Dong Hyun Kim
andydhkim97@gmail.com

Design Document: Blackjack

How to run code:

- On Ubuntu Linux, simply type the following into a terminal once in the correct directory:
  - python3 blackjack.py

Design choices:

- I focused on making the code as modular as possible, creating several different classes and utilizing object-oriented programming principles. Each class focuses on a distinct part of the game; for instance, the Card class is solely responsible for storing the appropriate suit, rank, and values. This independence will hopefully let programmers use the classes in blackjack.py for other card games that require a standard 52 card deck. Furthermore, due to the modularity, the main loop itself is relatively simple and debugging becomes less of a burden.
- I decided to utilize a list to hold my Card objects, since it is easer to pop from a deck and append to a hand. The time complexity for each action is also O(1), which brings an advantage in performance. The shuffle function that existed for a list also made me consider it more.
- I also decided to use a dictionary (hash table) for my values, since I wanted each card to be assigned a value quickly as they were added to the deck. Reading a value from a key in a dictionary is O(1), which also brings an advantage in performance. Furthermore, if the point value of a card needs to change, this dictionary can easily be altered.
- I declared global variables before my classes in order to get rid of any magic numbers. Some of the global variables were used to control the flow of the game within the main loop.

Tooling:

- Python was used due to its simple nature and prevalence of libraries. It is also the language that I am most comfortable with.
- The random library was utilized to shuffle the deck with ease.
- The time library was utilized to have a delay as the dealer draws cards from the deck. This allows the player to see each draw one by one, much like a real game.