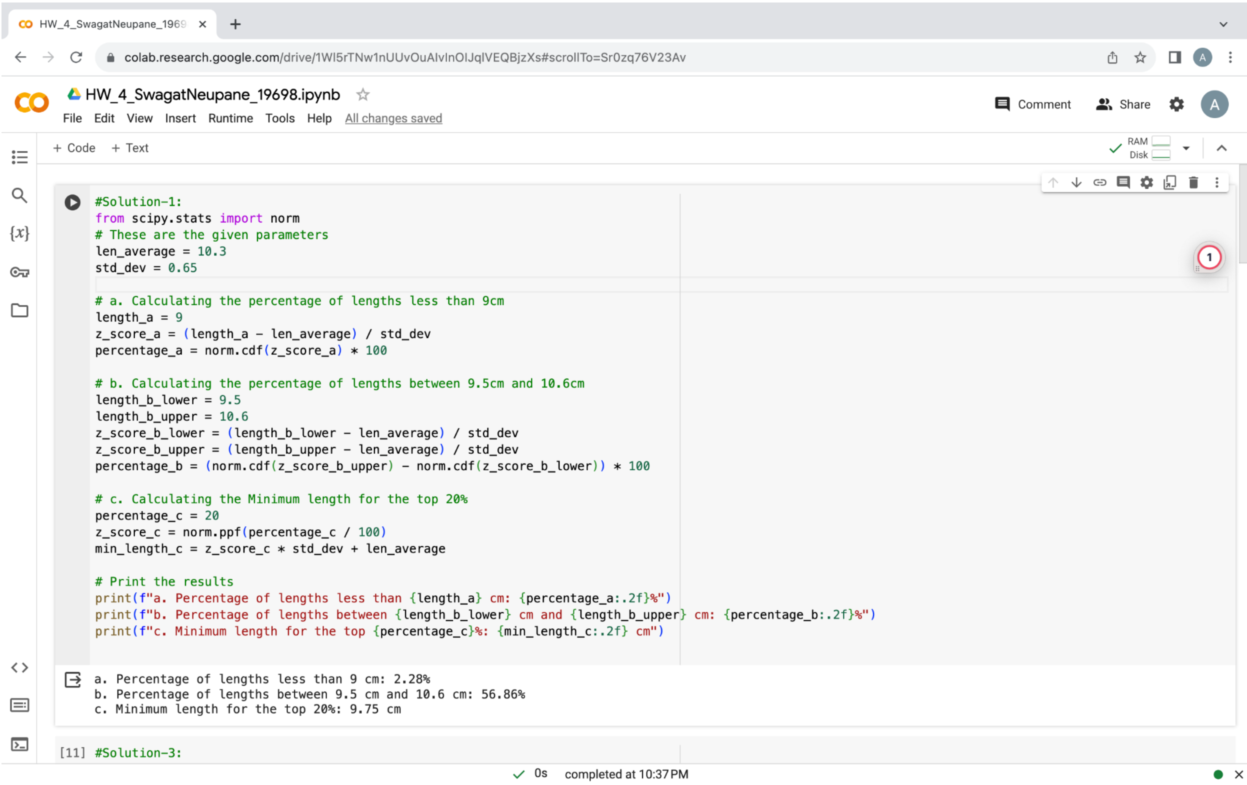
#Solution-1:




```python
#Solution-1:
from scipy.stats import norm
# These are the given parameters
len_average = 10.3
std_dev = 0.65

# a. Calculating the percentage of lengths less than 9cm
length_a = 9
z_score_a = (length_a - len_average) / std_dev
percentage_a = norm.cdf(z_score_a) * 100

# b. Calculating the percentage of lengths between 9.5cm and 10.6cm
length_b_lower = 9.5
length_b_upper = 10.6
z_score_b_lower = (length_b_lower - len_average) / std_dev
z_score_b_upper = (length_b_upper - len_average) / std_dev
percentage_b = (norm.cdf(z_score_b_upper) - norm.cdf(z_score_b_lower)) * 100

# c. Calculating the Minimum length for the top 20%
percentage_c = 20
z_score_c = norm.ppf(percentage_c / 100)
min_length_c = z_score_c * std_dev + len_average

# Print the results
print(f"a. Percentage of lengths less than {length_a} cm: {percentage_a:.2f}%")
print(f"b. Percentage of lengths between {length_b_lower} cm and {length_b_upper} cm: {percentage_b:.2f}%")
print(f"c. Minimum length for the top {percentage_c}%: {min_length_c:.2f} cm")
```

```
a. Percentage of lengths less than 9 cm: 2.28%
b. Percentage of lengths between 9.5 cm and 10.6 cm: 56.86%
c. Minimum length for the top 20%: 9.75 cm
```

```
[11] #Solution-3:
```


#Solution-2:

Solution:

If X and Y are independent normal distributions with means (μ) = 10 and standard deviations (σ) =3.

