Review test Intro

So you've made it :) You should have a very solid Python knowledge base. You're almost ready to get to the project part again.

However, before we set you off to dive into it, you should really go through the review test that is about to follow. It will test your knowledge from the previous 3 lessons. Depending on your score, you should consider revision of the particular lesson.

Good luck with the test as well as the project!

Review test 13-15

1/12

show questions

What are the benefits of comprehensions?

- [ ] They are always more readable
- [ ] Faster than for loops
- [ ] They do not generate all the items of a collection at once

- [ ] One line of code

Project description

Our aim is to build a blackjack game. Here are some basic rules we will want to translate into Python code. We have also a youtube link, where the game principles are demonstrated.:

The objective of the game

... is to beat the dealer in one of the following ways:

- Get 21 points on the player's first two cards (called a "blackjack" or "natural"), without a dealer blackjack;
- Reach a final score higher than the dealer without exceeding 21; or
- Let the dealer draw additional cards until their hand exceeds 21.

Each player that beats the dealer earnes the amount of the bet.

Any player, who goes over 21, including the dealer, looses. This is called **bust**. Also if the player and the dealer did not go over 21, but the dealer reached more points, the dealer wins.

In case the player and the dealer have the same amount of points lower or equal to 21, the player can keep the bet. This is called **push**.

How the game runs?

Let's limit the number of players at one table to 6 + the dealer.

1. Game starts by dealing 1 card to each player and the dealer. Then one more card is dealt to each player excluding the dealear. Players will have 2 cards, meanwhile the dealer only one.

2. Now the dealer asks each player, whether they want to draw more cards from the deck. The player can decide to **split** or **double down** (see below for more information). Of course, if the player has already 21 after dealing the frist two cards, the dealer does not ask there. This is called **blackjack**.

3. Once all the players have completed their hands, it is the dealer’s turn. The dealer must hit until the cards total 17 or more points.

To sum up, players win by not busting and having a total higher than the dealer, or getting a blackjack without the dealer getting a blackjack. If the player and dealer have the same total, this is called a "push", and the player typically does not win or lose money on that hand. Otherwise, the dealer wins.

Card values

One deck consists of 52 or 6 x 52 cards. There are 4 card **suits**: clubs (♣), diamonds (♦), hearts (♥) and spades (♠). Each suit includes an ace, king, queen and jack and ranks two through ten.

- Cards with numbers have the value corresponding to the number depicted on it.
- King, queen, jack have value of 10.
- Aces can have value of 1 or 11. If the hand value would exceed 21 due to aces, the player can choose value 1 for them. We will automatically expect that if the value would exceed 21 and player has aces on hand, their value will be changed to 1.

Splitting

If the first two cards of a hand have the same value, the player can split them into two hands, by moving in a second bet equal to the first. The player then plays out the two separate hands in turn.

Doubling down

After the first two cards are dealt, the player is allowed to increase the initial bet by up to 100% in exchange for committing to stand after receiving exactly one more card. So the player will have 3 cards at the end.

How will we proceed?

If we haven't played blackjack before, it can seem to us that there are **quite a lot of rules** to be incorporated into the program. **It can seem even overwhelming and discouraging** to continue with the program.

The best way, how to overcome a complicated task is to **divide it into smaller doable tasks**. And here we will speak about how to do that.

So let's try to list actions or tasks, that our program will have to perform.

Program tasks

PYTHON ACADEMY / PROJECT 4: BLACK JACK [H] / PROJECT / PROGRAM TASKS

A possible list of tasks, that our program will have to perform.

1. create deck
2. shuffle deck
3. register players + setup a table
4. putting a bet
5. serve players
6. play game
7. check hand - how many points are there?
8. get the card value
9. draw from the deck
10. show the current situation at the table

Designing our program

PYTHON ACADEMY / PROJECT 4: BLACK JACK [H] / PROJECT / DESIGNING OUR PROGRAM

Besides being able to rerun the same code again and again, functions allow for **better abstraction and program structuring**.

So we could plan our program writing down the function names and inside the definition insert `pass` keyword.

Plan Draft

Use dropdown feature below if you want to see, how we wrote the code.

