Projet *Car Model Names*

Inspiration:

- https://xkcd.com/1571/
- https://www.explainxkcd.com/wiki/index.php/1571:_Car_Model_Names

1. Pour une chaîne comportant des noms de modèle de voiture, faire un décompte de toutes les lettres stocké dans un tableau de 26 compteurs.
2. Appliquer la même méthode sur une autre chaîne de caractères de texte général (wikipedia?) pour avoir un point de comparaison.
3. En appliquant les comptages bruts, produire la liste des scores de chaque lettre obtenus par la formule Score(x) = Frequency_in_cars(x) - Frequency_in_French(x). *Essayer d'avoir des chaînes de longueurs comparables pour que les comptages soient eux-mêmes comparables.*
4. Calculer le score de quelques noms de marque existants.
5. Prendre des mots au hasard et calculer leur score.
6. Générer des mots possibles avec des scores élevés ou faibles.

TP Bonus : rendre le code pour chacune des questions, accompagné d'un document texte donnant les résultats produits par le programme et explicitant les choix effectués. Le texte ne doit pas faire plus de deux pages. Le tout doit être envoyé ou déposé sur iCampus avant le 28 novembre 2025 à 16h.

Quelques éléments en lien avec la discussion faite en cours le 21 novembre.

Pour la question 1, on peut envisager deux algorithmes présentés ici en pseudo-code:

mot est la chaîne dans laquelle on veut compter les occurrences de lettres

A. Version 1:

```
compteurs = []
pour chaque lettre x de l'alphabet:
   compteurs.append(nb_occurrences(x,mot))
```

B. Version 2:

```
compteurs = [0] * 26
pour chaque lettre l du mot:
    trouver le compteur correspondant et l'incrémenter
```

[1]:
```
# Mise en place version 1: on repart de la fonction
# qui compte le nombre de 'a' dans la chaîne _mot_
mot = "permenganate"
c = 0
for x in mot:
    c += (x == 'a')
print(c)
# On observe que le même code pourrait être généralisé,
# pour compter le nombre de l dans la chaîne s
```

```
2
```

[2]:
```
# D'où la fonction (sous-programme) définie de la manière suivante:
def nb_occurrences(l,s):
    c = 0
    for x in s:
        c += (x == l)
    return c
```

[3]:
```
print(nb_occurrences('a',"permenganate"))
```

```
2
```

[4]:
```
print(nb_occurrences('i','gratin à la dauphinoise'))
```

```
3
```

[5]:
```
# Le pseudo-code de la version 1 pourrait être implémenté ainsi:
# Ici avec les 5 premières lettres de l'alphabet
mot = "gratin à la dauphinoise"
compteurs = []
for lettre in ['a', 'b', 'c', 'd', 'e']:
    compteurs.append(nb_occurrences(lettre,mot))
print(compteurs)
```

```
[3, 0, 0, 1, 1]
```

[6]:
```
# Pour généraliser le code précédent, on peut utiliser
# les fonctions `ord()` et `chr()` qui permettent de mettre
# n correspondance les caractères et leur code (utf8 ou ascii).
mot = "gratin à la dauphinoise"
compteurs = []
for i in range(26):
    lettre = chr(i+ord('a'))
    compteurs.append(nb_occurrences(lettre,mot))
print(compteurs)
```

```
[3, 0, 0, 1, 1, 0, 1, 1, 3, 0, 0, 1, 0, 2, 1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0]
```

[7]:
```
# On peut facilement faire des vérification en utilisant les fonctions python.
print(len(mot))
print(sum(compteurs))
```

```
23
19
```

[8]:
```
# Pourquoi les deux nombres sont-ils différents ?
# Parce que les espaces et le a-accent grave ne sont pas comptés
```