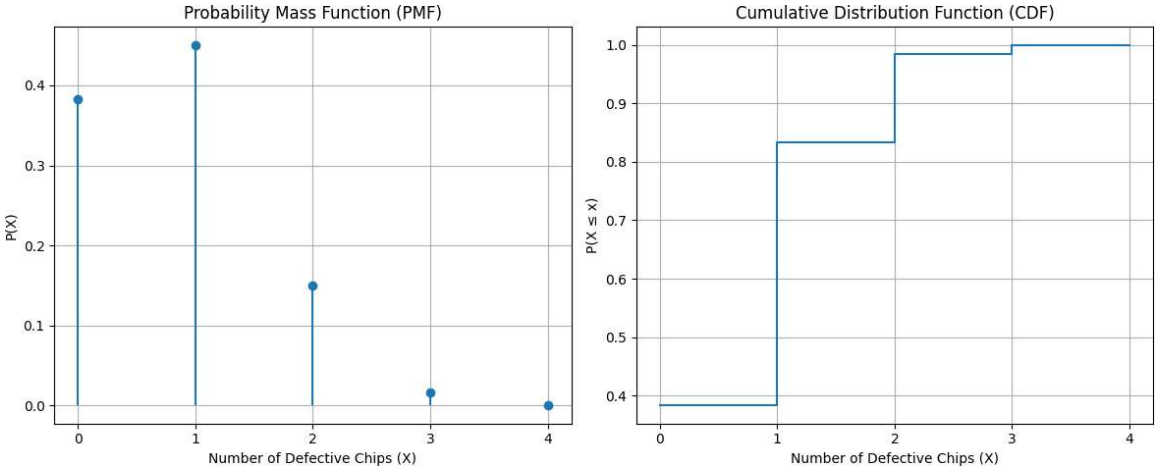
Q1. LP 2:Five defective µP-chips are accidentally mixed with twenty good ones. It is not possible to look at a chip and tell whether or not it is defective. Find the probability distribution of number of defective µP-chips, if four µP-chips are drawn at random from this lot. Graphically represent the probability function and cumulative distribution function.

In [ ]:

```
import matplotlib.pyplot as plt
from scipy.stats import hypergeom
import numpy as np

# Parameters
N = 25          # Total chips
D = 5           # Defective chips
n = 4           # Sample size

# Support: possible values of X
x = np.arange(0, 5)

# PMF and CDF
pmf = hypergeom.pmf(x, N, D, n)  #without replacement for Probablity distributio
cdf = hypergeom.cdf(x, N, D, n)

# Display values
print("PMF:")
for xi, pi in zip(x, pmf):
    print(f"P(X={xi}) = {pi:.4f}")

print("\nCDF:")
for xi, ci in zip(x, cdf):
    print(f"P(X<={xi}) = {ci:.4f}")

# Plotting
plt.figure(figsize=(12, 5))

# PMF plot
plt.subplot(1, 2, 1)
plt.stem(x, pmf, basefmt=" ") # This line was modified
plt.title("Probability Mass Function (PMF)")
plt.xlabel("Number of Defective Chips (X)")
plt.ylabel("P(X)")
plt.xticks(x)
plt.grid(True)

# CDF plot
plt.subplot(1, 2, 2)
```

```
plt.step(x, cdf,where='post')
plt.title("Cumulative Distribution Function (CDF)")
plt.xlabel("Number of Defective Chips (X)")
plt.ylabel("P(X ≤ x)")
plt.xticks(x)
plt.grid(True)

plt.tight_layout()
plt.show()
```

```
PMF:
P(X=0) = 0.3830
P(X=1) = 0.4506
P(X=2) = 0.1502
P(X=3) = 0.0158
P(X=4) = 0.0004

CDF:
P(X<=0) = 0.3830
P(X<=1) = 0.8336
P(X<=2) = 0.9838
P(X<=3) = 0.9996
P(X<=4) = 1.0000
```



Q2. A random variable X has the following probability distribution:

x : -2 -1 0 1 2

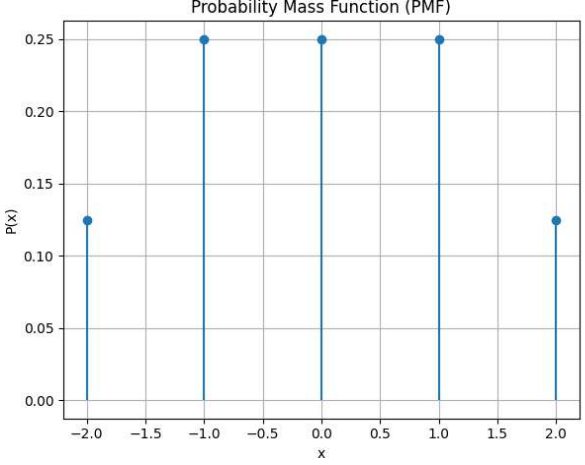
P(x): 1/8 2/8 k 2/8 1/8

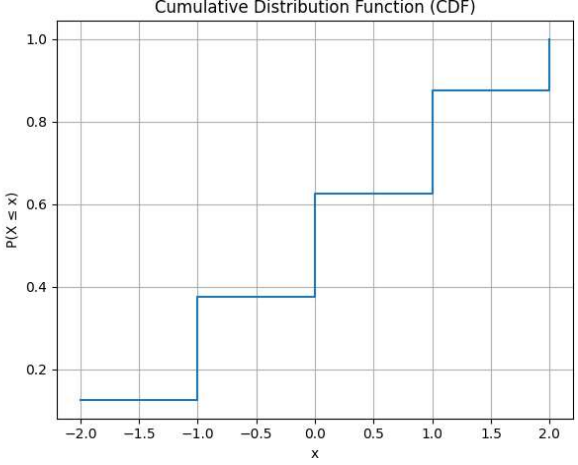
Find the (i) value of k (ii) mean and variance (iii) P(-1 < x ≤ 2) (iv) Express density function and c.d.f graphically.