Click to see our plan

Creating card deck

One deck consists of 52 cards. We distinguish 4 card **suits**:

- clubs (♣),
- diamonds (♦),

- hearts (♥) and
- spades (♠)

Each suit includes an ace, king, queen and jack, each depicted with a symbol of its suit and ranks two through ten.

We first need to generate the deck of cards, in order we can play the game.

Code Solution

Use dropdown feature below if you want to see, how we wrote the code.

Click to see our solution

Shuffling the deck

Our task is to create a function that will shuffle a card deck. Also, if the deck contains 6 x 52 cards, we would like to cut it after shuffling and introduce a blank card into the place of the cut.

- Will this function need some inputs?
- Will this function return any value?

The Shuffle and Cut

The dealer thoroughly shuffles portions of the pack until all the cards have been mixed and combined. He designates one of the players to cut, and the plastic insert card is placed so that the last 60 to 75 cards or so will not be used. (Not dealing to the bottom of all the cards makes it more difficult for professional card counters to operate effectively.)

Code Solution

Use dropdown feature below if you want to see, how we wrote the code.

Click to see our solution

Registering the players

We finally want to start playing. First we need to serve the cards to the players. But

- who are the players?
- what information do we need to keep about them?
- do we need to keep order in which we will interact with the players?
- max how many players can there be at one table?
- can players play more than one game at one table?

So let's try to create a function that registers the players asking for their names. Each player should get the default amount of money equal to 50. The function should return an object containing all the registered players, that will sit behind the table.

- do we want to keep **house** among the players?
- what will be the amount of money the house will have at its disposal?

We could run the function `register_players()` as follows:

```
1 players = register_players(2)
```

Code Solution

Use dropdown feature below if you want to see, how we wrote the code.

Click to see our solution

Click to see our solution

Bet before being served

Once we have our players at the table, we can begin to deal cards. In order we can do that, we need to answer few questions first:

- where will we store the information about what cards has each player on his/her hand?
- where will we store the information about the bet for the given game?
- how much has each player bet?

Maybe we could first ask the players to put their bets. That way we will know, who plays this game and who not. So let's create the function `put_bets()` .

Function `put_bet()` aim is to collect the bets, relate them to the player betting and create place for cards, that a player will have on hand. Once this is collected, the function will return a collection of players, their hands and bets.

Few questions to answer:

- how the function will know, who sits at the table?
- what will be the amount of min. bet amount?
- will we allow to bet for people, who do not have enough money to bet?
- what will we do with the people, who do not have enough money?
- how will the bet be reflected in player's personal bank?

Code Solution

Use dropdown feature below if you want to see, how we wrote the code.

Click to see our solution

Click to see our solution

Now we can serve

Once our players have their hands open and ready to collect cards in them, we can begin to deal the cards. The rule is that 2 cards are dealt to players and one to house in two rounds. In the first round everybody including the house will receive a card (house receives as the last one). Second card is then dealt only to players.

So questions concerning the function `serve()` are:

- what will be the function inputs?
- what will be the function outputs?

Code Solution

Use dropdown feature below if you want to see, how we wrote the code.

Click to see our solution

Do we want to draw cards?

To answer this question we need to know:

- how much do I already have on my hand?
- for that we would need to know, what is the value of each card?

Dealer does not want to ask me if I want another card, if I just got a blackjack. Also it would be nice for other players, if the computer calculated the points each player has. Also, if the player wants to split or double down, this is the time to tell the dealer.

Dealer needs to ask every player the same things and that means that we will need a loop during which we will:

1. check our hand
2. decide, whether we want to split or double down or draw or stand
3. if we want another card, dealer has to deal

Let's put this loop into a function called `play()`. This function will perform all the actions to be done during one blackjack game:

1. put bets
2. serve cards
3. ask all the players, whether they want more cards
4. evaluate the game

We have already created functions that cover the first two points, now we will turn our attention to the point number three.

Also note, that we do not count with registering the players that are currently at the table. It is because players at one table can play multiple games. Therefore their registration should take place outside the game itself.

Code Solution

Use dropdown feature below if you want to see, how we wrote the code.

