Exercices - Semaine 5

1. Pile ou face

Concevez un programme Python qui répète 10 fois l'expérience suivante :

- Le programme tire au hasard un résultat entre PILE ou FACE.
- On pose la question « PILE ou FACE ? » à l'utilisateur.
- Si la réponse est fausse, on recommence.

À la fin de ces 10 expériences, on affiche combien de fois une expérience s'est arrêtée après 0 réponse fausse, combien de fois une expérience s'est arrêtée après 1 réponse fausse, et ainsi de suite.

Un exemple d'interaction avec le programme est donné à la page suivante. Les entrées de l'utilisateur sont affichées en gras.

2. Pile ou face, façon automatisée

Modifiez le code obtenu à l'exercice précédent de manière à ne plus demander « PILE ou FACE » à l'utilisateur, mais considérez que l'utilisateur a 50% de chance de répondre juste.

Comme l'expérience peut être réalisée beaucoup plus rapidement, faites en sorte de ne pas seulement la faire 10 fois, mais plutôt 10'000 fois. Puis affichez les résultats.

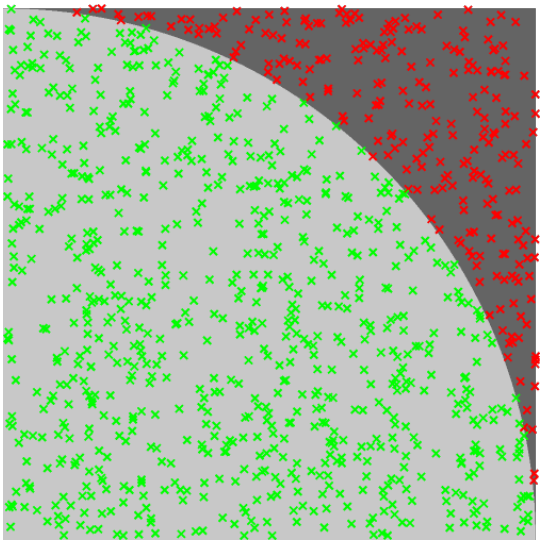
```
Entrez PILE ou FACE: PILE
Entrez PILE ou FACE: FACE
Entrez PILE ou FACE: FACE
Gagné après 2 erreur(s)
Entrez PILE ou FACE: PILE
Entrez PILE ou FACE: FACE
Entrez PILE ou FACE: PILE
Entrez PILE ou FACE: PILE
Gagné après 3 erreur(s)
Entrez PILE ou FACE: FACE
Entrez PILE ou FACE: FACE
Entrez PILE ou FACE: FACE
Entrez PILE ou FACE: PILE
Gagné après 3 erreur(s)
Entrez PILE ou FACE: PILE
Entrez PILE ou FACE: PILE
Entrez PILE ou FACE: PILE
Gagné après 2 erreur(s)
Entrez PILE ou FACE: PILE
Entrez PILE ou FACE: PILE
Entrez PILE ou FACE: PILE
Gagné après 2 erreur(s)
Entrez PILE ou FACE: PILE
Entrez PILE ou FACE: FACE
Gagné après 1 erreur(s)
Entrez PILE ou FACE: FACE
Entrez PILE ou FACE: FACE
Entrez PILE ou FACE: PILE
Gagné après 2 erreur(s)
Entrez PILE ou FACE: PILE
Gagné après 0 erreur(s)
Entrez PILE ou FACE: PILE
Entrez PILE ou FACE: PILE
Gagné après 1 erreur(s)
Entrez PILE ou FACE: FACE
Gagné après 0 erreur(s)
0 erreur(s): 2 fois
1 erreur(s): 2 fois
2 erreur(s): 4 fois
3 erreur(s): 2 fois
```

Exemple d'interaction avec le programme de l'exercice 1.

3. Estimer π par Monte-Carlo

Dans cet exercice, vous allez découvrir comment utiliser la génération de nombres pseudo-aléatoires pour estimer la valeur de π.

Pour cela, commencez par ouvrir et exécutez le programme fourni pour cet exercice. Vous devriez voir, dans le fichier de sortie, une image similaire à celle présentée ci-dessous.



