First homework report:

For the birthday paradox:

```
'''
Suppose that we have a class of n students. We assume that their birthdays
are uniformly distributed over the days of the year.
Question I: What is the probability that the class has two students having
the same birthday? Write python code to simulate this.
Question II: For what value of n (number of students in the class) the
above probability is 0.5? Use the Python code you developed in question I,
to find that value by trial and error (or a more sophisticated algorithm)
'''

students = 2
probability = 365 / 365
divisor = 365
for i in range(1, students+1):
   multiplier = (365 - i + 1)
   probability *= multiplier
   probability /= divisor
final_probability = 1 - probability
print(f'The probability of having {students} students share the same
birthday is :\n{final_probability*100}%')
while probability >= 0.5:
   students += 1
   multiplier = (365 - students + 1)
   probability *= multiplier
   probability /= divisor
fifty_percent_benchmark = 1 - probability
print(f"When there is {students} students, there shall be
{fifty_percent_benchmark*100}% chance of having same birthday(s) amongst
them.")
```

Results:

```
The probability of having 2 students share the same birthday is :
0.2739726027397249%
When there is 23 students, there shall be 50.72972343239856% chance of
having same birthday(s) amongst them.
```

The logic of these codes basically adopts the idea of finding the probability of not having the same birthday, then, I compute the complement of the former, and I obtain the probability of having same birthday.

For the Central Limit Theorem:

```
'''
Validate the CLT by simulation. Proceed as follows. Pick a number n
(n=1,2,3,...) 1. Pick some random variable.
Ideas: (a) coin flips (0,1) (b) die tossing (1,2,3,4,5,6) (c) U(0,1)
2. Sample from this random number n times (independently) 3. Calculate the
average
4. Calculate Z, based on a previous slide.
5. Repeat 1-4 many times (e.g. 10000) and make a histogram of the results.
Plot on the same histogram the Standard normal distribution.
'''

import random
import statistics as s
lst1 = []
loop_times = 100
Z_score = 0
lst2 = []
```

