I De quoi s'agit-il ?

On considère une suite infinie de lancers d'une pièce équilibrée. Deux joueurs s'affrontent selon les règles suivantes :

▷ Le joueur J est gagnant si la configuration « PILE-PILE-FACE » apparaît dans la suite des lancers avant que n'apparaisse la configuration « FACE-PILE-PILE » ;
▷ Le joueur J' est gagnant si la configuration « FACE-PILE-PILE » apparaît dans la suite des lancers avant que n'apparaisse la configuration « PILE-PILE-FACE » ;
▷ Si l'un des joueurs est gagnant, l'autre est perdant.

• ○ • ○ •

Curiosité : Le joueur J' a trois fois plus de chances de remporter la partie que le joueur J ; il s'agit de faire un programme en Python qui mette en évidence ce paradoxe.

II Des outils pour la simulation

II.1 Comparer des chaînes de caractères

Il est possible de comparer des chaînes de caractères. Par exemple, le test « `mot=="ppf"` » renverra VRAI (True) si la variable `mot` contient la chaîne "ppf".

II.2 Générer des nombres aléatoires

La commande `random()` renvoie un nombre réel compris entre 0 et 1 (de manière « uniforme » : c'est à dire qu'il y a autant « de chances » que le nombre renvoyé soit, *par exemple*, entre 0,2 et 0,3 que entre 0,6 et 0,7)

Remarque 1 Soit la distribution `Python` installée dispose déjà de la commande, soit il faut l'importer via l'instruction `from random import*`

III On est prêt !

Essayer de mettre en évidence le paradoxe de Penney.

Aide à lire si vous êtes dans l'impasse :

- On peut envisager une fonction sans paramètre qui construit une chaîne de caractères d'au plus 4 caractères et retourne la valeur 0 ou 1 suivant que la chaîne `'ppf'` ou la chaîne `'fpp'` a été obtenue.

- Demander à l'utilisateur du programme le nombre de parties qu'il souhaite effectuer et calculer la fréquence de victoires des deux joueurs.