Lab 6

Exercise 1

```
'''
Name: Zach Nichols
Program: ex1.py
Function: Calculates the probability of at least one pair of people out of n number of people having the same birtdhay
but checking 1000 times if any of the group of n people have a match.
'''

'''Imports random for deciding birthdays'''
import random

'''The main function of the program'''
def main():
    '''Initializes variables for a file, n (the number of people), a list of matching birthdays (T/F) and the list of
probabilities for each n value.'''
    f = open("Birthday Probability.csv", "w")
    f.write("%s, %s\n" %("Probability %", "Number of People"))
    n = 5

    while n <= 50: #Runs a loop for 5 < n < 50 times
        sum = 0.0
        mblist = []
        for i in range(999): #Runs a loop for 1000 times to get a decent probability
            blist = []
            for k in range(n + 1): #Creates a new birthday for each loop
                v = random.randint(1,366)
                blist.append(v)
            a = compare(blist, n) #Compares the list of birthdays then appends the value receieved to a newer list
            if a == True:
                sum += 1

        '''Calculates the probability (in percent).'''
        sum = probability(sum)

        '''Prints the value of n used alongside the probabiltiy of birthdays matching'''
        f.write("%f, %d\n"%(sum, n))

        n += 1

    '''Closes the file'''
    f.close()
```

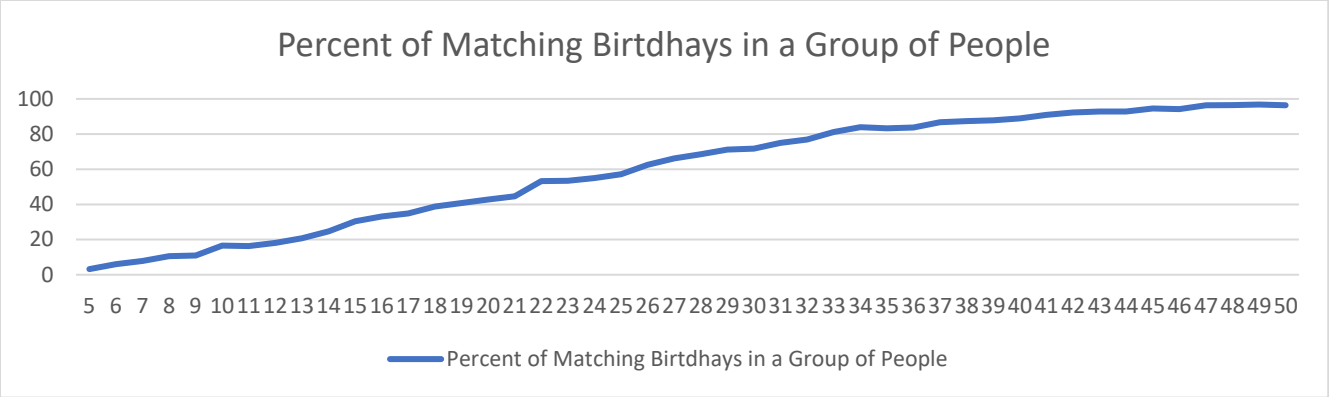
```
'''Compares each value of n, the algorithm sorts it from least to greatest to ensure the matches are right next to each other.
    If a match is found, the loop breaks and the function returns "True".'''
def compare(blist, n):
    matchingBirthdays = False
    blist.sort()
    for i in range(n):
        if blist[0] == blist[1]:
            matchingBirthdays = True
            return matchingBirthdays
        blist.pop(0)

    return matchingBirthdays

'''Calculates the probability (in percent).'''
def probability(sum):
    sum = sum / 10
    return sum

if __name__ == "__main__":
    main()
```

1. To my understanding, only one output was required.

This was the CSV file the program output.

Probability %	Number of People
3.2	5
6	6
7.8	7
10.6	8
10.9	9
16.6	10
16.3	11
18.1	12
20.8	13
24.7	14
30.4	15
33.1	16
34.9	17

38.8	18
40.8	19
42.8	20
44.6	21
53.2	22
53.4	23
55	24
57.2	25
62.5	26
66.2	27
68.5	28
71.2	29
71.7	30
75	31
76.9	32
81.2	33
83.9	34
83.3	35
83.7	36
86.7	37
87.4	38
87.8	39
88.9	40
90.9	41
92.3	42
92.8	43
92.8	44
94.6	45
94.2	46
96.4	47
96.5	48
96.8	49
96.4	50



X-axis – Number of people

Y-axis – Probability %

This is the graph of those outputs. As the number of people nears 50, the probability that two people have the same birthday is nearly 100%.