Click to see our solution

Checking the hand

In order a computer can calculate the amount of points on our hand, it first needs to be able to translate the cards rank into the corresponding value.

1. Our first task is to create a function `card_value()` that will translate the card rank into a value
2. This function can then be used by the function `check_hand()` that will sum all the values returned by `card_value()` .

The function `check_hand()` should also adjust the total amount for cases, when the player has one or more aces on the hand and the total sum would trespass the value of 21. In that case ace value should be adjusted to 1 instead of 11.

Code Solution

Use dropdown feature below if you want to see, how we wrote the code.

Click to see our solution

Asking the player

We know, how many points does the player have on the hand.

```
1  def play(players):
2      game = put_bets(players)
3      deck = shuffle_deck(generate_deck())
4      serve(game,deck)
5
6      for player in game:
```

```
7
8           on_hand = check_hand()
9
10          stand
11
12          split
13
14          double down
15
16          draw()
```

Now we need to ask the player few questions or check few facts before even asking:

1. whose turn is it? If it is the dealer, then just cards can be draw until 17 or more.
2. does the player have a blackjack?
3. does the player want to split?
4. or deos the player want to double down?
5. or just draw until bust or satisfied?
6. what information will we have about each player?

We will now need to set up few conditions...

Few notes

Player can split only if both cards on player's hands have the same rank. If the player decides to split, new hand and bet have to be created in the game. The newly created hand receives one of the two cards from the original hand. The bet has to be the same as the player's original bet. Do not forget to discount it from the player's bank.

If player decides to double down, the player's bet has to be doubled and only one more card can be dealt from the deck. Double down is permitted only if the player has less than 12 points on the hand.

Code Solution

Use dropdown feature below if you want to see, how we wrote the code.

Click to see our solution

Drawing a card

Dealer repeatedly asks the player, if another card should be drawn from the deck. The loop should terminate if the player can decide to stand (stop drawing) or the amount of points on hand has exceeded 21.

Each time a card is taken from the deck, the current amount of points as well as the cards on hand should be depicted to the player.

- what will be the inputs?
- will there be any outputs?

Code Solution

Use dropdown feature below if you want to see, how we wrote the code.

Click to see our solution

Game evaluation

Once all the players have taken their cards from the deck, it is time to evaluate wins and losses.

So when does the player win and how much does the player earn?

1. if got blackjack and the dealer did not get blackjack - player earns 1.5 of the original bet
2. if the dealer busted and the player's points do not exceed 21 - player earns the amoung of the original bet
3. if the player has more points than the dealer not exceeding 21 - player earns the amoung of the original bet

When does the player loose?

1. when player's cards exceeded 21
2. when the dealer has more on hand than the player and none of them exceed 21

No gain nor loss if the player and the dealer have the same amount of points not exceeding 21.

Let's create a function `evaluate_game()`, that will adjust player's banks, based on dealer's vs. their cards:

```
def play(players):
    ....
    ....

    return evaluate_game(game, players)
```

The function `evaluate_game()` should return the list of players at the table with their current balances:

```
[['Bob', 60], ['Ann', 40], ['house', 1000]]
```

Code Solution

Use dropdown feature below if you want to see, how we wrote the code.

Click to see our solution

Allowing to play multiple games

In order the program runs until no player has money to bet or until the player decides to terminate the game, we need to put the `play()` function into a loop. So games will be played until the above conditions will be met. Also, this could be a good place, where players could be registered.

Therefore we could call this function to be created `table()`. This will be the function we will call in order the whole program can be run.

Code Solution

Use dropdown feature below if you want to see, how we wrote the code.

Click to see our solution

Entire Code

So you'd like to see the whole thing, huh? Alright :) Below you can find:

1. the solution boxes for individual tasks,
2. next-to-last is the box with the entire code,
3. and last box starts the game.

Individual tasks

Game set up

Create deck

Shuffle deck

Register players

Putting a bet

Serve players

Get card value and Check hand

Draw

Show table

Play

Evaluate the game

Set up table for multiple games

Entire code & Game start

Entire code

START THE GAME

DALŠÍ LEKCE