Série des exercices : Révision DS N°2

Exercice n° 1 : *« Le crible d'Eratosthène »*

Un nombre est dit premier, s'il admet exactement 2 diviseurs distincts (lui-même et l'unité). 1 n'est donc pas premier.

On désigne sous le nom de crible d'Eratosthène une méthode de recherche des nombres premiers plus petits qu'un entier naturel *n* donné.

Pour ceci, on écrit un tableau de tous les nombres jusqu'à *n* (sauf 1).

- On commence par éliminer tous les multiples de 2.
- Puis on fait de même avec 3.
- On choisit alors le nombre suivant non éliminé ici 5 et on élimine tous ses multiples.
- On réitère le procédé jusqu'à la partie entière de la racine de n.

Les nombres non éliminés sont les nombres premiers jusqu'à n.

Ecrire une fonction Python nommée ***Eratosthène*** permettant de déterminer et retourner un tableau formé des nombres premiers entre 2 et n en utilisant « Le crible d'Eratosthène ».

Exercice n° 2 : *« Conjecture de Syracuse »*.

Soit x_n la suite entière telle que :

$$\begin{cases} x_{n+1} = \dfrac{x_n}{2} & \text{Si } x_n \text{ est pair ;} \\ x_{n+1} = 3.x_n + 1 & \text{Si } x_n \text{ est impair} \end{cases}$$

La conjecture de Syracuse affirme que cette suite passe toujours par 1.

Ecrire une fonction Python nommée ***Syracuse*** permettant de calculer et renvoyer sous la forme d'un tableau à une dimension les termes de la suite x_n obtenue jusqu'à ce que la suite correspondante atteigne 1.

Le premier terme x_0 doit être passé en paramètre avec $x_0 = 500$ comme valeur par défaut.

Exercice n° 3 : *«lambda»*.

Créer une fonction lambda qui permet de calculer l'expression suivante : $f(x) = x^4 - 4x^3 - x^2 + 16x - 12$

Proposer une fonction utilisant une structure de données composée de Python (autre que les tableaux) qui permet de calculer toutes les valeurs pour la fonction **f** sur l'intervalle $[-5,5]$.

Exercice n° 4 : *«lambda, map(), filter()»*

Créer une fonction Python permettant de :

- Générer une liste L contenant 20 entiers compris entre -10 et 10. En utilisant la fonction random() du module random. La fonction random() renvoie un nombre dans l'intervalle [0, 1[
- Créer une liste L2 contenant seulement les éléments supérieurs à 5 de L.
- Pour chaque élément de L2, calculer son nombre d'occurrence dans L. Le résultat sera dans un tableau tab à deux dimensions.

Exemple:

L = [1, 5, 8, 3, 2, 7, 7, 8]

L2 = [5, 8, 7, 7, 8]

tab= [[5,1], [8, 2], [7, 2]]

Exercice n° 5 : *«Tableaux 2D»*

Créer une fonction Python qui prend en paramètre une sequence d'entiers s de type quelconque (listes, tuples,…) et un entier c >= 2 et qui transforme cette séquence en un tableau à deux dimensions de c colonnes.

Exercice n° 6 : *«Valeur approchée»*

On se propose de programmer quelques fonctions pour le calcul d'une valeur approchée de π.

1. Méthode de Viete

Soit la suite U définie par : $\begin{cases} U_0 = \dfrac{1}{\sqrt{2}} \\ U_n = \sqrt{\dfrac{1}{2} + \dfrac{1}{2}U_{n-1}} \end{cases}$ La valeur de π est donnée par : $\pi = \dfrac{2}{\prod_{k=0}^{\infty} U_k}$

2. Méthode d'Euler
La valeur de π est donnée par : $\pi = 4 + \sum_{k=1}^{\infty} \frac{8}{1-16k^2}$

Travail demandé :

1. Ecrire la commande Python donnant une valeur approximative de π avec 20 chiffres significatifs.

2. Ecrire une fonction Python, nommée Viete, calculant à l'aide de la méthode de Viete une valeur approchée de π avec une précision ε donnée. La valeur approchée de π est atteinte lorsque la différence en valeur absolue entre deux termes successifs est inférieure à ε.

3. Ecrire une fonction Python, nommée Euler, retournant une valeur approchée de π avec une précision ε donnée. La valeur approchée de π est atteinte lorsque la différence en valeur absolue entre les valeurs obtenues suite à l'exécution de deux itérations successives est inférieure à ε.

Exercice n° 7 : *«Calcul polynômes»*

Soit n un entier naturel, on définit la suite polynomiale d'ordre n, par la relation de récurrence suivante :

$$\begin{cases} P_0(t) = 1 \\ P_1(t) = t \\ P_n(t) = \frac{2n-1}{n} t\, P_{n-1}(t) - \frac{n-1}{n} P_{n-2}(t) & \text{Pour } n > 1 \end{cases}$$

Ecrire une fonction Python nommée ***Legendre*** qui prend comme argument n, et retourne comme résultat le polynôme $P_n(t)$.

Exercice n° 8 : *«lambda»*
Ecrire un script Python qui permet de :
1- Saisir un entier **m** supérieur à 5.
2- Saisir deux listes **a** et **b** contenant **m** entiers, tel que les listes **a** et **b** ne présentent aucun élément en commun.
3- Définir la fonction f suivante : $f(x) = \sum_{n=1}^{m} (a_n \cos(n\pi x) + b_n \sin(n\pi x))$ en utilisant l'expression lambda
4- Créer une séquence S contenant les résultats de f(x), pour x entre 0 et 5 ;

Exercice n° 9 : *«chaines de caractères, try»*
Le but de cet exercice est de créer un script Python qui saisit une chaîne d'ADN valide et une séquence d'ADN valide et afficher la proportion de séquence dans la chaîne.

Chaine « valide » signifie qu'elle n'est pas vide et formée exclusivement d'une combinaison arbitraire de "a", "t", "g" ou "c".

1. Écrire une fonction python **valide** qui renvoie vrai si la saisie d'une chaine d'ADN donnée est valide, faux sinon.
2. Écrire une fonction python **saisie** qui effectue une saisie valide d'une chaine et renvoie la valeur saisie sous forme d'une chaîne de caractères.
3. Écrire une fonction python **proportion** qui reçoit deux arguments, la chaîne **ch** et la séquence **s** et qui retourne la proportion de séquence dans la chaîne, avec :
   Proportion de s dans ch = (nombre d'occurrences de s dans ch * 100.0) /nombre de caractères de la chaine.
4. Le programme principal appelle la fonction saisie pour la chaîne et pour la séquence et affiche le résultat.
   Exemple d'affichage : Il y a 13.33 % de "ca" dans votre chaîne.