L'image présente un carré gris foncé au-dessus duquel a été placé un quart de cercle de même rayon que le côté du carré. Par-dessus, une série de croix ont été placées. Le centre de chaque croix est un point (pseudo)-aléatoire du carré. La couleur de la croix (verte ou rouge) indique si le point se situe au-dessus du quart de cercle ou non.

De manière intéressante, on constate que la proportion de points verts est liée à la proportion de la surface du carré qui est couverte par le quart de cercle.

Soit r le rayon du quart de cercle (et aussi le côté du carré). L'aire du carré, C, ainsi que l'aire du quart de cercle, Q, obéissent aux équations suivantes :

$$C = r^2 \qquad\qquad Q = \frac{\pi r^2}{4}$$

Ainsi, de ces équations, il est possible de déduire l'égalité suivante :

$$\pi = 4\frac{\mathrm{Q}}{\mathrm{C}}$$

Modifiez le programme Python afin de calculer une estimation de la valeur de la proportion entre Q et C en calculant la proportion de point verts. À partir de cela, donnez une estimation de la valeur de π.

Vous pouvez bien entendu vous passez de la création de la visualisation avec PyTamaro, ce qui devrait vous permettre de simuler le placement de beaucoup plus de points !

4. Votations

Vous avez la lourde tâche de concevoir un programme Python pour la tenue de votations. Le programme sera utilisé par un scrutateur qui aura la charge de comptabiliser les votes.

De manière répétée, le programme devra demander à l'utilisateur d'entrer la proposition pour laquelle il faut enregistrer un vote.

Lorsque tous les votes ont été saisis, l'utilisateur laissera vide la proposition.

Finalement, les résultats du vote sont annoncés, tout d'abord en affichant les propositions par ordre décroissant de votes reçu, puis en indiquant la proposition gagnante (une de celles avec le plus grand nombre de votes).

En cas d'égalité, vous êtes libre de choisir arbitrairement le gagnant.

```
Entrez la proposition: A
Entrez la proposition: B
Entrez la proposition: A
Entrez la proposition: A
Entrez la proposition: D
Entrez la proposition:
A: 3 vote(s)
B: 1 vote(s)
D: 1 vote(s)
A gagne avec 3 vote(s)
```

Indice : Les fonctions d'agrégation comme `min` ou `max`, tout comme la fonction `sorted`, peuvent prendre un argument nommé optionnel appelé `key`, qui indique une fonction à appeler pour comparer les éléments.

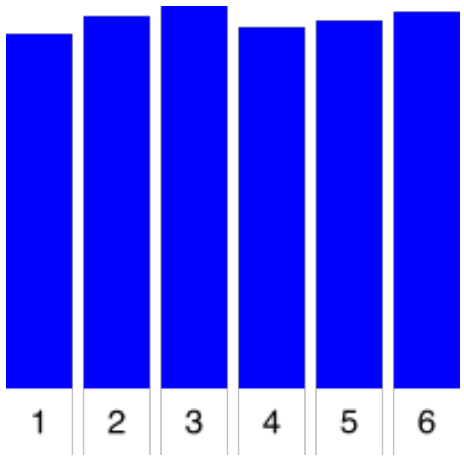
Ainsi, pour obtenir la clé avec la valeur minimale d'un dictionnaire, on peut faire :

```
mon_dictionnaire = {'Alice': 3, 'Bob': 5, 'Charlie': 2}
print(min(mon_dictionnaire, key=mon_dictionnaire.get))
```

