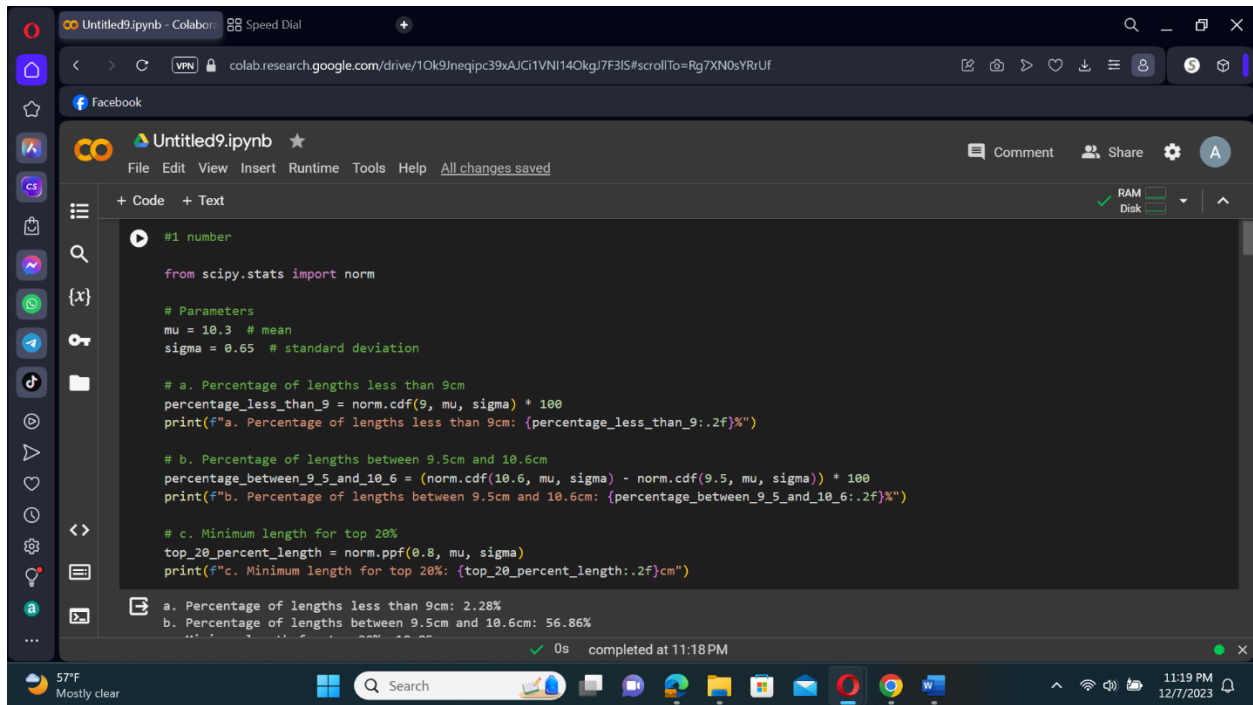


1. Assuming that the lengths of American anchovies appease the normal distribution with the mean $\mu = 10.3\text{cm}$ and standard deviation $\sigma = 0.65\text{cm}$, please find the percentages of the lengths in the population of American anchovies.

a. Less than 9cm.

b. Between 9.5cm and 10.6cm.

c. What is the minimum length if a restaurant claimed that the lengths of the sold anchovies are in the top of 20%?