In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt

# Define known values
```

```python
x_values = np.array([-2, -1, 0, 1, 2])
p_values = np.array([1/8, 2/8, None, 2/8, 1/8])  # k is unknown for x=0

# (i) Find value of k
k = 1 - sum([p for p in p_values if p is not None])
p_values[2] = k  # Set k at index corresponding to x=0

# (ii) Mean (μ = Σx * P(x)) and Variance (σ² = Σ(x - μ)² * P(x))
mean = np.sum(x_values * p_values)
variance = np.sum(((x_values - mean) ** 2) * p_values)

# (iii) P(-1 < x ≤ 2) = P(0) + P(1) + P(2)
p_range = p_values[2] + p_values[3] + p_values[4]

# (iv) Plot PMF and CDF
cdf_values = np.cumsum(p_values)

# Output
print(f"(i) Value of k: {k:.3f}")
print(f"(ii) Mean: {mean:.3f}")
print(f"     Variance: {variance:.3f}")
print(f"(iii) P(-1 < x <= 2): {p_range:.3f}")

# Plotting-
plt.figure(figsize=(12, 5))
# PMF
plt.subplot(1, 2, 1)
plt.stem(x_values, p_values, basefmt=" ")
plt.title("Probability Mass Function (PMF)")
plt.xlabel("x")
plt.ylabel("P(x)")
plt.grid(True)

# CDF
plt.subplot(1, 2, 2)
plt.step(x_values, cdf_values, where='post')
plt.title("Cumulative Distribution Function (CDF)")
plt.xlabel("x")
plt.ylabel("P(X ≤ x)")
plt.grid(True)

plt.tight_layout()
plt.show()
```

```
(i) Value of k: 0.250
(ii) Mean: 0.000
     Variance: 1.500
(iii) P(-1 < x ≤ 2): 0.625
```





Q3. Consider the random variable that represents the number of boys in a family. Out of 500 families with 5 children each, construct p.d.f and c.d.f with graph, Also find how many families would you expected to have

(i) One boy

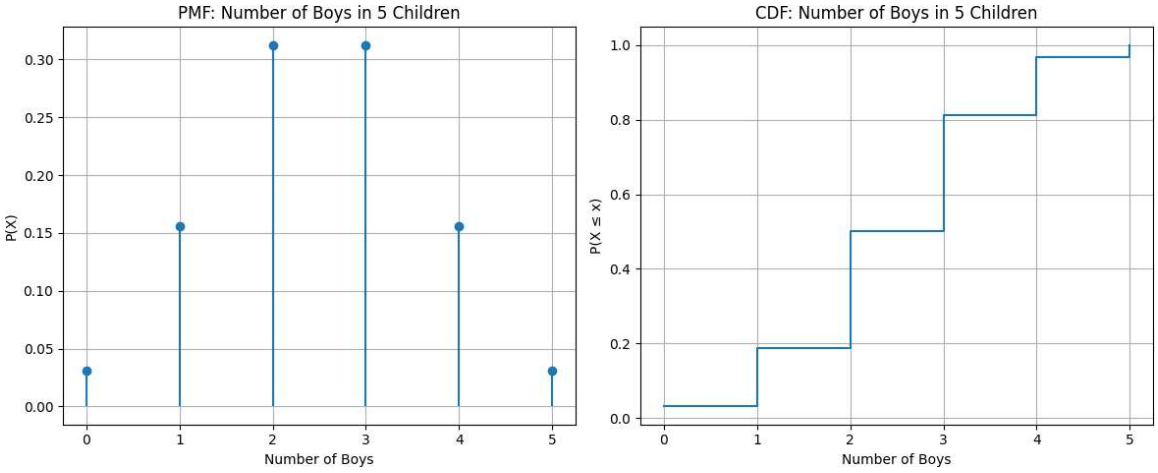
(ii) At most two girls.

In [ ]:

```python
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import binom
# Parameters
n = 5             # number of children
p = 0.5           # probability of a boy
families = 500    # total families
# Values of X: number of boys (0 to 5)
x = np.arange(0, n+1)
# PMF and CDF
pmf = binom.pmf(x, n, p)
cdf = binom.cdf(x, n, p)
# Expected families with 1 boy
expected_one_boy = families * pmf[1]
# At most 2 girls <=> at least 3 boys => X ≥ 3
expected_at_least_3_boys = families * sum(pmf[3:])
# Output
print("Probability Distribution Function (PMF):")
for xi, pi in zip(x, pmf):
    print(f"P(X = {xi}) = {pi:.4f}")

print("\n(i) exactly 1 boy:", round(expected_one_boy))
print("(ii) at most 2 girls (i.e., ≥3 boys):", round(expected_at_least_3_boys))
```

