INFO-F101 – Programmation
Projet 1
Blackjack

Année académique 2017–2018

Le Blackjack est un jeu de casino apparu en France au dix-huitième siècle sous le nom de vingt et un. Pour ce premier projet, nous vous demandons d'en réaliser une version simplifiée (pour un seul joueur) dont les règles sont décrites ci après.

Règles

Le jeu se joue avec des cartes à jouer classiques et oppose le joueur à la banque, qui est jouée par le croupier. La valeur des cartes est répartie comme suit :

— As : 1 ou 11 points, au choix du joueur
— Cartes de 2 à 10, la valeur faciale de la carte (le 2 vaut 2 points et ainsi de suite)
— Les figures (Valet, Dame ou Roi) valent chacune 10 points.

Le but est de battre la banque sans dépasser 21 points. Lorsque le joueur dépasse 21 points, on dit qu'il saute et il perd sa mise. Au début de chaque partie, le croupier distribue une première carte au joueur. Le joueur peut ensuite demander autant de cartes qu'il le souhaite. Une fois que le joueur a terminé, le croupier joue pour la banque. La banque joue toujours de la même manière : elle tire une carte supplémentaire tant qu'elle est en dessous de 17 et s'arrête à partir de 17. Si un as permet à la banque d'obtenir 17 ou plus en choisissant la valeur 11, elle ne tentera pas de tirer une carte supplémentaire.

Gains

— Si la banque dépasse 21, elle perd et les joueurs qui n'avaient pas sauté gagnent leur mise.
— Si la banque obtient un nombre entre 17 et 21, les joueurs en dessous du nombre perdent leur mise.

— En cas d'égalité avec la banque, le joueur récupère sa mise mais ne gagne rien. Il est à noter que 21 composé d'une carte valant 10 (un 10 ou une figure) avec un as se dit "Blackjack" et bat un 21 obtenu avec 3 cartes ou plus.
— Les joueurs ayant plus que la banque gagnent leur mise.

Modalités

Nous vous demandons d'implémenter ce jeu en python3 et de permettre à un joueur de jouer contre votre programme. Comme il n'y a qu'un seul joueur, lorsque celui-ci saute, la banque n'a pas besoin de jouer et empoche directement la mise du joueur. Afin de choisir aléatoirement une carte, vous ferez usage de la librairie `random`, dont voici un exemple d'utilisation :

```
>>> from random import randint
>>> print(randint(1,13))
# renvoie un nombre entier aléatoire inclus entre 1 et 13 (bornes comprises)
>>> 2
```

La fonction `randint(int,int)` vous permet d'obtenir un nombre aléatoire entre deux bornes, vous prendrez 1 et 13 comme bornes et considérerez que 1=as, 11=valet, 12=dame, 13=roi et les valeurs entre 2 et 10 la carte correspondante. Ces valeurs n'ont rien à voir avec les points que valent les cartes, ils vous permettent de tirer une carte aléatoirement. Vous devez prendre ces valeurs et aucunes autres.

En procédant de la sorte, deux exécutions successives du programme donneront des résultats différents. Ce qui n'est pratique ni pour débugger, ni pour corriger. Il vous est dès lors demandé d'initialiser le générateur de nombres aléatoires grâce à la fonction `seed(int)` du module `random`. Deux exécutions avec la même "graine" donneront alors les mêmes résultats. Voici un exemple d'utilisation de `seed()`

```
>>> from random import randint, seed
>>> graine=int(input("Entrez la graine: "))
>>> seed(graine)
```

Votre programme commencera par demander à l'utilisateur (grâce à la fonction `input()`) un nombre afin d'initialiser le générateur de nombre aléatoires. Le générateur de nombre aléatoires doit être initialisé une et une seule fois au début de l'exécution du programme et ce, avec la graine donnée par l'utilisateur. Il ne doit pas y avoir plus d'appels à `randint` que de cartes tirées.

La gestion du portefeuille du joueur est très simple. Lorsqu'il mise, le montant de la mise est diminuée du portefeuille et celui-ci peut avoir une valeur négative. Par exemple, si le joueur possède 10 euros, le programme ne l'empêchera pas de miser 30 euros, le portefeuille aura alors la valeur de -20. Lorsque le joueur a égalité ou a gagné, il récupère sa mise additionnée à l'éventuel gain.

