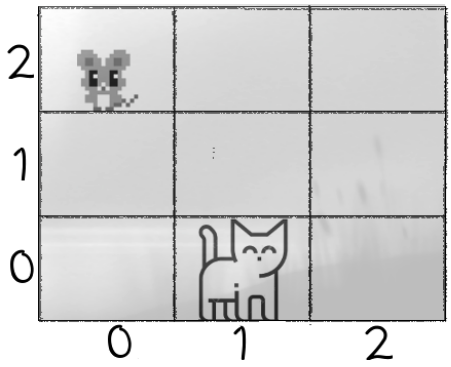
Le jeu du chat et de la souris

0.1 Présentation du jeu

Ce jeu se déroule sur un plateau de 9 cases où vont se déplacer la souris et le chat, d'une seule case à chaque lancer de dé. On repère chacune des cases par ses coordonnées $(x; y)$. Sur le plateau de jeu ci-dessous, la souris a pour coordonnée (0;2) et le chat (1;0).



Un plateau de jeu pendant une partie

La souris : Elle joue en premier et ne se déplace que vers le bas ou vers la gauche. Sa position de départ est la case de coordonnées (2;2).

Le chat : Il se déplace uniquement vers le haut ou vers la droite. Sa position de départ est la case de coordonnées (0;0).

Lancer du dé: Chacun leur tour, les joueurs lancent le dé.

⋆Si le chiffre est pair, le déplacement est horizontal

⋆Sinon, il est vertical.

Fin de partie : Le chat gagne s'il arrive sur la même case que la souris. La souris gagne dans le cas contraire.

0.2 On joue !

Prérequis: Les élèves ont testé et corrigé la partie avec les lancers imposés.

Avec votre binôme, effectuez au moins une dizaine de parties pour pouvoir répondre aux questions suivantes:

1. Combien de tour suffit-il de faire pour connaître le vainqueur ?
2. Quelle est la *fréquence* de l'évènement:"Le chat a gagné "?
3. Quelle méthode peut-on envisager pour déterminer la *probabilité* que la chat gagne ?

0.3 Simulation de parties

Prérequis:

Python: Les élèves connaissent l'interpréteur python et les structures de bases: affectation, condition, boucle *pour*.

Parité: Lors du chapitre Arithmétique, des test de parité du type `if n%2==0:` ont été effectués.

On se propose d'écrire un algorithme simulant un grand nombre de partie.

Dans cette partie, on assimile la souris à un point $S(x_s; y_s)$ et le chat à un point $C(x_c; y_c)$.

0.3.1 La souris

Complétez l'algorithme `déplacement de la souris` en pseudo-code, puis en `Python`:

1 Importer le module **random**
2 **si** *le chiffre obtenu est .....* **alors**
3 | $x_s \longleftarrow$ .....
4 **sinon**
5 | $y_s \longleftarrow$ .....
6 **fin**

Algorithme 1: la souris

```
from random import *
if randint(1, 6)%2==0:
    x_s = x_s-1
else:
    y_s = y_s-1
```

Algorithme 1: la souris (Python)

Testez cette séquence avec les coordonnées de départ $x_s = 2, y_s = 2$; vérifiez que les coordonnées x_s et y_s correspondent à une valeur possible sur le plateau. Vous pourrez utiliser pour cela:

1. `print("abscisse souris",` x_s`)` pour afficher x_s puis y_s
2. Utiliser le cadre *variables* de l'interpréteur.

Votre implémentation de l'algorithme fonctionne-t-elle ?

0.3.2 Le chat

Effectuez le même travail que précédemment pour obtenir une séquence d'instructions de déplacement du chat qui fonctionne (*Algorithme 2*).

0.3.3 Une partie

1. Comme indiqué en ***2.1***, il suffit maintenant de répéter deux fois les séquences précédentes afin d'obtenir une partie. Ecrivez cette séquence et testez-la pour obtenir un programme réalisant une partie.

2. A quelles conditions le chat a -t-il gagné ? Complétez l'algorithme suivant et implémentez-le en Python.

1 **si** ..... *et* ..... **alors**
2 | *victoire_chat* $\longleftarrow$ .....
3 **sinon**
4 | *victoire_souris* $\longleftarrow$ .....
5 **fin**

Algorithme 3: Fin de partie

0.3.4 Etudes des fréquences

Vous pouvez procéder de deux façons:

1. Si elles fonctionnent, répéter les séquences précédentes avec une boucle `pour`.
2. Dans le cas contraire:
   - **Importer** le module `chat_souris` *que vous aurez récupéré auprès de votre professeur.*
   - Utiliser la fonction `victoire_chat()`. Elle retourne la valeur 1 si le chat gagne une partie et 0 sinon.

Exemple:Partie où le chat a perdu.

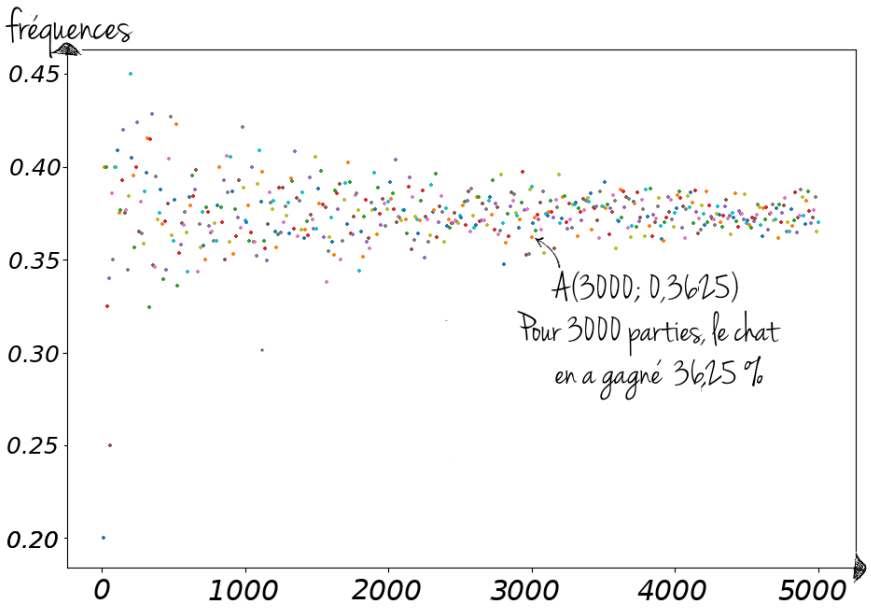
```
>>> from chat_souris import *
>>> victoire_chat()==1
True
```

Affecter 0 à la variable `nb_victoire`, et répéter 1000 parties en incrémentant `nb_victoire` à chaque victoire du chat.

 : `victoire_chat()` est une fonction et vous devez impérativement l'*appeler* avec des parenthèses !

0.3.5 Graphique des fréquences

- **Importer** le module `chat_souris` *que vous aurez récupéré auprès de votre professeur.*
- Utiliser la fonction `graph_frequence(n)` où n désigne le nombre de partie effectué. Elle retourne le nuage de points des fréquences de victoire du chat en fonction du nombres de parties. Qu'observe-t-on ?



Fréquences de victoire du chat en fonction du nombre de partie