Exercise 2

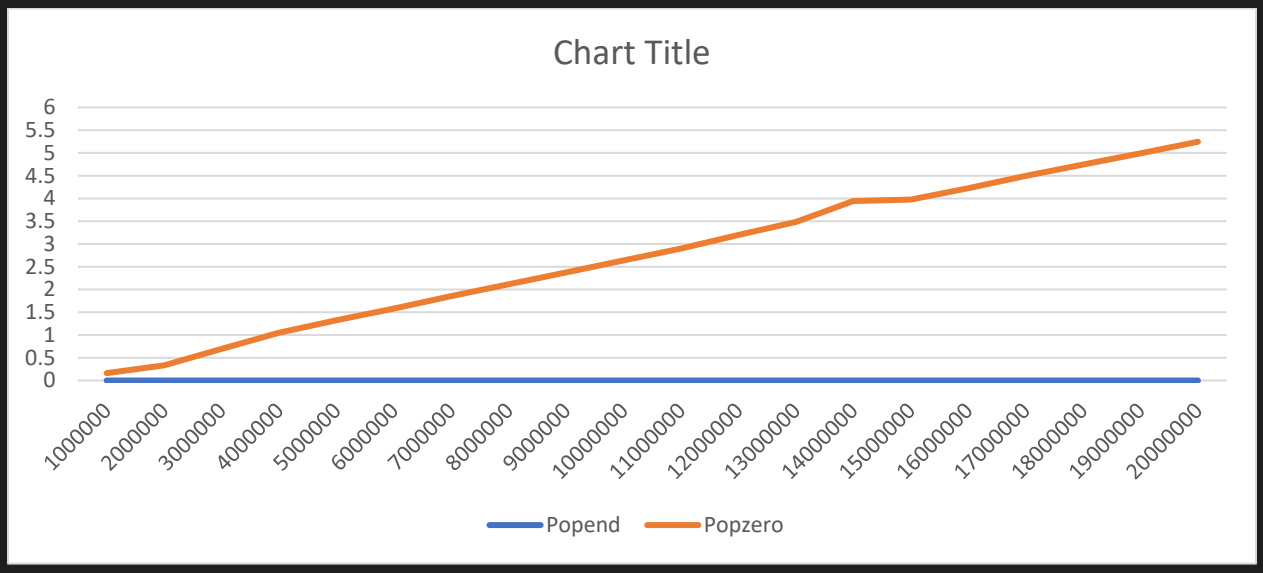
```
import timeit

f = open("times.csv", "w")
popzero = timeit.Timer("x.pop(0)", "from __main__ import x")
popend = timeit.Timer("x.pop()", "from __main__ import x")
for i in range(1000000,20000001,1000000):
    x = list(range(i))
    pt = popend.timeit(number=1000)
    x = list(range(i))
    pz = popzero.timeit(number=1000)
    print("%d, %f, %f" %(i,pz,pt))
    f.write("%d, %f, %f\n"%(i, pt, pz))
f.close()
```

1. This program will pop the item [0] in the list, as well as pop the last item in the list. As the list gets longer, I hypothesize (as we saw in class as well) that the time it will take to do popzero will increase, and popend will take the same consistent time. This is the csv file output. I added the headings for clarity's sake.

List Size	Popend	Popzero

1000000	0.000024	0.162268
2000000	0.000024	0.333627
3000000	0.000037	0.693666
4000000	0.000024	1.048559
5000000	0.000023	1.325113
6000000	0.000023	1.58162
7000000	0.000024	1.855145
8000000	0.000024	2.113911
9000000	0.000023	2.376669
10000000	0.000023	2.639487
11000000	0.000024	2.897781
12000000	0.000025	3.199498
13000000	0.000024	3.482389
14000000	0.000024	3.944531
15000000	0.000024	3.975934
16000000	0.000025	4.225311
17000000	0.000023	4.494579
18000000	0.000025	4.744671
19000000	0.000024	4.992319
20000000	0.000024	5.245187

Here is the graph of these outputs.



X-axis – List size

Y-axis - Time

While it is hard to see popend, it takes the same amount of time regardless of list, as it does not need to resort every item in the list. Popezero shows almost linear growth as the list size gets longer. This is due to popend having a Big-O notation of 1, while popzero has a Big-O of n since it has to reorder the entire list.