Import Code:

```
import numpy as np
from matplotlib import pyplot as plt
from scipy.stats import binom, geom, poisson, nbinom
```

Question 1: Binomial Distribution

Problem Setup:

- Probability of success (passing inspection): $p = \frac{4}{5} = 0.8$
- Number of trials (products tested): $N = 8$
- X = number of products that pass inspection
- Distribution: $X \sim \text{Binomial}(8, 0.8)$

Binomial PMF:

$$P_X(x) = \binom{n}{x} p^x (1-p)^{n-x} = \binom{8}{x} (0.8)^x (0.2)^{8-x}$$

Part (a): More than 5 products pass inspection

Setup: Find $P(X > 5) = P(X = 6) + P(X = 7) + P(X = 8)$
Step-by-step calculations:

$$P_X(6) = \binom{8}{6} (0.8)^6 (0.2)^2 = 28 \times 0.262144 \times 0.04 = 0.2936$$

$$P_X(7) = \binom{8}{7} (0.8)^7 (0.2)^1 = 8 \times 0.2097152 \times 0.2 = 0.3355$$

$$P_X(8) = \binom{8}{8} (0.8)^8 (0.2)^0 = 1 \times 0.16777216 \times 1 = 0.1678$$

Final answer:

$$P(X > 5) = 0.2936 + 0.3355 + 0.1678 = \boxed{0.7969}$$

There is a **79.69%** probability that more than 5 products will pass inspection.

Part (b): Factory fails more than once

Setup:

- “Fails more than once" (at least twice) means less than 7 products out of the 8 pass ($P(X < 7)$)
- Find $P(X < 7) = 1 - P(X \geq 7)$
- $P(X \geq 7) = P(X = 7) + P(X = 8)$

Calculation using complement:

$$\begin{aligned} P(X < 7) &= 1 - P(X \geq 7) \\ &= 1 - [P_X(7) + P_X(8)] \\ &= 1 - [0.3355 + 0.1678] \\ &= 1 - 0.5033 \\ &= \boxed{0.4967} \end{aligned}$$

Final answer: There is a **49.67%** probability that the factory will fail more than once.

Question 2: Geometric Distribution

Problem Setup:

- Probability of difficult quiz: $p = 0.35$
- X = number of quizzes taken until first difficult one
- $$P_X(x) = \begin{cases} p(1-p)^{x-1} & \text{where } x = 1, 2, ... \\ 0 & \text{otherwise} \end{cases}$$

Part (a): Plot PMF and CDF

```
# Question 2a - Original platform
p1 = 0.35  # Probability of difficult quiz

# Generate x values for plotting (adaptive range based on distribution)
x = np.arange(geom.ppf(0.01, p1)-2, geom.ppf(0.99, p1)+10).astype(int)

# Calculate PMF and CDF
pmf1 = geom.pmf(x, p1)
cdf1 = geom.cdf(x, p1)

# Plot PMF and CDF for original platform
plotStats(x, pmf1, cdf1, f'PMF (p={p1})', f'CDF (p={p1})', 'Number of quizzes (x)')
print(pd.DataFrame(np.array([x, pmf1]).T, columns=['x', 'PMF']))
```

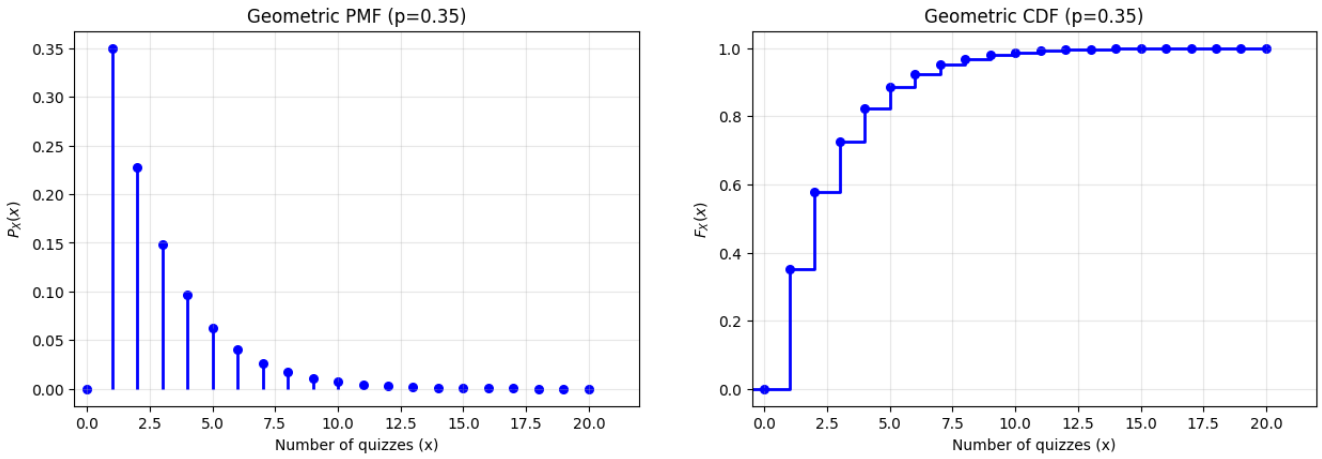

Figure 1: PMF and CDF for question 2 ($p = 0.35$)

