Séance 5: Jouer à Pierre-Feuille-Ciseaux

Université de Paris Cité

Objectifs:

- Un programme qui interagit avec l'utilisateur
- Création d'un petit jeu en plusieurs manches

L'objectif de ce TP est de programmer un jeu de Pierre-Feuille-Ciseaux dont les deux joueurs sont l'ordinateur et l'utilisateur. Une partie se jouera en une succession de manches et le programme maintiendra une fiche de score afin de déterminer le vainqueur à la fin.

Lors de ce TP, il faudra donc faire interagir l'utilisateur de votre programme avec l'ordinateur en lui demandant son coup. Il faudra interpréter sa réponse et réagir en conséquence. Il faudra également tenir à jour une fiche de score.

Pour cette séance, les programmes des exercices de 2 à 4 sont à réaliser dans un même fichier que vous enregistrerez dans le bon sous-répertoire.

Exercice 1 (Préliminaires : un générateur pseudo-aléatoire, $\star$)
Dans cet exercice nous allons nous familiariser avec la fonction `randint` *du module* `random`*, un générateur pseudo-aléatoire des nombres entiers. La but de l'exercice est de définir des procédures qui affichent les résultats de différentes utilisations de* `randint`*. La fonction* `randint` *sera ensuite utilisée dans les exercices suivants pour simuler le coup ordinateur dans le jeu Pierre-Feuille-Ciseaux.*

1. *Créez un fichier nommé* `des.py` *et écrivez le code suivant dans ce fichier :*

```
import random
def print_throw():
  n = random.randint(1, 6)
  print(n)
```

2. *Lancez l'interpréteur Python (avec la touche F5 si vous êtes dans le REPL de Python), et appelez plusieurs fois la procédure* `print_throw`*. Vous rappelez-vous ce que fait la fonction* `randint` *du module* `random` *? Que fait la procédure* `print_throw` *? Quelles valeurs peuvent être affichées par cette procédure ?*
3. *Modifiez la procédure* `print_throw` *de façon à ce qu'elle prenne un argument,* k*, et affiche le résultat de lancer un dé équitable avec* k *faces numérotées de* 1 *à* k*.*
4. *Écrivez une boucle qui permet de lancer un dé avec* 9 *faces* 42 *fois, et affiche les 42 résultats.*
5. *Dans le fichier* `des.py`*, écrire une procédure* `print_sum_throws`*, qui prend un argument* n*, qui simule le lancer d'un dé avec* 6 *faces* n *fois, et qui affiche la somme. Quelle valeur trouvez-vous si vous appelez* `print_sum_throws(1000)` *? Cela suggère-t-il qu'il s'agit d'un dé équitable (essayer plusieurs fois l'appel de* `sum_throws(1000)` *pour comprendre la type de distribution qu'elle génère) ?*

□

Exercice 2 (Jeu en une manche, $\star\star$)
Dans cet exercice, nous créerons un jeu de Pierre-Feuille-Ciseaux en une manche.

Rappel : *Le jeu de Pierre-Feuille-Ciseaux est un jeu de mains où chaque joueur choisit simultanément un élément parmi pierre, feuille et ciseaux. Le vainqueur est décidé selon les principes suivants : la pierre émousse les ciseaux, les ciseaux coupent la feuille, la feuille enveloppe la pierre.*

1. *Écrire une fonction* `convert` *qui prend en paramètre un entier* `n` *et renvoie une chaîne de caractères selon la valeur de* `n` *:*
   — *si* `n` *vaut* `1`*, alors* `convert(n)` *retourne* `"Pierre"`*,*
   — *si* `n` *vaut* `2`*, alors* `convert(n)` *retourne* `"Feuille"`*,*
   — *sinon* `convert(n)` *retourne* `"Ciseaux"`*.*
2. *Écrire une fonction* `tirage` *sans paramètre qui retourne dans une chaîne de caractères le choix – aléatoire– de l'ordinateur.*

   Contrat:
   Un appel de `tirage` *retourne une chaîne de caractères contenant aléatoirement* `"Pierre"`*,* `"Feuille"` *ou* `"Ciseaux"`*.*

