Comme d'habitude, le TD doit commencer par la copie des fichiers python fournis dans votre dossier personnel sur le réseau. Vous veillerez à bien tester chaque fonction créée.

I Polynômes

1.1 Encodage

Comme expliqué dans le devoir sur table, un polynôme $\mathrm{P(X)} = a_0 + a_1\mathrm{X} + \cdots + a_n\mathrm{X}^n$ est encodé en python par la liste $[a_0, \ldots, a_n]$ de longueur n + 1.

Exercice 1
Compléter les fonctions **zeros** et **elimine_zero** du fichier *polynome.py*.
Donner la raison de la présence du paramètre eps dans cette deuxième fonction.

1.2 Opérations

Exercice 2
Compléter les fonctions **somme** et **soustrait**.

On prendra garde au fait que ces opérations peuvent faire apparaître des coefficients nuls à la fin de nos polynôme. On souhaite les éliminer à chaque fois.

Exercice 3
On pose $\mathrm{P(X)} = \sum_{k=0}^{n} a_k\mathrm{X}^k$ et $\mathrm{Q} = \sum_{i=0}^{m} b_i\mathrm{X}^i$.

1. Ecrire sous forme d'une somme double (au brouillon) le produit $\mathrm{C(X)} = \mathrm{P(X)Q(X)} = \sum_{j=0}^{p} c_k\mathrm{X}^k$. Exprimer également p en fonction de n et m.
   Qu'en déduire pour la longueur de la liste représentant C(X) ?
2. Pour mémoire : à quel indice dans la liste représentant C est stocké le coefficient de X^{k+i} ?
3. Compléter la fonction produit en utilisant l'idée suivante : on commence par créer une liste C remplie de 0, et on ajoute pour chaque valeurs de k et i le produit $a_k b_i$ à l'indice convenable.

1.3 Evaluation

Le but est ici de calculer $\mathrm{P}(x_0)$ de manière rapide pour un polynôme donné P et un nombre x_0.

Exercice 4
Compléter la fonction `evalue` qui retourne la valeur $\mathrm{P}(x_0)$ étant donné la liste représentant P et la nombre x_0. On prendra bien garde à ne pas calculer x_0^k pour toutes les valeurs de k utiles, mais plutôt d'utiliser une variable locale contenant les valeurs successives des puissances de x_0.

Exercice 5
Pour **tester** la fonction précédente [1], tracer la courbes représentative de sin et celle de $x \mapsto x - \frac{x^3}{6} + \frac{x^5}{120}$ sur l'intervalle $[-\pi, \pi]$ et sur un même graphique.
Tracer ensuite, et toujours sur le même graphique la courbe représentative de $x \mapsto x - \frac{x^3}{6} + \frac{x^5}{120} - \frac{x^7}{5040}$. Les courbes doivent être presque confondues.
Les coefficients mystère : calculer, à l'aide de python s'il le faut, les nombres : $3!, 5!, 7!$.

II Tableaux triés

Le cadre cette fois est différent : on dispose d'un tableau T de nombres que l'on souhaite trier dans l'ordre croissant.

2.1 Algorithme

On note n la longueur de T. Une première idée peut-être la suivante (tri par sélection) :

1. Trouver l'indice i_0 du plus grand élément de T et échanger $\mathrm{T}[n-1]$ et $\mathrm{T}[i_0]$
2. Trouver l'indice $i_0$du plus grand élément de $\mathrm{T}[0 : n-1]$ (les n-1 derniers éléments) et échanger $\mathrm{T}[n-1]$ et $\mathrm{T}[i_0]$.
3. ...
4. Trouver l'indice i_0 du plus grand élément de $\mathrm{T}[0 : 2]$ et échanger $\mathrm{T}[i_0]$ et T[1].

Un problème de cette méthode est qu'elle ne conserve pas le tableau d'origine. Nous allons donc devoir le copier.

1. Toute réponse qui n'utilise pas la fonction `evalue` est donc irrecevable

2.1.1 En route

1. Créer un fichier **tri.py**.
2. Ecrire une fonction **copie** qui prend un tableau (ou une liste) en entrée et retourne une copie de cette liste.
3. Ecrire une fonction **echange** qui échange deux éléments d'indices donnés dans une liste.
4. Ecrire une fonction **maxi** à 3 paramètres : une liste L et deux indices j_0, j_1 et qui retourne l'indice du plus grand élément de la liste considérée entre les indices j_0 et j_1 (compris).
5. Implémenter enfin la fonction **tri** qui retourne une copie triée de la liste passée en paramètre.

2.2 Pour aller plus loin

Concevoir et implémenter un autre algorithme de tri de `L` sur le principe suivant :

- Pour k allant de 1 à `len(L) - 1` :
- par des échanges successifs, on place `L[k]` par échange successif avec les éléments précédents de telle manière que `L[0:k]` soit triée après les échanges nécessaires.