Part (b): Probability of more than 4 quizzes

Hand calculation:

$$\begin{aligned} P(X > 4) &= 1 - P(X \leq 4) \\ &= 1 - F_X(4) \\ &= 1 - [1 - (1 - 0.35)^4] \\ &= (0.65)^4 \\ &= 0.1785 \end{aligned}$$

Reading this from the plot we can see that $F_X(4) \approx 0.82$, so

$$P(X > 4) \approx 1 - 0.82 = 0.18$$

Part (c): New platform with p = 0.15

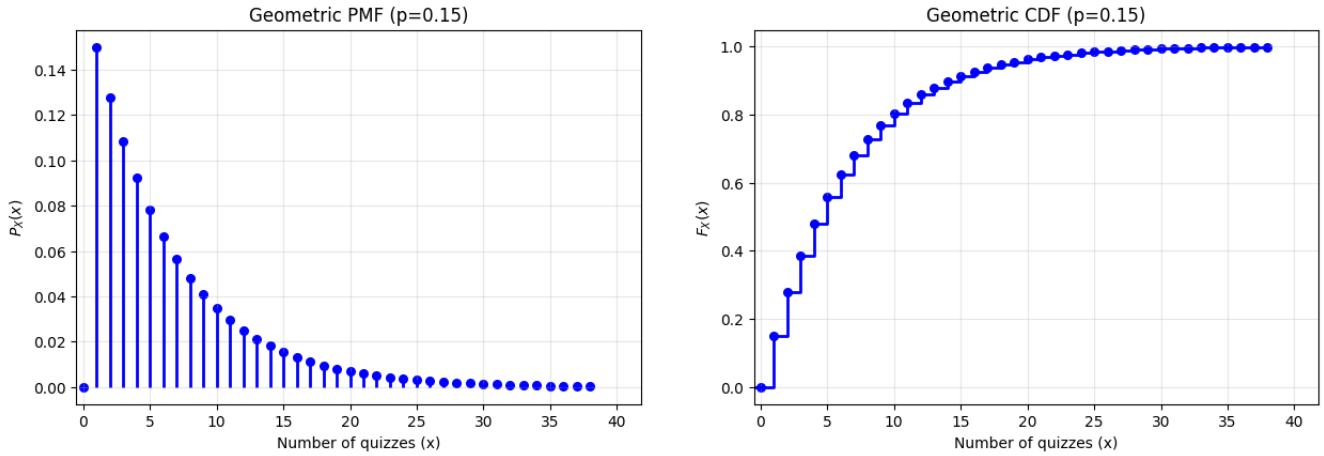

Figure 2: PMF and CDF for question 2 ($p = 0.15$)

Part (d): Probability of more than 4 quizzes (new platform)

Hand calculation:

$$\begin{aligned} P(X > 4) &= (1 - 0.15)^4 \\ &= (0.85)^4 \\ &= 0.5220 \end{aligned}$$

We can look at the CDF to verify that the probability of doing 4 or less quizzes is just below 0.5.

Question 3: Binomial Distribution

Problem Setup:

- Success probability: p (email delivered successfully)
- Number of transmissions: n
- K = number of successful deliveries

Part (a): PMF of K

Each email transmission is an independent Bernoulli trial with success probability p. The number of times K that the client receives the email is the number of successes in n transmissions. So we know that K has a binomial distribution with PMF

$$P_K(k) = \begin{cases} \binom{n}{k} p^k (1-p)^{n-k} & \text{where } k = 0, 1, ..., n \\ 0 & \text{otherwise} \end{cases}$$

Part (b): Plot for p = 0.6, n = 8

```
# Question 3b - Email transmission analysis
p = 0.6  # Probability of successful email delivery
n = 8    # Number of transmissions

# Generate k values for plotting (adaptive range based on distribution)
```

```
k = np.arange(binom.ppf(0.01, n, p)-2, binom.ppf(0.99, n, p)+10).astype(int)

# Calculate PMF and CDF
pmf = binom.pmf(k, n, p)
cdf = binom.cdf(k, n, p)

plotStats(k, pmf, cdf, f'PMF (p={p}, n={n})', f'CDF (p={p}, n={n})', 'Number of successful
    deliveries (k)')
print(pd.DataFrame(np.array([k, cdf]).T, columns=['k', 'CDF']))
```