```python
# Plotting
plt.figure(figsize=(12, 5))

# PMF
plt.subplot(1, 2, 1)
plt.stem(x, pmf, basefmt=" ")
plt.title("PMF: Number of Boys in 5 Children")
plt.xlabel("Number of Boys")
plt.ylabel("P(X)")
plt.xticks(x)
plt.grid(True)

# CDF
plt.subplot(1, 2, 2)
plt.step(x, cdf, where='post')
plt.title("CDF: Number of Boys in 5 Children")
plt.xlabel("Number of Boys")
plt.ylabel("P(X ≤ x)")
plt.xticks(x)
plt.grid(True)
plt.tight_layout()
plt.show()
```

```
Probability Distribution Function (PMF):
P(X = 0) = 0.0312
P(X = 1) = 0.1562
P(X = 2) = 0.3125
P(X = 3) = 0.3125
P(X = 4) = 0.1562
P(X = 5) = 0.0312

(i) Expected number of families with exactly 1 boy: 78
(ii) Expected number of families with at most 2 girls (i.e., ≥3 boys): 250
```



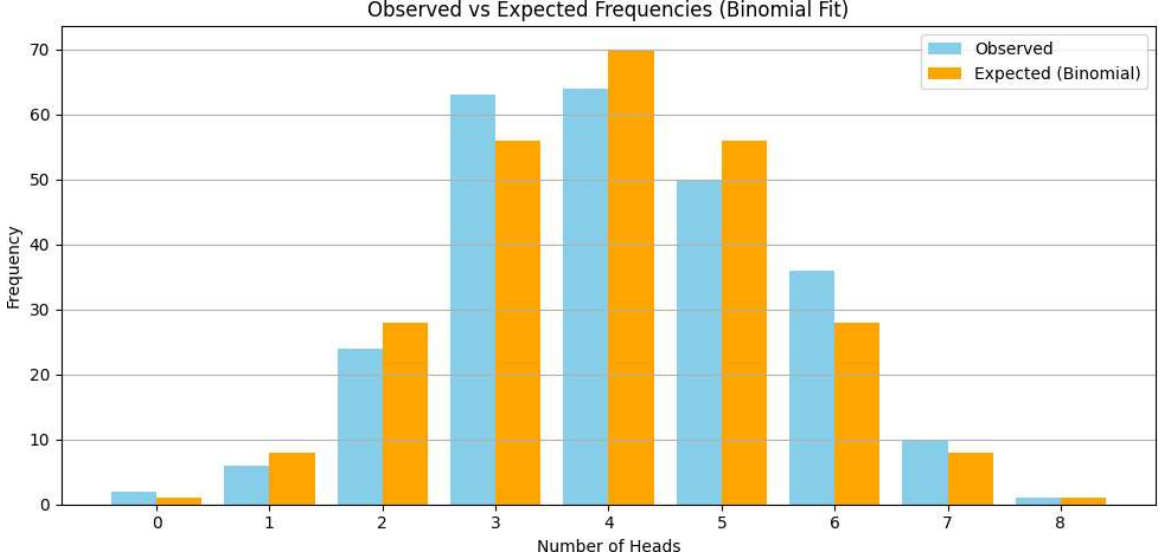
Q4.A set of 8 symmetrical coins was tossed 256 times and the frequencies of throws observed were as follows. Fit a Binomial distribution.

No. of heads 0 1 2 3 4 5 6 7 8

Frequency of throws 2 6 24 63 64 50 36 10 1

In [ ]:
```python
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import binom

# Data
x = np.arange(0, 9)  # Number of heads from 0 to 8
observed_freq = np.array([2, 6, 24, 63, 64, 50, 36, 10, 1])
total_throws = sum(observed_freq)

# Parameters for binomial distribution
n = 8      # number of coins
p = 0.5    # fair coins

# Calculate expected frequencies using binomial distribution
pmf = binom.pmf(x, n, p)
expected_freq = pmf * total_throws

# Print observed vs expected
print("x\tObserved\tExpected")
for xi, o, e in zip(x, observed_freq, expected_freq):
    print(f"{xi}\t{o}\t\t{e:.2f}")

# Plotting
plt.figure(figsize=(10, 5))
plt.bar(x - 0.2, observed_freq, width=0.4, label="Observed", color="skyblue")
plt.bar(x + 0.2, expected_freq, width=0.4, label="Expected (Binomial)", color="o
plt.xlabel("Number of Heads")
plt.ylabel("Frequency")
plt.title("Observed vs Expected Frequencies (Binomial Fit)")
plt.xticks(x)
plt.legend()
plt.grid(axis='y')
plt.tight_layout()
plt.show()
```

```
x       Observed        Expected
0       2               1.00
1       6               8.00
2       24              28.00
3       63              56.00
4       64              70.00
5       50              56.00
6       36              28.00
7       10              8.00
8       1               1.00
```



POISSON DISTRIBUTION

Q1.LP1: The number of failures of a testing instrument from contamination particles on the product is a Poisson random variable with a mean of 0.02 failure per hour.

(a) What is the probability that the instrument does not fail in an eight-hour shift?

(b) What is the probability of at least one failure in a 24-hour day?

In [ ]:

