Extra Credit Card Simulation

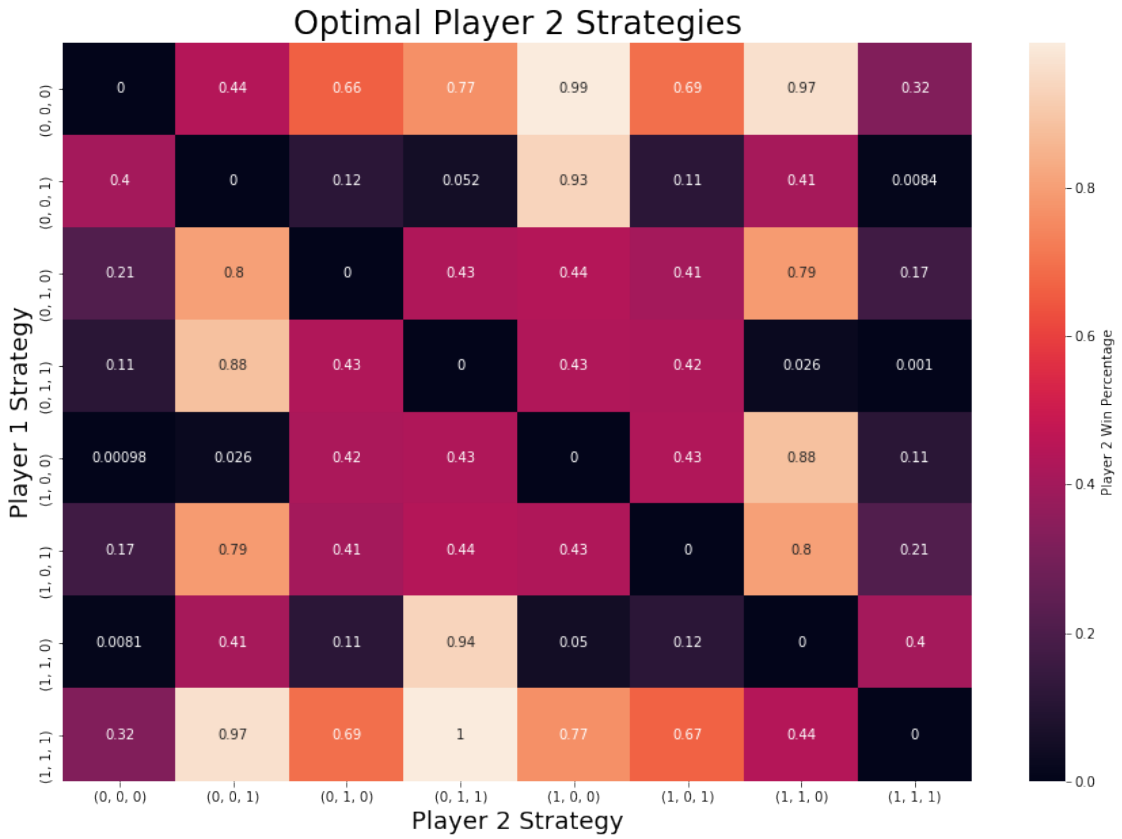
January 26, 2019

In [18]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from itertools import product
%matplotlib inline

pTwoStrats = list(product([0,1],repeat=3))
pOneStrats = list(product([0,1],repeat=3))

responseTable = pd.DataFrame(columns=pTwoStrats)

samples = 100000

def generateDeck():

    deck = np.random.permutation(52) % 2

    return(deck)

sampleDecks = []
for s in range(samples):
    sampleDecks.append(generateDeck())

for pOneSet in pOneStrats:
    stratWinDict = dict()
    for pTwoSet in pTwoStrats:
        if pTwoSet == pOneSet:
            stratWinDict[pTwoSet] = 0
        else:
            wins = []
            for deck in sampleDecks:
                pOneTricks = 0
                pTwoTricks = 0
                countSinceTrick = 2
                for i in range(2, len(deck)):
                    if countSinceTrick >= 2:
                        lastThree = (deck[i-2], deck[i-1], deck[i])
```

```
                                if lastThree == pOneSet:
                                    pOneTricks += 1
                                    countSinceTrick = 0
                                elif lastThree == pTwoSet:
                                    pTwoTricks += 1
                                    countSinceTrick = 0

                            else:
                                countSinceTrick += 1

                        wins.append(0 if pOneTricks >= pTwoTricks else 1)


                    stratWinDict[pTwoSet] = sum(wins)/samples

            maximum = max(stratWinDict, key=stratWinDict.get)
            responseTable = responseTable.append(stratWinDict, ignore_index=True)
            print("Best Pairing - P1 Strat:" + str(pOneSet) +
                  " P2 Strat:" + str(maximum) + " Win Pct: " +str(stratWinDict[maximum]))
        responseTable.index = pOneStrats

        fig, ax = plt.subplots(figsize=(15,10))

        sns.heatmap(responseTable, annot=True, cbar_kws={'label': 'Player 2 Win Percentage'})
        plt.xlabel('Player 2 Strategy', fontsize=18)
        plt.ylabel('Player 1 Strategy', fontsize=18)
        plt.title('Optimal Player 2 Strategies', fontsize=24)
```

```
Best Pairing - P1 Strat:(0, 0, 0) P2 Strat:(1, 0, 0) Win Pct: 0.99492
Best Pairing - P1 Strat:(0, 0, 1) P2 Strat:(1, 0, 0) Win Pct: 0.93496
Best Pairing - P1 Strat:(0, 1, 0) P2 Strat:(0, 0, 1) Win Pct: 0.80179
Best Pairing - P1 Strat:(0, 1, 1) P2 Strat:(0, 0, 1) Win Pct: 0.8837
Best Pairing - P1 Strat:(1, 0, 0) P2 Strat:(1, 1, 0) Win Pct: 0.88359
Best Pairing - P1 Strat:(1, 0, 1) P2 Strat:(1, 1, 0) Win Pct: 0.80193
Best Pairing - P1 Strat:(1, 1, 0) P2 Strat:(0, 1, 1) Win Pct: 0.93565
Best Pairing - P1 Strat:(1, 1, 1) P2 Strat:(0, 1, 1) Win Pct: 0.99506
```

```
Out[18]: Text(0.5, 1.0, 'Optimal Player 2 Strategies')
```


Optimal Player 2 Strategies
Player 1 Strategy
(0, 0, 0)
(0, 0, 1)
(0, 1, 0)
(0, 1, 1)
(1, 0, 0)
(1, 0, 1)
(1, 1, 0)
(1, 1, 1)
0 0.44 0.66 0.77 0.99 0.69 0.97 0.32
0.4 0 0.12 0.052 0.93 0.11 0.41 0.0084
0.21 0.8 0 0.43 0.44 0.41 0.79 0.17
0.11 0.88 0.43 0 0.43 0.42 0.026 0.001
0.00098 0.026 0.42 0.43 0 0.43 0.88 0.11
0.17 0.79 0.41 0.44 0.43 0 0.8 0.21
0.0081 0.41 0.11 0.94 0.05 0.12 0 0.4
0.32 0.97 0.69 1 0.77 0.67 0.44 0
(0, 0, 0) (0, 0, 1) (0, 1, 0) (0, 1, 1) (1, 0, 0) (1, 0, 1) (1, 1, 0) (1, 1, 1)
Player 2 Strategy
Player 2 Win Percentage
0.0 0.2 0.4 0.6 0.8