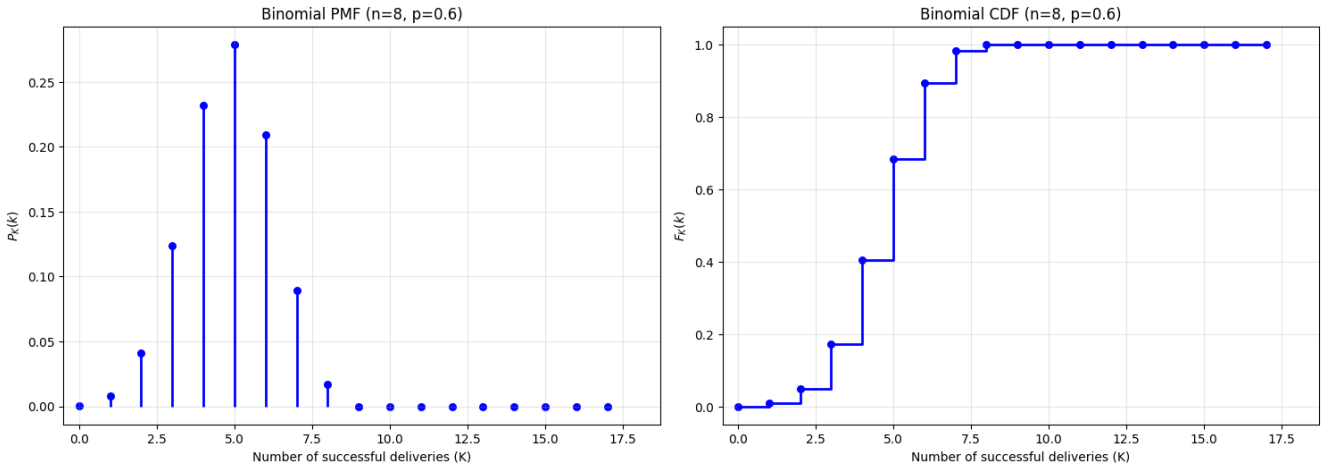

Figure 3: PMF and CDF for question 3b ($p = 0.6$, $n = 8$)

Part (c): Number of transmissions for 98% success

Setup: We want $P(K \geq 1) \geq 0.98$, which means $P(K = 0) \leq 0.02$.
Hand calculation:

$$\begin{aligned} P(K = 0) &= (1 - p)^n \leq 0.02 \\ (1 - 0.6)^n &\leq 0.02 \\ (0.4)^n &\leq 0.02 \\ n &\geq \log_{0.4}(0.02) \\ n &\geq \frac{\log(0.02)}{\log(0.4)} \\ n &\geq 4.26941 \end{aligned}$$

Note: The inequality flips because the base is smaller than 1 (but greater than 0)
Therefore, $n = 5$ transmissions are needed.
Verification with Python:

```
# Question 3c - Email transmission verification
p1 = 0.6  # Probability of successful email delivery
n = 5

# Generate x values for plotting (adaptive range based on distribution)
x = np.arange(binom.ppf(0.01, n, p1)-2, binom.ppf(0.99, n, p1)+10).astype(int)

# Calculate PMF and CDF
pmf1 = binom.pmf(x, n, p1)
cdf1 = binom.cdf(x, n, p1)

plotStats(x, pmf1, cdf1, f'PMF (p={p}, n={n})', f'CDF (p={p}, n={n})', 'Number of successful
    deliveries (k)')
```

```python
# Print F_K(4) for verification
f_k_4 = binom.cdf(4, n, p1)
print(f"F_K(4) = {f_k_4:.4f}")

# Print F_K(5) for verification
f_k_5 = binom.cdf(5, n, p1)
print(f"F_K(5) = {f_k_5:.4f}")
```

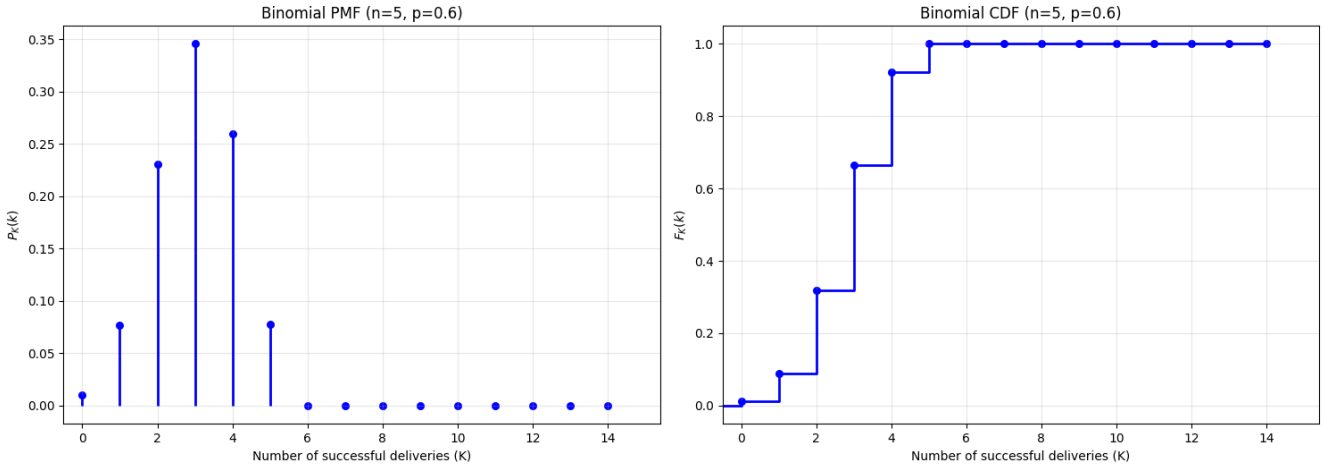

Figure 4: PMF and CDF for question 3b ($p = 0.6$, $n = 5$)

