Lab 1: Birthday Probabilty

Introduction:

This Notebook's purpose is to solve the "*Birthday Probability Problem*". What that means is that given a number of people, represented by the variable **g**, and a number of experiments to run, represented by the variable **n**, what is the probability that of **g** people, 2 of those people share the same birthday if **n** experiments are ran? In addition, This Notebook attempts to find what is the minimum number of people needed in a group to ensure that 2 of them share the same birthday more than 50% of the time.

Author: Nigel Nelson

The cell that is below contains the method **findBirthdayProb(g, n)**, which is given a group size, **g**, and a number of experiments to run, **n**, will return the percent chance that out of **g** number of people, at least 2 of them will share a birthday.

In [1]:
```python
import random


def findBirthdayProb(g, n):
    """
    This method takes g and n as an argument in order
    to calculate the % chance 2 people will share a
    birthdate
    :param g: the number of people in the group
    :param n: the number of experiments to run
    :return: double representing the percent chance out of "n" experiments
    2 people from the "g" number of people will share a birthdate
    :author: Nigel Nelson
    """
    numDuplicateBirthdays = 0

    for y in range(n):
        listBirthdays = set()
        for x in range(g):
            listBirthdays.add(random.randint(1,365))
        if(len(listBirthdays) != g):
            numDuplicateBirthdays += 1
    return round((numDuplicateBirthdays/n)*100, 2)
```

The method **findBirthdayProb(g, n)** contains a doctstring explaining its functionality, as such the docstring can be displayed by inputting this method into the arument of the **help()** method which is shown below:

In [2]:
```python
help(findBirthdayProb)
```

```
Help on function findBirthdayProb in module __main__:
```

```
findBirthdayProb(g, n)
    This method takes g and n as an argument in order
    to calculate the % chance 2 people will share a
    birthdate
    :param g: the number of people in the group
    :param n: the number of experiments to run
    :return: double representing the percent chance out of "n" experiments
    2 people from the "g" number of people will share a birthdate
    :author: Nigel Nelson
```

The cell that is below contains the method **printBirthdayProb(g, n)** that uses **findBirthdayProb(g, n)** to print out a message that explains that the percent represents the chance that 2 people share a birthday, and explains which values of **g** and **n** were used.

In [3]:
```
def printBirthdayProb(g, n):

    print("The probability of a group of size ", g," having "
          "a common birthday is" , findBirthdayProb(g, n), "%")
```

The following cell calculates the smallest group size to have a probability of greater than 50% that two people share the same birthday.

If no value of n is given it defaults to running 10,000 experiments, however this can be changed by editing the argument in the call to minSize() below

In [4]:
```
%%time
def minSize(n=10000):
    """
    This method takes n as an argument in order to
    calculate the minimum group size neccessary so that
    greater there is more than a 50% there is a shared
    birthday.
    :param n: the number of experiments to run
    :return: double representing the percent chance out of "n" experiments
    2 people from the "g" number of people will share a birthdate
    :author: Nigel Nelson
    """

    for x in range(365):
        percent = findBirthdayProb(x, n)
        if(percent > 50):
            print(x, " is the smallest group size")
            break

minSize()
```

```
23  is the smallest group size
Wall time: 2.68 s
```

Data Visualization

The code below finds the percent chance that 2 people share a birthday for all group sizes between 1 and 365. It runs 100 tests as a default number but that value can be changed by changing the method call below. The scatter plot graph displays the group size on the x axis and the percent chance of a shared birthday on the y axis.

