Le Jeu du Poulpe

Noël approche et vous êtes fauché comme un champ. ‘Santa’ ne passe pas chez les pauvres malheureusement. Vous avez une famille que vous avez négligé pour le travail[1] et si vous ne leur donnez pas de cadeaux iels risquent de vous abandonner…
Coup de bol, un jour quand vous vous baladez dans la rue un homme en costume vous propose de participer à un jeu pour gagner un maximum de blé! C’est le Jeu du Poulpe.

Le Jeu du Poulpe (JdP) se passe en plusieurs niveaux avec de plus en plus de règles. À chaque niveau vous allez devoir établir une stratégie et la soumettre sous forme de code python. ***Lisez attentivement les explications qui suivent.***

https://img.freepik.com/photos-premium/photographie-animaux-marins-camouflage-intelligent-poulpe_1032986-5332.jpg?w=2000

[1] Votre conjoint.e vous en veux de passer tous les samedis à faire du python.

Niveau 0: Tutoriel

> "*Les pieuvres se distinguent par leurs capacités intellectuelles étonnantes pour un invertébré.* [...] *Leur intelligence leur permet d'adopter des comportements faisant appel au camouflage, à l'innovation, à la tromperie.*"
>
> Wikipedia, "*Pieuvre*", accédé le 5 décembre 2025

Une petite histoire:

Deux petites pieuvres se rencontrent dans les profondeurs de l'océan pacifique. Elles ont faim, mais les proies sont bien trop grandes pour chasser en solitaire. En faisant équipe, les deux pieuvres arrivent à coincer un crabe fort appétissant. La première pieuvre, gourmande, se dit alors "Avec mon jet d'encre, je pourrais aveugler l'autre et dérober la proie! Quel festin ce sera."
Sans hésiter elle lance son attaque. Mais à sa grande surprise sa cible a eu exactement là même idée. Les deux pieuvres se trouvent avec la vision obscurcie par leur avarice. Pendant qu'elles sont perdues dans un nuage d'encre, le crabe ne se fait pas attendre et rentre discrètement chez lui.

Le principe du jeu est le suivant: vous allez programmer un robopoulpe (squidbot) qui va participer à un *tournoi* contre plusieurs autres robopoulpes. Dans chaque tournoi, votre robopoulpe va jouer un match contre chaque autre robopoulpe.
Un *match* consiste en un nombre aléatoire de rounds (entre 10 et 20). À chaque round votre robopoulpe chasse un crabe avec un autre participant. Il doit *choisir entre coopérer ou trahir*. Votre robopoulpe retient toutes les décisions prises pendant le match courant, et doit se baser là-dessus pour prendre sa prochaine décision.
À chaque round, vous gagnez un nombre des points en fonction du choix d'action de votre robopoulpe et de celui de l'adversaire.

- Si les deux robopoulpes coopèrent chacun gagne 3 points (le crabe est partagé en deux).
- Si un robopoulpe coopère et l'autre trahit, le traitre gagne 5 points (il n'a pas la place pour tout le crabe) et l'honnête robopoulpe ne gagne aucun point.
- Si les deux robopoulpes trahissent, +1 point chacun (le crabe s'échappe, mais pour des raisons techniques un petit poisson se fait avoir par l'encre).

Ceci est illustré dans le tableau qui suit (Tableau 1).