```python
from scipy.stats import poisson

# Given failure rate
lambda_1h = 0.02

# (a) No failures in 8-hour shift
lambda_8h = lambda_1h * 8
p_no_failure_8h = poisson.pmf(0, lambda_8h)

# (b) At least one failure in 24-hour day
lambda_24h = lambda_1h * 24
p_at_least_one_24h = 1 - poisson.pmf(0, lambda_24h)

# Output
```

```
print(f"(a) P(no failures in 8 hours) = {p_no_failure_8h:.4f}")
print(f"(b) P(at least one failure in 24 hours) = {p_at_least_one_24h:.4f}")
```

```
(a) P(no failures in 8 hours) = 0.8521
(b) P(at least one failure in 24 hours) = 0.3812
```

Q2. LP:9 In an automatic telephone exchange the probability that any one call is wrongly connected is 0.001. What is the minimum number of independent calls required to ensure a probability 0.9 that at least one call is wrongly connected?

Poisson Distribution: Let

$\lambda = np$ be the expected number of wrong connections.

The probability of at least one wrong connection is:

$P(X \geq 1) = 1 - P(X = 0) = 1 - e^{-\lambda}$

Set this equal to 0.9:

$1 - e^{-\lambda} \geq 0.9 \Rightarrow e^{-\lambda} \leq 0.1 \Rightarrow \lambda \geq -\ln(0.1) \approx 2.3026$ So we want:

$$n\ p \geq 2.3026 \Rightarrow n \geq 2.3026/\ 0.001$$

2302.6

So minimum

n=2303

In [ ]:
```
from scipy.stats import poisson
import numpy as np

# Desired threshold
threshold = 0.9
# Corresponding tail for P(X >= 1) >= 0.9 is P(X <= 0) <= 0.1
tail_prob = 1 - threshold

# Search for the smallest lambda such that P(X <= 0) <= 0.1
# i.e., poisson.ppf(0.1, lambda) >= 1

lam = 0
while True:
    if poisson.ppf(tail_prob, lam) >= 1:
        break
    lam += 0.001  # Small increment for precision
```

```
# Now get minimum n such that n * p >= lambda
p = 0.001
n = int(np.ceil(lam / p))

print("Minimum number of calls needed:", n)
```

```
Minimum number of calls needed: 2303
```

NORMAL DISTRIBUTION

Q1. LP:15 The average number of acres burned by forest and range fires in a large New Mexico county is 4,300 acres per year, with a standard deviation of 750 acres. The distribution of the number of acres burned is normal. What is the probability that between 2,500 and 4,200 acres will be burned in any given year?

Distribution: Normal

Mean (μ): 4300

Standard Deviation (σ): 750

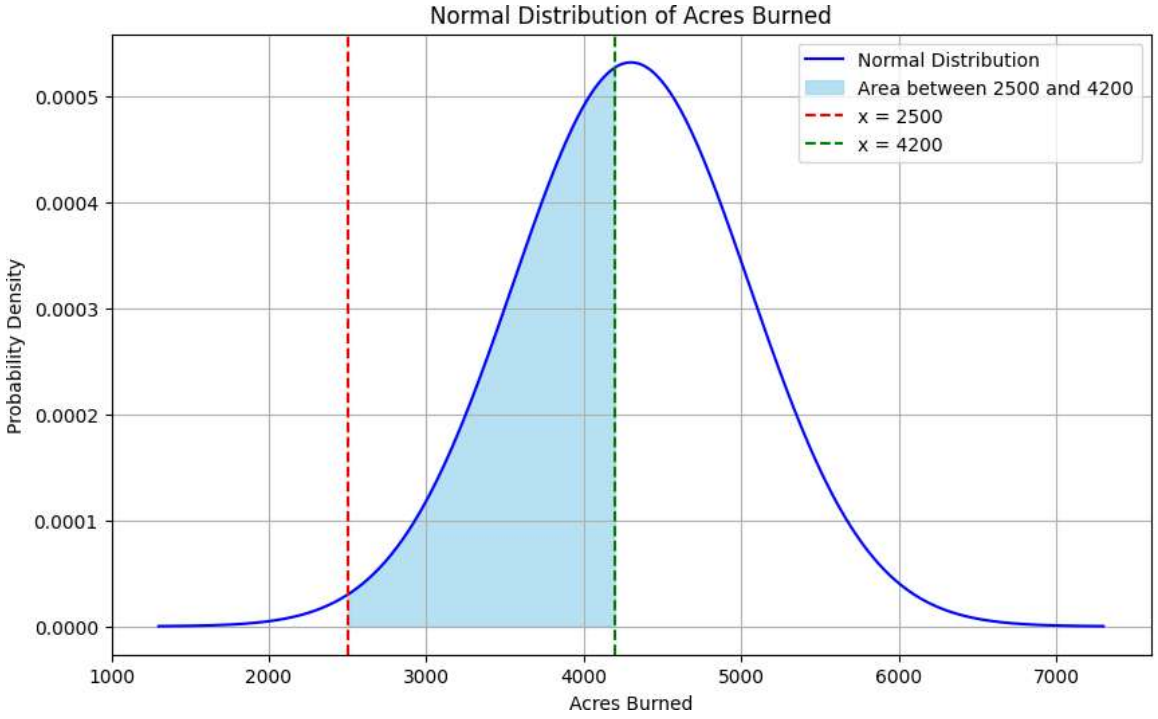
Find:

P(2500≤X≤4200)

In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import norm

# Parameters
mu = 4300
sigma = 750
x1 = 2500
x2 = 4200

# Probability calculation
p1 = norm.cdf(x1, mu, sigma)
p2 = norm.cdf(x2, mu, sigma)
probability = p2 - p1
print(f"Probability that between {x1} and {x2} acres will be burned: {probabilit

# Plot setup
x = np.linspace(mu - 4*sigma, mu + 4*sigma, 1000)
y = norm.pdf(x, mu, sigma)
```

```
# Plotting
plt.figure(figsize=(10, 6))
plt.plot(x, y, label='Normal Distribution', color='blue')
plt.fill_between(x, y, where=(x >= x1) & (x <= x2), color='skyblue', alpha=0.6,

# Annotations
plt.axvline(x1, color='red', linestyle='--', label=f'x = {x1}')
plt.axvline(x2, color='green', linestyle='--', label=f'x = {x2}')
plt.title('Normal Distribution of Acres Burned')
plt.xlabel('Acres Burned')
plt.ylabel('Probability Density')
plt.legend()
plt.grid(True)
plt.show()
```

```
Probability that between 2500 and 4200 acres will be burned: 0.4388
```



