

Blackjack

Описание проекта

Вам предлагается в рамках заключительного проекта по курсу “Python Core” реализовать консольную версию карточной игры blackjack. Здесь могут быть полезны все приобретенные знания и навыки в рамках курса.

Реализация данной игры в качестве заключительного проекта была выбрана не случайно: это довольно примитивная игра с довольно простыми правилами - от слушателей курса не потребуется особой алгоритмической подготовки. Мы намеренно немного упростили некоторые правила игры, исключили некоторые распространенные вариации хода игры, поэтому игра в этом проекте может немного отличаться от тех, что привыкли видеть в игорных домах и казино (мы исключили такие опции как сплит, страхование, а также вариации коэффициентов при выигрыше в зависимости от тех или иных условий). Тем не менее для успешного выполнения потребуется использовать многие аспекты объектно-ориентированного программирования, озвученные в курсе.

Результатом вашего труда будет полная картина о разработке отдельного проекта с использованием разных подходов и возможностей языка Python.

Описание игры blackjack

Blackjack

карточная игра, в которой игрок соревнуется с дилером. Исходом может быть проигрыш (игрок теряет ставку), ничья (игрок остается при своей ставке) и победа (игрок получает выигрыш равный удвоенной ставке). Используется стандартная колода из 52 карт

Список терминов:

Игрок Player

игрок (в нашем случае тот, за кого будем играть)

Дилер Dealer

сотрудник казино, принимающий участие в игре и принимающий ставки. Мы будем считать, что у дилера неограниченное количество фишек

Рука набор карт, которые получил игрок/дилер

Ставка целое число фишек, от которого зависит сколько игрок потеряет или получит в результате игры

Еще взять еще одну карту на руку

Хватит отказаться от дальнейшего набора карт (игрок в таком случае после stand добирать карты и не может)

Удвоить удвоить ставку и взять карту

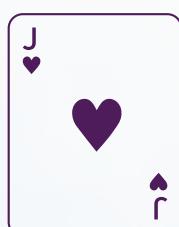
Фишка численный эквивалент ставки. Все ставки, проигрыши и выигрыши являются целым числом фишек

Блэкджек ситуация, когда сумма очков первых двух карт на руке - 21

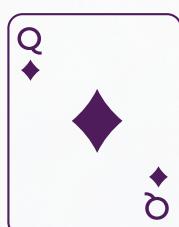
Очки на картах (от масти не зависит)

2, 3, 4, 5, 6, 7, 8, 9, 10

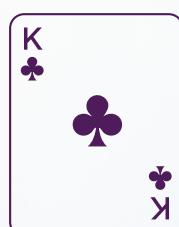
число очков соответствует номиналу карты (т.е. 1-1, 2-2 и т.д.)



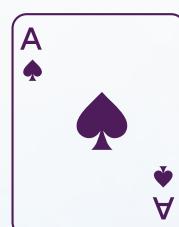
Jack
валет



Queen
королева



King
король



Ace
туз

10 очков

1 или 11 очков

если сумма очков на картах руки без этого туза меньше 11, то 11 очков, иначе - 1

Есть особое правило для дилера - он обязан брать карту, пока у него меньше 17 очков. Даже если игрок отказался от дальнейшего набора карт при меньшей сумме или имеет перебор.

Особое правило для игрока - если после раздачи двух карт в начале партии у него блэкджек, то он выигрывает автоматически независимо от того, какая карта у дилера.

Выигрывает тот, у кого очков на картах больше, чем у соперника и отсутствует перебор. При равном числе очков - ничья. При переборе у одного выигрывает другой. При переборе и у игрока, и у дилера - ничья.

Ход игры

В начале игры игрок получает две карты в закрытую (дилер их не видит). Дилер получает одну карту в открытую (игрок ее видит), и вообще рука дилера - открыта.

Если у игрока блэкджек - он выигрывает. Иначе дилер берет карту. Далее они по очереди добирают карты.