The screenshot shows a Jupyter Notebook titled 'Untitled9.ipynb' in a Colaboratory environment. The code defines parameters for a normal distribution and calculates the required percentages and minimum length.

```
#1 number

from scipy.stats import norm

# Parameters
mu = 10.3 # mean
sigma = 0.65 # standard deviation

# a. Percentage of lengths less than 9cm
percentage_less_than_9 = norm.cdf(9, mu, sigma) * 100
print(f"a. Percentage of lengths less than 9cm: {percentage_less_than_9:.2f}%")

# b. Percentage of lengths between 9.5cm and 10.6cm
percentage_between_9_5_and_10_6 = (norm.cdf(10.6, mu, sigma) - norm.cdf(9.5, mu, sigma)) * 100
print(f"b. Percentage of lengths between 9.5cm and 10.6cm: {percentage_between_9_5_and_10_6:.2f}%")

# c. Minimum length for top 20%
top_20_percent_length = norm.ppf(0.8, mu, sigma)
print(f"c. Minimum length for top 20%: {top_20_percent_length:.2f}cm")
```

The output of the code is displayed below the code cells:

```
a. Percentage of lengths less than 9cm: 2.28%
b. Percentage of lengths between 9.5cm and 10.6cm: 56.86%
```

The status bar at the bottom indicates that the code was completed at 11:18 PM on 12/7/2023.

```

from scipy.stats import norm

# Parameters
mu = 10.3 # mean
sigma = 0.65 # standard deviation

# a. Percentage of lengths less than 9cm
percentage_less_than_9 = norm.cdf(9, mu, sigma) * 100
print(f"a. Percentage of lengths less than 9cm: {percentage_less_than_9:.2f}%")

# b. Percentage of lengths between 9.5cm and 10.6cm
percentage_between_9_5_and_10_6 = (norm.cdf(10.6, mu, sigma) - norm.cdf(9.5, mu, sigma)) * 100
print(f"b. Percentage of lengths between 9.5cm and 10.6cm: {percentage_between_9_5_and_10_6:.2f}%")

# c. Minimum length for top 20%
top_20_percent_length = norm.ppf(0.8, mu, sigma)
print(f"c. Minimum length for top 20%: {top_20_percent_length:.2f}cm")

a. Percentage of lengths less than 9cm: 2.28%
b. Percentage of lengths between 9.5cm and 10.6cm: 56.86%
c. Minimum length for top 20%: 10.85cm

```

2. If the random variables X and Y are normal distributions with $\mu = 10$ & $\sigma = 3$ and $\mu = 15$ & $\sigma = 8$, namely, $X \sim N(10, 3)$ and $Y \sim N(15, 8)$, and they are independent, what is the probability distribution and statistical parameters of (1) $X + Y$ (2) $X - Y$ (3) $3X$ (4) $4X + 5Y$.

(1) $X + Y$

The mean is obtained by summing the individual means: $\mu_x + \mu_y = 10 + 15$, resulting in a mean of 25. The variance, calculated as the sum of squared standard deviations, is $3^2 + 8^2$, equaling 73. Therefore, the probability distribution is denoted as $N(25, \sqrt{73})$.

(2) $X - Y$

The mean, derived from the difference of individual means ($\mu_x - \mu_y = 10 - 15$), is -5. The variance remains consistent at 73. The corresponding probability distribution is represented as $N(-5, \sqrt{73})$.

(3) $3X$

With the mean of 3 times μ_x ($3 * \mu_x$), the result is $3 * 10 = 30$. The variance calculated as $(3 * \sigma_x)^2$, becomes $(3 * 3)^2$, resulting in 81. Consequently, the probability distribution is $N(30, 9)$.

(4) $4X + 5Y$

The mean for this expression ($4 * \mu_x + 5 * \mu_y$) is calculated as $4 * 10 + 5 * 15 = 40 + 75$, yielding a mean of 115. The variance, derived from $(4 * \sigma_x)^2 + (5 * \sigma_y)^2$, becomes $(4 * 3)^2 + (5 * 8)^2$, resulting in 344. The probability distribution is expressed as $N(115, \sqrt{344})$.

3. For the students in Engineering School, please write Python program to verify the mean $\mu = np$ and standard deviation $\sigma = \sqrt{npq}$ for $p=0.05$ and selecting any n greater than 50 in binomial distribution.