In [ ]:

```
from scipy.stats import norm

# Given parameters
mu = 4300       # mean
sigma = 750     # standard deviation

# Range
x1 = 2500
x2 = 4200

# Calculate probabilities using CDF
p1 = norm.cdf(x1, loc=mu, scale=sigma)
p2 = norm.cdf(x2, loc=mu, scale=sigma)

# Probability that X is between x1 and x2
probability = p2 - p1

print(f"Probability that between {x1} and {x2} acres will be burned: {probabilit
```

```
Probability that between 2500 and 4200 acres will be burned: 0.4388
```

Q6. LP:16 In a normal distribution, 31% of the items are under 45 and 10% are over 64. Find the mean and standard deviation of the distribution.

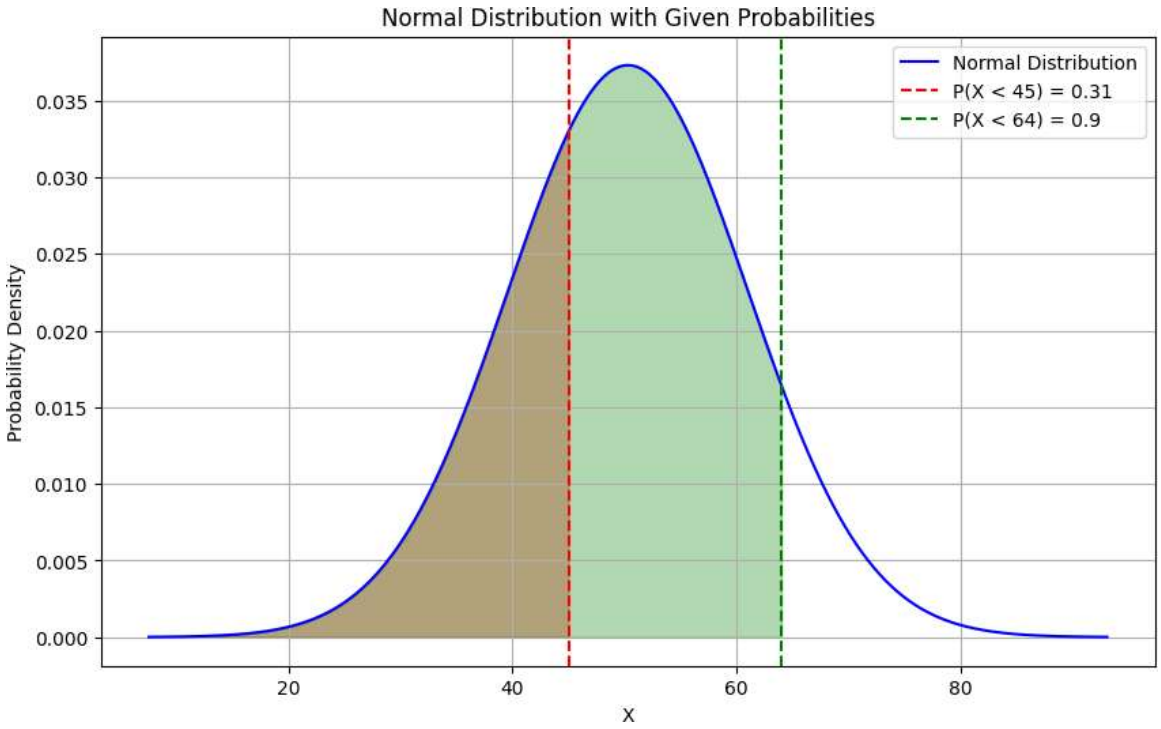
In [ ]:
```
from scipy.stats import norm

# Given probabilities
p1 = 0.31  # P(X < 45)
p2 = 0.90  # P(X < 64)



# Corresponding x values
x1 = 45
x2 = 64

# Corresponding z-scores
z1 = norm.ppf(p1) # ppf for finding Z value using prob value.
z2 = norm.ppf(p2)

# Solve the two equations:
# (x1 - μ)/σ = z1 => x1 = μ + z1 * σ
# (x2 - μ)/σ = z2 => x2 = μ + z2 * σ
# Subtract equations:
# (x2 - x1) = (z2 - z1) * σ

sigma = (x2 - x1) / (z2 - z1)
mu = x1 - z1 * sigma

# Output
print(f"Mean (μ): {mu:.2f}")
print(f"Standard Deviation (σ): {sigma:.2f}")
```

```
Mean (μ): 50.30
Standard Deviation (σ): 10.69
```

