***Série de révision Python – Exercices extraits de concours***

## Problème 1 :

Un nombre premier est un entier qui n'est divisible que par 1 et par lui-même. On se propose d'écrire un programme qui établit la liste de tous les nombres premiers compris entre 1 et 100, en utilisant ***la méthode du crible d'Eratosthène*** :

- Créer une liste de 100 éléments, chacun initialisé à la valeur 1,
- Parcourir cette liste à partir de l'élément d'indice 2: si l'élément analysé possède la valeur une, mettez à zéro tous les autres éléments de la liste, dont les indices sont les multiples de l'indice auquel vous êtes arrivé.

Lorsque vous aurez parcouru ainsi, toute la liste les indices des éléments qui seront restés à 1 seront les nombres premiers recherchés.

En effet : à partir de l'indice 2, vous annulez tous les éléments d'indices pairs 4, 6,8…Avec l'indice 3, vous annulez les éléments d'indices 6,9….et ainsi de suite. Seul resteront à 1 les éléments dont les indices sont effectivement des nombres premiers.

1. Ecrire une fonction **init** permettant d'initialiser une liste de 100 éléments à un.
2. Ecrire une fonction **multiple** permettant de mettre à zéro les éléments dont l'indice est un multiple d'un entier donné comme paramètre.
3. Ecrire une fonction **suivant** qui à partir d'un indice donné retourne le premier indice dont l'élément correspondant est différent de zéro.
4. Ecrire une fonction **crible** permettant d'appliquer la méthode présentée par la crible d'Eratosthène pour retourner une liste qui contient les nombres premiers (il suffit d'étirer jusqu'à la racine entière de 100).
5. Ecrire une fonction récursive **crible_recursive** permettant d'appliquer la méthode présentée pour retourner la liste initiale après les modifications.
6. Donner le schéma d'exécution de la fonction précédente (si on veut déterminer les entiers premiers qui sont <10).
7. Ecrire un programme qui fait appel aux fonctions déjà définies afin d'afficher les entiers premiers compris entre 1 et 100.

## Problème 2 :

On définit une suite de polynôme SP par :

$$\begin{cases} SP_0(x) = P(x) \\ SP_1(x) = -P'(x) \\ SP_{k+2}(x) = Q(x) * SP_{k+1}(x) + SP_k(x) \end{cases}$$

Avec : P(x) un polynôme de degré n (n≠0) tel que : $P(x)=\sum_{i=0}^{n} a_i x^i$

$P'(x)$ le polynôme dérivé de degré n-1 tel que : $P'(x)=\sum_{i=0}^{n-1}(i+1)a_{i+1}x^i$

Q(x) est le produit des deux polynômes $SP_{k+1}(x) et\ SP_k(x)$

Tout polynôme de degré « n » sera représenté sous forme d'une liste de taille « n+1 » tel que les coefficients sont les éléments et les degrés sont les indices

**Exemple :** La liste correspondante au polynôme P(x)=1+x-5$x^2$+$x^3$ − 2$x^4$+6$x^6$
est : L=[1,1,-5,1,-2,0,6]

On désire déterminer à partir de la suite ci-dessus le polynôme $SP_m$(x) avec m>=2.

1. Ecrire une fonction appelée **saisie_deg** qui permet de saisir un entier strictement positif.
2. Ecrire une fonction appelée **saisie_poly** permettant de saisir la représentation d'un polynôme P(x) de degré « n » dans une liste.
3. Ecrire une fonction appelée **derive** permettant de déterminer la représentation du polynôme dérivé d'un polynôme, représenté par une liste, le résultat sera une liste.
4. Ecrire une fonction appelée **opp_poly** permettant de déterminer la représentation du polynôme opposé, représenté par une liste, le résultat sera une liste.
5. Ecrire une fonction appelée **add_poly** qui prend comme paramètres deux polynômes P1 et P2 ainsi que leurs degrés et retourne la représentation du polynôme P1+P2.
6. Ecrire une fonction appelée **mul_poly** qui prend comme paramètres deux polynômes P1 et P2 ainsi que leurs degrés et retourne la représentation du polynôme P1*P2
   Sachant que si P1(x)=$\sum_{i=0}^{n1} a_i x^i$ et P2(x)=$\sum_{i=0}^{n2} b_i x^i$

