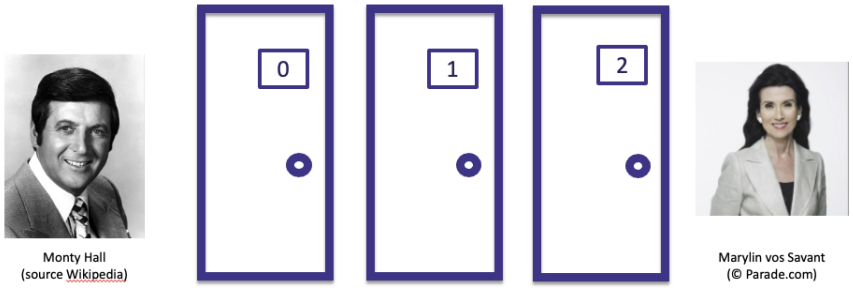
1. Faisons un deal !

1.1 Trois portes : une voiture, deux chèvres

Vous participez à un jeu télévisé, on vous demande de choisir librement une porte parmi trois, sachant que derrière l'une d'elles il y a un prix de valeur (une voiture par exemple), et derrière les deux autres se trouve une chèvre. Vous gagnerez ce qui se trouve derrière la porte que vous aurez choisie. Simple !

Vous annoncez la porte de votre choix :



Note : la numérotation 0, 1, 2 correspond à la programmation en python dans le section suivante.

Mais pas si vite ! Avant d'ouvrir votre porte, le présentateur Monty ouvre une des deux autres portes et révèle une chèvre (Monty sait où se trouve la voiture, il dispose donc dans tous les cas de la possibilité d'ouvrir une porte cachant une chèvre), et il vous demande : **"Avant de continuer, voulez-vous modifier votre choix ? "**

1.2 Question

Avez vous intérêt à conserver votre choix initial ou a choisir l'autre porte ?
Après tout vous aviez une chance sur trois de gagner et rien n'a changé, maintenant c'est une chance sur deux, et changer risque de vous faire perdre).

Par exemple : si vous aviez choisi la porte 2, et que Monty a ouvert la porte 1, devriez vous conserver la porte 2 ou bien choisir la porte 0 ?

Monty Hall a animé cette émission (Let's Make a Deal !) sur la télévision américaine entre 1963 et 1986, les participants faisaient face à cette situation, et utilisaient leur intuition. En 1990 le problème de Monty Hall a fait l'objet d'une controverse alors que Marylin vos Savant a proposé une réponse dans le magazine Parade.

2. Faisons le calcul

Il est assez facile de simuler le jeu avec quelques lignes de Python, on peut ainsi comparer les deux stratégies : (1) le joueur ne change jamais d'avis ou (2) le joueur change systématiquement d'avis.

On pourra utiliser ici la fonction `randint()` de la bibliothèque `random`. L'expression `randint(a,b)` renvoie un entier compris entre a et b (tous deux inclus), a doit être plus petit que b

Par exemple :

```
1 from random import randint
2 for i in range(50):
3     print(randint(0,2), end="")
```

022011112120002212222010101211210122102112222220200

Pour construire la simulation, on peut utiliser la démarche suivante :

1. Construire un programme qui joue 1000 `games`, où une `wining_door` et un `player_choice` sont choisis au hasard, un compteur `wins` est incrémenté à chaque fois que `wining_door` et `player_choice` sont identiques, on devrait trouver autour de 333 `wins`.

2. Ensuite, écrire un programme qui joue un certain nombre de parties où `wining_door` et `player_choice` sont choisis au hasard, et dans le cas où les deux sont identiques, on choisit au hasard une des deux autres portes (celle que Monty va ouvrir : `open_door`).
   La question est équivalente à : étant donné un nombre choisi dans {0,1, 2}, choisir un au hasard un autre nombre dans le même ensemble.

