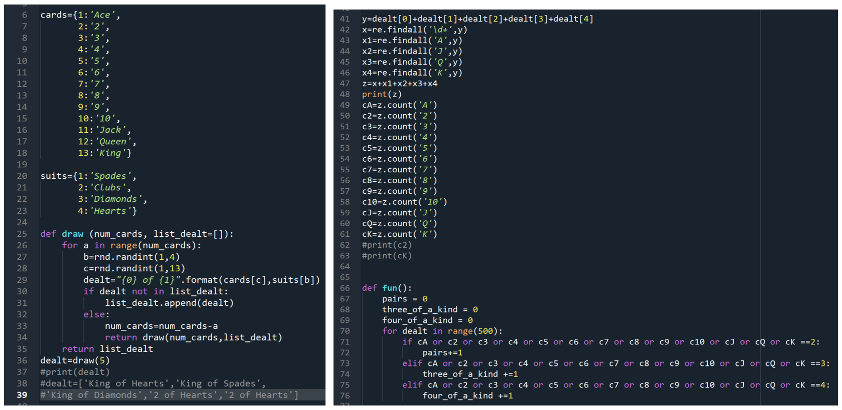
Final Project

Ken Petro

GitHub: kenlpetro/Math300_Petro/Python/Final_Project

Initial Approach

I went through many iterations before I arrived at my final code. My initial idea was to use a drawing function to give a hand of five cards using a random integer. It would then return the integers in the format "Two of Hearts", "Queen of Spades" etc. It ended up being useless because I then had to use "re.findall" to pick out just the rank, since the suits are unnecessary. I then used the count function to get the rank of each card in the hand. Next I tried to make a function to deal 500 hands, and use "if" statements to count the number of pairs. This approach ended up using many lines unnecessarily, and in the end I couldn't get it to work.



```python
cards={1:'Ace',
       2:'2',
       3:'3',
       4:'4',
       5:'5',
       6:'6',
       7:'7',
       8:'8',
       9:'9',
       10:'10',
       11:'Jack',
       12:'Queen',
       13:'King'}

suits={1:'Spades',
       2:'Clubs',
       3:'Diamonds',
       4:'Hearts'}

def draw (num_cards, list_dealt=[]):
    for a in range(num_cards):
        b=rnd.randint(1,4)
        c=rnd.randint(1,13)
        dealt="{0} of {1}".format(cards[c],suits[b])
        if dealt not in list_dealt:
            list_dealt.append(dealt)
        else:
            num_cards=num_cards-a
            return draw(num_cards,list_dealt)
    return list_dealt
dealt=draw(5)
#print(dealt)
#dealt=['King of Hearts','King of Spades',
#'King of Diamonds','2 of Hearts','2 of Hearts']
```

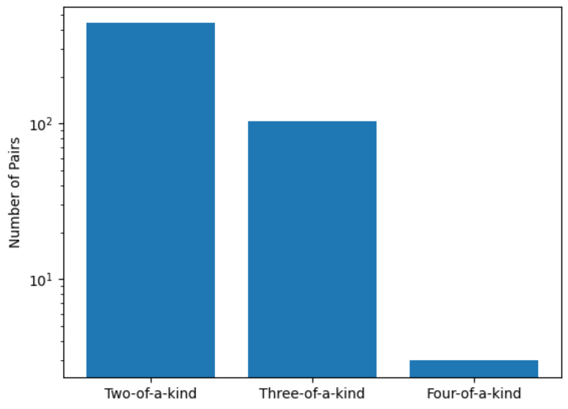
```python
y=dealt[0]+dealt[1]+dealt[2]+dealt[3]+dealt[4]
x=re.findall('\d+',y)
x1=re.findall('A',y)
x2=re.findall('J',y)
x3=re.findall('Q',y)
x4=re.findall('K',y)
z=x+x1+x2+x3+x4
print(z)
cA=z.count('A')
c2=z.count('2')
c3=z.count('3')
c4=z.count('4')
c5=z.count('5')
c6=z.count('6')
c7=z.count('7')
c8=z.count('8')
c9=z.count('9')
c10=z.count('10')
cJ=z.count('J')
cQ=z.count('Q')
cK=z.count('K')
#print(c2)
#print(cK)


def fun():
    pairs = 0
    three_of_a_kind = 0
    four_of_a_kind = 0
    for dealt in range(500):
        if cA or c2 or c3 or c4 or c5 or c6 or c7 or c8 or c9 or c10 or cJ or cQ or cK ==2:
            pairs+=1
        elif cA or c2 or c3 or c4 or c5 or c6 or c7 or c8 or c9 or c10 or cJ or cQ or cK ==3:
            three_of_a_kind +=1
        elif cA or c2 or c3 or c4 or c5 or c6 or c7 or c8 or c9 or c10 or cJ or cQ or cK ==4:
            four_of_a_kind +=1
```

Initial Approach

Result

After doing lots of research on different options and cutting/adding parts to the code, I came across a website that had a very simple solution that only used five lines. They just used the built-in shuffle function of Python to give random hands from a list of suits and ranks. I was then able to add in a for-loop at the start to reiterate each hand, as well as ”re.findall” from my previous try to get the ranks of each hand.

My final problem was to count the number of pairs, and I ended up with the idea to count the length of each hand to find the amount of unique characters. This would always give me the correct amount of two-of-a-kinds, since there would always be four unique and one duplicate character. I realize that this method will not be totally accurate with three and four-of-a-kinds, since there can be cases with combinations of pairs. For example: a hand could have two pairs, which would count as a three-of-a-kind, or a three-of-a-kind and a pair would count as a four-of-a-kind. I was unable to find a solution to this issue. Finally, I used ”matplotlib” to create a bar plot of the three kinds of pairs. Since the probability of getting four-of-a-kinds is so low, I added ”plt.yscale('log')” to make the results visible.



Results