In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import norm

# Given probabilities and corresponding x-values
p1 = 0.31
p2 = 0.90
x1 = 45
x2 = 64

# Corresponding z-scores
z1 = norm.ppf(p1)
z2 = norm.ppf(p2)

# Calculate sigma and mu
sigma = (x2 - x1) / (z2 - z1)
mu = x1 - z1 * sigma

# Print results
```

```
print(f"Mean (μ): {mu:.2f}")
print(f"Standard Deviation (σ): {sigma:.2f}")

# Generate x values for plotting
x_vals = np.linspace(mu - 4*sigma, mu + 4*sigma, 1000)
y_vals = norm.pdf(x_vals, mu, sigma)

# Plot the normal distribution
plt.figure(figsize=(10, 6))
plt.plot(x_vals, y_vals, label='Normal Distribution', color='blue')
plt.axvline(x1, color='red', linestyle='--', label=f'P(X < {x1}) = {p1}')
plt.axvline(x2, color='green', linestyle='--', label=f'P(X < {x2}) = {p2}')

# Fill area under curve up to x1 and x2
x_fill1 = np.linspace(mu - 4*sigma, x1, 500)
y_fill1 = norm.pdf(x_fill1, mu, sigma)
plt.fill_between(x_fill1, y_fill1, color='red', alpha=0.3)

x_fill2 = np.linspace(mu - 4*sigma, x2, 500)
y_fill2 = norm.pdf(x_fill2, mu, sigma)
plt.fill_between(x_fill2, y_fill2, color='green', alpha=0.3)

# Plot styling
plt.title('Normal Distribution with Given Probabilities')
plt.xlabel('X')
plt.ylabel('Probability Density')
plt.legend()
plt.grid(True)
plt.show()
```

```
Mean (μ): 50.30
Standard Deviation (σ): 10.69
```



Q7. LP:14 For a certain type of fluorescent light in a large building, the cost per bulb

of replacing bulbs all at once is much less than if they are replaced individually as they burn out. It is known that the lifetime of these bulbs is normally distributed, and that 60% last longer than 2500 hours, while 30% last longer than 3000 hours. What are the approximate mean and standard deviation of the lifetimes of the bulbs?

Let the lifetime of a bulb be X~N(μ,σ 2)

You are told:

P(X>2500)=0.60 → P(X≤2500)=0.40

P(X>3000)=0.30 → P(X≤3000)=0.70

Convert these to Z-scores: Z

$X - \mu / \sigma$

So we set up two equations:

2500 − μ/ σ = z1 , where z 1

norm.ppf (0.40)

3000 − μ/ σ

z 2 , where z 2

norm.ppf (0.70)

We can solve this system of two equations for μ and σ.

In [ ]:
```
from scipy.stats import norm

# CDF values
z1 = norm.ppf(0.40)  # Corresponds to X = 2500
z2 = norm.ppf(0.70)  # Corresponds to X = 3000

# Solve the system:
```

```
# (2500 - mu) / sigma = z1
# (3000 - mu) / sigma = z2

# Subtract equations:
# ((3000 - mu) - (2500 - mu)) / sigma = z2 - z1
# (500) / sigma = z2 - z1
sigma = 500 / (z2 - z1)
mu = 2500 - z1 * sigma

# Output
print(f"Approximate mean (μ): {mu:.2f}")
print(f"Approximate standard deviation (σ): {sigma:.2f}")
```

```
Approximate mean (μ): 2662.87
Approximate standard deviation (σ): 642.88
```

In [ ]:

BINOMIAL DISTRIBUTION USING LUNGS DATASET

In [ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [ ]:
```
df = pd.read_csv('/content/lungdata.csv')
```

In [ ]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 725 entries, 0 to 724
Data columns (total 6 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   LungCap    725 non-null    float64
 1   Age        725 non-null    int64  
 2   Height     725 non-null    float64
 3   Smoke      725 non-null    object 
 4   Gender     725 non-null    object 
 5   Caesarean  725 non-null    object 
dtypes: float64(2), int64(1), object(3)
memory usage: 34.1+ KB
```

In [ ]:
```
age_count=df['Age'].value_counts()
age_count
prob_age=age_count/725
prob_age.sort_index()
```

Out[ ]:

Age	count
3	0.017931
4	0.008276
5	0.027586
6	0.034483
7	0.051034
8	0.056552
9	0.055172
10	0.070345
11	0.080000
12	0.093793
13	0.095172
14	0.077241
15	0.088276
16	0.074483
17	0.059310
18	0.059310
19	0.051034

dtype: float64

In [ ]:
```
smokers = df['Smoke'].value_counts()
prob=smokers/len(df)
num_smokers = (df['Smoke'] == 'yes').sum()
num_non_smokers = (df['Smoke'] == 'no').sum()

print(prob)

# Probability that a randomly selected individual is a smoker
#prob_smoker = smokers / total_individuals
```

```
Smoke
no     0.893793
yes    0.106207
Name: count, dtype: float64
```

Binomial Distribution Questions

In [ ]:
```
df.shape
```

Out[ ]: (725, 6)

Q1. What is the probability that at least 10 out of 25 randomly selected individuals are smokers?

