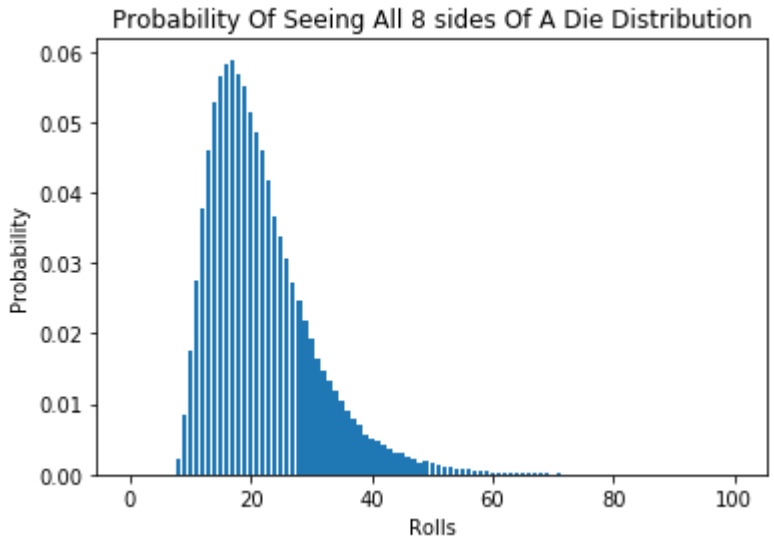
In [6]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from random import uniform
import random
```

1.1 Coupon Collector's Problem

In [44]:
```
# Part A
def FairCoinToss():
    if uniform(0,1)<0.5:
        return "H"
    else:
        return "T"

def EightSidedDieRolling():
    probability = uniform(0,1)
    if probability <= 0.125:
        return "1"
    elif probability <= 0.25:
        return "2"
    elif probability <= 0.375:
        return "3"
    elif probability <= 0.5:
        return "4"
    elif probability <= 0.625:
        return "5"
    elif probability <= 0.75:
        return "6"
    elif probability <= 0.875:
        return "7"
    else:
        return "8"

def BiasedCoinToss(alpha):
    if uniform(0,1) < alpha:
        return "H"
    else:
        return "T"
equalCounter = 0

def seen():
    valuesleft = ["1", "2", "3", "4", "5", "6", "7", "8"]
    count = 0
    while len(valuesleft) > 0:
        a = EightSidedDieRolling()
        if (a in valuesleft):
            valuesleft.remove(a)
        count = count + 1
    return count

attempts = []
for x in range(100000):
    attempts.append(seen())

#Part B
X = []
maximum = np.amax(attempts)

for x in range(maximum + 1):
    counter = 0
    for i in range(len(attempts)):
        if x == attempts[i]:
            counter = counter + 1
```

```
    X.append(counter / 100000)

#Part C
index = np.arange(len(X))
plt.bar(index, X)
plt.xlabel("Rolls")
plt.ylabel("Probability")
plt.title("Probability Of Seeing All 8 sides Of A Die Distribution")
mostfrequent = X.index(np.amax(X))

print("the most frequent number of attempts that occurred is : ",mostfrequent)


#Part D
total = sum(attempts)
average = total / 100000
print(average)

sumtotal = 0
for x in range(1,9):
    sumtotal = sumtotal + (1/x)
summationanswer = 8 * sumtotal
print(summationanswer)

# when you plug in ln(8) + 0.5772156449 into a calculator, you get 2.65
# multiple that by 8, you get 21.253
# add 0.5 = 21.75325
#they are all extremely close to each other which is expected
```