(1) X + Y:

Explanation:

The sum of X and Y is also a normal distribution. The mean of the sum is the sum of the means, and the variance of the sum is the sum of the variances.

Mean (μ_X+Y) = μ_X + μ_Y = 10 + 15 = 25

And, variance (σ^2_{x+y}) = μ_X^2 + μ_Y^2= 3^2 + 8^2 = 73

So,the probability distribution for is (X + Y)~ N(25, $\sqrt{73}$).

(2) X - Y:

The difference of two X and Y is a normal distribution.

Mean (μ_X-Y) = μ_X - μ_Y = 10 - 15 = -5

Variance (σ^2_{x+y})= 3^2 + 8^2 = 9 + 64 =73

So, the probability distribution for is (X - Y) ~ N(-5, $\sqrt{73}$).

(3) 3X:
Mean (μ_3X) = 3μ_X
= 3 * 10 = 30
Variance (σ^2_{3x}) =(3*3)^2= 81
So, the probability distribution for 3X is 3X ~ N(30, 9).
(4) 4X + 5Y:
Mean (μ_4X+5Y)
= 4μ_X + 5μ_Y
= 4*10 + 5*15
= 40 + 75
= 115
Variance (σ^2_{4x+5y})
= ((4.σx)^2 + (5.σ3)^2
= (4.3)^2 + (5.8)^2
= 344
Now,
SD = $\sqrt{344}$
So, the probability is N(115, $\sqrt{344}$).

#Solution-3:

HW_4_SwagatNeupane_19698.ipynb

File Edit View Insert Runtime Tools Help All changes saved

```
a. Percentage of lengths less than 9 cm: 2.28%
b. Percentage of lengths between 9.5 cm and 10.6 cm: 56.86%
c. Minimum length for the top 20%: 9.75 cm
```

[11]
```python
#Solution-3:
import numpy as np
from scipy.stats import binom


p = 0.05
n = 100  # Selecting n greater than 50

# Generating a binomial distribution
binomial_dist = binom(n, p)

# Calculating the mean and standard deviation
mean_calculated = n * p
std_dev_calculated = np.sqrt(n * p * (1 - p))

mean_from_dist = binomial_dist.mean()
std_dev_from_dist = binomial_dist.std()

print(f"Generated Binomial Distribution with p={p} and n={n}")
print(f"Expected Mean (μ): {mean_calculated}, Calculated Mean: {mean_from_dist}")
print(f"Expected Standard Deviation (σ): {std_dev_calculated}, Calculated Standard Deviation: {std_dev_from_dist}")
```

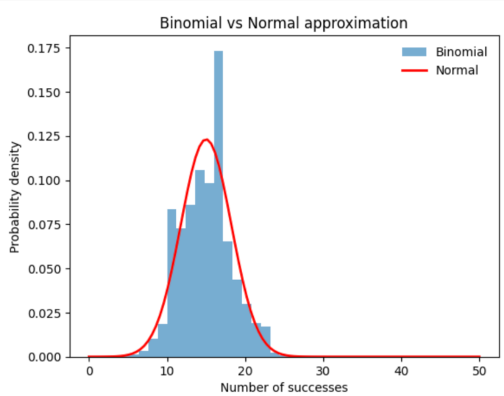
```
Generated Binomial Distribution with p=0.05 and n=100
Expected Mean (μ): 5.0, Calculated Mean: 5.0
Expected Standard Deviation (σ): 2.179449471770337, Calculated Standard Deviation: 2.179449471770337
```

```python
#Solution-4:
import numpy as np
import matplotlib.pyplot as plt
```

0s completed at 10:37 PM

#Solution-4:

```python
#Solution-4:
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import norm, binom

n = 50
p = 0.3

if n*p > 5 and n*(1-p) > 5:
    # Binomial distribution
    binom_dist = binom(n, p)

    # Normal distribution
    mu = n*p
    sigma = np.sqrt(n*p*(1-p))
    norm_dist = norm(mu, sigma)

    # Plotting histograms
    fig, ax = plt.subplots()
    ax.hist(binom_dist.rvs(size=10000), bins=20, density=True, alpha=0.6, label='Binomial')
    ax.plot(np.linspace(0, n, 100), norm_dist.pdf(np.linspace(0, n, 100)), 'r-', lw=2, label='Normal')
    ax.legend(loc='best', frameon=False)
    ax.set_xlabel('Number of successes')
    ax.set_ylabel('Probability density')
    ax.set_title('Binomial vs Normal approximation')
    plt.show()
else:
    print('n*p and n*(1-p) must both be greater than 5')
```




```python
#solution-7:
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import t
```

