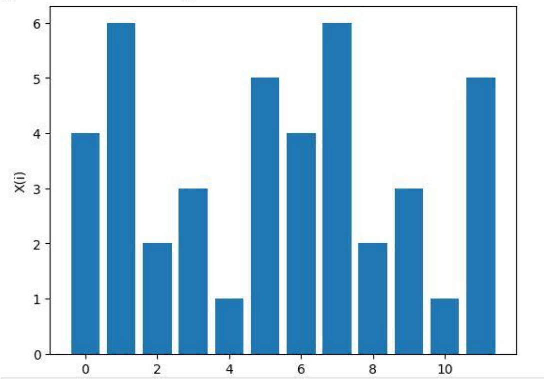
1ª LISTA DE EXERCÍCIOS

1)

$X_{i+1} = 5_{Xi} \, mod \, 7$
$X_0 = 4$
$X_1 = (5\,.\,4) \, mod \, 7 = 20 \, mod \, 7 = 6$
$X_2 = (5\,.\,6) \, mod \, 7 = 30 \, mod \, 7 = 2$
$X_3 = (5\,.\,2) \, mod \, 7 = 10 \, mod \, 7 = 3$
$X_4 = (5\,.\,3) \, mod \, 7 = 15 \, mod \, 7 = 1$
$X_5 = (5\,.\,1) \, mod \, 7 = 05 \, mod \, 7 = 5$
$X_6 = (5\,.\,5) \, mod \, 7 = 25 \, mod \, 7 = 4$
$X_7 = (5\,.\,4) \, mod \, 7 = 20 \, mod \, 7 = 6$

```
import numpy as np
import matplotlib.pyplot as plt

x=4
x1=np.array([x])
n=11
a=5
m=7
for i in range(n):
    x=(a*x)%m
    x1=np.append(x1,x)
print(x1)
ind=np.arange(n+1)
plt.bar(ind, x1)
plt.xlabel('amostra')
plt.ylabel('X(i)')
plt.show()
```

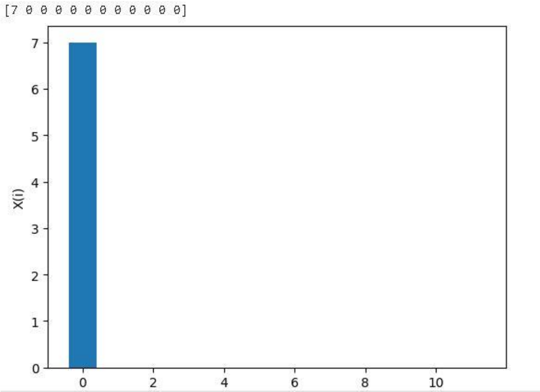
[4 6 2 3 1 5 4 6 2 3 1 5]



$X_0 = 7$
$X_1 = (5\,.\,7) \, mod \, 7 = 35 \, mod \, 7 = 0$
$X_2 = (5\,.\,0) \, mod \, 7 = 00 \, mod \, 7 = 0$
$X_3 = (5\,.\,0) \, mod \, 7 = 00 \, mod \, 7 = 0$

```
import numpy as np
import matplotlib.pyplot as plt

x=7
x1=np.array([x])
n=11
a=5
m=7
for i in range(n):
    x=(a*x)%m
    x1=np.append(x1,x)
print(x1)
ind=np.arange(n+1)
plt.bar(ind, x1)
plt.xlabel('amostra')
plt.ylabel('X(i)')
plt.show()
```

[7 0 0 0 0 0 0 0 0 0 0 0]



Ao comparar as duas sequencias conseguimos compreender que ao usar $X_0 = 7$, a sequência gerada consiste em apenas 0, divido a ser maior que m, resultando em um ciclo não aleatório. Já em gesta a sequência de $X_0 = 4$ obtivemos resultados aleatório dentro do período.