```
the most frequent number of attempts that occurred is :  17
21.74695
21.74285714285714
```



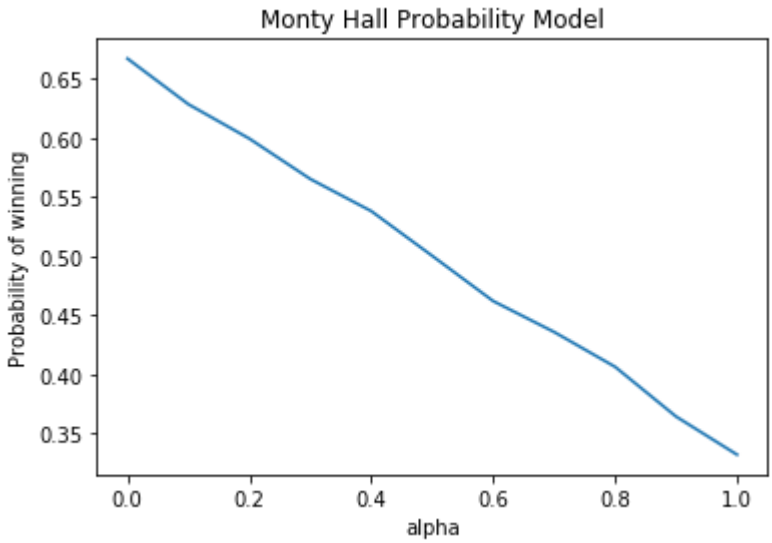
1.2 Monty Hall Problem

Part A: It is in your advantage to switch your choice. When you switch doors, you have a 67% change of winning, but if you stick with your first choice, then you will always have a 33% chance of winning. I think this happens because there are more situations in the sample space that when you switch, you will get the car, and there are less situations in the sample space where if you don't switch, you win.

In [76]:
```
#Part B
def MontyHallProblem(a):
    doors = [1, 2, 3]
    car = np.random.randint(1,4)
    contestantguess = np.random.randint(1,4)
    removed = 100
    if contestantguess == car:
        doors.remove(car)
        removed = doors[np.random.randint(0,2)]
    else:
        doors.remove(car)
        doors.remove(contestantguess)
        removed = doors[0]
    doors = [1, 2, 3]

    random = uniform(0,1)
    if random >= a:
        doors.remove(removed)
        doors.remove(contestantguess)
        contestantguess = doors[0]

    if contestantguess == car:
        return "WIN"
    else:
        return "LOSE"
```

In [107]:
```
results = []
for x in range(0, 11, 1):
    successes = 0
    for y in range(10000):
        answer = MontyHallProblem(x/10)
        if answer == "WIN":
            successes = successes + 1
    results.append(successes/10000)
n = np.arange(0, 1.1, 0.1)
plt.plot(n, results)
plt.title("Monty Hall Probability Model")
plt.xlabel("alpha")
plt.ylabel("Probability of winning")
```

Out[107]: Text(0, 0.5, 'Probability of winning')



The probabilty if you never switch (when alpha is 1) is 0.33. The probability if you always switch (when alpha is 0) is 0.67. This is in concordance with my initial guess in part A.

1.3 Coupon Collector Problem

Part A If n = 1, then P(X = "1) = 0. If only one person is in a group and you need two people to have the same birthdays, it is impossible. This person needs to get a friend. Part B If n = 366, then P(X = "1") = 1. Since there are only 365 unique days in a year, if there are 366 people in a group, at least one person has to share a birthday because 366 > 365.

In [52]:
```
#Part C
def BirthdayParadox(n):
    birthdays = set()
    for x in range(n):
        birthday = (np.random.randint(1,366))
        if birthday not in birthdays:
            birthdays.add(birthday)
        else:
            return 1
    return 0
```

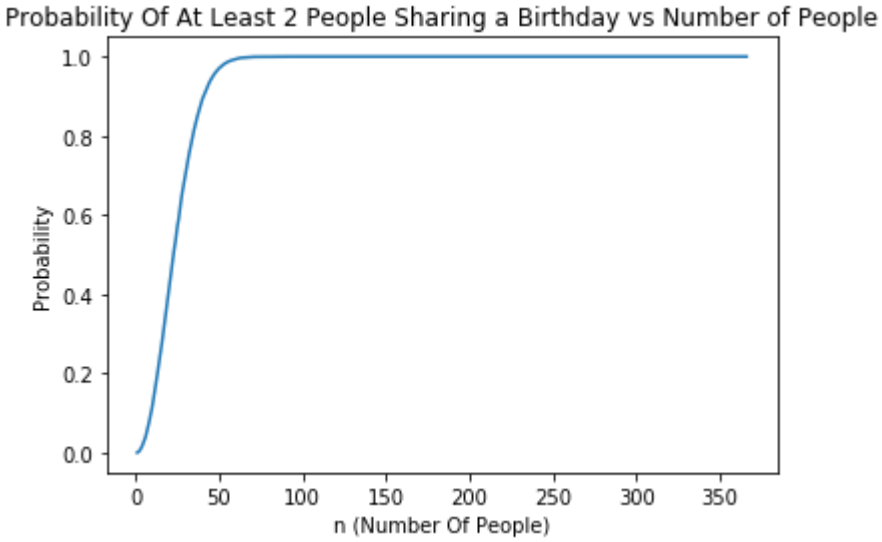
In [54]:
```
#Part D
answer = []
for x in range(1, 367):
    sum = 0
    for y in range(0, 100000):
        result = BirthdayParadox(x)
        sum = sum + result
    probability = sum / 100000
    answer.append(probability)
```

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
```

