Modelação de Sistemas Físicos - Aula Prática nº2

Regressão linear pelo método dos mínimos quadrados:

Para um dado conjunto de N pares de valoes $\{(x_i,y_i)\}$, com $i=1,\ldots,N$, a reta $y(x)=mx+b$ que minimiza a *distância* $\sum_i [y_i - y(x_i)]^2$ é obtida quando,

$$ m = \frac{N \sum_i x_i y_i - \sum_i x_i \sum_i y_i}{N \sum_i x_i^2 - (\sum_i x_i)^2}, $$$$ b = \frac{\sum_i x_i^2 \sum_i y_i - \sum_i x_i \sum_i x_i y_i}{N \sum_i x_i^2 - (\sum_i x_i)^2} $$
sendo a qualidade do ajuste dada pelo coeficiente de correlação,

$$ r^2 = \frac{(N \sum_i x_i y_i - \sum_i x_i \sum_i y_i)^2}{[N \sum_i x_i^2 - (\sum_i x_i)^2][N\sum_i y_i^2 - (\sum_i y_i)^2]}. $$
O erro associado ao declive e ordenada no orígem é dado por,

$$ \Delta m = |m|\sqrt{\frac{(1/r^2) - 1}{N-2}} $$$$ \Delta b = \Delta m \sqrt{\frac{\sum_i x_i^2}{N}} $$

Problema cap 1.5

Reset the kernel

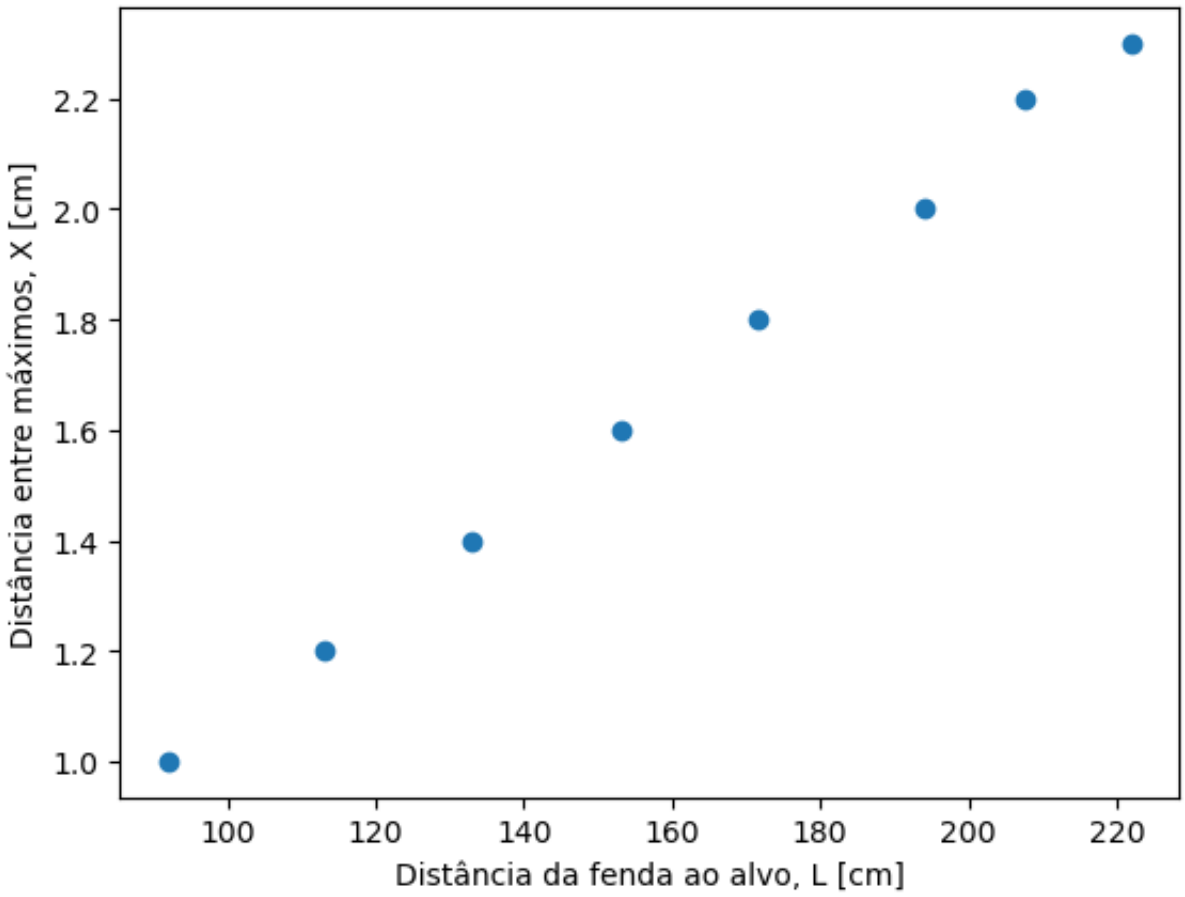
In [1]:
```
# Reset ipython kernel
#import os
#os._exit(00)
```

