# TME 9 : RL

Nous utiliserons dans ce TME la plateforme en python `gym` (de *open-ai*) ainsi que le paquet `flappy-bird-gym` . Installer ces paquets (ne pas oublier de configurer le proxy !) :

```
export https_proxy=proxy:3128
export http_proxy=proxy:3128
pip3 install --user gym
pip3 install --user flappy-bird-gym
```

Télécharger également le fichier code source sur le site de l'UE, les fonctions `test_gym()` et `test_flappy()` permettent de tester votre installation.

## Programmation dynamique

L'environnement `Taxi-v2` de `gym` est une tâche de RL où un taxi doit récupérer un client et le déposer à un endroit donné sur une grille de 5 par 5 : (cf `https://gym.openai.com/envs/Taxi-v3/`). Ci-dessous quelques fonctions utiles pour les environnements :

- initialiser un environnement : `env = gym.make('Taxi-v3')`
- `env.reset()` permet de ré-initialiser l'environnement
- `env.action_space.n` permet de connaître le nombre d'actions possibles (chaque action est un entier entre 0 et le nombre d'actions exclu) ;
- `obs,reward,done,info = env.step(action)` permet de jouer l'action passée en argument : `obs` contient le nouvel état, `reward` la récompense associée à l'action, `done` un booléen pour savoir si le jeu est fini, info contient des infos auxiliaires non importantes ;
- `env.render()` permet de visualiser le nouvel état ;
- `env.observation_space.n` : soit le nombre d'états si ce nombre est fini discret, soit l'espace de description de l'état si celui-ci est continu (sans le `.n`).
- `env.env.P` : le mdp de la tâche (quand il existe), sous la forme d'un dictionnaire : chaque clé est un état, chaque valeur un dictionnaire ; pour ce 2ème dictionnaire, chaque clé est une action et la valeur associée une liste de tuples `(proba,état,reward,done)`, la probabilité d'atteindre l'état, la récompense associée et si le jeu est fini ou pas. Ainsi, `env.env.P[state][action]` permet de connaître pour un état et une action la liste des états atteignables avec leur probabilité et la récompense associée.

Pour les politiques discrètes, on représente classiquement la politique par un tableau (ou un dictionnaire) associant chaque état à l'action. De même, les fonctions valeurs peuvent être codées par un tableau ou par un dictionnaire.
Coder :

- une fonction `eval_pol(pi, env, gamma, thresh)` qui permet d'estimer la fonction valeur associée à la politique `pi` pour un environnement `env`, avec un decay de la récompense `gamma`. Pour cela, appliquer l'opérateur de Bellman jusqu'à convergence (une différence de moins de `thresh` sur les nouvelles et anciennes valeurs de V, vous pouvez coder la version mdp stochastique pour être plus générique) :

$$V^\pi(s_t) \leftarrow r(s_t, \pi(s_t), s_{t+1}) + \gamma V^\pi(s_{t+1})$$

- une fonction `get_pol(V,env,gamma)` qui permet de déduire la politique optimale en fonction de la fonction valeur `V` : $\pi(s_t) = argmax_a\ r(s_t, a, s_{t+1}) + \gamma V^\pi(s_{t+1})$

- une fonction `policy_iteration(env,gamma,thresh)` qui permet de d'effectuer l'algorithme de policy iteration en alternant les deux fonctions précédentes.
- une fonction `value_iteration(env,gamma,thresh)` qui permet d'effectuer l'algorithme de value iteration :

$$V(s_t) \leftarrow \max_a r(s_t, a, s_{t+1}) + \gamma V(s_{t+1})$$

Tester vos algorithmes avec la classe `AgentPolicy` et comparer.

## Q-Learning

Lorsque le MDP n'est pas disponible, on ne peut plus appliquer les algorithmes précédents. Le QLearning utilise des episodes de jeux pour apprendre itérativement la q-value selon la mise-à-jour suivante : $Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha(r_{t+1}, +\gamma\ max_a\ Q_t(s_{t+1}, a) - Q_t(s_t, a_t))$. Afin d'obtenir des episodes, il est courant d'utiliser une politique $\epsilon$-greedy en suivant la valeur $Q_t$ estimée.
Coder l'algorithme de Q-Learning et appliquez-le à l'environnement `Taxi`. Comparer aux algorithmes précédents.

`FlappyBird` dispose lui d'états continues : il n'est pas possible d'appliquer tel quel le Q-Learning, il faut d'abord discrétiser les états. Proposer une discrétisation et écrire une classe `FlappyAgent` adaptée.