Question 4: Poisson Distribution

Problem Setup:

- Rate: $\lambda = \frac{1}{6}$ customers per minute
- For time interval T: $\alpha = \frac{T}{6}$
- C = number of customers in interval T

Part (a): Exactly 2 customers in 3 minutes

Setup: $T = 3$, so $\alpha = \frac{3}{6} = \frac{1}{2}$

$$P_C(c) = \begin{cases} \frac{(\alpha)^c e^{-\alpha}}{c!} & \text{where } c = 0, 1, ... \\ 0 & \text{otherwise} \end{cases}$$

Hand calculation:

$$\begin{aligned} P_C(2) &= \frac{\alpha^2 e^{-\alpha}}{2!} \\ &= \frac{(\frac{1}{2})^2 e^{-\frac{1}{2}}}{2} \\ &= \frac{\frac{1}{4} \times 0.6065}{2} \\ &= 0.0758 \end{aligned}$$

Python verification:

```
rate = 1/6
T = 3
alpha = rate * T

# Generate x values for plotting (adaptive range based on distribution)
x = np.arange(poisson.ppf(0.01, alpha)-2, poisson.ppf(0.99, alpha)+10).astype(int)

# Calculate PMF and CDF
pmf1 = poisson.pmf(x, alpha)
cdf1 = poisson.cdf(x, alpha)

plotStats(x, pmf1, cdf1, f'PMF (T={T}, alpha={alpha})', f'CDF (T={T}, alpha={alpha})', 'Number of
    customers (c)')
print(pd.DataFrame(np.array([x, pmf1]).T, columns=['c', 'PMF']))
print()
print(pd.DataFrame(np.array([x, cdf1]).T, columns=['c', 'CDF']))
# Find P(C=2)
prob_2 = poisson.pmf(2, alpha)
print(f"P(C=2): {prob_2:.4f}")
```

c	PMF
-2.0	0.000000
-1.0	0.000000
0.0	0.606531
1.0	0.303265
2.0	0.075816
3.0	0.012636
4.0	0.001580
5.0	0.000158
6.0	0.000013

Table 1: Table of c values and their corresponding PMF

Reading the probability off of the Python output for $c = 2$ yields 0.0758, which is the same as our calculated value.

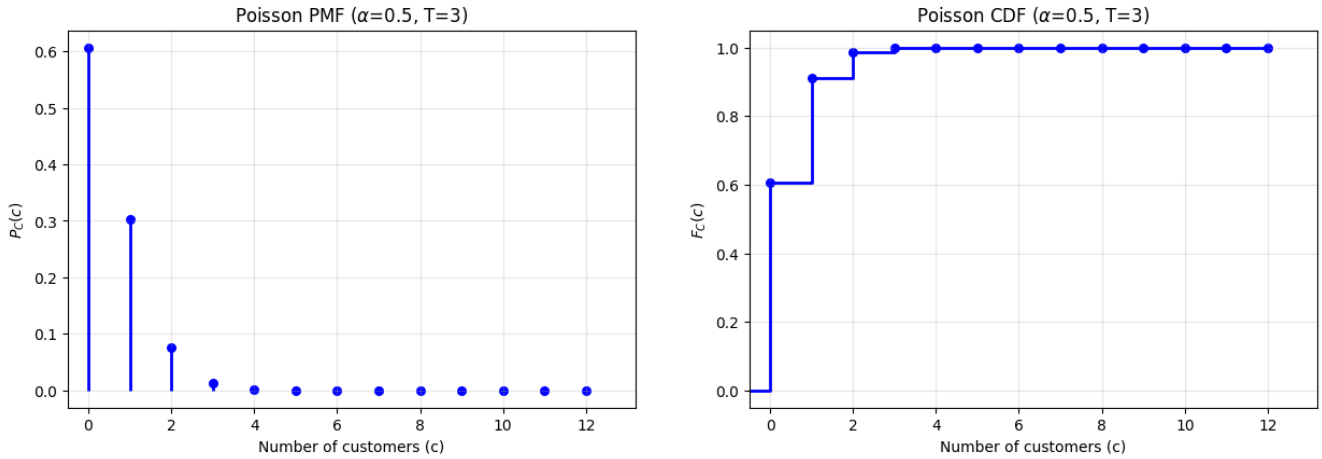

Figure 5: PMF and CDF for question 4a ($\lambda = \frac{1}{6}$, $T = 3$)

Part (b): PMF for 9-minute interval

Setup: $T = 9$, so $\alpha = \frac{9}{6} = \frac{3}{2}$
Reading the probability off of the Python output for $c = 3$ yields 0.1255, which is the same as our calculated value.