2) a)

$$P_r\,[X = X] = \frac{\lambda^x . e^{-\lambda}}{x!}$$
$$P_r\,[X = 0] = \frac{6^0 .\ e^{-6}}{0!}$$
$$P_r\,[X = 0] = e^{-6}$$
$$P_r\,[X = 0] = 0{,}002478\ ou\ 0{,}25\%$$

b)

$$P_r\,[X < 8] = P\,[X = 0] + P\,[X = 1] + P\,[X = 2] + P\,[X = 3] + P\,[X = 4] + P\,[X = 5] + P\,[X = 6] + P\,[X = 7]$$

$$P_r\,[X < 8] = e^{-6} + \frac{6^1 .\ e^{-6}}{1!} + \frac{6^2 .\ e^{-6}}{2!} + \frac{6^3 .\ e^{-6}}{3!} + \frac{6^4 .\ e^{-6}}{4!} + \frac{6^5 .\ e^{-6}}{5!} + \frac{6^6 .\ e^{-6}}{6!} + \frac{6^7 .\ e^{-6}}{7!}$$

$$P_r\,[X < 8] \approx 0{,}99999$$

```python
import numpy as np

lambda1=6 #Número médio de requisições
N=60 #Numero de amostras
value=0
count=0
av=np.array([])
x=np.random.uniform(0,1,N)
for ix in x:
    i = 0
    pr = np.exp(-lambda1)
    F=pr
    while ix>=F:
        pr=lambda1/(i+1)*pr
        F = F + pr
        i = i + 1;
    a1=i
    av=np.append(av,a1)

Pr0 = np.exp(-lambda1)

# Variância de C
variance_C = lambda1
# Desvio padrão de C
std_dev_C = np.sqrt(variance_C)
```

```python
for binvalue in av:
    if binvalue>=value:
        count=count+1
prob=count/N
Pro= Pr0*100

print("a) A probabilidade de não receber chamada é ≈",Pr0,"ou",Pro*100,"%")
print("b) A probabilidade do suporte receber menos de oito chamadas é ≈",prob)
print("c) Média de chamada por hora é:",variance_C)
print("d) A variância de C é:",variance_C)
print("e) O desvio padrão de C é:", std_dev_C)
```

```
a) A probabilidade de não receber chamada é ≈ 0.0024787521766663585 ou 24.787521766663584 %
b) A probabilidade do suporte receber menos de oito chamadas é ≈ 1.0
c) Média de chamada por hora é: 6
d) A variância de C é: 6
e) O desvio padrão de C é: 2.449489742783178
```

c) $\lambda = \frac{Total\ de\ chamadas}{Total\ de\ horas}$ $\quad \lambda = \frac{60}{10}$ $\quad \lambda = 6$

d) $Var\ C = \lambda = 6$

e) $\sigma = \sqrt{Var\ C}$ $\quad \sigma = \sqrt{6}$ $\quad \sigma \approx 2{,}45$

3) a)