Escreva um programa em python que calcule as quantidades anteriores.

a) Comece por representar os dados experimentais num gráfico

Use os dados os valores da tabela apresentados da aula prática, correpondentes às medições realizadas numa experiência de difração por uma fenda única de um feixe de luz [link], em que L é a distância da fenda ao alvo e X a distância entre máximos luminosos consecutivos da figura de difração.

In [2]:
```
# Importar os pacotes "numpy" e "matplotib", e definir respetivos acrónim
import numpy as np
import matplotlib.pyplot as plt

# Definir cadeias de valores {x_i} e {y_i}
L_i = np.array([222.0, 207.5, 194.0, 171.5 , 153.0, 133.0, 113.0, 92.0])
X_i = np.array([2.3, 2.2, 2.0, 1.8, 1.6, 1.4, 1.2, 1.0])

# Representar dados num grafico (usando o matplotlib)
plt.scatter(L_i, X_i)
plt.xlabel("Distância da fenda ao alvo, L [cm]")
plt.ylabel("Distância entre máximos, X [cm]")
plt.show()
```



b) Verifique as somas das expressões

$\sum_i x_i y_i = 2322.4$

$\sum_i x_i = 1286.0$

$\sum_i y_i = 13.5$

$\sum_i x_i^2 = 221719.5$

$\sum_i y_i^2 = 24.33$

In [3]:
```
print("sum_xy = {0:.1f}".format(np.sum(L_i * X_i)))
print("sum_x  = {0:.1f}".format(np.sum(L_i)))
print("sum_y  = {0:.1f}".format(np.sum(X_i)))
print("sum_x2 = {0:.1f}".format(np.sum(L_i ** 2)))
print("sum_y2 = {0:.2f}".format(np.sum(X_i ** 2)))
```

```
sum_xy = 2322.4
sum_x  = 1286.0
sum_y  = 13.5
sum_x2 = 221719.5
sum_y2 = 24.33
```