The screenshot shows a Jupyter Notebook titled 'Untitled9.ipynb' in a web browser. The code defines a function `verify_binomial_distribution(n, p)` that generates binomial data, calculates the mean and standard deviation, and compares them to theoretical values. The function is executed, and the output is displayed at the bottom of the cell.

```
# 3 number
import numpy as np
import scipy.stats as stats

def verify_binomial_distribution(n, p):
    # Generate a binomial distribution with n trials and probability p
    binomial_data = np.random.binomial(n, p, 10000)

    # Calculate mean and standard deviation from the generated data
    mean_calculated = np.mean(binomial_data)
    std_dev_calculated = np.std(binomial_data)

    # Calculate actual mean and standard deviation based on theory
    mean_expected = n * p
    std_dev_expected = np.sqrt(n * p * (1 - p))

    # Print the results
    print(f"Number of trials (n): {n}")
    print(f"Probability of success (p): {p}")
    print("\nVerification Results:")
    print(f"Calculated Mean: {mean_calculated:.4f}")
    print(f"Actual Mean (np): {mean_expected:.4f}")
```

0s completed at 11:18 PM

The screenshot shows the same Jupyter Notebook with the function `verify_binomial_distribution` called with `n_value = 90` and `p_value = 0.05`. The output displays the calculated and expected mean and standard deviation values.

```
print("\nVerification Results:")
print(f"Calculated Mean: {mean_calculated:.4f}")
print(f"Actual Mean (np): {mean_expected:.4f}")
print(f"Calculated Standard Deviation: {std_dev_calculated:.4f}")
print(f"Actual Standard Deviation (sqrt(npq)): {std_dev_expected:.4f}")

# Set values for n and p
n_value = 90
p_value = 0.05

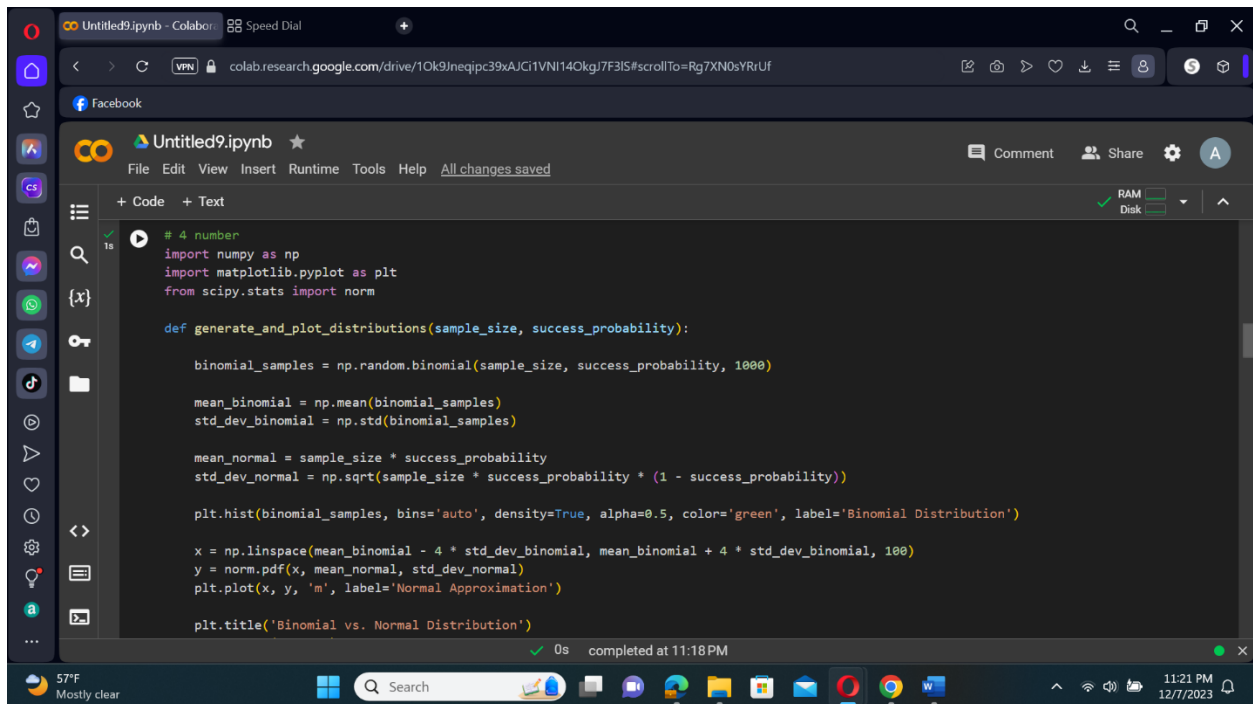
# Call the function to verify the binomial distribution
verify_binomial_distribution(n_value, p_value)
```

