Série exercices révision

Exercice 1 :

Soit la suite définie par :

$$Un = \begin{cases} 1 & \text{si } n=0 \text{ ou } n=1 \\ 3*Un-1+Un-2 & \end{cases}$$

Ecrire une fonction récursive permettant de retourner le nième terme de la suite.

Ecrire une fonction itérative permettant de retourner le nième terme de la suite.

Exercice 2 :

Un entier est dit distinct s'il est composé de chiffres distincts (différents).

Ecrire une fonction python **Verification** qui permet de vérifier si l'entier n est distinct ou non en affichant un message.

Exemple:

N=1273 est dit distinct car il est formé par les chiffres 1, 2, 7 et 3 qui sont tous distincts, donc, le programme affichera: cet entier est distinct

N=1565 est dit non distinct car il est formé par les chiffres 1, 5, 6, 5 qui ne sont pas tous distincts (le chiffre 5 se répète deux fois), donc le programme affichera: cet entier est non distinct.

Exercice 3 :

On se propose d'écrire un programme Python permettant de déterminer et d'afficher un code à partir d'un entier N strictement positif et supérieur à 100, selon le principe suivant :

- Calculer le produit P des chiffres qui composent le nombre N.
- Recommencer le calcul du produit des chiffres du produit obtenu P tant que celui-ci n'est pas compris entre 0 et 9.
- Le code sera le nombre formé par N auquel on place à sa gauche le dernier produit obtenu.

Exemple :

Pour N=9867, le programme affichera : le code est 09867

En effet : Pour N=9867 :

- Le 1er produit P vaut 3024 (car 9*8*6*7=3024)

- Le 2ème produit P vaut 0 (car 3*0*2*4=0)

Etant donné que le dernier produit P, qui vaut 0, est compris entre 0 et 9, le code sera 09867

1. Ecrire une fonction python **Saisie** qui permet de saisir un entier n supérieur à 100.
2. Ecrire une fonction python **Prod** qui permet de calculer et de retourner le produit des chiffres d'un entier n.

3. Ecrire une fonction python **Result** qui, en utilisant la fonction Prod, permet de retourner le dernier produit calculé des chiffres d'un entier n. Le résultat retourné doit être un entier compris entre 0 et9.
4. Ecrire une fonction python **Insert** qui permet de saisir un entier n >100, d'insérer à sa gauche le chiffre obtenu par la fonction Result et d'afficher le résultat final.

Exercice 4:

1. Ecrire une fonction **Saisie_long** qui permet de saisir **n** la longueur de la liste L.
2. Ecrire une fonction **Remplir_L** qui permet de remplir la liste L avec n entiers positifs.
3. Ecrire une fonction **Rech1** qui cherche la longueur de la plus longue séquence des nombres identiques d'une liste **L** en commençant par une position **p**.

Exemple : pour L=[1,1,1,1,2,3,4,5,4,3,1] et p=0 : on obtient la valeur 4 qui représente la longueur de la séquence 1,1, 1, 1.

4. Donner la complexité de la fonction Rech1.
5. Ecrire une fonction **Rech2** qui à partir d'une liste **L**, retourne une liste composée de tuples chacun constitué de l'élément redondant et de la longueur de la séquence.

Exemple : Pour L= [1,1,1,2,2,2,2,2,4,3,1], on obtient A= [(1,3),(2,5),(4,1),(3,1),(1,1)]

6. Ecrire une fonction **Rech3** qui retourne le tuple contenant l'élément redondant et la longueur de la séquence la plus longue d'une liste **L**.

Exemple : Pour L=[1,1,1,**2,2,2,2,2**,4,3,1], on obtient (2,5).

7. Écrire une fonction **Aff** qui prend en paramètres une liste L des tuples de la question 5 et un tuple separator de taille 4. La fonction devra renvoyer une chaîne de caractères construite de la manière suivante :

   1. Elle commence par le premier élément de separator qui est '[' ;

   2. et pour chaque tuple de la liste, on ajoute à la chaine le premier élément du tuple puis le deuxième élément de separator qui est ' -> ' puis le deuxième élément du tuple puis le troisième élément de separator qui est ',' (sauf pour le dernier tuple);

   4. Le chaine se termine par le quatrième élément de separator qui est ']'.

   Exemple : >>> Aff([(1,3),(2,5),(4,1),(3,1),(1,1)], ('[', '->', ',' , ']')) affichera « [1->3,2->5,4->1,3->1,1->1] »