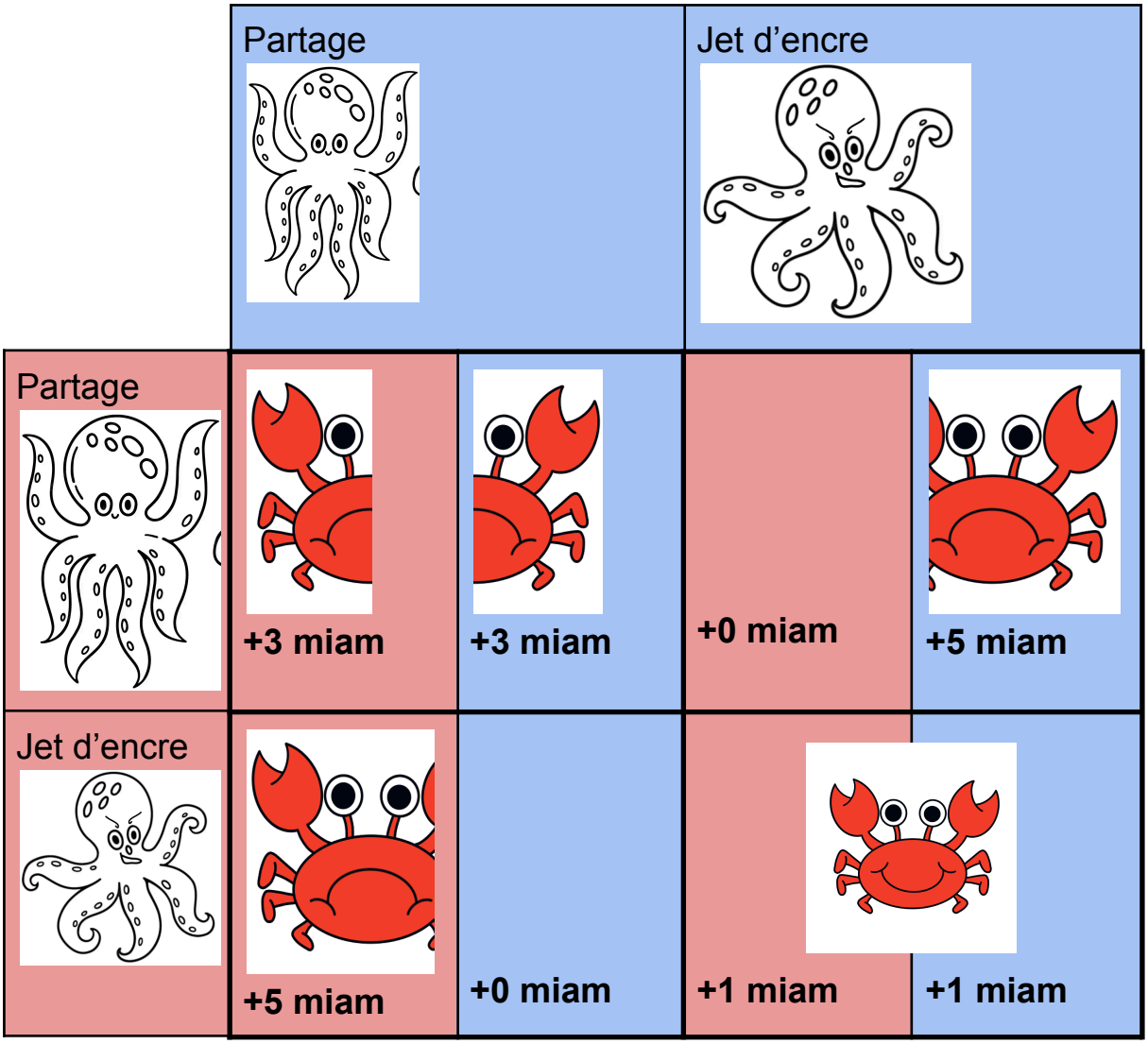

Tableau 1: matrice des gains d'un round du jeu du poulpe

Concrètement, vous allez devoir programmer une fonction `ma_stratégie` qui prend en entrée deux listes: `mon_historique` et `historique_adversaire`. Ses listes sont les décisions prises par les adversaires au rounds précédents: `True` pour coopérer et `False` pour trahir. Votre fonction doit renvoyer `True` si vous souhaitez coopérer au round actuel, et `False` si vous souhaitez trahir.

Exemple de deux stratégies:

```
import random

def jennybot(mon_historique, historique_adversaire):
    '''jenny'''
```

```
    #Je joue au hasard comme mon adversaire ou comme mon
dernier coup
    if not mon_historique: #au premier coup je coopère
        return True
    else:
        return random.randomchoice([mon_historique[-1],
historique_adversaire[-1]])

def ernestbot(mon_historique, historique_adversaire):
    ´´´ernest´´´
    #Je suis très fourbe
    if not mon_historique: #au premier coup je trahis
        return False
    if historique_adversaire[-1] and not mon_historique[-1]:
        #si j'ai reussi à arnaquer je continue
        return False
    else:
        return True
```

Instructions:

1. Téléchargez les fichier **my_test.py** et **niv0_local.py** depuis moodle.
2. Modifiez la fonction dans le fichier **student.py**. Cela doit rester une fonction à deux arguments (listes de booléens mon_historique et historique_adversaire) et qui renvoie un booléen.
3. Donnez un nom à votre stratégie en ajoutant un docstring à votre fonction: pour donner le nom ma_stratégie à votre stratégie ajoutez ´´´ma_stratégie´´´ comme première ligne dans votre fonction.
4. Testez votre code en lançant le programme **niv0_local.py**
5. Essayez de gagner contre les autres stratégies!

Niveau 1: Le Tournoi

Il est maintenant temps d'affronter les autres participant.es. Le code de votre robopoulpe va jouer contre les codes de tous les autres groupes. Suivez bien les instructions ci-dessous pour participer à la première épreuve.

Instructions:

1. Modifiez la fonction dans le fichier **student.py**. Cela doit rester une fonction à deux arguments (listes de booléens `mon_historique` et `historique_adversaire`) et qui renvoie un booléen.
2. Donner un nom à votre stratégie: pour donner le nom ma_stratégie à votre stratégie écrivez ´´´ma_stratégie´´´ comme première ligne dans votre fonction.
3. Testez votre code en lançant le programme **niv1_1tournoi.py** téléchargeable sur moodle.
4. Sauvegarder votre code dans un fichier appelé **group*X*.py** (remplacez *X* par le numéro de votre groupe).
5. Envoyez votre fichier sur moodle sous **Stratégie - Niveau 1**.