✅ This means: We want to compute:

$$P\ (\ X\ \geq\ 10\ )$$

$1 - P\ (\ X\ \leq\ 9\ )$ P(X≥10)=1−P(X≤9) Where:

X~Binomial(n=25,p=smoker rate)

In [ ]:
```
import pandas as pd
from scipy.stats import binom
import matplotlib.pyplot as plt
import numpy as np

total=len(df)
# Estimate smoker probability from dataset
p_smoker = (df['Smoke'] == 'yes').sum()/total

# Parameters
n = 25
k = 9  # We want P(X ≥ 10) = 1 - P(X ≤ 9)

# Calculate probability
p_at_least_10 = 1 - binom.cdf(k, n, p_smoker)

# Output
print(f"prob (p): {p_smoker:.3f}")
print(f"P(at least 10 smokers) = {p_at_least_10:.4f}")
```

```
Estimated smoker probability (p): 0.106
P(at least 10 smokers out of 25) = 0.0001
```

Q2. If the probability of a person being a smoker in this dataset is 0.2, what is the expected number of smokers in a sample of 50 individuals?

If

p=0.2 and

n=50, what is the expected number of smokers?

✅ Formula: For a binomial distribution:

Expected value (mean)= E[X] =n×p

In [ ]:
```python
import pandas as pd
from scipy.stats import binom
import matplotlib.pyplot as plt
import numpy as np

total=len(df)
# Estimate smoker probability from dataset
p_smoker = (df['Smoke'] == 'yes').sum()/total
print(p_smoker)

# Take a random sample of 50 individuals
sample = df.sample(n=50, random_state=42)

# Count number of smokers in sample (assuming 'yes' means smoker)
actual_smokers = sample['Smoke'].str.lower().eq('yes').sum()/total

# Calculate expected number of smokers (based on probability 0.2)
expected_smokers = 0.2 * 50

# Output results
print(f"Expected number of smokers: {expected_smokers}")
print(f"Actual number of smokers in the sample: {actual_smokers}")
```

```
0.10620689655172413
Expected number of smokers: 10.0
Actual number of smokers in the sample: 0.005517241379310344
```

In [ ]:
```python
import pandas as pd

total=len(df)

# Convert 'Smoke' column to binary: 1 if smoker, 0 otherwise
df['IsSmoker'] = df['Smoke'].apply(lambda x: 1 if x.lower() == 'yes' else 0)

# Calculate the probability of being a smoker
smoker_probability = df['IsSmoker'].sum()/total
print(f"Probability of being a smoker: {smoker_probability:.2f}")

# Given a sample of 50 individuals
sample_size = 50
expected_smokers = smoker_probability * sample_size
print(f"Expected number of smokers in a sample of 50: {expected_smokers:.1f}")
```

```
Probability of being a smoker: 0.11
Expected number of smokers in a sample of 50: 5.3
```

Q3. What is the probability that exactly 20 out of 40 randomly selected people are female?

A fixed number of trials (n = 40)

Two outcomes (female or not)

A constant probability of success (p = proportion of females in the dataset)

In [ ]:
```
import pandas as pd
from scipy.stats import binom
import matplotlib.pyplot as plt
import numpy as np


total=len(df)
# Estimate the probability of selecting a female
p_female = (df['Gender'] == 'female').sum()/total

# Parameters
n = 40
k = 20  # We want exactly 20 females

# Calculate binomial probability
prob_20_females = binom.pmf(k, n, p_female)

# Output
print(f"Estimated female probability (p): {p_female:.3f}")
print(f"Probability of exactly 20 females out of 40: {prob_20_females:.4f}")
```

```
Estimated female probability (p): 0.494
Probability of exactly 20 females out of 40: 0.1250
```

Q4. What is the probability that fewer than 5 out of 15 randomly selected people are male?

This means:

P(X<5)=P(X≤4) Where

X~Binomial(n=15,p=proportion of males)

In [ ]:
```
from scipy.stats import binom


total=len(df)
# Estimate male probability from dataset
p_male = (df['Gender'] == 'male').sum()/total

# Binomial parameters
n = 15
k = 4  # We want P(X < 5) = P(X <= 4)

# Compute cumulative probability
prob_less_than_5_males = binom.cdf(k, n, p_male)

# Output
print(f"Estimated male probability (p): {p_male:.3f}")
print(f"Probability of fewer than 5 males out of 15: {prob_less_than_5_males:.4f
```

```
Estimated male probability (p): 0.506
Probability of fewer than 5 males out of 15: 0.0538
```

Q5. What is the probability that more than 10 out of 60 individuals were born via Caesarean?