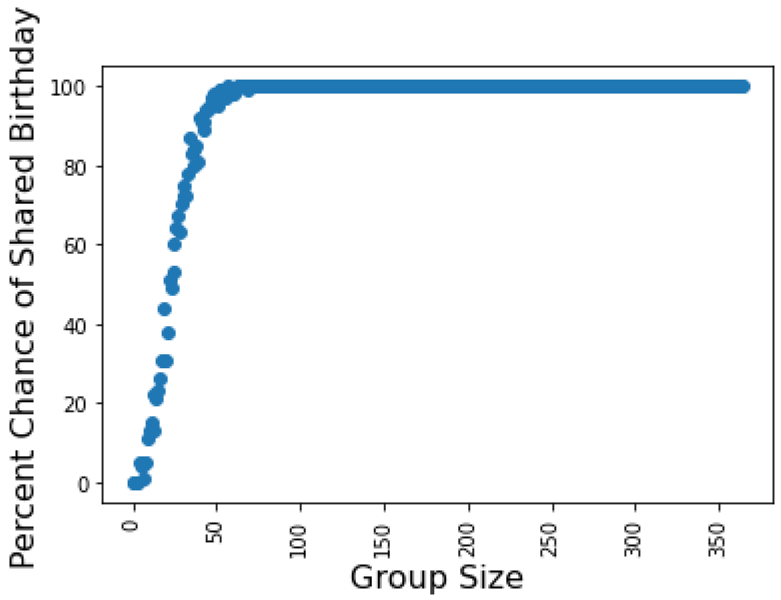
In [5]:
```
%%time
def getPercents(n = 100):
    values = []
    for x in range(365):
            values.append([x, findBirthdayProb(x, n)])
    return values

values = getPercents()
g_values = [f[0] for f in values]
percent_values = [f[1] for f in values]

import matplotlib.pyplot as plt
plt.scatter(g_values,percent_values)
plt.xlabel("Group Size", fontsize=16)
plt.xticks(rotation=90)
plt.ylabel("Percent Chance of Shared Birthday", fontsize=16)
```

```
Wall time: 6.5 s
```

Out[5]:
```
Text(0, 0.5, 'Percent Chance of Shared Birthday')
```



The above cell was found to be the longest running cell in this Notebook. As such benchmarking was conducted for the times to run different values of **n** for the above cell:

n size	laptop	Cluster
10	1.61 s	837 ms
100	15.1 s	5.69 s
10,000	2 min 30 s	57.1 s
100,000	27 min 18 s	9 min 46 s

As seen in the above benchmarking table, the time difference between the laptop and the Cluster are apparent even from an **n** size of just 10. Initially the cluster is approximately twice as fast at an **n** size of 10, but after that the cluster becomes about 3 times as fast as the laptop. I beleive that the reason for this is that at the small **n** value of 10 it is such a minimal amount of computing that the cluster strains to demonstrate all of its advantages. However, once there is an **n** size of 100 the operation size becomes significant enough tso that the cluster can take advatage of its increased core size, its greater cooling capacity, and overall computational prowess thanks to parralel processing.

Calculations

The following cell allows a user to change a group size variable **g** and a number of simulations variable **n** and prints the answer the question: “The probability of a group of size G having a common birthday is __%”

In [6]:
```
    try:
        g = int(input("Enter your desired group size: "))
        n = int(input("Enter the number of experiments you'd like to run: "))
        %time printBirthdayProb(g, n)
    except ValueError:
        print("Please enter an int!")
```

```
Enter your desired group size: 20
Enter the number of experiments you'd like to run: 1000
The probability of a group of size  20  having a common birthday is 43.3 %
Wall time: 22.3 ms
```

Conclusion

In this lab it was discovered that for a group size of 20 people, there is a **41.11%** chance that at least one pair will have the same birthday. In addition, It was found that when using a large enough number of experiments **n**, it was consitently found that the smallest group size to have a probability of greater than 50% that two people share the same birthday is **23**. It was found experimentally that **10,000** was the smallest value of **n** to give a consistent answer for the smallest group size question referenced in the previous sentence. However, it was also found experimentally that **500,000** was the smallest value of **n** to give a consistent answer (+- .1%) for the percent chance that a pair in a group **g** shared a birthday. Lastly, It was discovered that at the root of this "Birthday Problem" is hash collissions. The reason for this is that in the method used to calulcate percent chance of shared birthdates, **findBirthdayProb(g, n)**, random date place holders were added to a set **g** times. This is curcial because sets do not allow duplicate values to be added, when this is attempted a hash collision occurs, resulting in a single instance of the duplicate remaining in the set. This hash collision reults in the set size being smaller than **g**, which indicates that there is a shared birthday.