TP ÉVALUÉ D'INFORMATIQUE DU 18/12/2025

Instructions :

- Ouvrir le fichier à compléter TP_eval_nom.py (diaxens/PTSI2_Informatique/TP_evalue_S1) et renommez-le avec votre nom (par exemple TP_eval_Duval.py)
- Les temps indicatifs des **4 exercices** à traiter sont indiqués (il se peut que ce soit optimiste).
- Aucun document, numérique ou papier (hormis le sujet) n'est autorisé.
- En fin de séance, vous déposerez votre travail sur le dossier de dépôt moodle créé à cet effet. Ne pas quitter la salle sans que votrex professeur ait vérifié que le travail est bien déposé.
- Pour une grande partie des questions, le test pour vérifier votre travail est déjà proposé en commentaire. Décommenter pour tester.
- Si vous bloquez sur une fonction, considérez pour la suite qu'elle est faite. La grande majorité des questions est indépendante, profitez-en.

Exercice 1. Manipulation des listes (≈20 min)

Travail 1. Créer la liste de flottants `L=[5.0,5.2,5.4,…,9.8,10.0]` à l'aide d'une boucle `while`, en partant d'une liste vide.

Travail 2. Définir une fonction `moy_int(L)` qui prend en argument une liste **L** quelconque (composée d'éléments de tous types) et qui retourne la moyenne des entiers de cette liste. S'il n'y a pas d'entier, la fonction doit renvoyer le booléen `False`.

Par exemple, `moy_int([1,7.1,"a",0,8.0,9,"12"])` doit renvoyer **3.3333**.

Aide : Pour une variable `a`, l'instruction `type(a)` renvoie `int` si `a` est de type entier

Travail 3. Décommenter puis compléter la fonction `rech_dicho(L,val)` au niveau des '...' qui prend en argument une liste triée `L` et un entier `val`, et qui renvoie `True` si val est dans `L`, en effectuant une recherche dichotomique.

Exercice 2. Calcul des points au Scrabble (≈15 min environ)

On s'intéresse ici au jeu du SCRABBLE. Le dictionnaire donné dans le fichier à compléter `Dpoints` contient les points accordés au SCRABBLE pour chaque lettre de l'alphabet.

Travail 1. Deux erreurs se sont glissées dans ce dictionnaire. La lettre Z a été oubliée (elle vaut 10 points), et la lettre H vaut 4 points (au lieu de 3). Corriger cela en une instruction supplémentaire 'courte' pour chaque erreur. On s'interdira de recréer le dictionnaire entier, on apportera **juste une modification** du dictionnaire `Dpoints` déjà créé.

Travail 2. Définir une fonction `points(mot)` qui prend en argument une chaîne de caractères mot (uniquement constituée de lettres majuscules), et qui renvoie le nombre de points accordés pour ce mot. Aucun bonus particulier n'est à prendre en compte. Par exemple `points('CONCATENEZ')` doit renvoyer 23, ce qui correspond à la somme des points associés aux lettres du mot.

Afin de rendre son nombre de points attribués plus conséquent, un joueur indélicat souhaite augmenter de 1 tous les points associés à chaque lettre.

Travail 3. Définir les instructions modifiant `Dpoints` comme le souhaite le joueur indélicat. On impose d'utiliser une boucle **for** pour balayer tous les éléments du dictionnaire pour incrémenter de 1 le score de chaque lettre. Vérifier le résultat en s'assurant que `points('CONCATENEZ')` doit maintenant renvoyer 33.

Exercice 3. Etude de polynômes (≈35 min environ)

A un polynôme P de degré $n \in \mathbb{N}$ à coefficients réels est associée la liste de taille n+1 de ses coefficients comme suit :

- $1 - 2x + 3x^2$ est représenté par [1,−2, 3]
- $5 - 2x + x^4$ est représenté par [5,−2, 0, 0, 1]

Par ailleurs, on supposera toujours que le dernier élément de la liste est non nul. On confondra désormais un polynôme et la liste de ses coefficients.

Travail 1. Écrire la fonction `degre(P)` qui prend en argument un polynôme P et qui renvoie son degré. Par exemple, `degre([1,-2,3])` doit renvoyer 2.