У игрока **три опции**

У дилера **две опции**

После stand игрок или дилер не могут добирать карт - у того, кто выбрал stand, stand до конца игры

Замечание: дилер не может выбрать stand, если у него на руке менее 17 очков

В конце партии игрок получает удвоенную ставку в случае победы, ставку - в случае ничьей. В случае поражения игрока все фишки получает дилер

Примеры партий

```
...
game start
player's bet 10 chips, player's start hand: (10, A), dealer's start hand: (10)
game finish
results: player: 21 (blackjack), dealer: 10, player won
```

```
ooo
game start
player's bet 5 chips, player's start hand: (7, K), dealer's start hand: (8)
dealer's move: Hit
player's bet 5 chips, player's start hand: (7, K), dealer's start hand: (8, 5)
players's move: Hit
player's bet 5 chips, player's start hand: (7, K, 2), dealer's start hand: (8, 5)
dealer's move: Hit
player's bet 5 chips, player's start hand: (7, K, 2), dealer's start hand: (8, 5, A)
players's move: Stand
player's bet 5 chips, player's start hand: (7, K, 2), dealer's start hand: (8, 5, A)
dealer's move: Hit
player's bet 5 chips, player's start hand: (7, K, 2), dealer's start hand: (8, 5, A, 6)
dealer's move: Stand
player's bet 5 chips, player's start hand: (7, K, 2), dealer's start hand: (8, 5, A, 6)
game finish
results: player: 19, dealer: 20, player lost
```

Каждая карта содержит также масть. Мы ее не отображали, чтобы сфокусировать внимание на очках на картах. Вам же необходимо будет отображать масти карт. (Игра на одной колоде, повторяющихся карт нет)

Технические требования проекта

Структура

Решением слушателя курса должна быть директория со всеми необходимыми файлами, которая называется так
`python_core_final_project_<фамилия>_<имя>`
(надо подставить свои фамилию и имя латиницей)

Внутри директории должен присутствовать файл
`blackjack.py`
(это скрипт игры на python)

Разрешается и даже рекомендуется разбивать решение по модулям и пакетам

Интерфейс

Пользователь должен иметь возможность играть (за игрока), используя интерфейс командной строки.

При запуске игры первым делом выводится сообщение:

```
ooo
***Start blackjack game***
PLAYER'S BET: <INPUT>
```

с предложением ввести начальную ставку игрока,
куда будет вводится целое число фишек

После номинала каждой карты должны выводится ее масть (suit). Их 4 вида

clubs

diamonds

hearts

spades

Далее вводится информация при раздаче карт (пример)

```
...
__player's bet: 5 chips
__player's start hand: (7-clubs, K-spades)
__dealer's start hand: (8-hearts)
```

Затем ходы игрока и дилера. Игрок должен вводить
тип хода (hit, stand, double down) без учета регистра.

Далее все будет выглядеть, как в примере:

```
...
DEALER'S MOVE: Hit
__player's bet: 5 chips
__player's start hand: (7-clubs, K-spades)
__dealer's start hand: (8-hearts, 5-clubs)
PLAYER'S MOVE: Hit
__player's bet: 5 chips
__player's start hand: (7-clubs, K-spades, 2-hearts)
__dealer's start hand: (8-hearts, 5-clubs)
DEALER'S MOVE: Hit
__player's bet: 5 chips
__player's start hand: (7-clubs, K-spades, 2-hearts)
__dealer's start hand: (8-hearts, 5-clubs, A-diamonds)
PLAYER'S MOVE: Stand
__player's bet: 5 chips
__player's start hand: (7-clubs, K-spades, 2-hearts)
__dealer's start hand: (8-hearts, 5-clubs, A-diamonds)
DEALER'S MOVE: Hit
__player's bet: 5 chips
__player's start hand: (7-clubs, K-spades, 2-hearts)
__dealer's start hand: (8-hearts, 5-clubs, A-diamonds, 6-spades)
DEALER'S MOVE: Stand
__player's bet: 5 chips
__player's start hand: (7-clubs, K-spades, 2-hearts)
__dealer's start hand: (8-hearts, 5-clubs, A-diamonds, 6-spades)
***Game finish***
RESULTS:
__player: 19
__dealer: 20
__player lost
```

Если для руки игрока или дилера есть особая ситуация в конце
игры (blackjack, bust), то эта информация должны быть выведена
в скобках сразу после количества очков (пример):

```
ooo  
RESULTS:  
__player: 21 (blackjack)  
__dealer: 10  
__player won
```