58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114

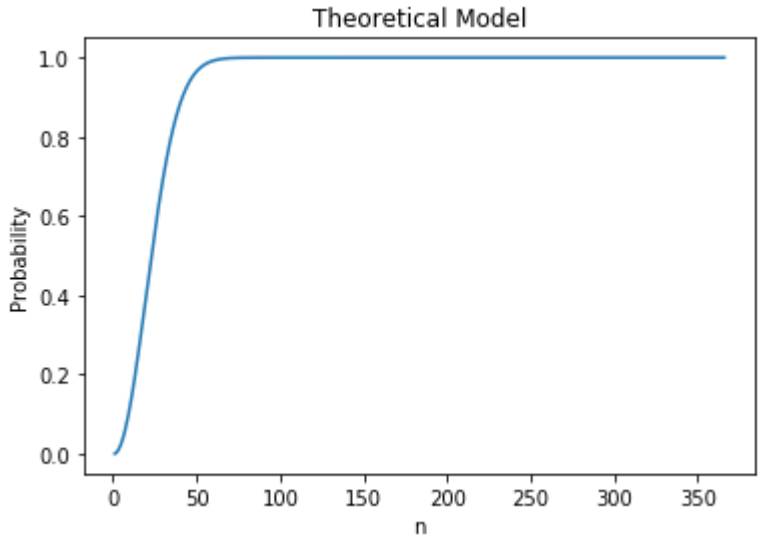
In [60]:
```
#Part E
n = np.arange(1,367)
plt.plot(n,answer)
plt.xlabel("n (Number Of People)")
plt.ylabel("Probability")
plt.title("Probability Of At Least 2 People Sharing a Birthday vs Number of People")
```

Out[60]:
```
Text(0.5, 1.0, 'Probability Of At Least 2 People Sharing a Birthday vs Number of People')
```



In [66]:
```
n = np.arange(1,367,1)
yvalues = []
for x in n:
    yvalues.append(1 - np.math.exp(-1*(x*(x-1)/730)))
plt.plot(n, yvalues)
plt.title("Theoretical Model")
plt.xlabel("n")
plt.ylabel("Probability")
```

Out[66]: Text(0, 0.5, 'Probability')



In [ ]:
```
The curves look identical. They both reach a probability of 1.0 at around 60-7
0.
```

2.1 Coin Toss - Die Rolling

In [95]:
```
#Part A
def CoinTossDieRolling(alpha):
    answer = []
    x = BiasedCoinToss(alpha)
    answer.append(x)
    if x == 'T':
        answer.append(random.randint(1,8))
    else:
        answer.append(random.randint(1,4))
    return tuple(answer)
```

In [98]:
```
#Part B
results = []
for x in range(10000):
    results.append(CoinTossDieRolling(0.5))
```