Number of trials (n): 90
Probability of success (p): 0.05

Verification Results:
Calculated Mean: 4.4949
Actual Mean (np): 4.5000
Calculated Standard Deviation: 2.0754
Actual Standard Deviation (sqrt(npq)): 2.0676

0s completed at 11:18 PM

4. In general, if $np > 5$ and $nq > 5$ in binomial distribution, binomial probabilities can be approximated using the normal distribution. Please select any big enough n and p 's values to verify in Python program or Excel and plot the histogram.



```
# 4 number
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import norm

def generate_and_plot_distributions(sample_size, success_probability):

    binomial_samples = np.random.binomial(sample_size, success_probability, 1000)

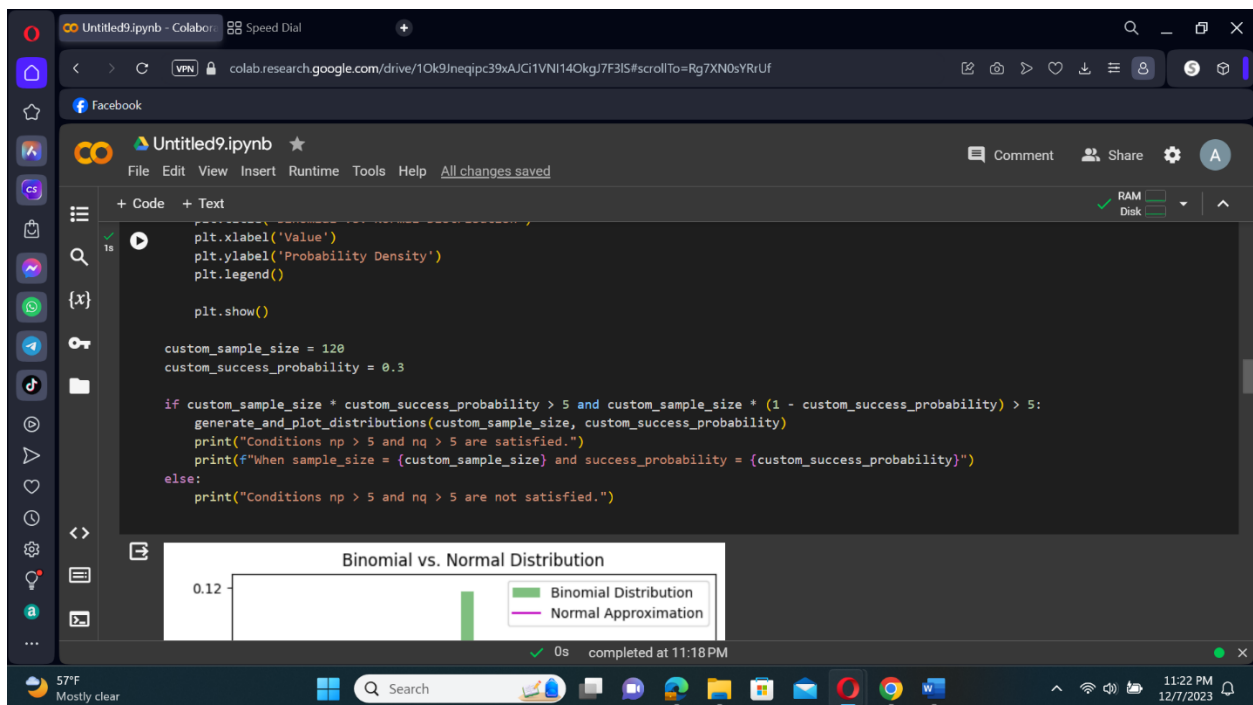
    mean_binomial = np.mean(binomial_samples)
    std_dev_binomial = np.std(binomial_samples)

    mean_normal = sample_size * success_probability
    std_dev_normal = np.sqrt(sample_size * success_probability * (1 - success_probability))

    plt.hist(binomial_samples, bins='auto', density=True, alpha=0.5, color='green', label='Binomial Distribution')

    x = np.linspace(mean_binomial - 4 * std_dev_binomial, mean_binomial + 4 * std_dev_binomial, 100)
    y = norm.pdf(x, mean_normal, std_dev_normal)
    plt.plot(x, y, 'm', label='Normal Approximation')

    plt.title('Binomial vs. Normal Distribution')
```