Hand calculation verification:

$$\begin{aligned} P_C(3) &= \frac{\alpha^3 e^{-\alpha}}{3!} \\ &= \frac{(\frac{3}{2})^3 e^{-\frac{3}{2}}}{6} \\ &= 0.1255 \end{aligned}$$

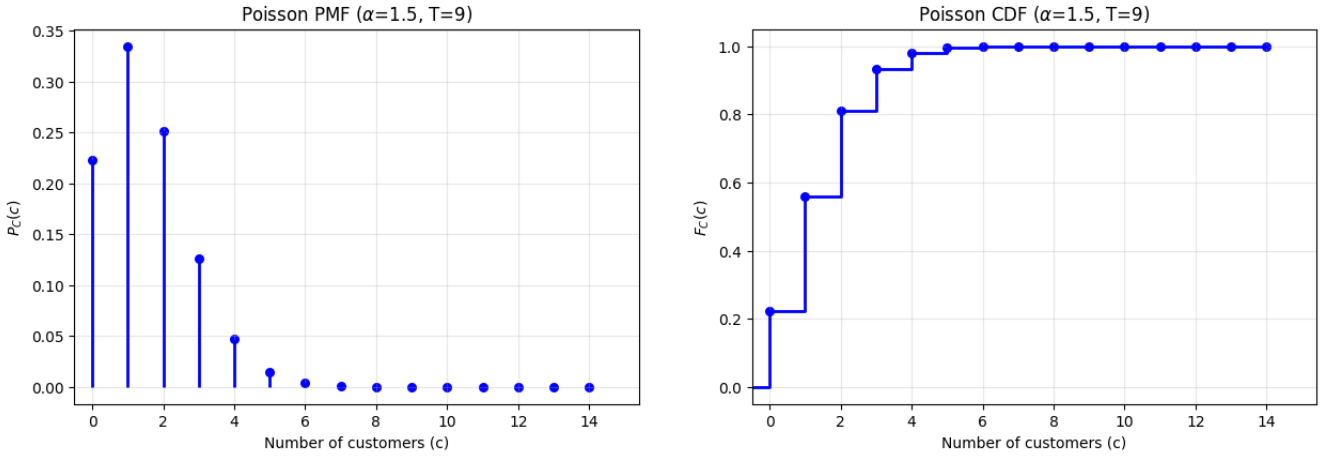

Figure 6: PMF and CDF for question 4a ($\lambda = \frac{1}{6}$, $T = 9$)

Reading the probability off of the Python output for $c = 3$ yields 0.1255, which is the same as our calculated value.

Part (c): Time for P(C ≥ 1) = 0.95

Setup: We want $P(C \geq 1) = 1 - P(C < 1) = 1 - P(C = 0) = 0.95$
Hand calculation:
We want $P(C \geq 1) = 0.95$, which means $P(C = 0) = 0.05$

$$\begin{aligned} P(C = 0) &\leq 0.05 \\ e^{-\alpha} &\leq 0.05 \\ -\alpha &\leq \ln(0.05) \\ -\alpha &\leq \ln(\frac{1}{20}) \\ -\alpha &\leq -\ln(20) \\ \alpha &\geq \ln(20) \\ \alpha &\geq 2.996 \end{aligned}$$

Since $\alpha = \frac{T}{6}$:

$$\begin{aligned} \frac{T}{6} &= 2.996 \\ T &= 6 \times 2.996 = 17.98 \\ T &\approx 18 \text{ minutes} \end{aligned}$$

c	PMF
-2.0	0.000000
-1.0	0.000000
0.0	0.049787
1.0	0.149361
2.0	0.224042
3.0	0.224042
4.0	0.168031
5.0	0.100819
6.0	0.050409
7.0	0.021604
8.0	0.008102
9.0	0.002701
10.0	0.000810
11.0	0.000221