(последняя строка результата принимает одно из трех значений: player lost, draw, player won)

После строк

```
ooo  
PLAYER'S BET:  
DEALER'S MOVE:
```

ожидается ввод игрока на этой же строке

Пример целой игры:

```
ooo  
***Start blackjack game***  
PLAYER'S BET: 5  
__player's bet: 5 chips  
__player's start hand: (7-clubs, K-spades)  
__dealer's start hand: (8-hearts, 5-clubs)  
DEALER'S MOVE: Hit  
__player's bet: 5 chips  
__player's start hand: (7-clubs, K-spades)  
__dealer's start hand: (8-hearts, 5-clubs)  
PLAYER'S MOVE: Hit  
__player's bet: 5 chips  
__player's start hand: (7-clubs, K-spades, 2-hearts)  
__dealer's start hand: (8-hearts, 5-clubs)  
DEALER'S MOVE: Hit  
__player's bet: 5 chips  
__player's start hand: (7-clubs, K-spades, 2-hearts)  
__dealer's start hand: (8-hearts, 5-clubs, A-diamonds)  
PLAYER'S MOVE: Stand  
__player's bet: 5 chips  
__player's start hand: (7-clubs, K-spades, 2-hearts)  
__dealer's start hand: (8-hearts, 5-clubs, A-diamonds)  
DEALER'S MOVE: Hit  
__player's bet: 5 chips  
__player's start hand: (7-clubs, K-spades, 2-hearts)  
__dealer's start hand: (8-hearts, 5-clubs, A-diamonds, 6-spades)  
DEALER'S MOVE: Stand  
__player's bet: 5 chips  
__player's start hand: (7-clubs, K-spades, 2-hearts)  
__dealer's start hand: (8-hearts, 5-clubs, A-diamonds, 6-spades)  
***Game finish***  
RESULTS:  
__player: 19  
__dealer: 20  
__player lost
```

Обработка ввода

В случае некорректного ввода пользователя программа не должна аварийно прерываться (исключением является комбинация клавиш `ctrl-C`). Вместо этого должно выводиться сообщение о неверном вводе с подсказкой и предложением ввести данные заново.

Пример:

```
***Start blackjack game***
PLAYER'S BET: -45

Error. WRONG INPUT:
__player's bet must be positive integer
TRY AGAIN:

PLAYER'S BET: hello

Error. WRONG INPUT:
__player's bet must be positive integer
TRY AGAIN:
PLAYER'S BET: 5
__player's bet: 5 chips
__player's start hand: (7-clubs, K-spades)
__dealer's start hand: (8-hearts, 5-clubs)
```

И так далее...

ООП

Все сущности используемые в логике, такие как карты, колода, игрок, дилер и т.д. должны быть описаны соответствующими классами. Причем каждый класс должен быть в отдельном модуле. (Некоторые логично будет сгруппировать в пакеты - на усмотрение разработчика).

Сущности должны быть логично и ясно именованы
(карта - Card, колода - Deck и т.д.)

Логирование (опционально)

Необходимо, чтобы информация выводилась не только в терминал, но и записывалась в таком же виде в файл `blackjacklog.txt` в директории проекта.

Критерии оценивания проекта

Суммарно за весь проект можно набрать **25 баллов**

Минимальное количество баллов
необходимых для сдачи проекта **15 баллов**

Структура проекта

Структура **4 балла**
соответствие пункту “структуре” из раздела
“технические требования проекта”

Интерфейс и работоспособность

Работоспособность **5 баллов**
игра запускается и отрабатывает до
конца с верным подсчетом результатов
при корректном вводе пользователя

Соответствие описанному интерфейсу **5 баллов**
соответствие пункту “интерфейс” из
раздела “технические требования проекта”

Обработка ввода

Обработка некорректного ввода пользователя **5 баллов**

ООП

Соответствие пункту “ООП” **4 балла**
соответствие пункту “ООП” из раздела
“технические требования проекта”

Логирование (опционально)

Соответствие пункту “логирование” **2 балла**
соответствие пункту “логирование” из
раздела “технические требования проекта”