This means:

$P(X > 10) = 1 - P(X \leq 10)$

Where:

$X \sim \text{Binomial}(n=60, p=\text{probability of Caesarean birth})$

In [ ]:
```
from scipy.stats import binom


total=len(df)
# Estimate the probability of Caesarean birth from the dataset
p_csection = (df['Caesarean'] == 'yes').sum()/total

# Parameters
n = 60  # sample size
k = 10  # threshold

# Calculate probability that more than 10 were born via Caesarean
prob_more_than_10 = 1 - binom.cdf(k, n, p_csection)
print(f"Estimated Caesarean birth probability (p): {p_csection:.3f}")
print(f"Probability of more than 10 Caesarean births out of 60: {prob_more_than_
```

```
Estimated Caesarean birth probability (p): 0.226
Probability of more than 10 Caesarean births out of 60: 0.8279
```

Poisson Distribution

In [ ]:
```
df.head(725)
```

Out[ ]:

	LungCap	Age	Height	Smoke	Gender	Caesarean
0	6.475	6	62.1	no	male	no
1	10.125	18	74.7	yes	female	no
2	9.550	16	69.7	no	female	yes
3	11.125	14	71.0	no	male	no
4	4.800	5	56.9	no	male	no
...	...	...	...	...	...	...
720	5.725	9	56.0	no	female	no
721	9.050	18	72.0	yes	male	yes
722	3.850	11	60.5	yes	female	no
723	9.825	15	64.9	no	female	no
724	7.100	10	67.7	no	male	no

725 rows × 6 columns

Q1. what is the probability that in a random group of 50 individuals, no one smokes?

In [ ]:
```
import pandas as pd
from scipy.stats import poisson
import numpy as np

n_total = len(df)
n_sample = 50

n_smokers = (df["Smoke"] == "yes").sum()
p_smoker = n_smokers / n_total

lambda_sample = n_sample * p_smoker

p_no_smokers = poisson.pmf(0, mu=lambda_sample)


print(f"Number of smokers: {n_smokers}")
print(f"Expected number of smokers (λ) in a sample of 50: {lambda_sample:.4f}")
print(f"Probability that no one smokes in a group of 50: {p_no_smokers:.4f}")
```

```
Number of smokers: 77
Expected number of smokers (λ) in a sample of 50: 5.3103
Probability that no one smokes in a group of 50: 0.0049
```

Q2. What is the probability that more than 15 individuals smoke in a group of 50?

```
n_total = len(df)
n_sample = 50

n_smokers = (df["Smoke"] == "yes").sum()

p_smoker = n_smokers / n_total

lambda_sample = n_sample * p_smoker

p_more_than_15 = 1 - poisson.cdf(15, mu=lambda_sample)

print(f"P(smoker): {p_s ex moker:.4f}")
print(f"Expected λ (for 50 people): {lambda_sample:.4f}")
print(f"Probability that more than 15 individuals smoke: {p_more_than_15:.5f}")
```

NORMAL DISTRIBUTION

Q1. What is the probability that a randomly selected individual has a Lung Capacity greater than 10 liters?

In [ ]:
```
import pandas as pd
from scipy.stats import norm

lung_cap = df["LungCap"]

mean_lungcap = lung_cap.mean()
std_lungcap = lung_cap.std()

x = 10

# P(X > 10) = 1 - CDF(10)
prob_greater_than_10 = 1 - norm.cdf(x, loc=mean_lungcap, scale=std_lungcap)

# Output
print(f"Mean Lung Capacity: {mean_lungcap:.4f}")
print(f"Standard Deviation: {std_lungcap:.4f}")
print(f"Probability that LungCap > 10 liters: {prob_greater_than_10:.6f}")
```

```
Mean Lung Capacity: 7.8631
Standard Deviation: 2.6620
Probability that LungCap > 10 liters: 0.211068
```

Q2. What percentage of individuals are expected to have a Height between 60 and 70 inches?

In [ ]:
```
import pandas as pd
from scipy.stats import norm


height = df["Height"]

mean_height = height.mean()
std_height = height.std()

# range
lower = 60
upper= 70

# Compute probabilities
p_between = norm.cdf(upper, loc=mean_height, scale=std_height) - \
            norm.cdf(lower, loc=mean_height, scale=std_height)

# Convert to percentage
percentage_between = p_between * 100

# Output
print(f"Mean Height: {mean_height:.2f}")
print(f"Standard Deviation: {std_height:.2f}")
print(f"Percentage of individuals with height between 60 and 70 inches: {percent
```

```
Mean Height: 64.84
Standard Deviation: 7.20
Percentage of individuals with height between 60 and 70 inches: 51.24%
```

In [ ]:

What is the 90th percentile of Height?

In [ ]:
```
# 90th percentile value
height_90 = norm.ppf(0.90, loc=df['Height'].mean(), scale=df['Height'].std())
print(f"90th percentile of Height: {height_90:.2f} inches")
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-1-dded7646c8a4> in <cell line: 0>()
      1 # 90th percentile value
----> 2 height_90 = norm.ppf(0.90, loc=df['Height'].mean(), scale=df['Height'].st
d())
      3 print(f"90th percentile of Height: {height_90:.2f} inches")

NameError: name 'norm' is not defined
```