Modalités du test :

Dès réception du mail contenant l'entièreté de l'exercice, vous aurez jusqu'à **lendemain minuit** pour compléter les exercices.

Tous les exercices devront être hébergés sur un répertoire GIT de votre choix dont il faudra nous communiquer l'URL (vous avez le droit de préparer le répertoire à l'avance).

Vous pouvez créer un répertoire par exercice ou un seul répertoire pour tous les exercices.

Si un commit dépasse l'heure de rendu (**lendemain minuit** après l'horaire de réception du mail) il ne sera pas comptabilisé.

Assurez-vous donc d'effectuer des commits fréquents.

Critères d'évaluation :

- Maintien et lisibilité du/des répertoire(s) GIT. Présence de README, gitignore, structure...
- Qualité de programmation (nom des variables, commentaires, DRY-Don't Repeat Yourself, respect des normes du langage).
- Temps nécessaire pour réaliser la totalité des exercices.
- Attention tout tentative de triche en copiant un algorithme déjà existant entraînera votre disqualification. C'est relativement facile à détecter.

Exercice 1 d'analyse et visualisation

Ci-joint deux fichiers csv, « tous les codes » et « codes disponibles », sans entré dans les détails de la signification de chaque type de codes, crée les fonctions python qui permettent de:

- Calculer le pourcentage des codes disponibles par rapport à tous les codes.
- Calculer la fréquence d'un code disponible donné.
- Afficher un graphique qui compare les fréquences de 5 codes disponibles.
- Afficher un graphique qui permet de visualiser le maximum, minimum, médiane, et la moyenne des fréquences des codes disponibles.

Exercice 2, Algorithmie:

Implémentation de l'algorithme de diviser pour mieux régner.

Le principe de cet algorithme : Vous avez un nombre aléatoire que vous devez insérer dans une liste déjà triée. Au lieu de parcourir naïvement et de façon séquentielle toutes les cases du tableau pour trouver la bonne position, il est plus efficace pour les grands tableaux de tester le milieu du tableau vis à vis du nombre à insérer. On sait ensuite si on doit l'insérer dans le demi tableau de droite, ou de gauche. On se retrouve donc avec un nouveau tableau (le demi tableau que nous avons sélectionné) et on peut appliquer le même processus jusqu'à que l'on trouve la bonne position. C'est un exercice classique d'algorithmie. Vous pouvez trouver plus d'informations sur la problématique en ligne.

Votre programme devra disposer de trois constantes au début permettant de définir la LONGUEUR du tableau, les valeurs MIN et MINX admises dans ce tableau.

Générez un tableau de la LONGUEUR définie en générant un nombre aléatoire compris entre MIN et MAX.

Puis vous pouvez utiliser une fonction native python pour trier le tableau de façon chronologique.

Ensuite générez un nombre aléatoire entre MIN et MAX. Ce sera le nombre à insérer.

Appelez ensuite votre fonction DPMR (Diviser Pour Mieux Régner). Cette fonction (idéalement récursive) devra vous renvoyer l'indice tu tableau où le chiffre devrait être inséré (tous les autres nombres présent dans le tableau après l'indice trouvé et y compris le nombre occupant l'indice trouvés seraient déplacés à droite).

Bonus (si vous avez du temps en plus) : Créez une fonction permettant de tester XX fois votre fonction et de comparer ses résultats pour vous assurer qu'elle fonctionne bien.

Exercice 3, Machine learning

Ci-joint le dataset « text_code » :

- En utilisant « Bert » et « RoBerta », entraîner deux classificateurs qui permettent de reconnaître le bon code.
- Afficher un graphique qui compare les performances des modèles entraînés.

"I'm not a great programmer; I'm just a good programmer with great habits."
— **Martin Fowler, Refactoring: Improving the Design of Existing Code**