5. In coin tossing experiments, please find the probability of the exact 6 heads from 12 tossing by ONLY using the normal distribution method.

```
# 5 number
import scipy.stats as stats
import math

# Given values
n_tosses = 12
p_heads = 0.5
k_heads = 6

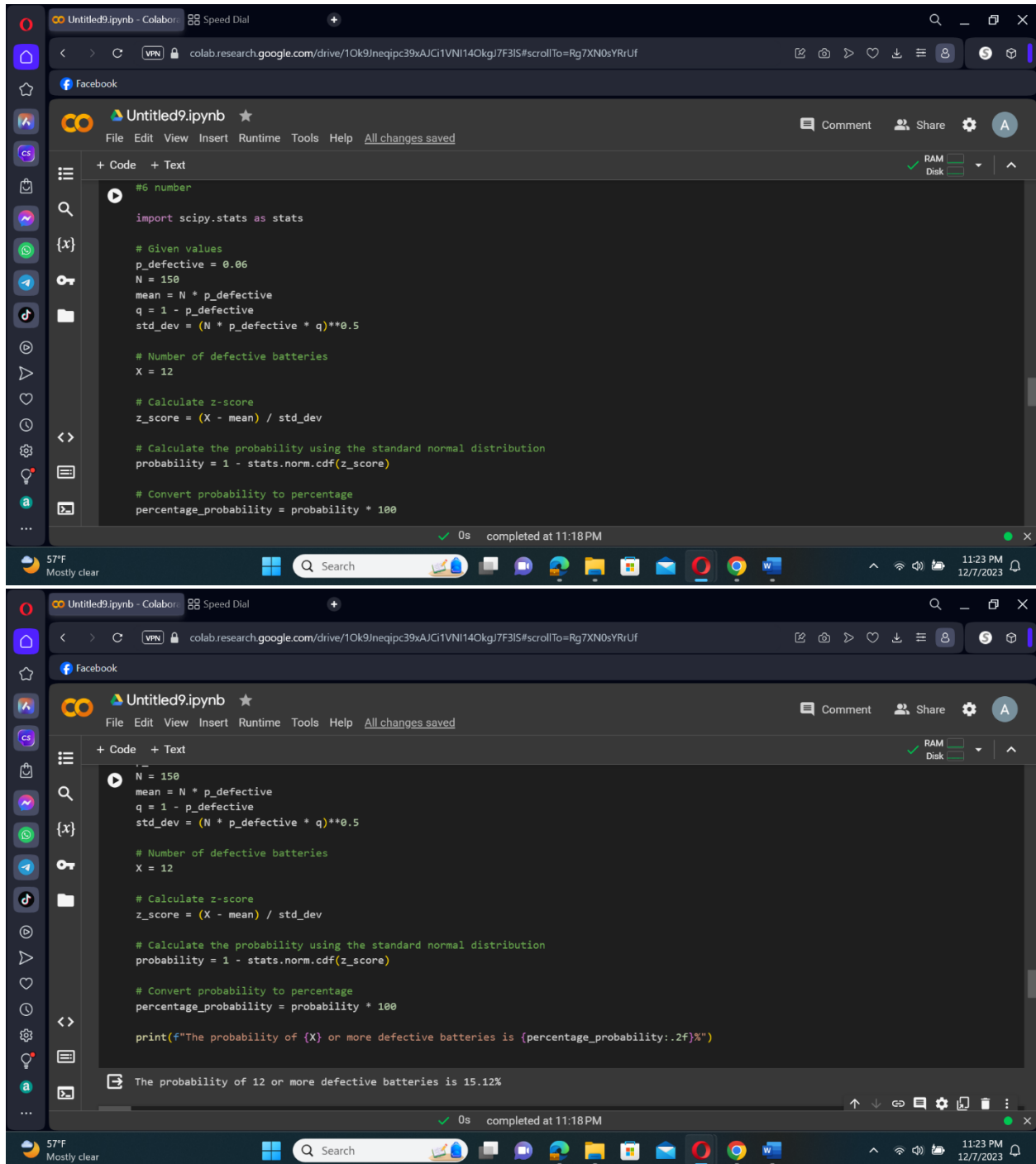
# Calculate mean ( $\mu$ ) and standard deviation ( $\sigma$ )
mean = n_tosses * p_heads
std_dev = math.sqrt(n_tosses * p_heads * (1 - p_heads))

# Calculate the probability using the normal distribution formula
probability = 1 / (math.sqrt(2 * math.pi) * std_dev) * math.exp(-(k_heads - mean)**2 / (2 * std_dev**2))

print(f"The probability of getting exactly {k_heads} heads in {n_tosses} coin tosses is {probability:.4f}")
```

