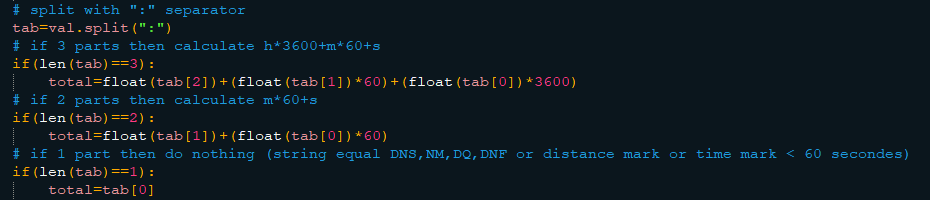
**Étapes d’enrichissement et de preprocessing tout au long du PROJET**

**1\_convert\_time\_in\_seconds.py :**

Dans le dataset collecté sur World Athletics, la variable “Mark” prends plusieurs formes :

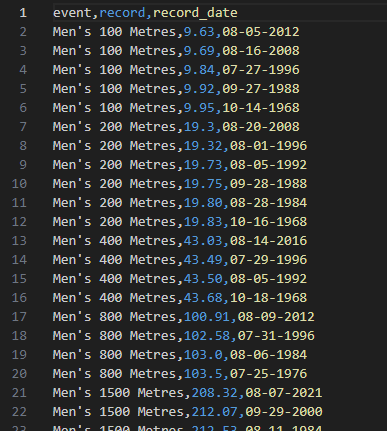
* des temps en h/m/s, m/s, s
* des distances en mètres

Le dataset initial est donc divisé en deux (temps et distance), et sur le dataset temps, nous appliquons une conversion de la variable Mark en secondes. Après le remplacement des caractères h/m par “:” afin d’uniformiser, et la suppression des espaces et caractères parasites, on décompose (split) les heures:minutes:secondes, puis nous calculons le total en secondes

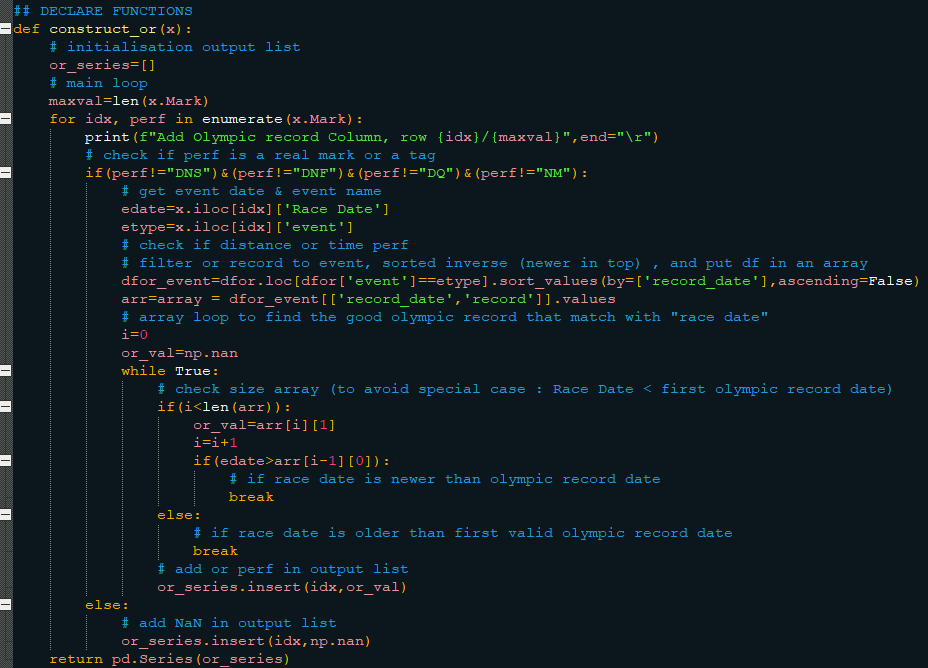
****

**2\_add\_olympic\_record.py :**

Voulant déterminer pour chaque performance (‘Mark’) du dataset, si elle est inférieure ou supérieure au record olympique en vigueur à la date de l’observation, nous avons constitué manuellement un tableau ne contenant que les dates et valeurs des records olympiques à partir des données Wikipédia :