Niveau 2: Le Super Tournoi Evolution

« On comprend facilement qu'un naturaliste [...] en vienne à la conclusion que les espèces n'ont pas été créées indépendamment les unes des autres [...] Toutefois, en admettant même que cette conclusion soit bien établie, elle serait peu satisfaisante jusqu'à ce que l'on ait pu prouver comment les innombrables espèces habitant la Terre, se sont modifiées de façon à acquérir cette perfection de forme et de coadaptation qui excite à si juste titre notre admiration. »

Charles Darwin, 1859, *L'origine des espèces*

Maintenant que vous avez compris les principes de base d'un tournoi du jeu du poulpe, voici que le vrai combat commence. Nous allons simuler l'évolution du poulpe dans les océans avec votre stratégie de robopoulpe[2] de la facon suivante:

- Au départ, chaque stratégie est représentée de manière égale dans la population de poulpes.
- Un tournoi du jeu du poulpe est joué entre chaque stratégie.
- Une nouvelle génération de poulpes est créée en fonction des résultats du tournoi. Plus une stratégie a fait un bon score, plus elle sera représentée dans la nouvelle population.
- On recommence ce processus 250 fois et on regarde quel type de poulpe à eu le plus de succès.

Notez que votre stratégie joue également contre elle-même! Il n'est donc pas forcément optimal d'être très méchant.

Instructions:

1. Modifiez la fonction dans le fichier **student.py**. Cela doit rester une fonction à deux arguments (listes de booléens `mon_historique` et `historique_adversaire`) et qui renvoie un booléen.
2. Sauvegardez votre code dans un fichier appelé **group*X*.py** (remplacez *X* par le numéro de votre groupe).
3. Testez votre code en lançant le programme **niv2_tournoi_evolution.py** téléchargeable sur moodle.
4. Envoyez votre fichier sur moodle sous **Stratégie - Niveau 2**.

[2] Le type louche qui vous a proposé de l'argent (cf. intro) complote peut-etre de prendre le controle des océans avec une espèce de poulpe génétiquement modifiée?

Niveau 3: Erreurs de Communication

Nous apportons maintenant une modification au déroulement du tournoi. Il est bien connu que les poulpes sont un peu sourds. Ainsi, lorsque deux poulpes doivent se coordonner pour attaquer une proie simultanément, il se peut que le signal de départ ne soit pas entendu par un des deux partis.
Pour modéliser ceci dans le tournoi nous allons introduire une probabilité d'erreur de communication. Lorsque vous décidez de coopérer il y a 10% de chance que vous trahissez tout de même votre adversaire. Ceci vaut aussi pour votre adversaire, donc réfléchissez deux fois avant d'être trop rancunier.

Instructions:

1. Écrivez une fonction à deux arguments (listes de booléens `mon_historique` et `historique_adversaire`) et qui renvoie un booléen.
2. Sauvegardez votre code dans un fichier appelé **group*X*.py** (remplacez *X* par le numéro de votre groupe).
3. Testez votre code en lançant le programme **niv3_bruit.py** téléchargeable sur moodle.
4. Envoyez votre fichier sur moodle sous **Stratégie - Niveau 3**.

Niveau 4: Réputation

Au village, sans prétention,
J'ai mauvaise réputation ;
Que je me démène ou je reste coi,
Je pass' pour un je-ne-sais-quoi.
Georges Brassens, 1952, *La mauvaise réputation*

Elle m'a pas follow back quand je l'ai follow (Pou-loulou)
Niska, 2017, *Réseaux*

Les robopoulpes sont des êtres très sociaux. Lorsque vous ne coopérez pas, cela déteint sur votre réputation et vos congénères le retiendront.
Dans cette épreuve, nous vous donnerons accès à la réputation de vos adversaires. Elle est calculée comme suit. Au début d'un tournoi (au début de chaque génération) la réputation de chaque robopoulpe est initialisée à 0. À chaque fois que vous trahissez, votre réputation diminue de 1, et à chaque fois que vous coopérez votre réputation augmente de 1.
Vous allez donc devoir modifier votre fonction pour qu'elle prenne en compte deux arguments supplémentaires: `ma_reputation` et `opp_reputation`. Vous pouvez utiliser ces informations pour définir une stratégie plus fine.

Instructions:

1. Modifiez votre fonction `my_strategy` dans **my_test.py** pour qu'elle accepte **quatres arguments**: `mon_historique` et `historique_adversaire` (listes de booléens), et `ma_reputation` et `opp_reputation` (entiers). Le code ci-dessous illustre la définition de fonction nécessaire.

```
def my_strategy(my_hist, opp_hist, my_rep, opp_rep) ->
bool:
     """Ma_strategie"""
    ##########################################
    ############# TON CODE ICI ###############
    ##########################################
```

2. Sauvegardez votre code dans un fichier appelé **group*X*.py** (remplacez *X* par le numéro de votre groupe).
3. Testez votre code en lançant le programme **niv4_reputation.py** téléchargeable sur moodle.
4. Envoyez votre fichier sur moodle sous **Stratégie - Niveau 4**.