1 Boite à outil

Pour les exercices de cette feuille, vous aurez besoin, entre autres, des outils suivants :

1. Boucles while et boucle for : voir les TP 1 et 2
2. Fonction `int` qui convertit un décimal en entier (en tronquant ce qu'il y a après la virgule), ou qui transforme une chaîne de caractères en sa valeur.
   Par exemple `int('2')` va donner l'entier 2 (au lieu de la chaine '2').
3. Instruction `input` dont la syntaxe est `var=input(text)` qui demande à l'utilisateur d'écrire quelque chose et qui enregistre ce que l'utilisateur a écrit dans la variable intitulée `var`. L'argument `text` permet d'introduire la demande par un texte.

   Par exemple :
   `age=input('quel age avez vous ? ')`
   a pour effet d'afficher le texte "quel age avez vous ?" et python attend la réponse de l'utilisateur. La réponse est alors enregistrée *avec le type* `str` dans une variable intitulée `age`.
4. La fonction `randint(a,b)`, à importer depuis le module `random`. Chaque fois que l'on exécute `randint(a,b)`, un nombre entier au hasard entre a et b est répondu. Pour importer la fonction randint depuis random :

   ```
   from random import randint
   ```

5. L'équivalent de la fonction `randint` pour des nombres réels est `uniform(a,b)`.

Exercice 1 (Échauffement)
L'informatique est souvent utilisé pour simuler des phénomènes aléatoires : on crée des programmes qui se comportent de telle façon que leurs réponses sont "au hasard".

Si on tape `randint(1,6)`, cela simule ainsi parfaitement un lancer de dé.

1. Essayez dans le shell pour constater que cela donne bien des nombres "au hasard" entre 1 et 6. On verra dans un futur TP comment mieux vérifier l'aspect uniforme...
2. Simulez 100 lancers de dé, et vérifiez à l'oeil nu que cela donne quelque chose d'aléatoire.
3. Testez de même la fonction uniform entre `e` et `pi`.
4. Tirez un nom au hasard dans la liste Ruby, Ourane, Elena. Écrire une fonction qui prend en entrée une liste de prénom et en renvoie un de manière aléatoire (on appellera ce programme "Courtepaille" et le perdant sera dévoré)

Exercice 2 (Qui veut gagner des millions)

1. Une phase de jeu "qui veut gagner des millions" se déroule comme suit :
   a) Le présentateur pose une question à 4 possibilités, numerotées de 1 à 4.
   b) Le joueur répond un nombre de 1 à 4.
   c) Le présentateur demande si c'est le dernier mot du joueur, et tant que le joueur ne répond pas par l'affirmative, on repose la question et on laisse le joueur choisir une autre réponse.

   Programmer une phase de jeu : l'ordinateur est le présentateur et votre voisin sera le joueur (c'est vous qui aurez écrit la question et les propositions de réponses). On utilisera l'instruction `print` pour afficher le texte de la question et `input` pour demander la réponse du joueur.
2. Bonus (à faire après l'exercice 3 si vous êtes en avance) : comment gérer plusieurs questions ? Créer une banque de dix questions et faire en sorte que le programme pose une question prise au hasard dans cette banque de questions.

Exercice 3 (simuler des lancers de dés)
1. Ecrire un programme qui simule la répétition de lancer de dés jusqu'à obtenir un 6 et qui retourne le nombre de lancers qui a été nécessaire à l'obtention de ce 6.
2. Proposer un programme qui estime le nombre moyen de lancers nécessaires à l'obtention d'un 6.

Exercice 4 (Cambriolage)
Vous êtes un cambrioleur et voulez ouvrir un coffre fort dont la combinaison est un nombre entier entre 1 et 100. Vous disposez de 6 essais avant que l'alarme ne se déclenche, mais vous disposez d'un outil qui vous dit si votre proposition est plus petite ou plus grande que la combinaison du coffre

Le but de l'exercice est d'écrire un mini jeu simulant ce cambriolage : l'ordinateur générera un nombre au hasard entre 1 et 100, et le joueur devra le trouver. L'ordinateur affichera après chaque essai si la proposition du joueur est plus petite ou plus grande que le code.

1. Créez ce jeu, dans un premier temps sans prendre en compte la limite de 6 essais, puis en ajoutant cette contrainte.
2. Bonus mathématique : combien d'essais est-il raisonnable de donner si la combinaison est entre 1 et 1000 ?

Exercice 5 (Shi-fu-mi) Le jeu du Shifumi, ou "pierre-feuille-ciseaux", est un jeu de hasard à deux joueurs (avec variantes possibles à plus). Les joueurs choisissent simultanément de jouer Pierre, Feuille ou Ciseaux, et on compare les résultats avec les règles suivantes :

— La Pierre casse les Ciseaux (la Pierre gagne)
— Les Ciseaux coupent la Feuille (les Ciseaux gagnent)
— La Feuille recouvre la Pierre (la Feuille gagne)

Pour cet exercice, on va jouer contre l'ordinateur, que l'on va faire jouer totalement au hasard. On considère qu'on aura gagné la partie si on remporte 3 manches.

Programmez ce jeu.

1 Boite à outil

Pour les exercices de cette feuille, vous aurez besoin, entre autres, des outils suivants :

1. Boucles while et boucle for : voir les TP 1 et 2
2. Fonction `int` qui convertit un décimal en entier (en tronquant ce qu'il y a après la virgule), ou qui transforme une chaîne de caractères en sa valeur.
   Par exemple `int('2')` va donner l'entier 2 (au lieu de la chaine '2').
3. Instruction `input` dont la syntaxe est `var=input(text)` qui demande à l'utilisateur d'écrire quelque chose et qui enregistre ce que l'utilisateur a écrit dans la variable intitulée `var`. L'argument `text` permet d'introduire la demande par un texte.

   Par exemple :
   `age=input('quel age avez vous ? ')`
   a pour effet d'afficher le texte "quel age avez vous ?" et python attend la réponse de l'utilisateur. La réponse est alors enregistrée *avec le type* `str` dans une variable intitulée `age`.
4. La fonction `randint(a,b)`, à importer depuis le module `random`. Chaque fois que l'on exécute `randint(a,b)`, un nombre entier au hasard entre a et b est répondu. Pour importer la fonction randint depuis random :

   ```
   from random import randint
   ```

5. L'équivalent de la fonction `randint` pour des nombres réels est `uniform(a,b)`.

Exercice 1 (Échauffement)
L'informatique est souvent utilisé pour simuler des phénomènes aléatoires : on crée des programmes qui se comportent de telle façon que leurs réponses sont "au hasard".

Si on tape `randint(1,6)`, cela simule ainsi parfaitement un lancer de dé.

1. Essayez dans le shell pour constater que cela donne bien des nombres "au hasard" entre 1 et 6. On verra dans un futur TP comment mieux vérifier l'aspect uniforme...
2. Simulez 100 lancers de dé, et vérifiez à l'oeil nu que cela donne quelque chose d'aléatoire.
3. Testez de même la fonction uniform entre `e` et `pi`.
4. Tirez un nom au hasard dans la liste Ruby, Ourane, Elena. Écrire une fonction qui prend en entrée une liste de prénom et en renvoie un de manière aléatoire (on appellera ce programme "Courtepaille" et le perdant sera dévoré)

Exercice 2 (Qui veut gagner des millions)

1. Une phase de jeu "qui veut gagner des millions" se déroule comme suit :
   a) Le présentateur pose une question à 4 possibilités, numerotées de 1 à 4.
   b) Le joueur répond un nombre de 1 à 4.
   c) Le présentateur demande si c'est le dernier mot du joueur, et tant que le joueur ne répond pas par l'affirmative, on repose la question et on laisse le joueur choisir une autre réponse.

   Programmer une phase de jeu : l'ordinateur est le présentateur et votre voisin sera le joueur (c'est vous qui aurez écrit la question et les propositions de réponses). On utilisera l'instruction `print` pour afficher le texte de la question et `input` pour demander la réponse du joueur.
2. Bonus (à faire après l'exercice 3 si vous êtes en avance) : comment gérer plusieurs questions ? Créer une banque de dix questions et faire en sorte que le programme pose une question prise au hasard dans cette banque de questions.

Exercice 3 (simuler des lancers de dés)

1. Ecrire un programme qui simule la répétition de lancer de dés jusqu'à obtenir un 6 et qui retourne le nombre de lancers qui a été nécessaire à l'obtention de ce 6.
2. Proposer un programme qui estime le nombre moyen de lancers nécessaires à l'obtention d'un 6.

Exercice 4 (Cambriolage)
Vous êtes un cambrioleur et voulez ouvrir un coffre fort dont la combinaison est un nombre entier entre 1 et 100. Vous disposez de 6 essais avant que l'alarme ne se déclenche, mais vous disposez d'un outil qui vous dit si votre proposition est plus petite ou plus grande que la combinaison du coffre

Le but de l'exercice est d'écrire un mini jeu simulant ce cambriolage : l'ordinateur générera un nombre au hasard entre 1 et 100, et le joueur devra le trouver. L'ordinateur affichera après chaque essai si la proposition du joueur est plus petite ou plus grande que le code.

1. Créez ce jeu, dans un premier temps sans prendre en compte la limite de 6 essais, puis en ajoutant cette contrainte.
2. Bonus mathématique : combien d'essais est-il raisonnable de donner si la combinaison est entre 1 et 1000 ?

Exercice 5 (Shi-fu-mi) Le jeu du Shifumi, ou "pierre-feuille-ciseaux", est un jeu de hasard à deux joueurs (avec variantes possibles à plus). Les joueurs choisissent simultanément de jouer Pierre, Feuille ou Ciseaux, et on compare les résultats avec les règles suivantes :

— La Pierre casse les Ciseaux (la Pierre gagne)
— Les Ciseaux coupent la Feuille (les Ciseaux gagnent)
— La Feuille recouvre la Pierre (la Feuille gagne)

Pour cet exercice, on va jouer contre l'ordinateur, que l'on va faire jouer totalement au hasard. On considère qu'on aura gagné la partie si on remporte 3 manches.

Programmez ce jeu.