$P[X = x] = \binom{n}{x}\ q^x\ (1 - q)^{n-x}$

X=0

$P[X = 0] = \binom{8}{0} . (0{,}15)^0 . (1 - 0{,}15)^8$

$P[X = 0] = \left(\frac{8!}{0!\,(8-0)!}\right) . (0{,}15)^0 . (0{,}85)^8$

$P[X = 0] = 1 . (0{,}15)^0 . (0{,}85)^8$

$P[X = 0] \approx 0{,}085$

X=1

$P[X = 1] = \binom{8}{1} . (0{,}15)^1 . (1 - 0{,}15)^7$

$P[X = 1] = \left(\frac{8!}{1!\,(8-1)!}\right) . (0{,}15)^1 . (0{,}85)^7$

$P[X = 1] = 8 . (0{,}15)^1 . (0{,}85)^7$

$P[X = 1] \approx 0{,}275$

X=2

$P[X = 2] = \binom{8}{2} . (0{,}15)^2 . (1 - 0{,}15)^6$

$P[X = 2] = \left(\frac{8!}{2!\,(8-2)!}\right) . (0{,}15)^2 . (0{,}85)^6$

$P[X = 2] = 28 . (0{,}15)^2 . (0{,}85)^6$

$P[X = 2] \approx 0{,}322$

$P[X \leq 2] = P[X = 0] + P[X = 1] + P[X = 2]$

$Pa[X \leq 2] = 0{,}085 + 0{,}275 + 0{,}322$

$Pa[X \leq 2] \approx 0{,}682$

b)
X=6

$P[X = 6] = \binom{8}{6} . (0{,}15)^6 . (1 - 0{,}15)^2$

$P[X = 6] = \left(\frac{8!}{6!\,(8-6)!}\right) . (0{,}15)^6 . (0{,}85)^2$

$P[X = 6] = 28 \,.\, (0{,}15)^6 \,.\, (0{,}85)^2$

$P[X = 6] \approx 0{,}236$

X=7

$P[X = 7] = \binom{8}{7} \,.\, (0{,}15)^7 \,.\, (1 - 0{,}15)^1$

$P[X = 7] = \left(\frac{8!}{7!\,(8-7)!}\right) \,.\, (0{,}15)^7 \,.\, (0{,}85)^1$

$P[X = 7] = 8 \,.\, (0{,}15)^7 \,.\, (0{,}85)^1$

$P[X = 7] \approx 0{,}061$

X=8

$P[X = 8] = \binom{8}{8} \,.\, (0{,}15)^8 \,.\, (1 - 0{,}15)^0$

$P[X = 8] = \left(\frac{8!}{8!\,(8-8)!}\right) \,.\, (0{,}15)^8 \,.\, (0{,}85)^0$

$P[X = 8] = 1 \,.\, (0{,}15)^8 \,.\, (0{,}85)^0$

$P[X = 8] \approx 0{,}002$

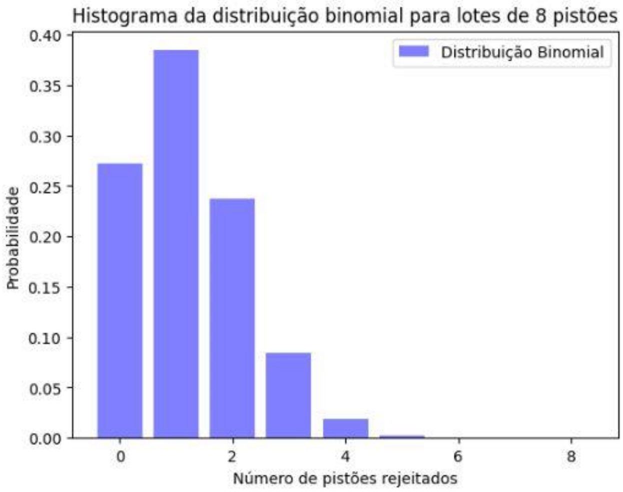
$Pb[X \geq 6] = P[X = 6] + P[X = 7] + P[X = 8]$

$Pb[X \geq 6] = 0{,}236 + 0{,}061 + 0{,}002$

$Pb[X \geq 6] \approx 0{,}299$

Probabilidade de não mais que 2 pistões rejeitados 0,682; Probabilidade de pelo menos 6 pistões rejeitados 0,299.

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

# Definindo os parâmetros
n = 8
p = 0.15
q = 1 - p

# (a) Calculando a probabilidade de não mais que 2 rejeitados
prob_a = sum(binom.pmf(k, n, p) for k in range(3))

# (b) Calculando a probabilidade de pelo menos 6 rejeitados
prob_b = sum(binom.pmf(k, n, p) for k in range(6, 9))

print("Probabilidade de não mais que 2 rejeitados:", prob_a)
print("Probabilidade de pelo menos 6 rejeitados:", prob_b)