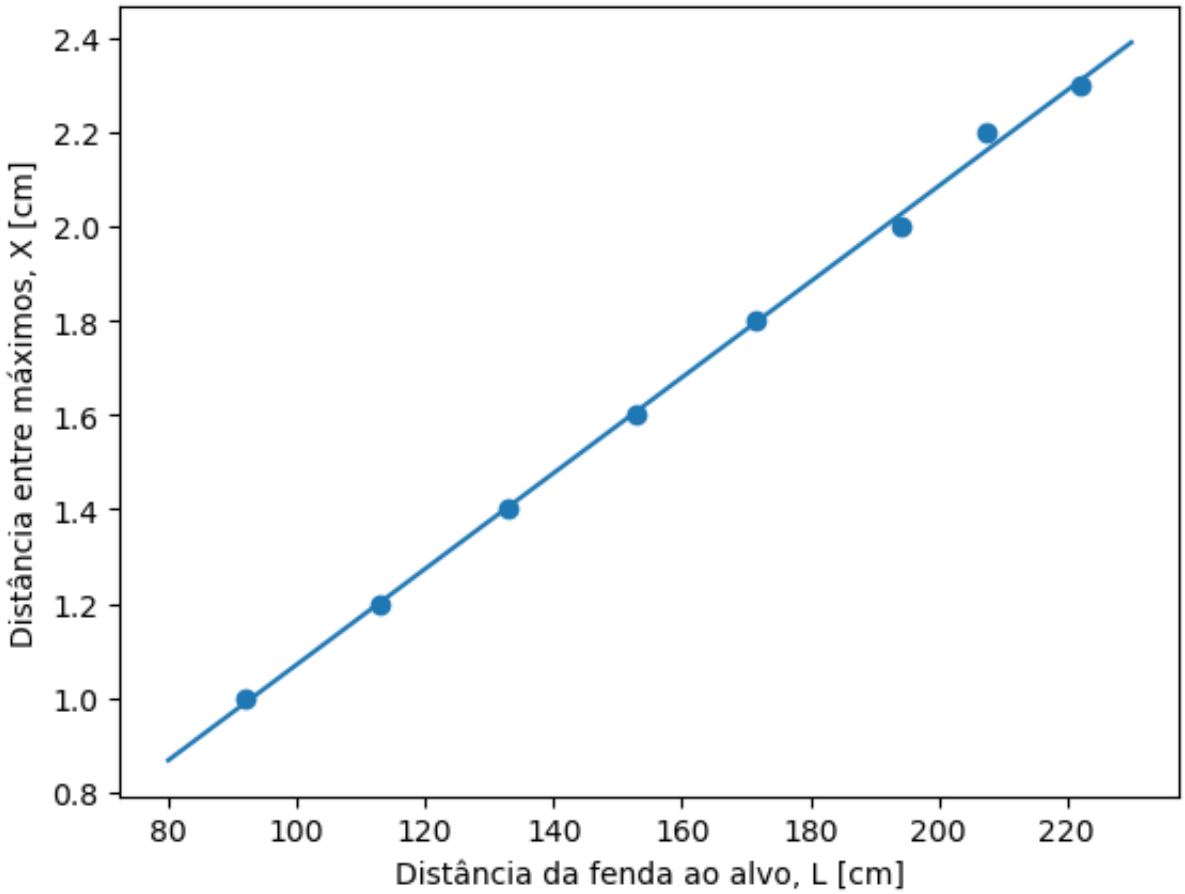
c) De seguida calcule o declive, a ordenada na origem e o coeficiente de determinação ou de correlação r^2.

In [4]:
```python
# Função minimos quadrados
# Argumentos de entrada:
#     x: cadeia de valores x_i
#     y: cadeia de valores y_i
# Argumentos de saída:
#     m: declive
#     b: ordenada na orígem
#     r2: coeficiente de correlação ** 2
#     dm: erro do declive
#     db: erro da ordenada na orígem
def minimos_quadrados(x, y):
    # numero de pares {(x_i, y_i)}. Assumimos que x.size = y.size
    N = x.size

    # Definir somas elementares (escalares)
    sum_x = np.sum(x)
    sum_y = np.sum(y)
    sum_x2 = np.sum(x ** 2)
    sum_y2 = np.sum(y ** 2)
    sum_xy = np.sum(x * y)

    # Calcular declive
    m = (N * sum_xy - sum_x * sum_y) / (N * sum_x2 - sum_x ** 2)

    # Calcular ordenada no origem
    b = (sum_x2 * sum_y - sum_x * sum_xy) / (N * sum_x2 - sum_x ** 2)

    # Calcular coeficiente de correlação
    r2 = (N * sum_xy - sum_x * sum_y) ** 2 / ((N * sum_x2 - sum_x ** 2) *

    # Calcular erro do declive
    dm = np.absolute(m) * np.sqrt((1 / r2 - 1) / (N - 2))

    # Calcular erro da ordenada na orígem
    db = dm * np.sqrt((sum_x2) / N)

    # Devolver os argumentos desejados
    return m, b, r2, dm, db


# Executar a função (calcular melhor reta)
m, b, r2, dm, db = minimos_quadrados(L_i, X_i)

# Imprimir resultados
print("m = {0:.4f}".format(m))
print("b = {0:.2f} cm".format(b))
print("r² = {0:.4f}...".format(r2))
print("Δm = {0:.4f}".format(dm))
print("Δb = {0:.2f} cm".format(db))
```

```
m = 0.0102
b = 0.06 cm
r² = 0.9985...
Δm = 0.0002
Δb = 0.03 cm
```

d) faça um gráfico com os pontos experimentais e a reta cujos parâmetros m e b calculou anteriormente.