Table 2: Table of c values and their corresponding PMF

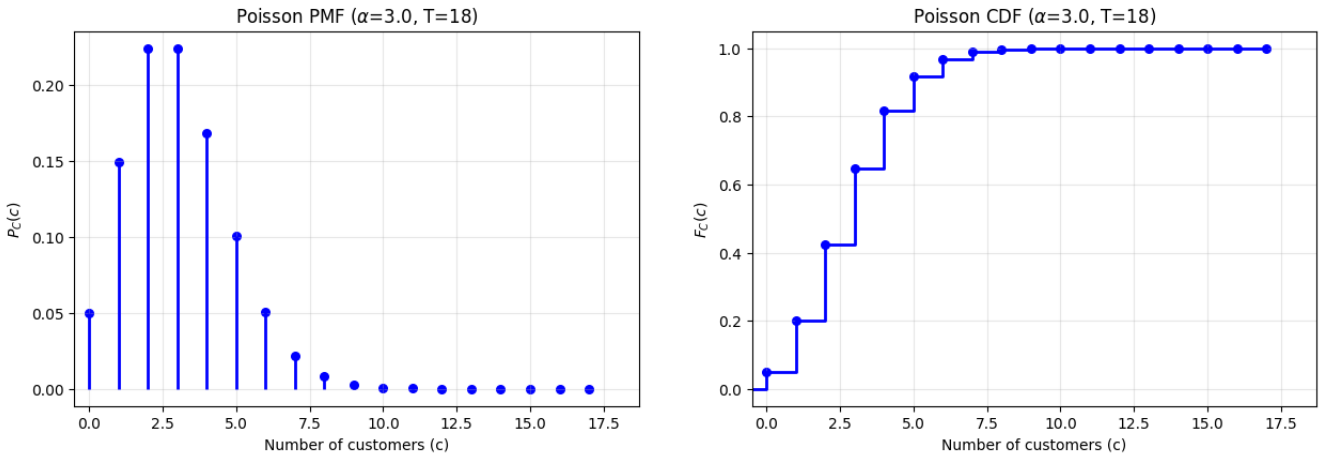

Figure 7: PMF and CDF for question 4a ($\lambda = \frac{1}{6}$, $T = 18$)

Reading the PMF printed out in Python, we can see that the probability of no customers arriving is 0.049787, which is close to our answer.

Question 5: Pascal (Negative Binomial) Distribution

Problem Setup:

- Success probability: $p = 0.12$
- X = number of projects before nth breakthrough
- PMF: $P_X(x) = \binom{x-1}{n-1} p^n (1-p)^{x-n}$ for $x = n, n+1, ...$

Part (a): PMF of X

The random variable X follows a Pascal (Negative Binomial) distribution:

$$P_X(x) = \binom{x-1}{n-1} (0.12)^n (0.88)^{x-n} \quad \text{for } x = n, n+1, n+2, ...$$

where x is the total number of projects needed to achieve n breakthroughs.

Part (b): First breakthrough on 5th project

Setup: $n = 1$, $x = 5$
This is equivalent to a geometric distribution since we want the first success.
Hand calculation:

$$\begin{aligned} P_X(5) &= \binom{5-1}{1-1}(0.12)^1(0.88)^{5-1} \\ &= \binom{4}{0}(0.12)(0.88)^4 \\ &= 1 \times 0.12 \times 0.5997 \\ &= 0.0720 \end{aligned}$$

Alternatively, using geometric distribution formula:

$$P_X(5) = (0.12)(0.88)^{5-1} = 0.12 \times 0.5997 = 0.0720$$

Python verification:

```
# Question 5b
n = 1
p = 0.12

# Generate x values for plotting (adaptive range based on distribution)
x = np.arange(nbinom.ppf(0.01, n, p)-2, nbinom.ppf(0.99, n, p)+10).astype(int)

# Calculate PMF and CDF
pmf_y = nbinom.pmf(x, n, p)
cdf_y = nbinom.cdf(x, n, p)

# Plot PMF and CDF
plotStats(x+n, pmf_y, cdf_y, f'PMF (n={n}, p={p})', f'CDF (n={n}, p={p})', 'Number of projects (x)
    ')

plt.show()
print(pd.DataFrame(np.array([x+n, pmf_y]).T, columns=['x', 'PMF']))
print()
print(pd.DataFrame(np.array([x+n, cdf_y]).T, columns=['x', 'CDF']))

# Print specific value
prob_5 = nbinom.pmf(5-n, n, p)  # x=5 means 4 failures before first success
print(f"P(X=5): {prob_5:.4f}")
print()
```

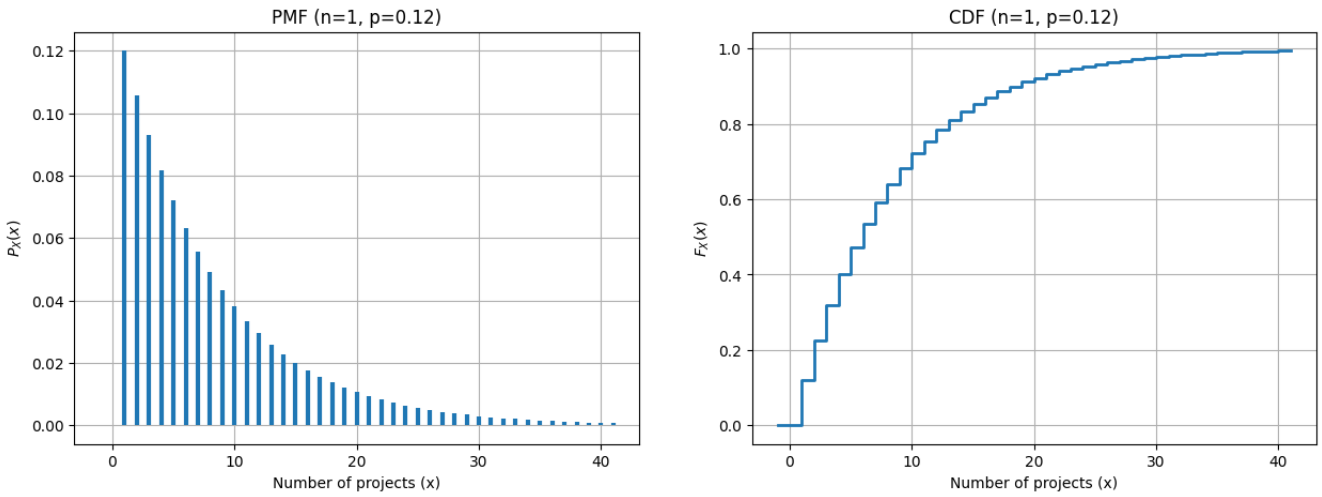

Figure 8: PMF and CDF for question 5b ($n = 1$, $p = 0.12$)

