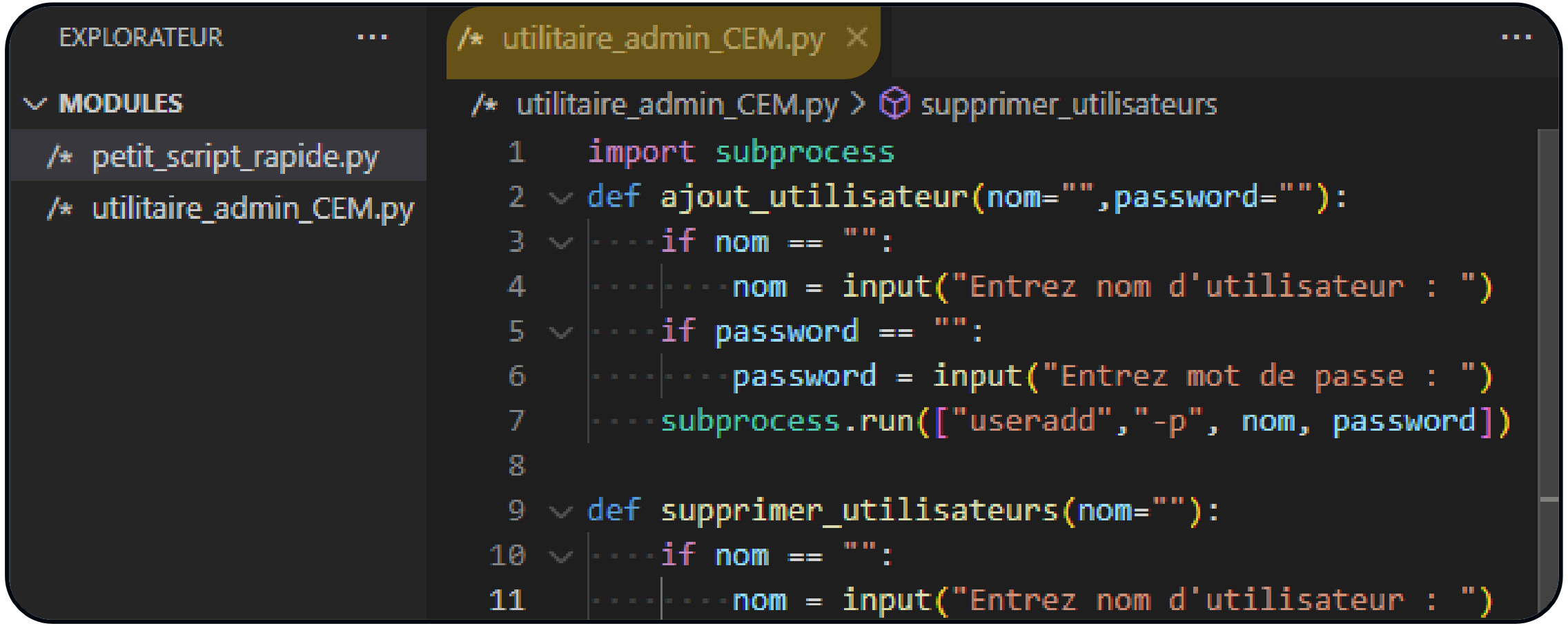
Modules qu'on a vu et utilisés

- **os** qui regroupe des fonctions pour le système d'opération
- **csv** qui regroupe des fonctions pour traiter les données d'un fichier csv
- **requests** qui regroupe des fonctions pour faire des requêtes http
- **json** qui regroupe des fonctions pour traiter ce type de données

Création d'un module

- Si on réutilise le même code dans plusieurs scriptes, on va faire un nouveau module afin de s'assurer de toujours utiliser exactement le même code.
- Vous pouvez donc créer votre propre module pour regrouper des fonctions sur un sujet.
- Créer un nouveau module est aussi simple que d'écrire un script

Modules


EXPLORATEUR
MODULES
petit_script_rapide.py
utilitaire_admin_CEM.py
utilitaire_admin_CEM.py
utilitaire_admin_CEM.py > supprimer_utilisateurs
1 import subprocess
2 def ajout_utilisateur(nom="",password=""):
3 if nom == "":
4 nom = input("Entrez nom d'utilisateur : ")
5 if password == "":
6 password = input("Entrez mot de passe : ")
7 subprocess.run(["useradd","-p", nom, password])
8
9 def supprimer_utilisateurs(nom=""):
10 if nom == "":
11 nom = input("Entrez nom d'utilisateur : ")

Importer le module et utiliser ses fonctions

petit_script_rapide.py

```python
#importe notre module fait maison
import utilitaire_admin_CEM
import csv
#ouvre un csv et exécute une fonction provenant du module
with open("liste_nouveaux_utilisateurs.csv","r",) as fichier_users:
    csv_read = csv.reader(fichier_users)
    for ligne in fichier_users:
        nom = ligne[0]
        mot_de_passe = ligne[1]
        utilitaire_admin_CEM.ajout_utilisateur(nom, mot_de_passe)
```

Importer le module

Utiliser une de ses fonctions

Tester les cas limites

- Quand vous écrivez une fonction, vous devez tester les cas limites pour vérifier que tout est ok.
- Les cas limites sont les valeurs qui pourraient causer des problèmes.
- Par exemple, si votre fonction prend en paramètre un entier entre 1 et 10, les cas limites seraient:
  - Passer 1 en paramètre
  - Passer 10 en paramètre
  - Passer -1 en paramètre
  - Passer 11 en paramètre
  - Ne rien passer en paramètre
  - Passer 'patate' en paramètre

Nous allons voir plus de possibilités dans ces cas plus tard, en utilisant les try...except

Tester les cas limites pour toutes les fonctions de vos modules

- Quand vous écrivez un module, c'est très important de tester les cas limites des fonctions qui sont dans vos modules.
- On s'attend à ce qu'un module ait été bien testé et qu'il soit vraiment fonctionnel.
- Si une fonction pose problème, cette erreur existera dans tous les scripts utilisant ce module.

Un fichier : module ET script

- Lorsqu'on importe un module, l'interprétateur pas au travers de chacune des lignes du module.
- Toutes les lignes seront exécutés.
- Pour utilisé un même fichier en module et script, il faut inclure les lignes a exécuter en script doivent être inclues dans un bloc conditionnel :

```
if __name__ == "__main__":
    
    while True:
        var = input("Entrez le nomb
        if var.isdigit() :
```