In [5]:
```
# Definir dois pontos (x1, y1) e (x2, y2) para representar melhor reta
L = np.array([80.0, 230.0])
X = m * L + b

# Representar dados num grafico (usando o matplotlib)
plt.scatter(L_i, X_i)
plt.plot(L, X)
plt.xlabel("Distância da fenda ao alvo, L [cm]")
plt.ylabel("Distância entre máximos, X [cm]")
plt.show()
```



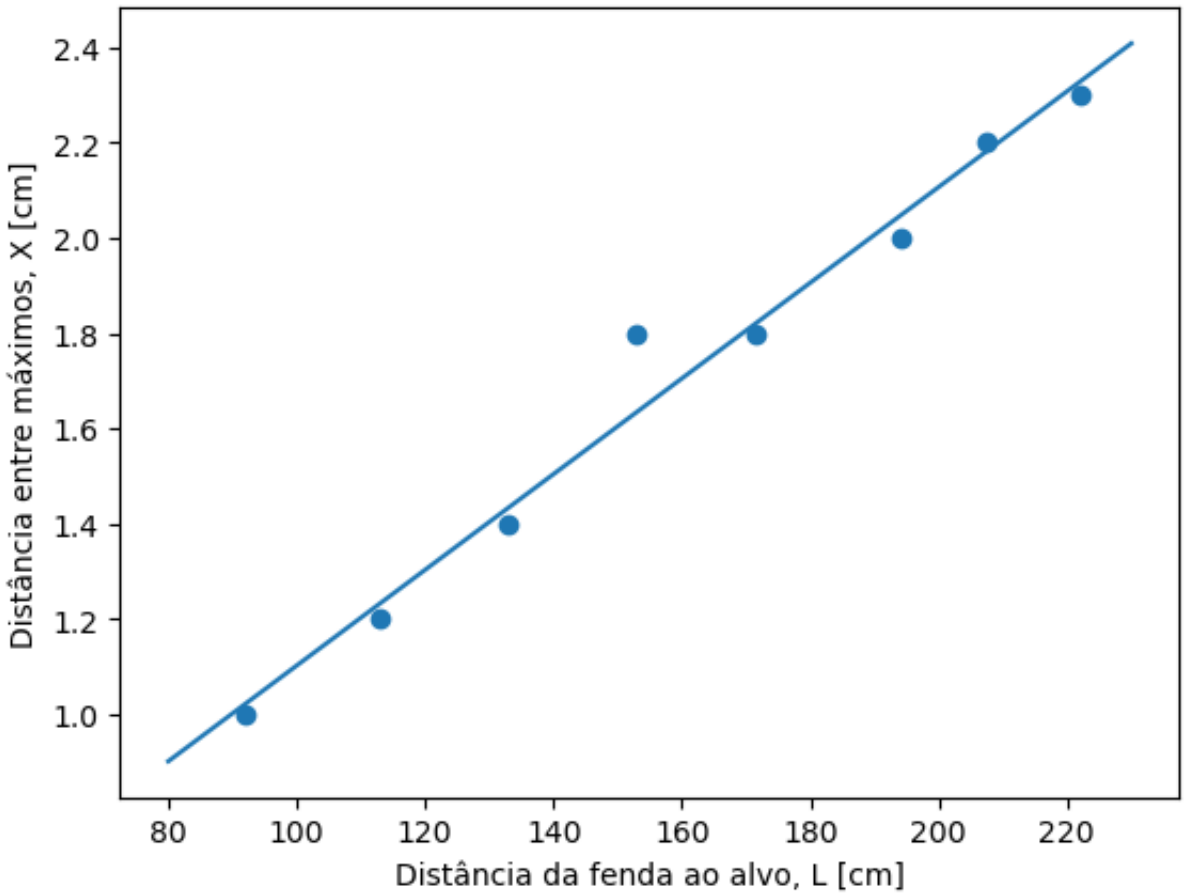
e) