Card Game Documentation

1. Author: Brian(Kangjian) Ma (kangjianma@gmail.com)
2. Date: September 8, 2020
3. Programming Language: Python 3
4. Project Interpreter: Python 3.6
5. Card Game Introduction:
   1) A Card Game is designed with 3 operations including randomly shuffle, deal the top card, and sort in assigned order, which supports two players to determine a winner.
   2) Game Play Rule:
      2 players play the game. They will draw 3 cards by taking turns.
      Whoever has the high score wins the game. (color point calculation, e.g. red = 3, yellow =2, green = 1) the point is calculated by color point * number in the card.

6. Design Assumptions:
   1) User can assign the number of decks to play, which should be at least 1.
   2) User can assign suit types in a deck while the assigned suit type should be unique, e.g. suits = ['red', 'yellow', 'green'], suits = [‘spade’, ‘club’, ‘heart’, ‘diamond’] are valid, while suits = ['red', red, 'green'] will induce an Error.
   3) When initiated, each suit has user-assigned number of cards, which can be same or different to other suits. The range of the number on a card of one suit is [0, total number of cards of the suit - 1]
   4) User can assign suit-point matching pattern to calculate points in the play, e.g. points = {'red': 3, 'yellow': 2, 'green': 1}
   5) There is one type of deck of cards in each play, and different decks of cards cannot mix in a play.

7. Classes Design:
   There are two classes designed.
   1) Class Card: A class to represent a deck of cards with assigned suits and number of cards for each suit.
   2) Class CardGame: A class to represent a card game with randomly shuffle, deal a card from top, sort in assigned order, and play methods, which supporting 2 players to determine a winner.

   The relationship between class CardGame and class Card is composition. A CardGame object has at least one Card object to play.

8. Methods Design:
   The designed Card Game supports 3 operations and two players can play the game.
   1) Shuffle cards in the deck: randomly mix the cards in the card deck, and return a whole deck of cards with a mixed order.
   2) Get a card from the top of the deck: get one card from top of the card deck, return a card, and if there is no card left in the deck return error or exception.

3) Sort cards: take a list of color as parameter and sort the card in that color order. Numbers should be in ascending order.
   i.e. If the deck has a card contains with following order
   (red, 1), (green, 5), (red, 0), (yellow, 3), (green, 2)
   Sort cards([yellow, green, red]) will return the cards with following order
   (yellow, 3), (green, 0), (green, 5), (red, 0), (red, 1)