Université Paris Ouest Nanterre
Licence d'Info-Com
2015-2016

Sujets de projet
"Introduction à l'algorithmique"

1 Sujet 1 : jeu de carte "Bataille"

On souhaite jouer à la bataille entre deux joueurs. On dispose d'un jeu de 54 cartes. On distribue les cartes de manière à ce que chaque joueur reçoive 27 cartes. À chaque tour, chaque joueur retourne la carte du dessus du paquet. Le joueur ayant la carte la plus forte empoche les 2 cartes ainsi retournées. Le but du jeu est de récupérer toutes les cartes.

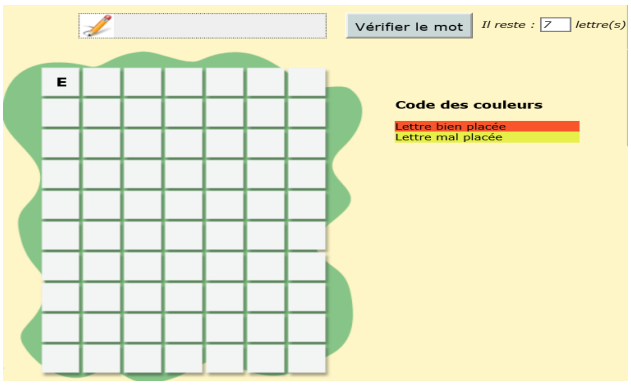
On dit qu'il y a bataille quand les 2 cartes retournées sont de même force (la couleur des cartes n'intervient pas dans le jeu). Quand il y a bataille, on recommence le processus en retournant chacun une carte de son paquet et en comparant leur valeur. Le joueur ayant la plus grosse carte empoche toutes les cartes retournées depuis le début de la bataille. Les cartes sont stockées dans un tableau noté cartes. Chaque joueur possède donc un tableau contenant, au début du jeu, 27 éléments. Le premier élément du tableau représente le dessus du paquet.

On rappelle que pour ajouter ou retirer un élément dans un tableau on utilise :
- tab.append(element) qui ajoute element au tableau tab ;
- tab.remove(element) qui retire element au tableau tab.

Les joueurs sont 2 variables globales du programme (représentés par un tableau) notées joueur1 et joueur2 (initialement vide).

1. Écrire une fonction melange(cartes) qui prend un jeu de 54 cartes en entrée et renvoie le jeu mélangé.
   `Indice :` Le premier tableau contiendra le jeu de carte trié (c'est-à-dire les 4 couleurs placé dans l'ordre 2,3,4,5,6,7,8,9,10,'V','D','R','A'). On pourra créer un deuxième tableau. On choisit aléatoirement une carte dans le tableau cartes, on la retire de ce tableau (ce qui revient à retirer une carte choisie aléatoirement) et on l'ajoute au deuxième tableau (cette fois ci en fin de tableau).
   La méthode *tab.remove(elem)* retire la première occurence de *elem* dans le tableau *tab*. Si un choix aléatoire vous invite à supprimer la deuxième ou troisième occurence d'une carte, la méthode *remove* ne permet de retirer que la première occurence de la carte ; néanmoins comme on ne distingue pas les 4 couleurs d'un jeu, on suppose que cette solution est acceptable.
2. Écrire une procédure **distribue(cartes)** qui distribue les cartes à chaque joueur. Indice : parcourir tout le tableau *cartes* et ajouter une carte successivement au *joueur1* puis au *joueur2*.
3. Écrire une fonction **plusFort(carte1, carte2)** qui renvoie vrai si *carte1* est supérieure à *carte2*. Les 2 cartes sont considérées différentes. L'ordre croissant des cartes est : 2, 3, 4, 5, 6, 7, 8, 9, 10, 'V', 'D', 'R', 'A'.
4. Écrire une procédure **joue1fois()** qui fait jouer les 2 joueurs une seule fois. Pour cela on compare la valeur de la première carte de chaque tableau grâce à la fonction *plusFort()*. Celui qui a la plus grosse carte retire la carte du tableau de l'autre joueur et l'ajoute à son tableau.
5. Écrire une fonction **perdu(joueur1)** qui renvoie vrai si le joueur 1 a perdu.
6. Écrire une fonction **gagne(joueur1)** qui renvoie vrai si le joueur 1 a gagné.
7. Écrire une procédure **joue()** qui joue une partie jusqu'à ce qu'un des deux joueurs perde, et affiche le joueur qui a gagné.

2 Sujet 2 : jeu Motus



2.1 Objectif :

Créer un programme en langage Python qui permette de jouer au **jeu Motus** en solo.

2.2 Règles et déroulement :

Le but du jeu est de retrouver un mot de 7 lettres (pas de verbe conjugué) en un nombre d'essais limités.

La première lettre du mot vous est donnée. Il faut alors proposer un mot et en déduire les lettres qui le composent à l'aide du code couleur suivant :

- Une lettre colorée en ROUGE est dans le mot et est bien placée
- Une lettre colorée en JAUNE est dans le mot mais est mal placée
- Une lettre NON COLORÉE n'est pas dans le mot

Le joueur a le droit à 10 essais, après quoi il a perdu et le mot mystère lui est donné.

2.3 Cahier des charges :

- Écrire un programme en python permettant à l'utilisateur de jouer à ce jeu via une interface graphique créé avec Tkinter. Vous pourrez avoir une fenêtre de saisie et en-dessous faire apparaître au fur et à mesure les mots proposés avec les codes couleurs décrits comme précédemment.
- Vous devrez créer un fichier texte contenant un certain (grand !) nombre de mots à 7 lettres. Un mot sera alors choisi au hasard dans cette liste. L'utilisateur devra alors trouver ce mot avec pour indice de départ la première lettre.
- Le joueur a alors 10 essais en tout avec en indices les lettres bien placées ou celles présentes dans le mot.
- Si le joueur trouve le mot en 10 coups ou moins, lui annoncer qu'il est gagnant et lui proposer une nouvelle partie.
- Si le mot n'est pas trouvé au bout des 10 essais, dire au joueur qu'il a perdu et lui proposer une nouvelle partie.
- Il est fortement conseillé de décomposer votre programme à l'aide de fonctions.

3 Sujet 3 : Le Mastermind

Le projet consiste à écrire un programme en python permettant de jouer au Mastermind.

3.1 Règles et déroulement :

3.1.1 Règles :

Mastermind est un jeu de logique opposant deux joueurs : un *codeur* et un *décodeur*. On dispose d'un stock de pions de 6 couleurs différentes.

Le *codeur* définit en secret une combinaison de 4 pions de couleur pris dans le stock de pions disponibles. Il a le droit de prendre plusieurs fois la même couleur.

Le *décodeur* a pour but de deviner la combinaison secrète des couleurs. Pour cela, il propose des suites de 4 pions de couleur au codeur. Ce dernier compare la proposition du décodeur à sa propre combinaison secrète et doit indiquer :

- combien de pions de couleur de la proposition sont bien placés
- combien de pions de couleur de la proposition sont mal placés

Le *décodeur* dispose de 8 propositions au maximum pour trouver la bonne combinaison de couleurs définie par le codeur.

3.1.2 Déroulement :

a) Pour des raisons de facilité de programmation, on remplacera les couleurs par des chiffres de 0 à 5 et les combinaisons seront donc des suites de 4 chiffres.
Exemple :
- la combinaison secrète du *codeur* est 1 2 1 3
- la combinaison proposée par le *décodeur* est 3 3 1 2
- le nombre de chiffre(s) bien placé(s) est 1 (le 1 en troisième position)
- le nombre de chiffre(s) mal placé(s) est 2 (les chiffres 3 et le 2)

Attention :
- le 1 bien placé ne doit pas être compté une deuxième fois dans les mal placés bien qu'il apparaisse à deux endroits dans la combinaison secrète
- les deux 3 mal placés ne comptent que pour un seul car il n'y a qu'un seul 3 dans la combinaison secrète

Vous devrez veiller à ce que votre programme ne soit pas bloqué si un joueur humain fait une fausse manœuvre (par exemple s'il fait une proposition contenant autre chose que des chiffres compris entre 1 et 8).

b) Dans un deuxième temps, vous allez modifier votre programme afin qu'il joue complètement le rôle du *codeur*, c'est-à-dire qu'il définisse lui même de façon aléatoire une combinaison secrète.

3.2 Cahier des charges :

- Écrire un programme en python permettant à l'utilisateur de jouer à ce jeu via une interface graphique créé avec Tkinter.
- Le *décodeur* a 6 essais en tout, pour trouver la bonne combinaison. Notez que la partie s'arrêtera dans trois cas :
  - **gain** : le nombre de couleurs bien placées est égale à 4
  - **fin des essais :** sur les 6 essais possible le nombre de couleurs bien placées est < 4
  - **abandon** : l'utilisateur décide de quitter la partie
- **Comptage des points :** Sur plusieurs parties, ajoutez 2 compteurs qui indiquent le nombre de parties gagnées et le nombre de parties perdues, ainsi que le nombre moyen de propositions effectuées par partie.