Travail 2. Écrire une fonction `eval(P,x)` qui renvoie la valeur P(x). Par exemple, `eval([1,-2,3],5)` doit renvoyer **66**.

Travail 3. Écrire une fonction **`deriv(P)`** qui renvoie le polynome dérivée correpsondant. Par exemple, **`eval([1,-2,3])`** doit renvoyer **`[-2,6]`** car la dérivée de $1 - 2x + 3x^2$ est $-2 + 6x$

Travail 4. Écrire une fonction **`LxLy(P,a,b,n)`** qui renvoie les deux listes suivantes :

- **`[a,a+pas,a+2pas,…,a+(n-2)pas,a+(n-1)pas]`** la liste des n valeurs d'abscisses x comprises entre a et b inclus car $a + (n-1)pas = b$.
- **`[P(a),P(a+pas),P(a+2pas),…,P(a+(n-2)pas),P(a+(n-1)pas)]`** la liste des n valeurs correspondantes du polynôme P.

Aide : $pas = \frac{b-a}{n-1}$

Travail 5. Écrire la fonction **`trace(P,a,b,n)`** qui donne la représentation graphique de P(x) pour $x \in [a, b]$, en définissant *n* points. On utilisera la fonction **`LxLy(P,a,b,n)`.** On pourra s'aider de l'annexe.

Un projectile en chute libre, avec une impulsion initiale, a une évolution en altitude *h(t)* du type :

$$h(t) = at^2 + bt + c$$

avec $a = -\frac{9.81}{2}$, *b = 15* et *c = 5* en unités SI (mètres, secondes …)

Travail 6. Tracer alors l'évolution de l'altitude du projectile pour un temps compris entre 0 et 4 secondes, en prenant 500 points. On modifiera la fonction **`trace()`** de manière à obtenir un tracé « scientifiquement » acceptable, c.a.d. contenant des axes documentés avec unités, et un titre.

Exercice 4. Comptage et tri de listes (≈20 min environ)

On cherche à trier une liste L d'entiers naturels strictement inférieurs à un entier n donné. Pour la méthode proposée, il est tout d'abord nécessaire de créer une fonction **comptage(L,n)**, d'arguments L et n, renvoyant une liste de n éléments dont le k-ième élément désigne le nombre d'occurrences de l'entier k dans la liste L.
Par exemple, **comptage([0,2,3,2,0,2],4)** doit renvoyer la liste **[2,0,3,1]**, car il y a 2 fois la valeur 0, 0 fois la valeur 1, 3 fois la valeur 2 et 1 fois la valeur 3.

Travail 1. Coder la fonction **comptage(L,n)**.

Travail 2. Décommentez puis compléter la fonction **tri_comptage (L, n)**, qui renvoie une liste **Ltriee**, contenant les valeurs de L, triées dans l'ordre croissant. Il faut bien sûr supprimer les « … » pour que cela fonctionne. On s'interdira évidemment d'utiliser les fonctions pythons de tri déjà codées (par exemple, la méthode **.sort()**) et on s'imposera de faire appel à la fonction **comptage** précédente. Tester le résultat avec l'instruction proposée,

Travail 3. Créer une fonction **`Ltest(toto,n)`** qui renvoie une liste d'entiers, de taille **`toto`**. Chaque entier doit être compris aléatoirement entre 0 et n-1 inclus. On pourra utiliser la méthode **`randint(a,b)`** de la bibliothèque **`random`** (déjà importée) qui renvoie un entier aléatoire compris entre les entiers **a** et **b** inclus.

```
print(« FIN DE L'ENONCE »)
```

Annexe MATPLOTLIB

La librairie *matplotlib* permet de tracer des nuages de points dont les abscisses et ordonnées seront données sous la forme de listes.

1) Écrire en en-tête de programme la commande permettant d'importer la librairie :
   import matplotlib.pyplot as plt
   L'utilisation de la librairie se fera donc grâce à l'instance de classe nommée plt.
2) **Courbe simple :** Soient les abscisses stockées dans la liste notée LX1, et les ordonnées dans la liste LY1. Pour tracer la courbe, il faut alors écrire :

```
plt.plot(LX1,LY1,'g-') # déclaration des listes abscisse (ici LX1) et ordonnée (ici LY1), g
pour green, - pour ligne continue
plt.show()  # affiche la courbe
```