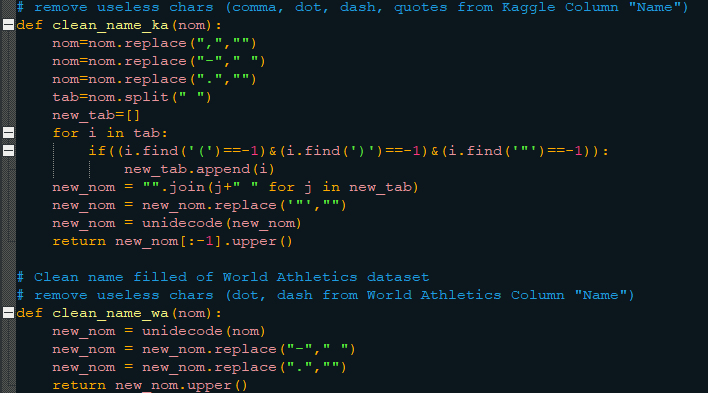
En parcourant l’ensemble des observations du dataset World Athletics, nous ajoutons le record correspondant à l’épreuve, et la date de la performance (‘Race date’),



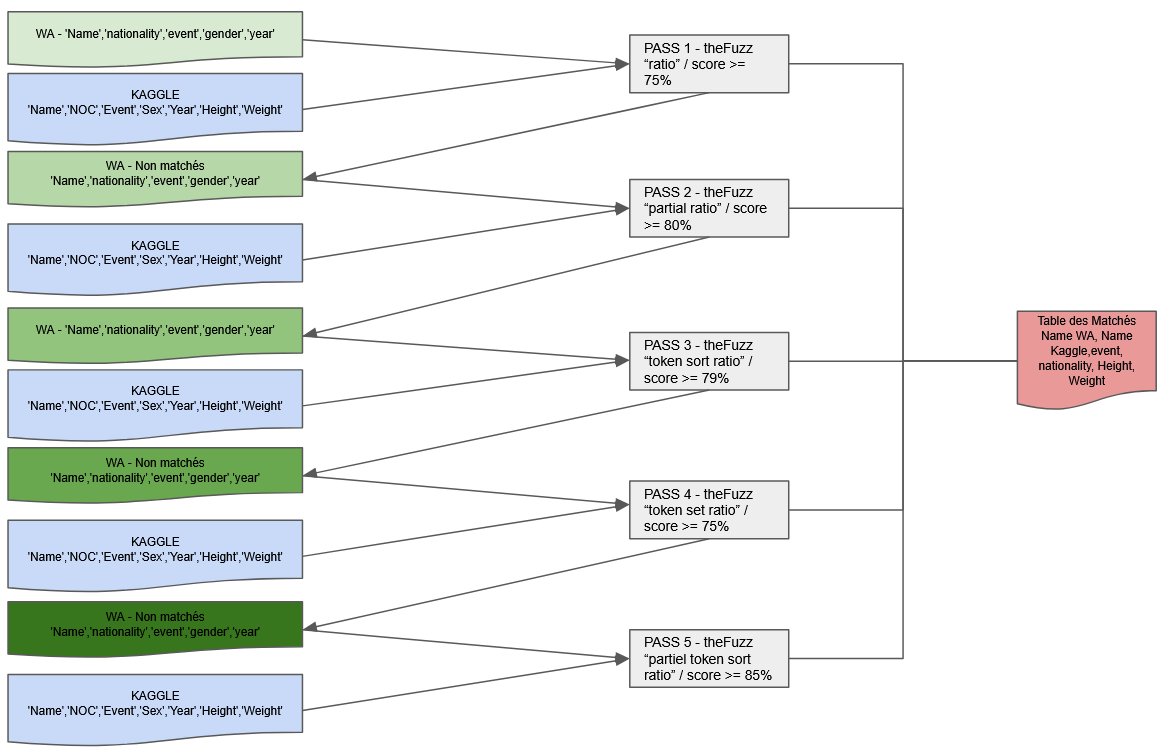
**3\_add\_height\_weight.py :**

Afin d’enrichir le dataset World Athletics, nous cherchons la taille et le poids des athlètes dans un dataset récupéré sur Kaggle, mais qui ne référence que les athlètes ayant déjà participé aux JO. La difficulté majeure est de faire une jointure sur la variable contenant le prénom / nom des athlètes, celle-ci est très différente entre les datasets (caractères et accents,surnoms entouré de doubles quotes, présence ou pas de plusieurs prénoms, etc).

