

EXAMEN DE FIN DE SEMESTRE1

Matière : INFORMATIQUE

Classes : PB1

Durée : 2h

EXERCICE 1 :

Calcul de la racine cubique d'un réel (**X**) donné au moyen de la suite convergente suivante :

$$U_0 = 1$$

$$U_{n+1} = (2U_n + X/U_n^2)/3$$

On arrête les calculs lorsque l'erreur relative entre 2 termes successifs est inférieure à ϵ nombre infinitésimal donné, on dit alors que U_n est la **racine cubique** de **X** à ϵ près.

Condition de convergence : $|U_{n+1} - U_n|/U_n < \epsilon$

Ecrire un programme **Python** permettant de :

- saisir un réel **X** quelconque
- saisir un réel positif ϵ (**EPS**) tel que $10^{-8} < \epsilon < 10^{-5}$
- calculer la racine cubique au moyen de la suite ci-dessus
- afficher le résultat

EXERCICE 2 :

Soit **N** et **M** sont deux entiers non nuls.

On considère l'ensemble des couples (**xi**, **yi**), où **xi** est un entier compris entre 1 et **N** et **yi** un entier compris entre 1 et **M** ;

Ecrire un Programme **Python** qui:

1- Permet de saisir les entiers **N** et **M**.

2- Permet, en utilisant **deux boucles imbriquées** :

- d'afficher tous les **couples (xi, yi)** selon l'exemple suivant, qui correspond à **N = 3** et **M = 2** :

(1 , 1)

(1 , 2)

(2 , 1)

(2 , 2)

(3 , 1)

(3 , 2)

- de créer et d'afficher trois couples contenant la somme, le produit et la moyenne des **xi** et des **yi**.

Ce qui correspond, pour **N = 3** et **M = 2**, à l'affichage suivant:

Le couple Somme est: (12 , 9)

Le couple Produit est : (36 , 8)

Le couple Moyenne est: (4.0, 4.5) **Rq** : (4.0=12/3 et 4.5=9/2)

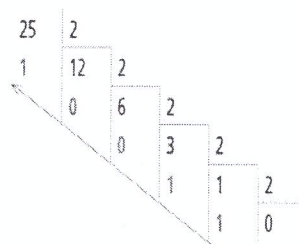
EXERCICE 3 :

Les nombres que nous utilisons habituellement sont ceux de la **base 10** (système décimal). Le binaire est le mode de comptage non plus en base 10 mais en **base 2**. Il est utilisé par les ordinateurs, car les machines ne peuvent comparer que deux valeurs : des 1 et des 0. On rappelle le principe de conversion décimal-binaire et celui binaire-décimal :

I- Conversion décimal-binaire :

- On a notre nombre en décimal.
- On le divise par 2 et on note le reste de la division (c'est soit un **1** soit un **0**).
- On refait la même chose avec le quotient précédent, et on met de nouveau le reste de côté.
- On répète la division, et ce jusqu'à ce que le quotient est **0**.
- Le nombre en binaire apparaît : il faut lire de bas en haut **les restes des divisions** qui représentent les **bits** du mot binaire. Le premier bit à placer est celui de poids fort (MSB) du gauche vers la droite jusqu'à le bit du poids faible (LSB).

Exemple : $(25)_{10} = (11001)_2$



II- Conversion binaire-décimal : Pour convertir un mot binaire en décimal, on procède de la manière suivante : on multiplie par 2^0 la valeur du rang 0, par 2^1 la valeur du rang 1, par 2^2 la valeur du rang 2, etc... et on fait la somme.

$$N = \sum_{i=0}^{n-1} a_i 2^i$$

Avec a_i sont les **bits** du mot binaire.

Travail demandé : on désire exprimer les deux sens de codage par un programme.

Ecrire un programme **Python** qui permet de :

- 1- saisir un entier $N > 0$.
- 2- créer et afficher une liste **binaire**. Cette liste sera remplie par les **bits** de la représentation binaire de N qui sont calculés selon la méthode donnée.

Exemple : si $N=25$ alors la liste **binaire** = $[1,1,0,0,1]$.

Indication : vous pouvez utiliser l'une des deux commandes suivantes :

- **append** qui permet d'ajouter un élément **à la fin** de la liste.
 - **insert** qui permet d'ajouter un élément à une **position donnée**.
- 3- déterminer, **à partir de la liste binaire obtenue**, si le nombre est pair ou impair.
 - 4- calculer et afficher le nombre de **0**, et le nombre de **1** dans la liste **binaire**.
 - 5- effectuer le codage **binaire décimal** : on suppose maintenant que la liste **binaire** est donnée. Calculer et afficher la valeur décimale correspondante aux **bits** de la liste **binaire** selon la méthode décrite.

Exemple : si **binaire** = $[1,1,0,0,1]$ alors $N=25$