The probability of getting exactly 6 heads in 12 coin tosses is 0.2303

6. Given that the defective rate of a product of the batteries in a manufacturing company is 6%, 150 batteries are randomly selected from the population. Please find the probability of 12 or more defective ones in them by ONLY using the normal distribution method.



The image displays two screenshots of a Jupyter Notebook interface, showing the process of calculating the probability of 12 or more defective batteries using the normal distribution method.

Top Screenshot: The notebook contains the following code:

```
#6 number

import scipy.stats as stats

# Given values
p_defective = 0.06
N = 150
mean = N * p_defective
q = 1 - p_defective
std_dev = (N * p_defective * q)**0.5

# Number of defective batteries
X = 12

# Calculate z-score
z_score = (X - mean) / std_dev

# Calculate the probability using the standard normal distribution
probability = 1 - stats.norm.cdf(z_score)

# Convert probability to percentage
percentage_probability = probability * 100
```

The code calculates the mean and standard deviation for the number of defective batteries, then calculates the z-score for 12 defective batteries. The probability is calculated using the standard normal distribution function, and the result is converted to a percentage.

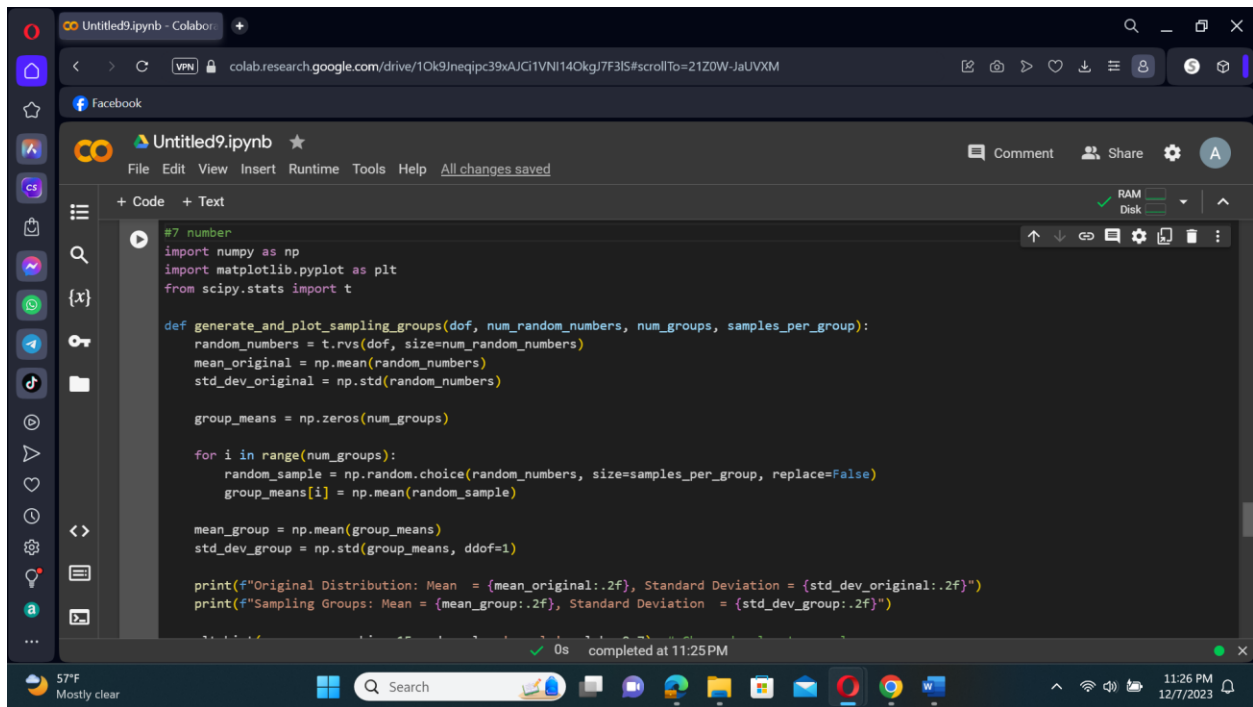
Bottom Screenshot: The notebook contains the same code as the top screenshot, but with an additional line of code to print the result:

```
print(f"The probability of {X} or more defective batteries is {percentage_probability:.2f}%")
```

The output of the code is displayed at the bottom of the notebook:

```
The probability of 12 or more defective batteries is 15.12%
```

7. For the students in Engineering School, please write a Python program by calling functions in the following link to create 100 random numbers in T distribution with $df = 10$ (degree of freedom) and calculate the mean μ and standard deviation σ . After that, the 30 samples will be randomly selected from these random numbers in each sampling group. A total of 15 sampling groups should be created. Based on Central Limit Theorem (CLT), the mean value \bar{x} in total 15 sampling group is roughly the mean μ of 100 random numbers and $\sigma_{\bar{x}} = \sigma/\sqrt{n}$. Please verify it and plot the histogram, which should be normal distribution. For Business school students, complete the above process in Excel.



```

#7 number
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import t

def generate_and_plot_sampling_groups(dof, num_random_numbers, num_groups, samples_per_group):
    random_numbers = t.rvs(dof, size=num_random_numbers)
    mean_original = np.mean(random_numbers)
    std_dev_original = np.std(random_numbers)

    group_means = np.zeros(num_groups)

    for i in range(num_groups):
        random_sample = np.random.choice(random_numbers, size=samples_per_group, replace=False)
        group_means[i] = np.mean(random_sample)

    mean_group = np.mean(group_means)
    std_dev_group = np.std(group_means, ddof=1)

    print(f"Original Distribution: Mean = {mean_original:.2f}, Standard Deviation = {std_dev_original:.2f}")
    print(f"Sampling Groups: Mean = {mean_group:.2f}, Standard Deviation = {std_dev_group:.2f}")
  
```