Nous appliquons une première étape de nettoyage sur la variable Name de chaque dataset (utilisation de la fonction replace, et de la fonction unidecode pour les accents) :

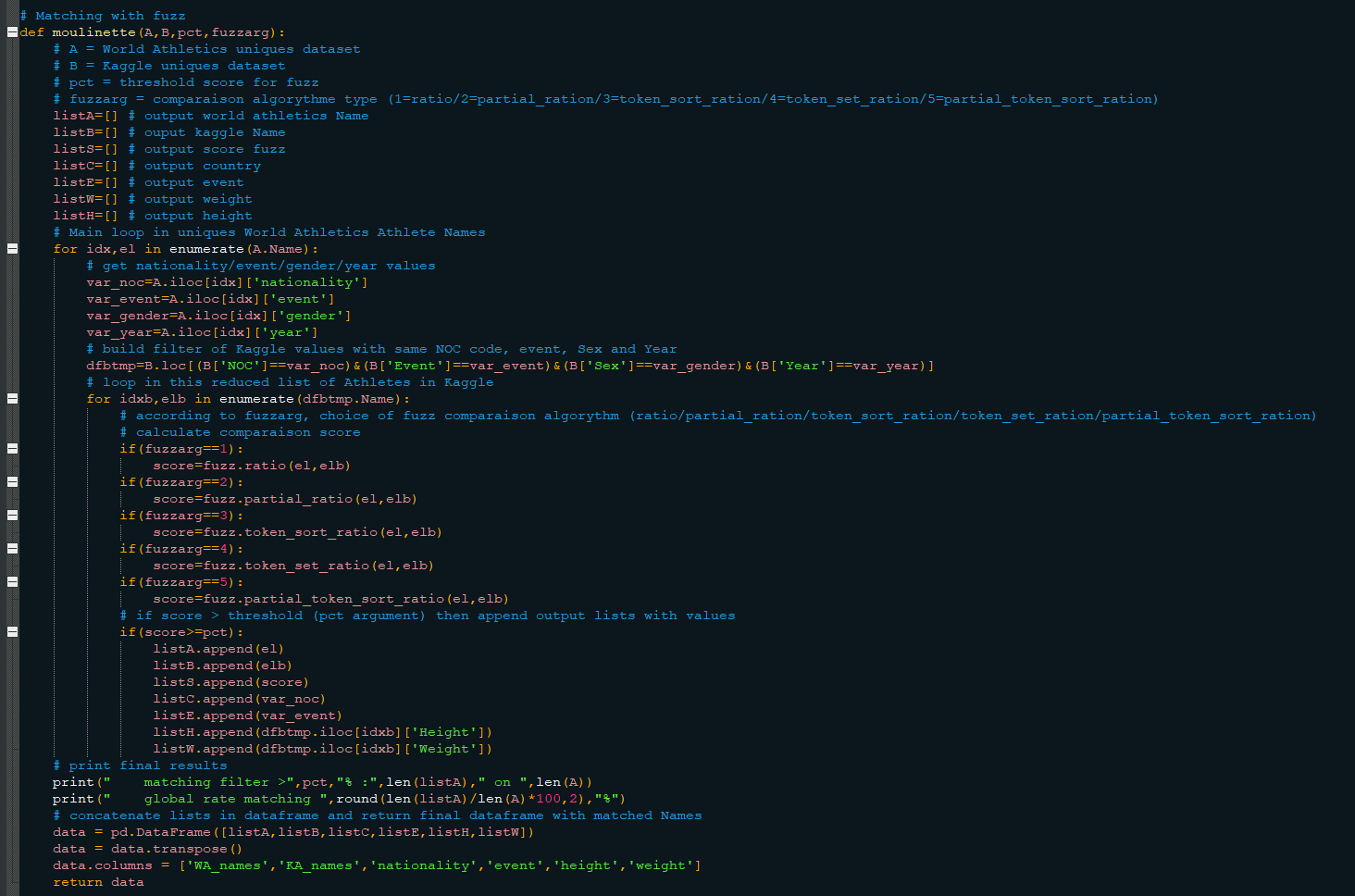


Nous réduisons les datasets initiaux, en filtrant sur les athlètes ayant déjà participés aux JO, et en retirant les observations d’épreuves combinées, afin de ne garder qu’une liste d’athlètes avec le genre, la nationalité, le type d’épreuve et ajoutons un champs avec l’année des performances, puis nous retirons les duplicatas.Ensuite, grâce à une boucle, nous parcourons les athlètes du dataset réduit de World Athletics, et utilisons la fonction fuzz de la librairie thefuzz pour comparer les noms afin de trouver les correspondances. Puis nous excluons par étape les athlètes world athletics que l’on a trouvé dans kaggle (les matching seront stockés dans une table avec la taille et le poids correspondant). En utilisant les différentes options de fuzz, nous faisons 5 passages :



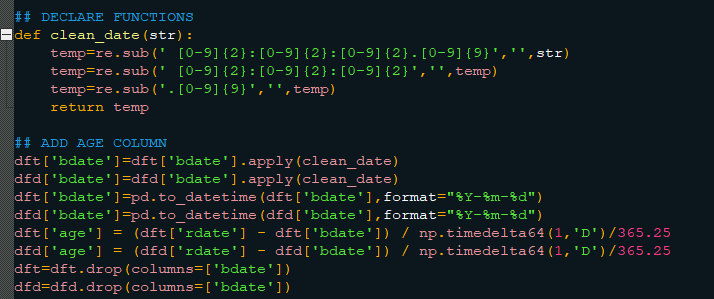
Ensuite, nous faisons une jointure gauche avec le dataset initial et cette table d’athlètes “matchés”. Il reste alors de nombreux NaN sur les colonnes que nous remplissons en prenant la moyenne des athlètes du même type d’épreuve afin de garder une cohérence globale. Nous rencontrons toutefois une difficulté, nous obtenons une taille/poids différent pour un même athlète lorsque celui ci participe à plusieurs type d’épreuves (saut en hauteur et lancé du poids par exemple), nous devons alors prendre une décision, et prenons pour chaque athlète une seule valeur du poids/taille, et nous l’appliquons à toutes les épreuves à laquelle il participe, afin d’avoir le même poids/taille pour chaque athlète.