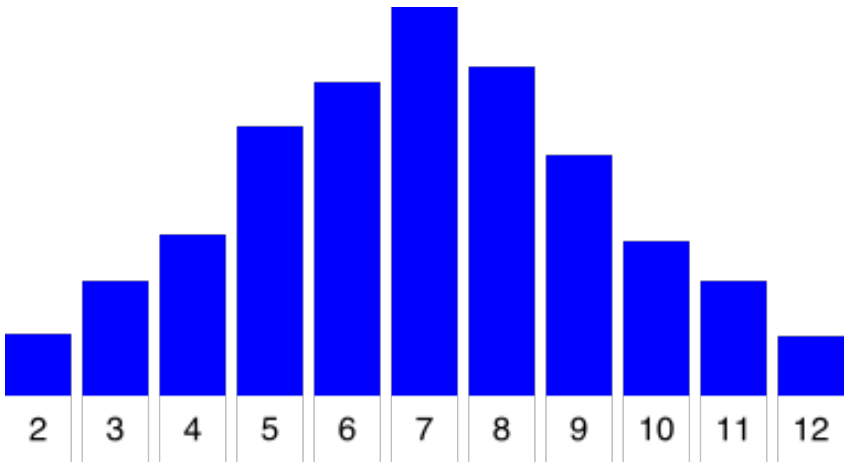
5. Somme de dés

De nombreux jeux de plateau utilisent des dés. Ils permettent, par exemple, de déterminer le nombre de cases à parcourir, ou bien si une action d'un joueur est réussie.

Lorsque l'on jette un unique dé à 6 faces, on tombe, s'il est bien équilibré, de façon uniforme sur les différentes faces. En répétant suffisamment de fois l'expérience, on s'attend à tomber sur un nombre similaire de 1 et de 2, tout comme de 6. Ce qui peut se visualiser de la façon suivante :



Parfois, dans certains jeux, les joueurs sont amenés à lancer plusieurs dés simultanément et à considérer la somme des faces des dés comme le résultat du lancer. Ainsi, lorsqu'on lance deux dés à 6 faces, on obtient un résultat entre 2 et 12. De manière intéressante, la distribution des résultats n'est pas uniforme, mais ressemble à plutôt à cela :

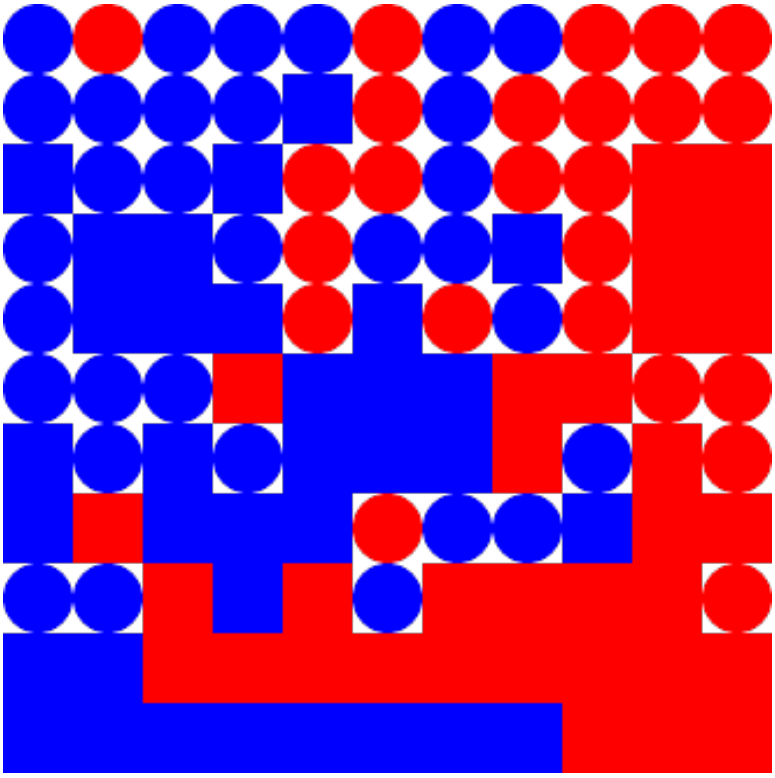


Votre but dans cet exercice est de concevoir un programme Python qui demande à l'utilisateur combien de dés il souhaite lancer, simule un grand nombre de lancers, puis produit une visualisation de la distribution des résultats. Ainsi, ci-dessous est présentée l'interaction avec le programme qui a généré la visualisation ci-dessus.

```
Nombre de dés à lancer ? 2
```

6. Mosaïque

Concevez un programme qui réalise des mosaïques (pseudo)-aléatoires comme celle présentée ci-dessous.



Dans cette mosaïque, formée de 11 par 11 pièces, chaque pièce est soit bleue soit rouge, et a pour forme soit un cercle soit un rectangle.

Plus la pièce est à une position haute, plus elle a de chance d'être un cercle. Plus une pièce est située à gauche, plus elle a de chance d'être bleue.