In [99]:
```
#Part C & D
countHeads = 0
countTails = 0
countHeadsOnes = 0
countHeadsFives = 0
countTailsOnes = 0
countTailsFives = 0
for x in range(len(results)):
    temp = results[x]
    if temp[0] == 'H':
        countHeads = countHeads + 1
        if temp[1] == 1:
            countHeadsOnes = countHeadsOnes + 1
        if temp[1] == 5:
            countHeadsFives = countHeadsFives + 1
    elif temp[0] == 'T':
        countTails = countTails + 1
        if temp[1] == 1:
            countTailsOnes = countTailsOnes + 1
        if temp[1] == 5:
            countTailsFives = countTailsFives + 1
print("P('1' | 'H')= " ,countHeadsOnes/countHeads)
#Part D
print("P('5' | 'H')= " ,countHeadsFives/countHeads)
print("P('1' | 'T')= " ,countTailsOnes/countTails)
print("P('1' | 'H')= " ,countTailsFives/countTails)
```

```
P('1' | 'H')=  0.24440411373260737
P('5' | 'H')=  0.0
P('1' | 'T')=  0.12279309660781591
P('1' | 'H')=  0.1267605633802817
```

The theoretical value for P("1" | "H") is 1/4 which makes sense because when you roll a 4 sided die, the probability of getting a 1 is 1/4.

In [101]:
```
#Part E
headFiveCount = 0
tailFiveCount = 0
fiveCounter = 0
oneCounter = 0
headOneCount = 0
tailOneCount = 0
for x in range(len(results)):
    temp = results[x]
    if temp[1] == 5:
        fiveCounter = fiveCounter + 1
        if temp[0] == 'H':
            headFiveCount = headFiveCount + 1
        elif temp[0] == 'T':
            tailFiveCount = tailFiveCount + 1
    elif temp[1] == 1:
        oneCounter = oneCounter + 1
        if temp[0] == 'H':
            headOneCount = headOneCount + 1
        elif temp[0] == 'T':
            tailOneCount = tailOneCount + 1
print("P('H' | '5')= ", headFiveCount/fiveCounter)
print("P('T' | '5')= ", tailFiveCount/fiveCounter)
print("P('T' | '1')= ", tailOneCount/oneCounter)
print("P('H' | '1')= ", headOneCount/oneCounter)
```

```
P('H' | '5')=  0.0
P('T' | '5')=  1.0
P('T' | '1')=  0.3380666302566903
P('H' | '1')=  0.6619333697433096
```

In [103]:
```
#Part F
results = []
for x in range(10000):
    results.append(CoinTossDieRolling(0.25))
countHeads = 0
countTails = 0
countHeadsOnes = 0
countHeadsFives = 0
countTailsOnes = 0
countTailsFives = 0
for x in range(len(results)):
    temp = results[x]
    if temp[0] == 'H':
        countHeads = countHeads + 1
        if temp[1] == 1:
            countHeadsOnes = countHeadsOnes + 1
        if temp[1] == 5:
            countHeadsFives = countHeadsFives + 1
    elif temp[0] == 'T':
        countTails = countTails + 1
        if temp[1] == 1:
            countTailsOnes = countTailsOnes + 1
        if temp[1] == 5:
            countTailsFives = countTailsFives + 1
print("results for alpha = 0.25")
print("P('1' | 'H')= ", countHeadsOnes/countHeads)
print("P('5' | 'H')= ", countHeadsFives/countHeads)
print("P('1' | 'T')= ", countTailsOnes/countTails)
print("P('5' | 'T')= ", countTailsFives/countTails)
print()
headFiveCount = 0
tailFiveCount = 0
fiveCounter = 0
oneCounter = 0
headOneCount = 0
tailOneCount = 0
for x in range(len(results)):
    temp = results[x]
    if temp[1] == 5:
        fiveCounter = fiveCounter + 1
        if temp[0] == 'H':
            headFiveCount = headFiveCount + 1
        elif temp[0] == 'T':
            tailFiveCount = tailFiveCount + 1
    elif temp[1] == 1:
        oneCounter = oneCounter + 1
        if temp[0] == 'H':
            headOneCount = headOneCount + 1
        elif temp[0] == 'T':
            tailOneCount = tailOneCount + 1
print("P('H' | '5')= ", headFiveCount/fiveCounter)
print("P('T' | '5')= ", tailFiveCount/fiveCounter)
print("P('T' | '1')= ", tailOneCount/oneCounter)
print("P('H' | '1')= ", headOneCount/oneCounter)
```

```
results for alpha = 0.25
P('1' | 'H')=  0.25877712031558187
P('5' | 'H')=  0.0
P('1' | 'T')=  0.127662424648359
P('5' | 'T')=  0.12993971868720697

P('H' | '5')=  0.0
P('T' | '5')=  1.0
P('T' | '1')=  0.5922933499067744
P('H' | '1')=  0.4077066500932256
```