   Alors le produit est P1(x)* P2(x)= $\sum_{k=0}^{n1+n2} c_k x^k$ avec $c_k = \sum_{i+j=k} a_i b_j$

7. Ecrire le programme principal permettant de :

   - saisir le degré « n » d'un polynôme
   - saisir la liste qui représente le polynôme P
   - déterminer et afficher la liste D relative à la dérivée du polynôme P
   - déterminer et afficher la liste O relative à l'opposé du polynôme P
   - déterminer et afficher la liste A relative à l'addition de P et sa dérivée
   - déterminer et afficher la liste M relative à la multiplication de P avec sa dérivée

## Problème 3 :

L'objectif de ce problème est d'écrire un programme de gestion de livres d'une bibliothèque (saisie, emprunt, affichage…)

Chaque livre est caractérisé par un numéro un titre et un nombre d'exemplaires. Les caractéristiques relatives à tous les livres sont représentées par les trois listes suivantes :

- **Lnum** pour tous les numéros des livres [num1,num2,….]
- **Ltitre** pour tous les titres des livres [T1,T2,……]
- **Lexp** pour tous les nombres des exemplaires [nb1, nb2,…..]

Le livre dont le numéro est Lnum[k] a pour titre Ltitre[k] et un nombre d'exemplaires disponibles Lexp[k].

Chaque numéro de livre et nombre d'exemplaires est un entier strictement positif chaque titre est une chaîne de caractères.

Pour la gestion des emprunts des livres, on utilise la liste de listes **LE** où chaque sous_liste correspond à un emprunt chaque sous_liste contient sept éléments selon cet ordre :

- **CIN** : un entier strictement positif composé de 8 chiffres
- **Num** : un numéro de livre qui doit correspondre à un numéro de livre existant dans la bibliothèque
- **JE** : le jour de l'emprunt
- **ME** : le mois de l'emprunt
- **JR** : le jour de retour prévu après 10 jours de la date de l'emprunt
- **MR** : le mois de retour
- **P** : indique si un emprunteur est pénalisé (P prend la valeur 1) ou non (P prend la valeur 0)

Pour la gestion des pénalités, on utilise une liste **LP** contenant les identifiant des emprunteurs pénalisés ayant rendu leurs livres.

**Hypothèse de travail :**

- Un emprunteur n'a pas le droit d'emprunter plus qu'un livre à la fois
- Un emprunteur pénalisé n'a pas le droit à un nouvel emprunt
- On suppose que l'année est non bissextile (février contient 28 jours)

**Travail demandé**

1. Ecrire une fonction **Saisie** permettant de saisir un entier strictement positif.
2. Ecrire une fonction **ajout_titre** qui saisit un numéro de livre à ajouter, son titre et le nombre des exemplaires en faisant les contrôles nécessaires.
3. Ecrire une fonction **etat_biblio**, qui affiche pour chaque livre son titre ainsi que le nombre d'exemplaires restants.
4. Ecrire une fonction **aut_emprunt** qui à partir du CIN d'un étudiant, vérifie s'il est autorisé d'emprunter un livre.
5. Ecrire une fonction **calcul_date** qui calcule la date de retour prévue.
6. Ecrire une fonction **emprunt** qui saisit le CIN de l'emprunteur et vérifier s'il est autorisé à emprunter, dans le cas de non pénalité saisir le numéro du livre. si le livre est disponible en stock on doit saisir la date de l'emprunt, réduire le stock et mettre à jour la liste des emprunts.
7. Ecrire une fonction **MAJ_penalite** qui à partir d'une date donnée met à jour la liste des emprunts ayant dépassé cette date limite de retour puis fait la mise à jour de la liste des pénalités en y ajoutant les CIN des emprunteurs pénalisés.
8. Ecrire une fonction **retour** permettant de faire les mises à jour nécessaire en cas de retour d'un livre.