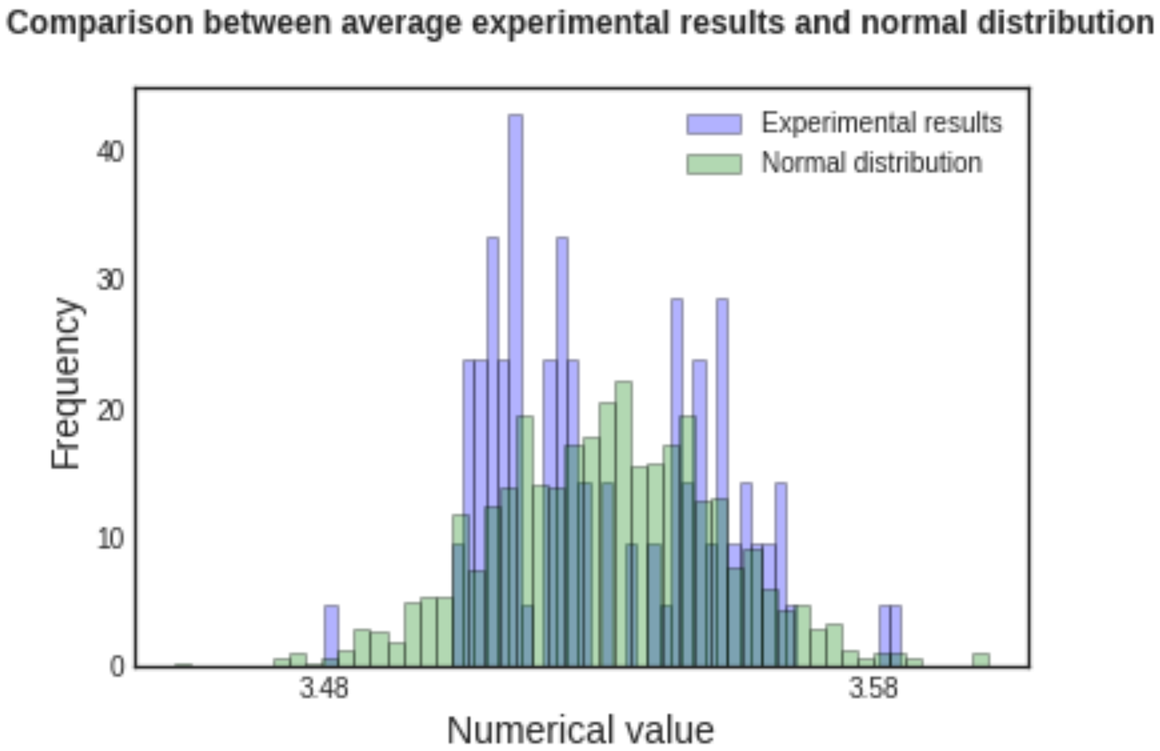
```
lst3 = []
lst4 = []
z = 0.5
z_score = 0
for i in range(loop_times):
    for j in range (100):
        appendixer = random.randint(1,6)
        lst1.append(appendixer)
    average = (sum(lst1)) / (len(lst1))
    lst2.append(average)
    standard_d = s.stdev(lst1)
    lst4.append(standard_d)
standard_deviation = s.stdev(lst2)
lst3.append(standard_deviation)
average_average = sum(lst2)/len(lst2)
print(f"The average result of rolling dices {loop_times} times is
{average_average}.")
average_standard_deviation = sum(lst3)/len(lst3)
average_standard_d = sum(lst4)/len(lst4)
positive_sigma = average_average + average_standard_d
negative_sigma = average_average - average_standard_d
positive_2_sigma = average_average + 2*average_standard_d
negative_2_sigma = average_average - 2*average_standard_d
positive_z_sigma = average_average + z*average_standard_d
negative_z_sigma = average_average - z*average_standard_d
print(f"The standard deviation in this rolling dice experiment is
{average_standard_d}, which means 68% of dice results are amidst the range
of {negative_sigma}~{positive_sigma}, 95% of dice results are amidst
{negative_2_sigma}~{positive_2_sigma}.")
print(f"The standard deviation in the average of all rolling dice
experiments is {average_standard_deviation}")
for k in range(len(lst1)):
    if lst1[k] < positive_z_sigma:
        z_score += 1
probability_z_score = z_score/len(lst1)
print(f"The Z score for z = {z} is {probability_z_score*100}%")

import numpy as np
import matplotlib.pyplot as plt
import pylab
from pylab import xticks
plt.style.use('seaborn-white')
x1 = lst2
x2 = np.random.normal(average_average,average_standard_deviation,1000)
kwargs1 = dict(bins=50, density=True, alpha=0.3, histtype="bar",
color="blue", edgecolor="black")
kwargs2 = dict(bins=50, density=True, alpha=0.3, histtype="bar",
color="green", edgecolor="black")
plt.title('Comparison between average experimental results and normal
distribution\n',
          fontweight ="bold")
plt.xticks(np.arange(min(x1), max(x2)+1, 0.1))
pylab.rc("axes", linewidth=8.0)
pylab.rc("lines", markeredgewidth=2.0)
plt.xlabel('Numerical value', fontsize=14)
plt.ylabel('Frequency', fontsize=14)
pylab.xticks(fontsize=10)
pylab.yticks(fontsize=10)
plt.hist(x1, **kwargs1,label = "Experimental results",
range=[min(x1),max(x1)])
plt.hist(x2, **kwargs2, label = "Normal distribution",
```

```
range=[min(x2),max(x2)])
plt.legend(prop ={'size': 10})
plt.show()
```

Results:

```
The average result of rolling dices 100 times is 3.5310718384172826.
The standard deviation in this rolling dice experiment is
1.7176811837099404, which means 68% of dice results are amidst the range of
1.8133906547073422~5.248753022127223, 95% of dice results are amidst
0.09570947099740179~6.966434205837164.
The standard deviation in the average of all rolling dice experiments is
0.020387951693106646
The Z score for z = 0.5 is 66.19%
```



The logic of these codes tries to simulate 100 times the event that we throw the dice for 100 times. I calculate the average and standard deviation of those 10,000 results, and plot an histogram. I would consider the two histograms are quite similar, the overall trend is roughly the same, it is just that my experimental-results-derived plots are somehow inconsistent than the actual normal distribution.