Simuler pour trouver une probabilité

Le problème

Un jeu télévisé se déroule face à trois portes identiques. On a placé une voiture derrière l'une des portes, et une chèvre derrière chacune des deux autres.
Le candidat est placé devant les trois portes, il en choisit une en se plaçant devant.
Le présentateur ouvre alors l'une des deux autres portes, derrière laquelle il y a une chèvre, et il demande au candidat :
« Voulez-vous modifier votre choix ? »
Le candidat a alors deux possibilités :
– soit il ouvre la porte qu'il avait choisie en premier ;
– soit il ouvre une autre porte (celle qui n'a pas été ouverte).
Quelle est la meilleure stratégie pour le candidat ?

Complément culturel

Ce jeu télévisé a réellement existé aux États-Unis dans les années 1970.
Le candidat repartait avec une belle voiture, avec rien ou avec une belle chèvre (*a goat* en anglais). Le paradoxe probabiliste associé à ce jeu est connu sous le nom de « problème de Monty Hall » (Monty Hall était le présentateur du jeu).

Les programmes

Python

Un raisonnement probabiliste peut fournir la solution. En l'absence d'un tel raisonnement, on peut « faire des expériences ». Avec Python, on a accès des échantillons de très grande taille. On peut par exemple lancer le programme « thegoat » ci-contre en demandant de reproduire dix millions de fois l'expérience (n = 10 000 000).

le programme

```
>>> from random import randint
>>> def thegoat(n):
        i=0
        k=0
        while i<n:
                a=randint(1,3)
                if a==1:
                        b=randint(2,3)
                if a==2:
                        b=1
                if a==3:
                        b=1
                if b==1:
                        k=k+1
                i=i+1
        return(k/n)
```

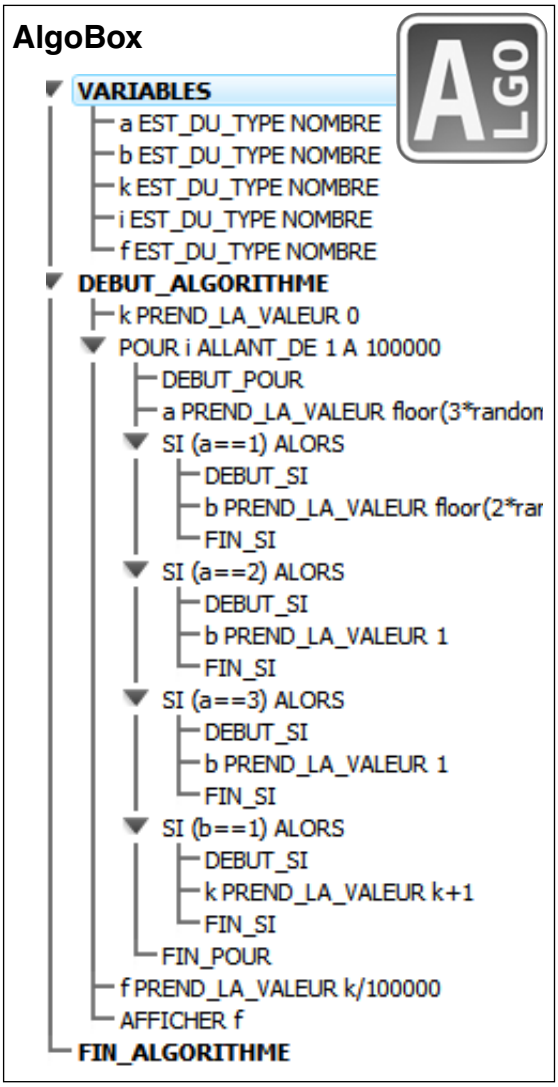
le résultat

```
>>> thegoat(10000000)
0.66664069999999997
```

Si le candidat s'obstine dans son premier choix, il conserve une chance sur trois de gagner. S'il choisit de modifier son choix suite à l'intervention du présentateur, on ne sait pas pour l'instant quelle est la probabilité qu'il gagne. Pour se donner une « idée » de cette probabilité, on simule n fois l'expérience en lui faisant **modifier systématiquement son premier choix** (n = 10 000 000 avec Python, 500 avec Scratch, 100 000 avec AlgoBox).
On numérote les portes 1, 2 et 3, et on place la voiture derrière la porte n°1. Le candidat choisit ensuite au hasard une porte (variable a), il modifie son choix après l'intervention du présentateur (variable b). Le programme renvoie la fréquence de jeux gagnants.
La variable i comptabilise le nombre d'expériences effectuées, et le compteur k recense quant à lui le nombre de fois où le candidat gagne au cours des cent mille répétitions de l'expérience enfin. La variable f donne la fréquence observée de l'événement « Le candidat gagne » sur les cent mille expériences. On notera la rapidité d'exécution des programmes.
Ainsi, l'algorithmique sert aussi à faire des probabilités en dehors du champ de la simulation. En effet, si vous avez bien compris la construction de l'algorithme, il vous est facile de voir pourquoi la probabilité de gagner lorsque le candidat change de porte est 2 / 3.

Scratch

AlgoBox



Prolongement*

Imaginons une nouvelle version du jeu, avec quatre portes, une voiture derrière l'une d'elles, des ânes derrière les autres. Le présentateur joue le même rôle que dans la version précédente.
Le candidat doit-il modifier son premier choix ?