   Indice : *utiliser la fonction* `randint` *du module* `random`*, déjà discutée dans l'Exercice 1 et la fonction* `convert` *décrite ci-dessous.*
3. *À l'aide d'une boucle, tester la fonction* `tirage` *un grand nombre de fois pour vérifier qu'elle semble bien uniformément aléatoire.*
4. *Écrire une fonction* `coupJoueur` *sans paramètre qui utilise la fonction* `input()` *pour demander le choix du joueur et le retourne dans une chaîne de caractères. Si le joueur n'entre pas un coup valide, il conviendra de lui demander de recommencer.* ***Indice :*** *pour cela, rappelez vous la boucle* `while` *qui permet de répéter des commandes tant qu'une condition est vrai. . .*

   Contrat:
   Un appel de `coupJoueur` *demande à l'utilisateur de choisir entre Pierre, Feuille et Ciseaux et retourne une chaîne de caractères correspondant à ce choix.*
5. *Écrire une procédure* `uneManche` *qui fait jouer une partie en une manche de Pierre-Feuille-Ciseaux entre l'utilisateur et l'ordinateur. Le programme affichera un message donnant les coups joués et le résultat de la partie (victoire du joueur, de l'ordinateur ou partie nulle).*

□

Exercice 3 (Jeu en plusieurs manches, $\star$)
Dans cet exercice, nous créerons le jeu de Pierre-Feuille-Ciseaux en plusieurs manches.

1. *En vous basant sur la procedure* `uneManche`*, écrivez une nouvelle fonction* `manche` *qui fait jouer à Pierre-Feuille-Ciseaux l'ordinateur contre l'utilisateur et qui renvoie une chaîne de caractères indiquant le vainqueur.*

   Contrat:
   Un appel de `manche()` *fait jouer une manche à l'utilisateur et l'ordinateur et retourne le résultat sous la forme d'une chaîne de caractères :*
   — `"J"` *si le joueur remporte la manche*
   — `"O"` *si l'ordinateur remporte la manche*
   — `"E"` *s'il y a égalité*
2. *Écrire une procedure* `chifoumi(n)` *qui fait jouer une partie de Pierre-Feuille-Ciseaux en* `n` *manches entre l'utilisateur et l'ordinateur. La procedure doit afficher le vainqueur, son nombre de victoires et un résumé de la partie sous forme de chaîne de caractères.*

   Contrat:
   Un appel de `chifoumi(5)` *fait jouer* `5` *manches de Pierre-Feuille-Ciseaux à l'ordinateur contre l'utilisateur, et il affiche à la fin le nom du vainqueur ainsi que le résultat de chaque manche sur une seule ligne :* `JOEJJ` *par exemple si le joueur utilisateur a gagné la première, la quatrième et la cinquième manche, que la seconde manche a été nulle et que la troisième a été une égalité.*

□

Exercice 4 ((Bonus) Pour aller plus loin, $\star$— $\star\star\star$)
Dans cet exercice, nous ajoutons quelques fonctionnalités à notre jeu. Chaque question est indépendante.

Pour des raisons de sécurité, il est conseillé de travailler sur une copie `chifoumi2(n)` *du programme de l'exercice précédent.*

1. *Modifier votre programme pour que le joueur choisisse le nombre de manches avant le début de la partie.*
2. *Modifier votre programme pour qu'il propose de rejouer une partie lorsqu'une partie est terminée.*
3. *Proposez une (ou plusieurs) fonction(s)* `coupOrdi` *qui choisi(sen)t le coup de l'ordinateur selon une autre stratégie que le tirage aléatoire uniforme. Vous pouvez utiliser, ou non, un ou plusieurs paramètres, selon vos envies. Testez vos stratégies face à la stratégie du tirage aléatoire uniforme.*
4. *Modifier votre programme pour qu'il vous fasse jouer à Pierre-Feuille-Ciseaux-Lézard-Spock.*
   Les premières règles s'appliquent toujours, mais il faut ajouter que le lézard mange le papier, empoisonne Spock, est écrasé par la pierre et est décapité par les ciseaux. Spock vaporise la pierre, casse les ciseaux, et il est discrédité par le papier.

□