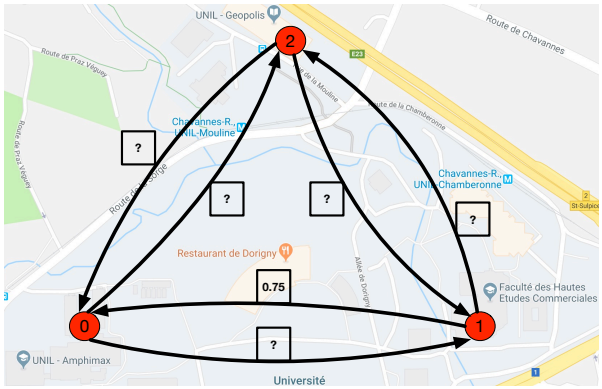
Exo 1 :
"Une chaîne de Markov est un processus aléatoire portant
sur un nombre fini d'états, avec des probabilités de transition sans mémoire."
(**http://benhur.teluq.ca/SPIP/inf6460/spip.php?article108)**

Par exemple, si vous modéliser vos déplacement sur le campus de l'UNIL:

- les états représentent les différents bâtiments de l'UNIL (e.g., Amphimax, Internef)
- Les transitions représentent le déplacement d'un bâtiment à l'autre

La figure ci-dessous montre 3 états (bâtiments) et 6 transitions possibles. Comme vous pouvez le voir sur la figure, la probabilité d'aller à l'Amphimax depuis l'Internef est de 75%. Dans une chaîne de Markov, la somme des probabilités des arêtes sortantes doit être 100%. Vous pouvez donc déduire que la probabilité d'aller de l'Internef au Géopolis est de 25%. Ainsi, une chaîne de Markov peut être utilisée pour prédire le prochain état en fonction de l'état actuel.



1.1 Les déplacements de Bob sont encodés sous forme d'une liste d'entiers où Amphimax est représenté par le numéro 0, Internef par le numéro 1 et Geopolis par le numéro 2. Développer un programme Python qui construit une matrice de transition ("state-transition matrix") pour les déplacement ci-dessous:
Deplacement_bob = [1,2,0,1,0,1,0,2,0,1,0,1,2,0,1,2,1,0,1,0,1,0,1,0,2,1,2,1, 0,1,0,1,0,2,0,2,0,1,0,1,0,2]

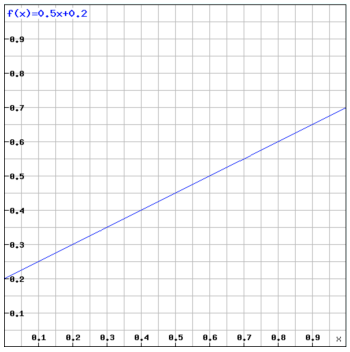
Aide1: [1,2,0,…] signifie que Bob s'est déplacé: de l'Internef, au Géopolis, puis à l'Amphimax
Aide2: La Figure ci-dessus possède la réponse correcte pour les déplacements depuis l'Internef (0.75).
Aide 3: une matrice de transition est une matrice indiquant les probabilités d'aller d'un état à l'autre. Ci-desssous, la matrice de transition partielle qu'il s'agit de construire à l'aide de votre programme python (en entier).

	Internef	Géopolis	Amphimax
Internef	0	.25	.75
Géopolis		0	
Amphimax			0

1.2 Considérant la variable Deplacement_bob donnée ci-dessus, que est le prochain déplacement le plus probable de Bob?

Exo 2 :

On veut calculer l'aire sous la courbe d'une fonction en approximant la valeur par un algorithme de Monte Carlo. Calculer l'aire pour la fonction f:x → $\frac{1}{2}$ x + 0.2 pour un plan de 0 à 1 (f:[0,1] →[0,1]). Conseil : utiliser la méthode random du module random pour générer un nombre aléatoire entre 0 et 1.

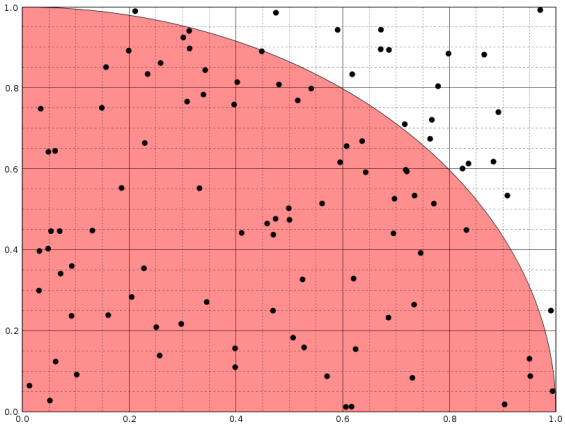


Exo3:

Il est possible d'approximer la valeur de pi en utilisant une approche probabiliste. Imaginez un plan cartésien où 0 < x < 1 et 0 < y < 1 sur lequel on met un quart de cercle de centre 0,0 et de rayon 1 (en rouge ci-dessous). On sait que les points noirs sont dans le cercle si $x^2 + y^2 \leq 1$. Vous pouvez obtenir une approximation de pi avec la formule suivante:

$$\frac{nombre\_points\_dans\_cercle}{nombre\_points\_tot} * 4 \simeq \pi .$$

Construisez un algorithme qui fait une approximation de pi à l'aide de 10'000 points.



Exo 4 :

Un checksum (ou somme de contrôle) est une chaîne de charactères de taille fixe “qu'on ajoute à un message à transmettre pour permettre au récepteur de vérifier que le message reçu est bien celui qui a été envoyé.” (https://fr.wikipedia.org/wiki/Somme_de_contr%C3%B4le). Le code python ci-dessous simule un checksum pour un fichier téléchargé sur le web. La ligne “file_downloaded” simule le fichier téléchargé est “checksum_from_website” est le checksum publique qui est visible sur le site sur lequel le fichier a été téléchargé.

- Quel est l’avantage, en termes de sécurité, de mettre à disposition un checksum?
- Pourquoi comparer le checksum plutôt que le fichier qui est téléchargé?
- Pourquoi est-ce considéré comme une technique probabiliste?

```
import hashlib
file_downloaded =
"01010000101011110101011010111011000010010101010010101011101010110101110110000100101010100101
01011101010110101110110000100101010100101010111010101101011101100001001010101001010101110101
11010111011000010010101010010101011101010110101110110000100101010100101010111"

checksum_from_website = "71b83f836adc7fcfbe88b0c4d2f0a8a7"

# hexdigest return the digest value as a string of hexadecimal digits.
checksum_calculated = hashlib.md5(file_downloaded.encode('utf-8')).hexdigest()

print('checksum from website: {0}'.format(checksum_from_website))
print('checksum calculated: {0}'.format(checksum_calculated))
print('test if equal: {0}'.format(checksum_from_website==checksum_calculated))
```