**Todo: yes, we are going to tryhard sweat this python stuff**

**\*Feel free to work on your own empty python file for functions, we can put it into the main file later**

Difficulty from LOW-HIGH, starting from TODO’s top to bottom

* ~~Touch up decorative parts of text printing(Allen)~~
* ~~Add additional UI info to make it easier for the player to read and play(Allen)~~
* ~~Print user hand in ordered suit~~s

~~- Make it so that the bots can randomly pick a name from a name bank~~

~~- Give the CPUs the ability to taunt/ say mean things(Allen)~~

* Add error handling for:
* ~~User input when playing a card~~
* ~~Choosing hand at beginning~~
* ~~Number of players at beginning~~
* Add ‘gamestatus.txt’ logging
* ~~Review aF tweaking rules (i.e. decrease safe plays if had any card that could win, not just trump card)~~

* ~~Review CPU discard logic, discard from lowest rank of all suits, not from lowest of most abundant suit.~~
* ~~Allow CPU to count cards that are in play/have been played:~~
* ~~Create a list of DEFINITELY safe cards to play given suit matches leading or “null~~
* ~~Given a surplus of winning cards, burn some to maintain win streak~~
* ~~“Overtaking” Let CPU play card w/ intention of winning round without playing trump card. Based on how close it is to the end (i.e. 1 player has to play after CPU), CPU will play a card less valuable than if CPU were further from the end (i.e. 3 players will play after CPU) since there is more risk from those 3 players vs 1 player.~~
* ~~“Baiting” Let CPU play a card that barely wins (using the ‘mWC’ func) not to win a round, but to bait other players into playing a higher card (basically, raising the stakes if the other players want to win the round).~~ *~~I.e. If the current highestCard played is 5, CPU2 has cards: 7-9-10, and Human has cards: 6-K. By playing 7 as CPU, Human is forced to play off K to win, allowing CPU2 to win future rounds regarding that suit with cards 9-10 (since they are all higher than Human’s card 6)~~*
* Make a method that examines the current cards in play/cards played/value-vs-risk to add a temporary boost or decrease in aF

**FUNCTIONS TO MAKE**

CPU Decision Making *(you wish it were ai)*

1. def adjustAF(self, cardsInPlay):

*Create a function that can observe what cards are currently in play, using probability to either add or subtract from aF.*

*Examples: If the number of highValue cards played < total cards played, it should likely concede the round to winningPlayer instead of compete by wasting a better card*

1. ~~def burnSurplus(self, activeCards):~~

*~~Make function that lets CPU realize it has more than one always winnable card of a certain suit (i.e. CPU has all three Ace, King, and Queen of some suit) and can burn off either card to win some rounds~~*

* *~~The ‘activeCards’ parameter will be supplied by the ‘getActiveCards’ function~~*

**CPU Utilities**

1. ~~def getActiveCards(self, cardsPlayed):~~

*~~Create function that monitors cards that have yet not been played, excluding those in CPU’s hand.~~*

* *~~The ‘cardsPlayed’ parameter can be a constantly running list of cards players have discarded throughout rounds, will be in main()~~*

**Game Appearance**

1. ~~def taunt(self, tauntBank):~~

*~~Make a function that lets CPU say some mean things whenever it plays a card to WIN~~*

*~~- ”You all came out the wrong way”~~*

*~~-”You smoll brains”~~*

*~~-”Bruh”~~*

*~~-”Why you do y’all even try?”~~*

1. def lament(self, lamentBank):

*Make a function that lets CPU say defeated/acknowledging phrases when it SAVES*

* *“Time to off myself”*
* *“Why has RNG failed me…”*
* *“It’s like an invisible hand has turned the difficulty down for <winningPlayer>”*
* *“Don’t worry, this is all part of the plan”*

1. ~~def randomName(self, nameBank):~~

*~~Make a function and custom nameBank that randomly chooses a name and returns it~~*

* *~~“Joe”~~*
* *~~“<Insert Crude Humor>”~~*
* *~~“Big Chungus”~~*
* *~~“Ben Dover”~~*
* *~~“<Insert a creative name here>”~~*

1. ~~def displayPlayerHand(self, hand):~~

*~~Create a function that displays a players hand, organizing cards into the four respective suits.~~*

* *~~‘hand’ is a Deck class object:~~*
* *~~Using ‘self.hand.getDeck()’ will return the player’s hand as a list. From there you can use default python functions to modify the list (splice, append, remove, etc.)~~*
* *~~Using ‘card.getSuit()’ returns suit of card~~*
* *~~Using ‘card.getRank()’ returns rank of card~~*

*~~E~~*~~x: ♡ 1, 2~~

~~♤~~

~~♧ 9~~

~~♢ king, Ace~~

**\*Bugs**

\* put “in progress” or “to be done” or “done” or “need help” depending on the bug’s status