OMIS 30 - Fall 2020
Midterm Project

Logistics:

Assigned: Tuesday, October 27, 2020
Due: Sunday, November 14, 2020 end of day

Objective:

Write a self contained program to play Black Jack.

The requirements for the blackjack program are:

- Build a deck of cards
- Use a 2 deck game
- 'Bet' before the deal
- Deal the initial hands
- Ask the user to hit/stand
- Play out the dealer hand based off the rules
- Keep track of winnings

Double, split, surrender, and insurance and splitting are not necessary.

Documentation:

Make sure to include a documentation file. Include what rules you intended to write into the program (aka, how your program "should" work). This should not be more than a couple pages long. But, do include a definition of each function (what it does) and object (deck of cards, hand of cards, etc)

Resources:

https://www.bicyclecards.com/how-to-play/blackjack/
https://wizardofodds.com/games/blackjack/basics/
The 2008 movie "21" was loosely about/inspired-by the MIT Black Jack team (though the second half of the movie is just a bad, b-rated Hollywood drama). But, if you'd like to meet a real-life person that was involved with the MIT black-jack team, visit Professor Phil Kesten in the Physics Department. He will inevitably have some great stories to share!

Collaboration:

This is a group project. Each group will turn in ONE submission. Groups should not have any external code in this project.

Submission:

- Name your final file <your_username>_project2_fall2018.py (mine would look like dvrdoljak_project2_fall2018.py), where the username belongs to the person submitting.
- Do not create this as a Jupyter Notebook! It must be a stand-alone Python script.
- Put all team members' names at the top of the code as a comment.
- Make sure it runs completely and correctly on your computer
- Submit it via Camino
- (We will run your program on our computer to test your answers)

Grading Rubric:

Section	**Grade**	**Criteria**
Deck of cards & Deal	10%	Randomness, 2 decks, order
Betting & Hit/Stand inputs	10%	User inputs, error validation
Dealer play	10%	Following dealer rules, determine winner
Tracking winnings	10%	Chip stack vs bet
Code Quality	20%	Quality of code, follows best practices
Ease of use	20%	Prompts well defined; Error handling done Visually appealing Speed
Use of comments & Readability	20%	Documentation of author & dates; Explanation of steps Use of whitespace; Use of new lines; Naming convention of variables