# Plotando o histograma da distribuição binomial
x = np.arange(0, n+1)
probabilidade = binom.pmf(x, n, p)
plt.bar(x, probabilidade, alpha=0.5, color='blue', label='Distribuição Binomial')
plt.xlabel('Número de pistões rejeitados')
plt.ylabel('Probabilidade')
plt.title('Histograma da distribuição binomial para lotes de 8 pistões')
plt.legend()
plt.show()
```



---

4)

$$P_r\,[X = X] = \frac{\lambda^x .\ e^{-\lambda}}{x!}$$

$$P_r\,[X = 0] = \frac{3^0 .\ e^{-3}}{0!} = 0.049787$$

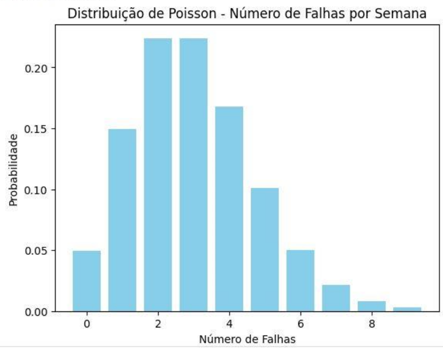
$$P_r\,[X = 1] = \frac{3^1 .\ e^{-3}}{1!} = 0.149361$$

Então, a probabilidade de pelo menos 2 falhas em uma semana é
$1 - P_r[X = 0] - P_r[X = 1]$

$1 - 0.049787 - 0.149361$ = 0,800852

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

# Média de falhas por semana
lambd = 3

# Calculando a probabilidade de 0 e 1 falha
p_0 = poisson.pmf(0, lambd)
p_1 = poisson.pmf(1, lambd)

# Calculando a probabilidade de pelo menos 2 falhas
p_at_least_2 = 1 - p_0 - p_1

print("Probabilidade de pelo menos 2 falhas em uma semana:", p_at_least_2)
print(p_0)
print(p_1)
# Plotando o histograma
x = np.arange(0, 10)
plt.bar(x, poisson.pmf(x, lambd), color='skyblue')
plt.title('Distribuição de Poisson - Número de Falhas por Semana')
plt.xlabel('Número de Falhas')
plt.ylabel('Probabilidade')
plt.show()
```

```
Probabilidade de pelo menos 2 falhas em uma semana: 0.8008517265285442
0.049787068367863944
0.14936120510359185
```



---

5)

$$f(x) = \lambda e^{-\lambda x}$$

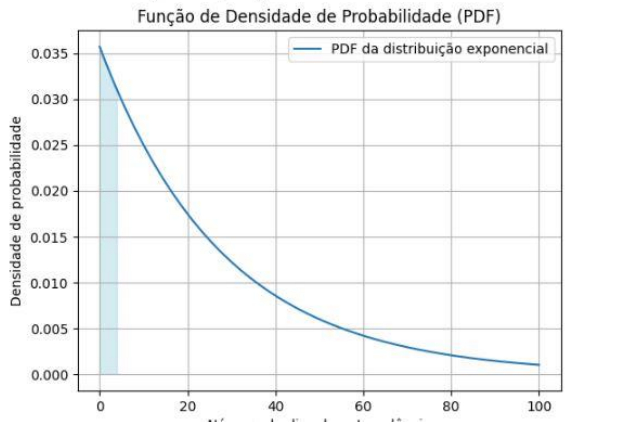
$$f(x) = \frac{1}{28} e^{-\frac{x}{28}}$$