Bonus : Réalisez une animation qui fait défiler plusieurs de ces magnifiques mosaïques.

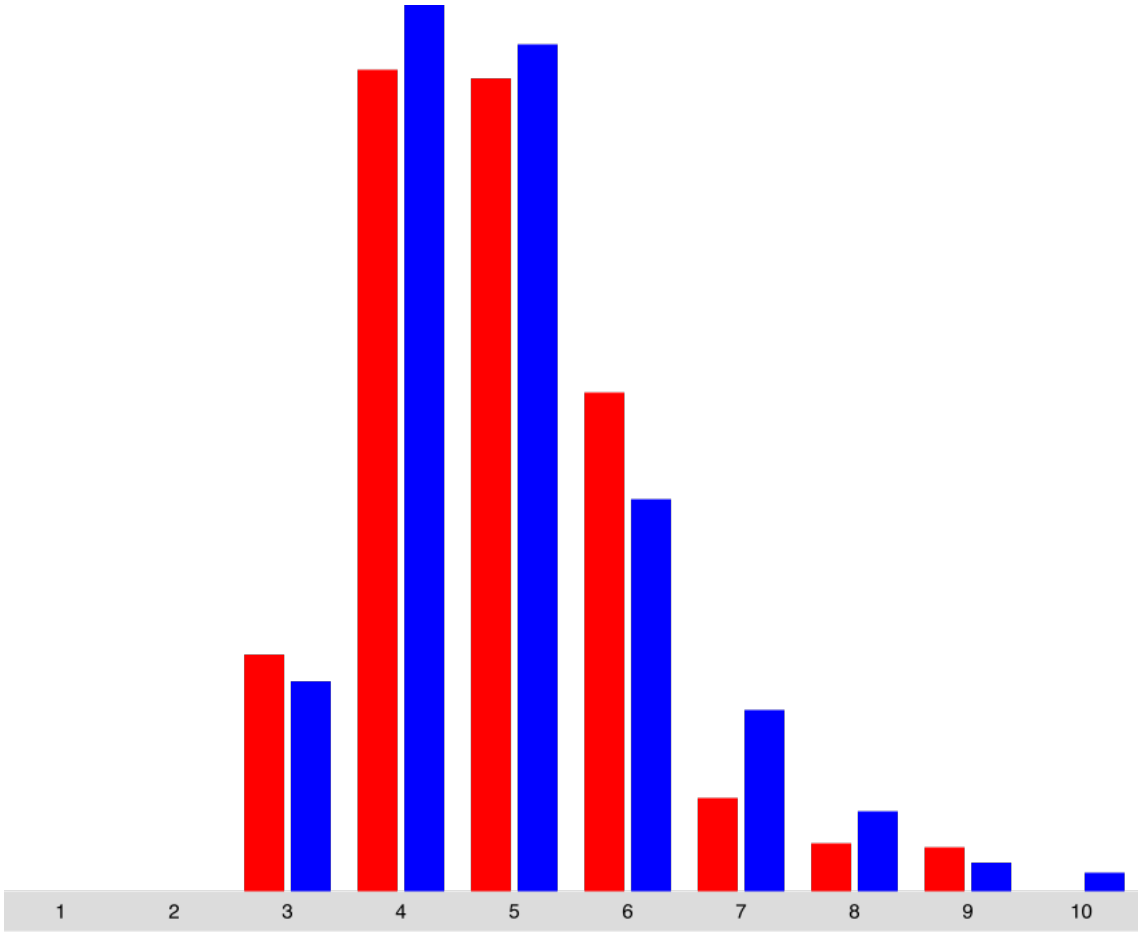
7. Tout sur les prénoms

Vous trouverez sur Moodle deux fichiers contenant le top 200 des prénoms les plus donnés à des nouveau-nés en Suisse en 2023. Un fichier correspond aux prénoms féminins, l'autre aux prénoms masculins.

Chaque fichier contient une entrée par ligne. Chaque entrée contient trois informations :

1. Le prénom
2. Le rang
3. Le nombre

Affichez, en une unique visualisation, la différence de distribution entre la longueur des prénoms féminins et celle des prénoms masculins.



Bonus : Réalisez une même visualisation pour la dernière lettre du prénom entre les filles et les garçons.