**Poker python code**

besHand(hands) function will find the winner from the input hands.

It will create Poker object, then call poker.scoreHand() function

to calculate the score, and compare ranks if score is same.

**def bestHand(hands):**

pk\_1 = Poker(hands[0])

bestScore = pk\_1.scoreHand()

bestH = hands[0]

for i in range(1,len(hands)):

pk = Poker(hands[i])

currentScore = pk.scoreHand()

# python can use compare **operator (<,>,==) to compare two Lists**

# (with Dictionary elements), that simplifies the code

if (bestScore < currentScore):

bestScore = currentScore

bestH = hands[i]

elif (bestScore == currentScore):

#k need to change bestH from string to list, so I can append tie hand

bestH = [bestH]

bestH.append(hands[i])

return bestH

return [bestH]

Poker class will parse each hand to create data structure:

- String hand,

- suits array of chars,

- ranksDict Dictionary with rank key, and rank count value pair.

Then scoreHand() will calculate score and create valuesDict Dictionary

valuesDict is reversed sorted keys, then reversed sorted values,

return [ score, valuesDict]

**class Poker:**

def **\_\_init\_\_(**self, hand):

self.hand = hand # string

self.suits = self.parseHand(True)

# Dictionary: key=rank, value=rank count, with ranks are reversed sorted

self.ranksDict = self.createRanksDict()

#self.scoreAry = self.scoreHand() # for local testing

pass

def **parseHand**(self, flag): # works

hand\_1 = self.hand.replace('10', 'T') # convert '10' to 'T'

hand\_2 = hand\_1.replace(' ', '') # take out all space

handAry = list(hand\_2)

# get suits and ranks array

suits = []

ranks = []

for i in range(len(handAry)):

if (i%2 != 0):

suits.append(handAry[i])

else:

ranks.append(handAry[i])

if flag:

return suits

else:

return ranks

# convert ranks to ranks values, then convert string to int, finally sort it ASC.

def **sortRanks(self):**

ranks = []

ranks = self.parseHand(False) # get ranks

ranks\_v = []

findex = **lambda c: '--23456789TJQKA'.index(c)** # convert ranks to values

for i in ranks:

ranks\_v.append(int(findex(i))) # convert string to int

#sorted(ranks\_v) # built-in sorted()

ranks\_v.sort() # this work

return ranks\_v

# python dictionary is unordered key,value structure, so even thought I try

# to order keys in DESC, but when ranksDict is created in keys in ASC

# create ranksDict : key=rank, value=count; work

def **createRanksDict(self):**

ranksSorted = []

ranksSorted = self.sortRanks()

# ranksSorted.reverse() => DESC ranskSorted array;

# howver it is still give me ranksDict with ASC ranks keys

# howver later when I create dictionary hand1 = reverse values sorted ranksDict

# this function will also reverse sorted ranks keys

# Dictionary valuesDict = sorted(self.ranksDict.items(), key=lambda r: (-r[1], -r[0]))

# create ranksDict : key=rank, value=count; work

ranksDict = {}

# has\_key() is removed in python3, now use if i in ranksDict

**for i in ranksSorted**:

if i in ranksDict:

ranksDict[i] += 1

else:

ranksDict[i] = 1

return ranksDict

**def scoreHand(self):**

# flush; work

**flush = all( self.suits[0] == s for s in self.suits )**

# ranksSorted = ranks array sorted ASC, need it for straight check

ranksSorted = self.sortRanks()

# special straight[2,3,4,5,14], change ranksDict= { 1:1, 2:2, 3:3, 4:4, 5:5}; work

if ( ranksSorted == [ 2,3,4,5,14]):

# also need to change ranskSorted to [1,2,3,4,5], so straight = True

ranksSorted = [1,2,3,4,5]

# change ranksDict

self.ranksDict = { 1:1, 2:2, 3:3, 4:4, 5:5}

# check for straight

len1 = len(ranksSorted)

straight = **all( ranksSorted[0]+i == ranksSorted[i] for i in range(len1) )**

# get kinds array = ranksDict valuses ([4,1])

kinds = list(self.ranksDict.values())

# use **list to convert ranksDict.values() to array**

kinds.sort(reverse=True)

# flush & straight => score = 8

if flush and straight:

score = 8

elif kinds == [4,1]:

score = 7

elif kinds == [3,2]:

score = 6

elif flush:

score = 5

elif straight:

score = 4

elif kinds == [3,1,1]:

score = 3

elif kinds == [2,2,1]:

score = 2

elif kinds == [2,1,1,1]:

score = 1

else:

score = 0

# create hand1 dictory with reverse sorted rank key,

# then reverse sorted rank counts value

valuesDict = []

# current ranksDict = ASC sorted ranks keys

# this function will create

# Dictionary valuesDict = DES sorted ranks keys, DES sorted rank count values

valuesDict = sorted(self.ranksDict.items(), **key=lambda r: (-r[1], -r[0]))**

# return array with [ score, valuesDict dictionary]

return [score] + valuesDict

# test for tie

#hands\_8 = ["9S 8S 7S 6S 5S", "9D 8D 7D 6D 5D", "4D 4S 4H QS KS"]

#bestHand(hands)