Ici le code de la fonction qui permet de faire la comparaison avec theFuzz :



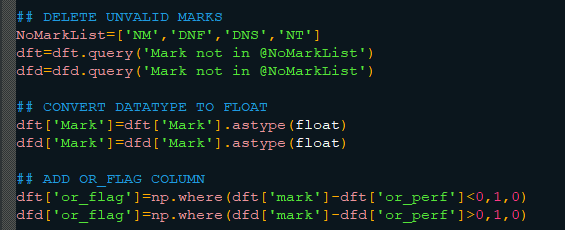
**4\_add\_age.py :**

Toujours dans le cadre de l’enrichissement, et disposant de la date de naissance et de la date de chaque performance dans le dataset World Athletics, nous calculons l’âge précisément (en retirant les heures/minutes/secondes/millisecondes qui étaient présentes dans le format datetime) en soustrayant les deux dates (date de la course - date de naissance) :



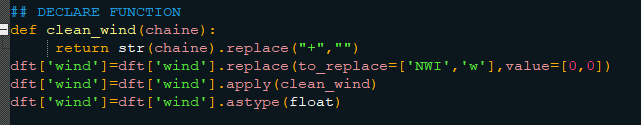
**5\_add\_olympic\_record\_flag.py :**

Afin de disposer d’une variable catégorielle pour les modèles de classifications, nous ajoutons une variable à 2 modalités : (1 = record olympique battu, 0 = record olympique non battu ). Pour cela nous faisons la différence entre la performance (‘Mark’) et le record olympique en vigueur (‘or\_perf’).



**6\_clean\_wind.py :**

la variable correspondants à la mesure du vent en course (‘wind’) contenant un caractère + ou -, nous retirons le symbole ‘+’ et les mesures NaN afin convertir la mesure du vent en float :



**7\_add\_perf\_nb\_mean\_max.py :**

Pour enrichir davantage le dataset, nous ajoutons plusieurs variables qui puisent dans les performances qui précèdent la performance de chaque observation. Pour chaque athlète et épreuve, nous calculons sur les 365 jours précédents :

- le nombre de performances

- la moyenne des performances

- la performance maximale