In [105]:
```
results = []
for x in range(10000):
    results.append(CoinTossDieRolling(0.75))
countHeads = 0
countTails = 0
countHeadsOnes = 0
countHeadsFives = 0
countTailsOnes = 0
countTailsFives = 0
for x in range(len(results)):
    temp = results[x]
    if temp[0] == 'H':
        countHeads = countHeads + 1
        if temp[1] == 1:
            countHeadsOnes = countHeadsOnes + 1
        if temp[1] == 5:
            countHeadsFives = countHeadsFives + 1
    elif temp[0] == 'T':
        countTails = countTails + 1
        if temp[1] == 1:
            countTailsOnes = countTailsOnes + 1
        if temp[1] == 5:
            countTailsFives = countTailsFives + 1

print("results for alpha = 0.75")
print("P('1' | 'H')= ", countHeadsOnes/countHeads)
print("P('5' | 'H')= ", countHeadsFives/countHeads)
print("P('1' | 'T')= ", countTailsOnes/countTails)
print("P('5' | 'T')= ", countTailsFives/countTails)
print()
headFiveCount = 0
tailFiveCount = 0
fiveCounter = 0
oneCounter = 0
headOneCount = 0
tailOneCount = 0
for x in range(len(results)):
    temp = results[x]
    if temp[1] == 5:
        fiveCounter = fiveCounter + 1
        if temp[0] == 'H':
            headFiveCount = headFiveCount + 1
        elif temp[0] == 'T':
            tailFiveCount = tailFiveCount + 1
    elif temp[1] == 1:
        oneCounter = oneCounter + 1
        if temp[0] == 'H':
            headOneCount = headOneCount + 1
        elif temp[0] == 'T':
            tailOneCount = tailOneCount + 1
print("P('H' | '5')= ", headFiveCount/fiveCounter)
print("P('T' | '5')= ", tailFiveCount/fiveCounter)
print("P('T' | '1')= ", tailOneCount/oneCounter)
print("P('H' | '1')= ", headOneCount/oneCounter)
```

```
# The numbers rolls that should not appear remain at 0, but P ("T or H" | numb
er) change drastically
```

```
results for alpha = 0.75
P('1' | 'H')=  0.2514698022447889
P('5' | 'H')=  0.0
P('1' | 'T')=  0.1212241653418124
P('5' | 'T')=  0.12917329093799682

P('H' | '5')=  0.0
P('T' | '5')=  1.0
P('T' | '1')=  0.13946044810242342
P('H' | '1')=  0.8605395518975766
```

2.2 Pregnancy Test

Part A

P(+|”P”) = (a and y)/y

P(−|”P”) = (b and y)/y

P(+|”N”) = (a and y^c)/y^c

P(−|”N”) = (b and y^c)/y^c

Part B

P(”P”|+) = (y and a)/a

P(”P”|+) = (y and b)/b

P(”P”|+) = (y^c and b)/b

P(”P”|+) = (y^c and b)/b

In [112]:
```
#Part C
pregnancyData = np.loadtxt(open("PregnancyData.csv","rb"),dtype = np.str,delim
iter=",")
print(pregnancyData)
print(pregnancyData[1])
print(pregnancyData[1][0])
for x in range (1, 10002)
```

```
[['Subject_ID' 'Preg/NonPreg' 'Test_result']
 ['0' 'N' '-']
 ['1' 'N' '+']
 ...
 ['9997' 'N' '+']
 ['9998' 'N' '-']
 ['9999' 'N' '+']]
['0' 'N' '-']
0
```

In [109]:
```
#Part D
```

```
[1, 2, 3]
```

In [ ]:
```

```