$$P(X < 4) = \int_0^4 \frac{1}{28} e^{-\frac{x}{28}}\, dx$$

$$P(X < 4) = 0{,}1331221$$

```python
# Função de densidade de probabilidade da distribuição exponencial
def expon_pdf(x, lamb):
    return (1 / lamb) * np.exp(-x / lamb)

# Parâmetro da distribuição exponencial (tempo médio)
lamb = 28

# Função para calcular a integral da função de densidade de probabilidade
def integral_expon_pdf(a, b):
    return quad(expon_pdf, a, b, args=(lamb))[0]

# Probabilidade de comprar uma passagem com menos de 4 dias de antecedência
probabilidade = integral_expon_pdf(0, 4)

print("Probabilidade de comprar uma passagem com menos de 4 dias de antecedência:", probabilidade)

# Traçar o gráfico da função de densidade de probabilidade
x = np.linspace(0, 100, 1000)
pdf = expon_pdf(x, lamb)

plt.plot(x, pdf, label='PDF da distribuição exponencial')
plt.fill_between(x, pdf, where=(x < 4), color='lightblue', alpha=0.5)
plt.title('Função de Densidade de Probabilidade (PDF)')
plt.xlabel('Número de dias de antecedência')
plt.ylabel('Densidade de probabilidade')
plt.legend()
plt.grid(True)
plt.show()
```

```
Probabilidade de comprar uma passagem com menos de 4 dias de antecedência: 0.1331221
```



---

6) Utiliza o método da inversão da função de distribuição acumulada:

$$f(x) = 1 - (1 - p)^x$$

```python
import numpy as np
```

```python
def gerador_das_variaveis_aleatorias(p, num_samples):
    u = np.random.rand(num_samples)
    x = np.ceil(np.log(1 - u) / np.log(1 - p))
    return x.astype(int)

# Probabilidade de sucesso
p = 20 / (30 + 20)  # Probabilidade de retirar uma bola preta

# Simulação para calcular a probabilidade de que a 6ª bola retirada com reposição seja a primeira bola preta
num_simulations = 10000

# Gera as variáveis aleatórias geométricas
variaveis_geometricas = gerador_das_variaveis_aleatorias(p, num_simulations)

# Calcula a probabilidade de que a 6ª bola retirada com reposição seja a primeira bola preta
count_suc = np.sum(variaveis_geometricas == 6)
prob = count_suc / num_simulations

print("Probabilidade de que a 6ª bola retirada com reposição seja a primeira bola preta:", prob)
```

```
Probabilidade de que a 6ª bola retirada com reposição seja a primeira bola preta: 0.0317
```

---

7) invertendo a função $F(x)$ para encontrar $F^{-1}\ (u)$, onde $u = \frac{e^x - 1}{e^2 - 1}$ para x.