Vous ne devez pas prendre en considération le cas où l'utilisateur entrerait une valeur aberrante, par exemple `quit` au lieu de 2.

Le programme doit être exécutable via la commande suivante aux salles machines du bâtiment NO :

```
~>python3 projet1.py
```

Voici un exemple d'exécution du programme attendu :

```
Bienvenue au Blackjack
Entrez la graine : 12
Veuillez entrer la quantité d'argent en votre possession : 100
Veuillez entrer votre mise (vous avez : 100) : 10
La carte tirée est : 8
Souhaitez-vous une carte ? (1: oui, 2: non) 1
La carte tirée est : 5
Souhaitez-vous une carte ? (1: oui, 2: non) 1
La carte tirée est : valet
Vous avez sauté

Souhaitez-vous jouer une autre partie ? (1: oui, 2: non) 1
Veuillez entrer votre mise (vous avez : 90) : 10
La carte tirée est : 9
Souhaitez-vous une carte ? (1: oui, 2: non) 1
La carte tirée est : valet
Souhaitez-vous une carte ? (1: oui, 2: non) 2
Vous avez obtenu 19 points
La banque joue :
La carte tirée est : 6
La carte tirée est : 3
La carte tirée est : 7
La carte tirée est : as
La banque a obtenu 17 points
Vous gagnez 10

Souhaitez-vous jouer une autre partie ? (1: oui, 2: non) 2
```

Quelques jours après la publication de cet énoncé, un module sur UpyLab (`http://upylab.ulb.ac.be`) vous permettra de tester votre solution. Veuillez à reproduire à l'identique l'output de l'exemple de fonctionnement ci-dessus afin de permettre l'exécution de tests automatiques sur votre solution. Un fichier contenant les chaînes de caractères utilisées vous est fourni sur l'Université virtuelle : `projet1.py`, il vous est demandé de l'utiliser. Vous ne devriez pas avoir besoin d'autres chaînes de caractères. Afin d'afficher une chaîne de caractères sans passer ensuite à la ligne, vous transmettrez un second paramètre à la fonction `print()` de la manière suivante :

```
print(maChaine,end="")
```

Pour afficher à l'écran plusieurs chaînes de caractères sur une même ligne, vous pouvez également concaténer celles-ci grâce à l'opérateur + de la manière suivante :

```
print(maChaine+uneAutre+" "+str(88)+" fois "+str(val))
```

Notez l'utilisation de `str()` avec un nombre en paramètre afin de transformer celui-ci en chaîne de caractère pouvant être concaténée.

Veillez à commenter votre programme avec pertinence.

Conseil

Le cas de l'As pourrait vous présenter quelque difficulté, si c'est le cas, ne perdez pas trop de temps dessus de prime abord. Réalisez d'abord une version simplifiée où l'As vaut toujours 1 et attaquez-vous à ce problème en dernier, lorsque tout le reste aura été implémenté. Notez que le joueur (ou la banque dans l'exemple ci-dessus) ne spécifie pas explicitement la valeur qu'il choisit, la valeur à son avantage est automatiquement choisie.

Consignes pour la remise du projet

Le projet devra être remis via deux canaux : une version imprimée au secretariat étudiant (Maryka Peetroons - P.2N8.104) et une version électronique sur l'Université virtuelle. Pour le document imprimé, n'utilisez pas de farde et/ou chemise en plastique. Si il comporte plusieurs pages, agrafez les. Veillez à soigner la présentation de ce document. Même imprimé en noir et blanc, utilisez la coloration syntaxique afin d'en améliorer la lisibilité et évitez les retours à la ligne non indentés pour vos commentaires. Les consignes pour la remise du projet sont disponibles en ligne sur la page du cours sur l'Université Virtuelle. Les consignes sont à respecter *scrupuleusement* ; relisez-les attentivement avant la remise !

Votre projet sera testé à l'aide de la commande suivante : `python3 projet1.py`.

Pour toute question concernant l'énoncé, nous vous invitons à vous adresser à Cédric Ternon (cternon@ulb.ac.be)

Date limite de remise. Le lundi 16 octobre 2017 à 13h.