#Solution-5:
Given that,
Probability(p)=6
n=12

Formula to find probability using normal distribution is,

P(X=k) = $\frac{1}{\sigma\sqrt{2\pi}} exp(-\frac{(x-\mu)^2}{2\sigma^2})$

Here We know that,

Mean(μ) = np

Standard deviation (σ)=$\sqrt{npq}$

Now,

The mean of the number of heads is μ = np = 12 * 0.5 = 6.
The Standard Deviation of head is σ=$\sqrt{npq} \Rightarrow \sqrt{12 * 0.5 * 0.5} = \sqrt{3}$

Now, let's substitute the value in above formula,

P(X=k)= $\frac{1}{\sigma\sqrt{2\pi}} exp(-\frac{(x-\mu)^2}{2\sigma^2})$

P(X= 6) = $\frac{1}{\sqrt{3}\sqrt{2\pi}} exp(-\frac{(6-6)^2}{2*3^2})$

$= \frac{1}{\sqrt{6\pi}}$ ~ 0.1508

#Solution-6:

When the defective rate of a product of the batteries in a company is 6%,
Then, the probability of a defective battery is p=6% = 0.06.
And, the probability of battery not being defective(q) can be calculated using this formula,
q= 1-p
= 1-0.006
=0.94
now , we know that,
P = 0.06 and n=150
then , Mean μ = np = 150 * 0.06 = 9
And, the standard deviation(σ)= $\sqrt{npq}$ $\Rightarrow$ $\sqrt{150 * 0.06 * 0.94}$
=$\sqrt{8.46}$ $\Rightarrow$ ~ 2.908

The variance of X is σ^2 = npq

= 150 * 0.06 * 0.94

= 8.46

The probability of 12 or more defective batteries can be calculated using the normal approximation to the binomial distribution. With the mean subtracted and the standard deviation divided, we may standardize the random variable X:

Z = (X - μ) / σ = (12 - 9) / 2.908 ≈ 1.031

Using standard normal distribution:

P ($X \geq$ 12) = 1 – P (X <12)

= 1 – 0.8485 ~ 0.1515.

Therefore, the probability of 12 or more defective batteries is 0.1515.

#Solution-7:

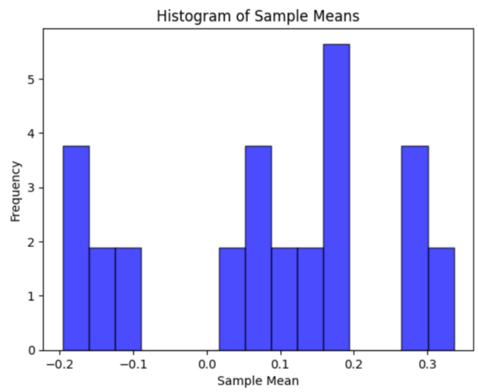
HW_4_SwagatNeupane_19698.ipynb

```python
#solution-7:
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import t

df = 10
# Creating 100 random numbers from the T distribution
random_numbers = t.rvs(df, size=100)
# Calculating mean and standard deviation of the 100 random numbers
mu = np.mean(random_numbers)
sigma = np.std(random_numbers)
num_groups = 15
samples_per_group = 30
# Creating an array to store sample means
sample_means = np.zeros(num_groups)
# Generating sampling groups and calculate sample means
for i in range(num_groups):
    # Randomly select 30 samples from the 100 random numbers
    samples = np.random.choice(random_numbers, size=samples_per_group, replace=False)
    # Calculate the sample mean
    sample_means[i] = np.mean(samples)
sigma_x = sigma / np.sqrt(samples_per_group)
print(f"Mean of 100 random numbers: {mu:.4f}")
print(f"Standard deviation of 100 random numbers: {sigma:.4f}")
print(f"Mean of sample means: {np.mean(sample_means):.4f}")
print(f"Expected standard deviation of sample means based on CLT: {sigma_x:.4f}")

plt.hist(sample_means, bins=15, density=True, alpha=0.7, color='b', edgecolor='black')
plt.title('Histogram of Sample Means')
plt.xlabel('Sample Mean')
plt.ylabel('Frequency')
plt.show()
```

```
Mean of 100 random numbers: 0.1152
```

```
plt.ylabel('Frequency')
plt.show()
```

```
Mean of 100 random numbers: 0.1152
Standard deviation of 100 random numbers: 1.0642
Mean of sample means: 0.0831
Expected standard deviation of sample means based on CLT: 0.1943
```



0s completed at 10:37 PM