This next year we will be attending the 705 statistics class.  In the spirit of this new adventure, I bring you this new years gift.  Enjoy.

\*\*\* HOW TO RUN: \*\*\*

==================

- Copy the entire "Cards" folder to a directory accessible to your python installation.

- "python war\_stats.py" from within that folder.

BACKSTORY:

===========

As you may know, I studied Python on my own in Spring of 2019 prior to starting this masters program.  This happened to coincide with my Intro to Stats course in the same semester.  One of the problems I struggled with (but really loved) was the probability of drawing a certain poker hand.  I studied numerous YouTube videos to even vaguely understand how the probabilities broke down, and don't even remember it today.  However, it inspired me to use that example to practice my new Python skills.  At the time, I was trying to learn how to create objects in Python.

 I created a card object model, called cardobjects.py, and wrote some code to simulate 1000 hands and evaluate how many types of poker hands I had received.  If you wish to run those, you simply need to run one of the four programs:

singlehandstats.py

twohandstats.py

fourhandstats.py

fourhandstats\_ser.py  - This illustrates if how you deal the cards affects the results (without getting yourself shot).

(btw, this early card code is kind of rough)

LATEST UPDATE:

===============

Over winter break, I decided to expand this to include the card game "War".  This is specifically sent out to new, or soon to be new parents, who's children may one day discover this game and beg you as parents to play.  And play.  And play again.  As an adult, the novelty starts  to wear out after playing 2 or 3 times.  At this point, you may find yourself wondering, "How long does this game typically go?".

THIS PROGRAM ANSWERS THAT QUESTION!

There are two programs you  can run

war\_game\_print.py  - This is the code with printing that only runs a single game.  This allows you to examine the dynamics of how cards are traded.

war\_stats.py - This is the same code without printing that runs a specified number of games.  The code then calculates the mean, median, min, and max number of rounds of all  games.  The median and mean are not exact, but you cannot play half a round, so the round value is truncated as "close enough".

Variables for these two games include:

Number of players - 2-10.

Number of decks - 2-5 (card count must be evenly divisible with players)

Draws for war - default = 3 cards

These variables are set within the .py files and should allow you to explore different variations of play.

Rules reference: <https://www.pagat.com/war/war.html>

=====================================================================

LESSONS LEARNED - FURTHER EDUCATION

I was driven to complete this so did not go back and polish it.  You will find room for improvement within.  Here are some lessons learned in the process:

Functions vs Objects - I started by trying to use functions to play directly with cardobjects.py, but soon found that tracking game state was difficult to manage.  I ended up creating objects (war\_objects.py) to manage individual player state, make managing game play easier, and automatically evaluate battles and wars.  Some lessons learned with objects:

Naming objects, their properties, and methods is quite an art form that requires a lot of planning.  The war\_objects were my attempt to obfuscate the cardobjects from the main code.  However, I often grew confused with the data structures and had a lot of errors trying to iterate objects that were not iterable.  If I do this again, I will be keeping a notebook of all my objecs with their methods and properties.

The model frame of reference is also tricky to keep straight.  In my cardobject model, the methods "add" and "take" are from the frame of reference of the deck.  However, the player object methods "take" and "give" are from the opposite frame of reference.  So you end up with player.take(deck.take()) which is confusing.  The verbs "take", "give", "add", "remove", etc all depend on a frame of reference, so choosing an ideal method verb that is universally clear is difficult.   I will likely go back and set those methods in the player objects to match those of the card objects.

Finally, the corresponding .docx is also a "plain writing" exercise from my 735 class.