x	PMF
1	0.1200
2	0.1056
3	0.0929
4	0.0818
5	0.0720
6	0.0633
7	0.0557
8	0.0490
9	0.0432
10	0.0380

Table 3: Table of x values and their corresponding PMF

From the Python output, $P(X = 5) = 0.0720$, confirming our hand calculation, that the probability that they get their first breakthrough on the fifth project is 0.0720.

Part (c): 3 breakthroughs with 15 projects budget

Setup: $n = 3$, want $P(X \leq 15)$
Hand calculation:

$$\begin{aligned} F_X(15) &= P(X \leq 15) \\ &= \sum_{x=3}^{15} \binom{x-1}{3-1} (0.12)^3 (0.88)^{x-3} \\ &= \sum_{x=3}^{15} \binom{x-1}{2} (0.12)^3 (0.88)^{x-3} \\ &= 0.2654 \end{aligned}$$

Expected value:

$$E[X] = \frac{n}{p} = \frac{3}{0.12} = 25 \text{ projects}$$

Analysis: With an expected value of 25 projects but only a budget of 15 projects, the probability of success is quite low at 26.54%. The company would need significantly more funding to have a reasonable chance of achieving 3 breakthroughs.
Python code:

```
# Question 5c - 3 breakthroughs with 15 projects budget
n = 3
p = 0.12
budget = 15

# Generate x values for plotting (adaptive range based on distribution)
x = np.arange(nbinom.ppf(0.01, n, p)-2, nbinom.ppf(0.99, n, p)+10).astype(int)

# Calculate PMF and CDF
pmf_y = nbinom.pmf(x, n, p)
cdf_y = nbinom.cdf(x, n, p)

# Plot PMF and CDF
plotStats_w_values(x+n, pmf_y, cdf_y, budget, nbinom.cdf(budget-n, n, p), f'PMF (n={n}, p={p})', f
    'CDF (n={n}, p={p})', 'Number of projects (x)')

plt.show()
print(pd.DataFrame(np.array([x+n, pmf_y]).T, columns=['x', 'PMF']))
print()
```

```
print(pd.DataFrame(np.array([x+n, cdf_y]).T, columns=['x', 'CDF']))

# Calculate P(X <= 15)
prob_15 = nbinom.cdf(budget-n, n, p)  # budget-n = 15-3 = 12 failures
print(f"P(X <= {budget}): {prob_15:.4f}\n")

dist = nbinom(n, p, loc=n)
expected_scipy = dist.mean()
print(f"Expected value E[X] = {expected_scipy:.2f} projects")
```

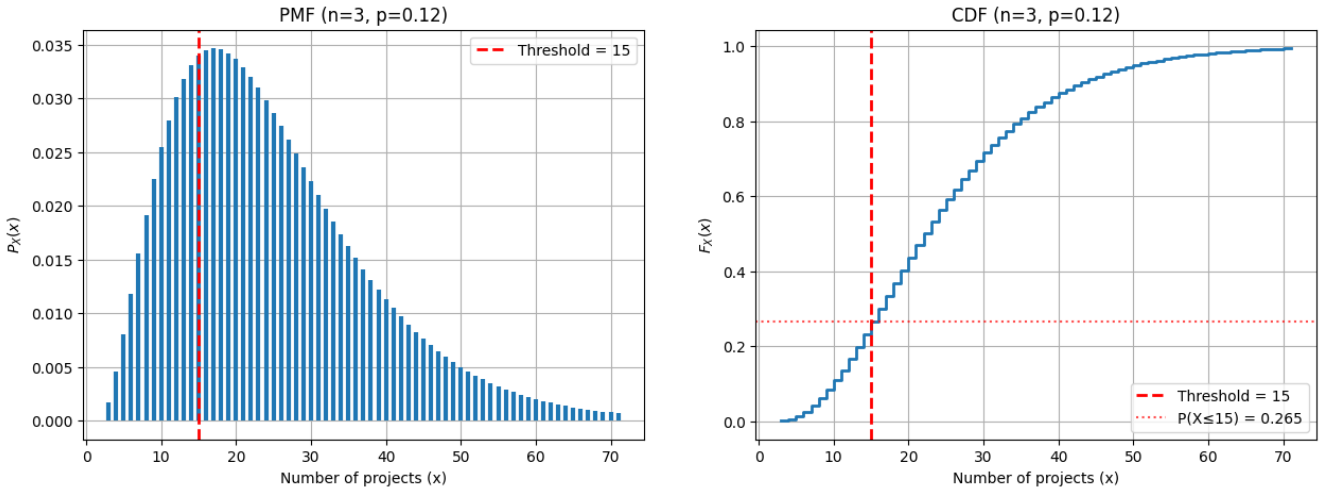

Figure 9: PMF and CDF for question 5c ($n = 3$, $p = 0.12$)