4 Sujet 4 : Jeu des allumettes

Le projet consiste à écrire un programme en python permettant de jouer au jeu des allumettes.

4.1 Règles et déroulement :

Le jeu des allumettes se joue à deux joueurs et 21 allumettes : à tour de rôle, chacun doit retirer soit 1, soit 2, soit 3 allumettes. Le joueur qui prend la dernière allumette a perdu.

Pour votre programme vous pouvez choisir de jouer :
- Soit contre l'ordinateur
- Ou contre un autre joueur

4.2 Cahier des charges :

- Écrire un programme en python permettant à l'utilisateur de jouer à ce jeu via une interface graphique créé avec Tkinter.
- Votre interface doit intégrée les points suivants :
  - 3 boutons **"1 allumette"**, **"2 allumettes"** ou **"3 allumettes**" en fonction du nombre d'allumettes qu'un joueur veut prendre
  - Un bouton **"Pass"** qui permet de passer la main à l'autre joueur (ou à l'ordinateur si vous jouez contre l'ordinateur) après avoir choisi le nombre d'allumettes à retirer
  - Un bouton **"Nouvelle partie"** qui permet de passer à une nouvelle partie
  - Un bouton **"Quitter"** qui va permettre de quitter la partie
- **Comptage des points :** Ajoutez 2 compteurs qui indiquent le nombre de parties gagnées et le nombre de parties perdues.

5 Sujet 5 : Jeu "Pierre, Feuille, Ciseau"

5.1 Objectif :

Il s'agit de réaliser une version simple d'un jeu de pierre feuille ciseau en python avec l'interface graphique Tkinter.

5.2 Règles et déroulement :

- À chaque tour de jeu le joueur choisit pierre feuille ou ciseau ; l'ordinateur aussi. Si les deux ont choisi la même chose ils sont ex-aequo aucun point n'est marqué. Sinon le joueur marque un point s'il gagne et perd un point si c'est l'ordinateur qui gagne.
- On rappelle que la pierre l'emporte sur le ciseau qui l'emporte sur la feuille qui elle même l'emporte sur la pierre.
- Pour faire jouer l'ordinateur aléatoirement, on mettra `from random import *` au début du programme
- À la fin de chaque coup, l'ordinateur demande à l'utilisateur s'il veut continuer et il y a un affichage des scores.

5.3 Cahier des charges :

- Écrire un programme en python permettant à l'utilisateur de jouer à ce jeu via une interface graphique créé avec Tkinter.
- Votre interface doit intégrée les points suivants :
  - Afficher les trois boutons : "Pierre", "Feuille" et "Ciseau", afin de permettre à l'utilisation de sélectionner son choix
  - Deux cases permettant d'afficher le choix du joueur et de l'ordinateur
  - Afficher 2 compteurs qui indiquent le score des deux joueurs (utilisateur et ordinateur)
  - Les boutons **"Recommencer"**, permettant de recommencer une nouvelle partie et le bouton **"Quitter"** pour quitter le jeu

Exemple d'interface Tkinter :

6 Projet 6 : Bandit manchot

6.1 Objectif :

On souhaite réaliser une application sous Tkinter permettant de jouer au bandit manchot.

6.2 Règles et déroulement :

Un bandit manchot est une machine de casino. L'application devra respecter les règles du jeu suivantes : Pour jouer, on mise une certaine somme. Puis trois rouleaux (ou n) tournent et s'arrêtent chacun aléatoirement sur l'un des symboles qu'ils portent. Si tous les symboles sont égaux, on gagne le jackpot (un certain multiple de la mise). Si la moitié ou plus des rouleaux sont sur la même position, on gagne exactement sa mise. Sinon, le gain est nul.

6.3 Cahier des charges :

- Écrire un programme en python permettant à l'utilisateur de jouer à ce jeu via une interface graphique créé avec Tkinter.
- Votre interface doit intégrée les points suivants :
  - Une zone de saisie de la mise engagée
  - Un bouton pour actionner le mouvement des trois rouleaux
  - 3 cases permettant d'afficher le résultats des trois rouleaux
  - Une zone d'affichage pour le gain du joueur et le gain du casino
  - Afficher également la statistique suivante : nombre de fois où chaque rouleau s'est arrêté sur chaque position.
  - Un bouton **"Nouvelle partie"** qui permet de passer à une nouvelle partie
  - Un bouton **"Initialiser"** qui permet de de remettre les compteurs à zéro
  - Un bouton **"Quitter"** qui va permettre de quitter la partie