3. Puis écrire un programme qui joue un certain nombre de parties, pour lesquelles `wining_door` et `player_choice` sont choisis au hasard et :
   — si les choix sont identiques, alors `open_door` est choisie au hasard parmi les deux autres (c'est le cas ci dessus),
   — si les choix sont différents, alors `open_door` est calculé de manière à désigner la troisième porte (différente de `wining_door` et `player_choice`).
   C'est équivalent à trouver une formule qui donne `open_door` en fonction de `wining_door` et de `player_choice` avec les règles :

```
player_choice 0 0   1 1   2 2
wining_door   1 2   0 2   0 1
————————————— ———   ———   ———
open_door     2 1   2 0   1 0
```

4. Finalement, utilisons le programme pour évaluer le nombre de victoires du joueur pour chacune des stratégies :
   — avec la stratégie (1) le joueur ne change jamais d'avis,
   — avec la stratégie (2) le joueur change systématiquement d'avis.

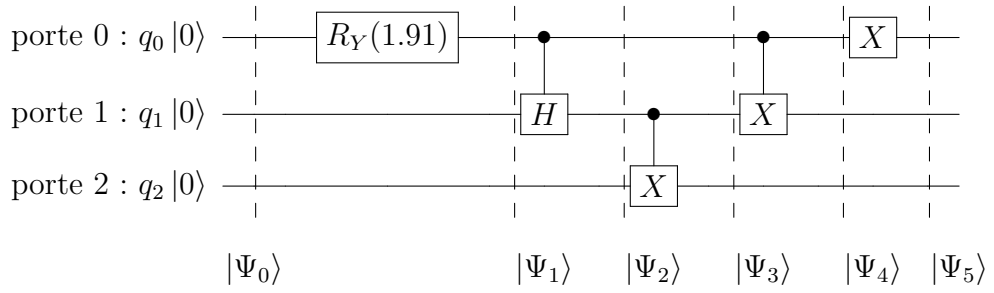
   Vous trouverez les taux de victoires pour chacune des stratégies.

3. Avec un ordinateur quantique

Le jeu est identique. Nous allons utiliser trois qubits q_0, q_1 et q_2 un pour chacune des portes, et sa valeur sera à $|1\rangle$ si il s'agit de la porte gagnante.

3.1 Mise en place de la voiture derrière une porte

On considère le circuit suivant :



Analyser le circuit : calculez les états depuis $|\Psi_0\rangle$ jusqu'à l'état en sortie : $|\Psi_5\rangle$

Conclusion ?

3.2 Décision de Monty

A nouveau on suppose (sans perte de généralité) que le joueur annonce son intention d'ouvrir la porte numéro 2. Les situations (a), (b) et (c) sont celles décrites plus haut (la voiture est respectivement derrière la porte 0, 1 et 2).

On ajoute maintenant un quatrième qubit q_3 à ce circuit, ce qubit va déterminer laquelle des portes (0 ou 1) Monty va ouvrir, de la manière suivante :

— si l'état de q_3 est à $|0\rangle$ alors Monty ouvre la porte 0,
— si l'état de q_3 est à $|1\rangle$ alors Monty ouvre la porte 1,
— ce codage sur un qubit suffit, car bien évidemment Monty n'a la possibilité d'ouvrir que la porte 0 ou la porte 1 (le joueur ayant choisi la porte 2).

Pour modifier l'état de ce qubit, on veut que :

— dans la situation (a) voiture derrière la porte 0 : Monty va ouvrir la porte 1 en d'autres termes si q_0 est à $|1\rangle$ on veut mettre q_3 à $|1\rangle$, et si q_0 est à 0 (la voiture n'est pas derrière la porte 0 : on laisse q_3 à $|0\rangle$
— dans la situation (b)voiture derrière la porte 1 : on veut ouvrir la porte 0, donc on veut que q_3 soit à $|0\rangle$ si q_1 est à $|1\rangle$, comme q_3 est déjà à $|0\rangle$, on ne fait rien.
— dans la situation (c) voiture derrière la porte 2, alors Monty va ouvrir au hasard la porte 0 ou la porte 1, c'est à dire : si q_2 est à 1, on veut mettre q_3 en superposition (état $|+\rangle$).

3.3 Quelle est la meilleure stratégie ?

1. Complétez le circuit avec le qubit q_3 et les portes nécessaires.
2. Calculer l'état en sortie de ce nouveau circuit.
3. En déduire qu'en effet, la meilleure stratégie est celle où le joueur change d'avis.