The expected value of X is over 15, which tells us it is unlikely that they will make 3 breakthroughs on their 15th project. To print the mean in Python, use

```
from scipy.stats import nbinom
dist = nbinom(n, p, loc=n)  # loc=n shifts distribution to start at n
expected_scipy = dist.mean()
print(f"Expected value E[X] = {expected_scipy:.2f} projects")
```

This will print out 25.

Part (d): Alternative research area (p = 0.22, budget = 12)

Setup: $n = 3$, $p = 0.22$, want $P(X \leq 12)$
Hand calculation:

$$\begin{aligned} F_X(12) &= P(X \leq 12) \\ &= \sum_{x=3}^{12} \binom{x-1}{2} (0.22)^3 (0.78)^{x-3} \\ &= 0.5114 \end{aligned}$$

Expected value:

$$E[X] = \frac{n}{p} = \frac{3}{0.22} = 13.64 \text{ projects}$$

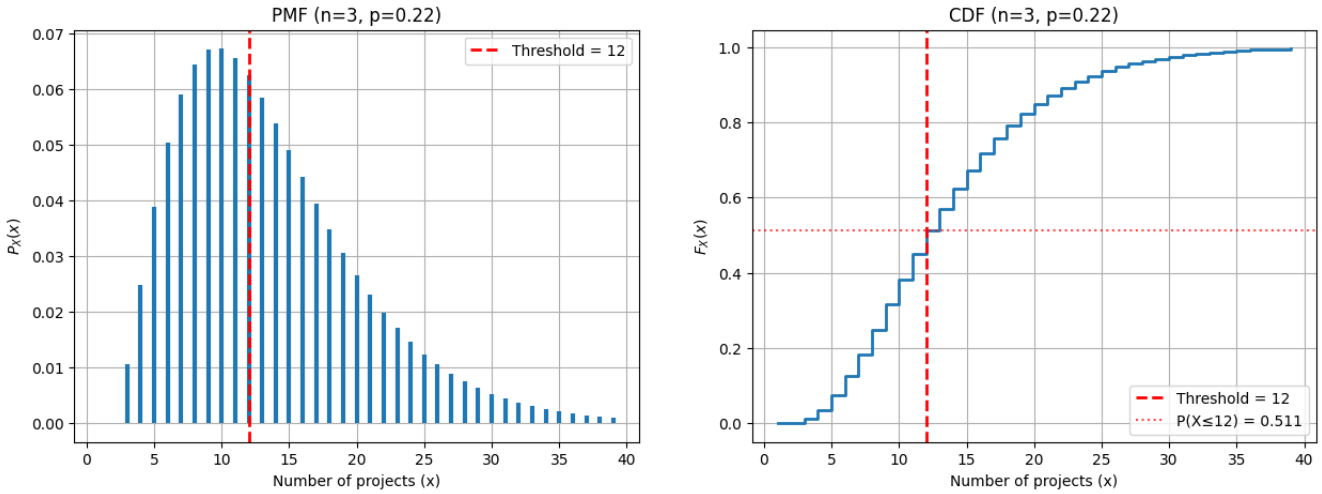

Figure 10: PMF and CDF for question 5d ($n = 3$, $p = 0.22$)

Using CDF we can see that $F_X(12) \approx 0.51$ (really 0.5114), so they have a much better chance if they move their research area. The expected value of X is

$$E[X] = 13.64$$

The expected value is much closer to 12, which tells us they have a better chance if they change their research area.

Comparison and Recommendation

Research Area	Original	Alternative
Success probability per project	12%	22%
Available budget (projects)	15	12
Expected projects needed	25.0	13.6
P(success within budget)	26.5%	51.14%

Analysis:

- The **alternative area** now has a significantly higher success probability (51.1% vs 26.5%)
- Both budgets are well above and below their respective expected values, but the alternative is much closer to feasibility

Conclusion: With limited budgets, the alternative research area offers the company's best chance of achieving their 3 breakthrough goal. The higher success rate per project compensates for the smaller budget, making it the strategically superior choice.

Summary of Python Implementation

All Python code segments combine into complete scripts for each question using these key libraries:

- `scipy.stats.binom` for binomial distribution (Questions 1, 3)
- `scipy.stats.geom` for geometric distribution (Question 2)
- `scipy.stats.poisson` for Poisson distribution (Question 4)
- `scipy.stats.nbinom` for negative binomial/Pascal distribution (Question 5)

Each implementation provides `.pmf()`, `.cdf()`, plotting capabilities, and verification of analytical results.