Python pour l'IA

TP1

Mme I.Yahiaoui

Exercice 1 : Quelques fonctions prédéfinies

- Créer une fonction anonyme qui reçoit des noms de dossiers et retourne le chemin complet (un path) .
- Utiliser la fonction « enumerate » pour énumérer les lettres de l'alphabet de 1 à 26.
- Utiliser la fonction « enumerate » pour écrire un script permettant transformer une phrase en CamelCase https://fr.wikipedia.org/wiki/Camel_case .
- Soit une liste comportant des noms de fichiers. Utiliser les fonctions « any » et « all » pour vérifier si la liste comporte au moins un fichier « jpeg » et si tous les fichiers sont des fichiers « pdf » .
- Utiliser la fonction « sorted », pour trier une liste en fonction de la longueur des chaînes de caractères qu'elle comporte.
- Utiliser la fonction « sorted », pour trier une liste de nombres en fonction du reste de la division de chaque élément par 7 (du plus grand au plus petit).
- Utiliser la fonction « zip » pour afficher la position dans l'alphabet de chaque lettre d'une variable contenant une chaine alphabétique.
- Utiliser la fonction « zip » pour créer un dictionnaire à partir de ces deux listes :

```
lst1=["CNN", "GAN" ,"RNN", "GRU", "LSTM"]
lst2=["Convolutional neural network", "Generative adversarial network", "Recurrent neural network", "Gated recurrent unit", "Long short-term memory"]
```

Exercice 2 : La fonction « map »

- Soit la liste [2.43, 9.05, 4.12, 3.77, 5.16, 1.81, 4.29]. Utiliser la fonction « map » pour imprimer le carré de chaque nombre arrondi à trois décimales.
- Soit une liste contenant des chaînes de caractères. En utilisant la fonction « map », créer une deuxième liste comportant les longueurs des éléments de la première.
- Soit la liste suivante :

```
Villes = ['Chalons_en_Champagne', 'Epernay', 'Rethel', 'Bezannes', 'Reims','Vitry_le_François', 'Cernay_les_Reims','Witry_les_Reims']
```

  Utiliser la fonction « map » pour créer une liste contenant le nombre d'occurrences de la voyelle « e » (e ou E) par nom de ville.
- Soient les deux listes suivantes :

```
lst1=[25, 1.35, 59, 402, 576, 43, 12.89, 1234]
lst2=[1,21,100,111,10020,67.56,34,145]
```

  Utiliser la fonction « map » pour créer une liste lst_max qui comporte la valeur max des deux listes à chaque position.

```
lst_max = [25, 21, 100, 402, 10020, 67.56, 34, 1234]
```

- Utiliser la fonction « map » pour créer une liste comportant la valeur max à une position donnée de deux listes l1, et l2.
- Soient les trois listes suivantes :

```
Civilite = ["M.", "Mme", "M.", "M.", "Mlle", "Mlle", "Mme"]
Noms_Famille = ["Bourcier", "Joly", "Bernard", "Durand", "Martin",
"Dubois", "Thomas"]
Prenoms = ["Pierre", "Martine", "Adam", "Gabriel", "Alice",
"Rose", "Emma"]
```

Utiliser la fonction « map » pour créer la liste Noms.

```
Noms = ['M. Pierre Bourcier', 'Mme Martine Joly', 'M. Adam Bernard',
'M. Gabriel Durand', 'Mlle Alice Martin', 'Mlle Rose Dubois', 'Mme
Emma Thomas']
```

Exercice 3 : La fonction « filter »

- Utiliser la fonction « filter » pour récupérer les nombres négatifs dans une liste.
- Utiliser la fonction « filter » pour récupérer tous les chiffres présents dans une chaîne de caractères.
- Utiliser la fonction « filtre » pour garder uniquement les noms de pays qui ont au plus six lettres.

```
pays = ["Allemagne", "Angola", "Bolivie", "Canada", "Chine",
"Cuba", "France", "Inde", "Mexique","Turquie", "Ukraine"]
```

- Utiliser la fonction « filter » pour récupérer toutes les voyelles d'une chaîne de caractères.
- Utiliser la fonction « filter » pour trouver les valeurs communes à deux listes de nombres.

Exercice 4 : La fonction « reduce »

- Utiliser la fonction « reduce » pour retourner la plus grande valeur dans une liste.
- Utiliser la fonction « reduce » pour calculer la factorielle d'un entier N saisi par l'utilisateur.
- Utiliser la fonction « reduce » pour reproduire le comportement de la fonction « join »
- Soient deux vecteurs mathématiques représentés par deux listes de nombres. Utiliser la fonction « reduce » pour calculer leur produit scalaire.

Exercice 5 : Itérateurs

- Plusieurs fonctions parmi celles utilisées dans les exercices précédents retournent des itérateurs. Remplacer le cast « list » ou « dict » par l'utilisation de la méthode « next » pour récupérer les résultats obtenus.
- Soit la liste suivante : lt = [(2,45,4,34) , (25,12) , (13,89,43,56,23,9,46,23) , (98,15,23) , (65,22,31,123,11)]
    - Utiliser la fonction map pour trouver la somme des nombres dans chaque tuple.
    - Utiliser l'itération manuelle pour imprimer les trois premiers résultats fournis par l'objet map résultant.
- Reprendre la première question en utilisant une expression génératrice.
- Écrire une classe d'itérateur « iter_inverse », qui prend une liste et l'itère dans le sens inverse.
- Écrire une classe d'itérateur « iter_melange », qui prend une chaine de caractères et retourne les éléments de cette chaine mélangée.

Exercice 6 : Les générateurs

- Créer un générateur qui génère les cubes de nombres jusqu'à un certain nombre N.

- Créer un générateur qui donne « n » nombres aléatoires entre un nombre de départ et un nombre d'arrivée (qui sont les paramètres d'entrée).
- Écrire un générateur qui donne « n » nombres aléatoires entre un nombre de départ et un nombre d'arrivée qui sont les paramètres de la fonction.
- Écrire un générateur qui prend une chaine de caractères et retourne les éléments de cette chaine mélangée.
- Écrire un générateur qui renvoie les éléments de la série de Fibonacci. Ces éléments sont calculés à l'aide de la formule suivante : Les deux premiers chiffres de la série sont toujours égaux à 1, et chaque nombre consécutif renvoyé est la somme des deux derniers chiffres.
- Écrire un générateur de nombres premiers nommé ***generateur_nombre_premier()*** ; Le générateur doit prendre une limite pour donner la taille maximale de la boucle que vous utilisez pour générer les nombres premiers. Vous devriez être en mesure d'exécuter le code suivant :

```
nombre = input('Merci de saisir un nombre:')

if nombre.isnumeric():

    num = int(nombre)

    if num <= 2:

         print('Le nombre doit être plus grand que 2')

    else:

         for premier in generateur_nombre_premier(num):

             print(premier, end=', ')

else:
    print('Le nombre doit être positif')
```

- Créer maintenant le générateur ***generateur_nombre_premier_infini*** (); ce générateur n'a pas de limite et continuera à générer des nombres premiers jusqu'à ce qu'il ne soit plus utilisé. Vous devriez être en mesure d'utiliser ce générateur de nombres premiers comme suit :

```
prime = generateur_nombre_premier_infini()

print(next(prime))

print(next(prime))

print(next(prime))

print(next(prime))

print(next(prime))
```