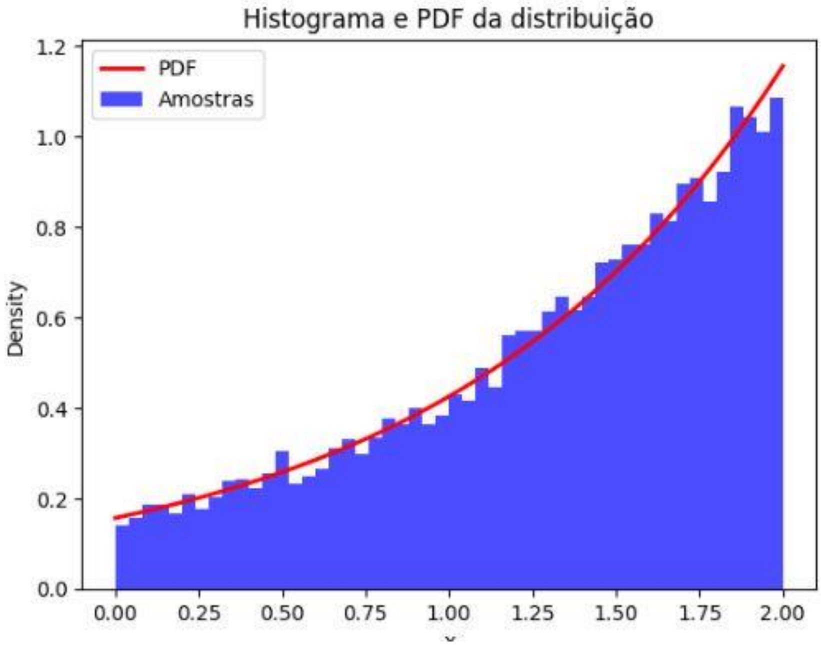
$u(e^2 - 1) = e^x - 1$

$u(e^2 - 1) + 1 = e^x$

$x = \ln(u(e^2 - 1) + 1)$

```python
import numpy as np
import matplotlib.pyplot as plt
```

```python
# Função inversa
def inverse_F(u):
    return np.log(u * (np.exp(2) - 1) + 1)

# Gerando valores de u uniformemente distribuídos
num_samples = 10000
u_values = np.random.rand(num_samples)

# Aplicando a função inversa aos valores de u
samples = inverse_F(u_values)

# Plotando o histograma das amostras
plt.hist(samples, bins=50, density=True, alpha=0.7, color='b')

# Plotando a função de densidade de probabilidade (pdf) para comparação
x = np.linspace(0, 2, 1000)
pdf = np.exp(x) / (np.exp(2) - 1)
plt.plot(x, pdf, 'r', linewidth=2)

plt.xlabel('x')
plt.ylabel('Density')
plt.title('Histograma e PDF da distribuição')
plt.legend(['PDF', 'Amostras'])
plt.show()
```

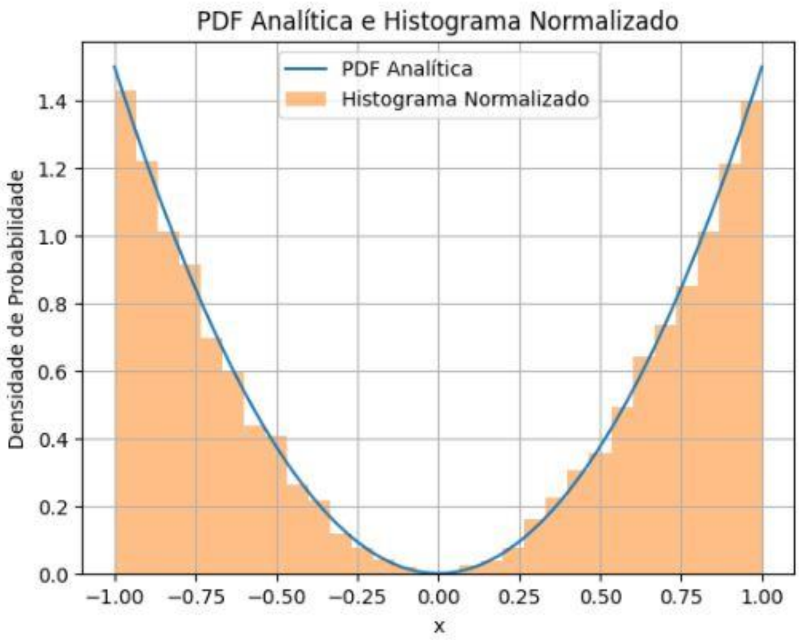


8)

```
[23] import numpy as np
     import matplotlib.pyplot as plt
```

```
# Função alvo
def f(x):
    return 1.5 * x**2

# Função de envelope
def g(x):
    return 2 * np.ones_like(x)

# Método de aceitação/rejeição
def acceptance_rejection_sampling(n_samples):
    samples = []
    while len(samples) < n_samples:
        x = np.random.uniform(-1, 1)
        u = np.random.uniform(0, 2)  # Amplitude de g(x)
        if u <= f(x) / g(x):
            samples.append(x)
    return np.array(samples)

# Número de amostras
n_samples = 10000

# Gerar amostras
samples = acceptance_rejection_sampling(n_samples)
```

```
# Plotar PDF analítica
x_values = np.linspace(-1, 1, 1000)
plt.plot(x_values, f(x_values), label='PDF Analítica')

# Plotar histograma normalizado
plt.hist(samples, bins=30, density=True, alpha=0.5, label='Histograma Normalizado')

plt.title('PDF Analítica e Histograma Normalizado')
plt.xlabel('x')
plt.ylabel('Densidade de Probabilidade')
plt.legend()
plt.grid(True)
plt.show()
```


PDF Analítica e Histograma Normalizado
PDF Analítica
Histograma Normalizado
Densidade de Probabilidade
x
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00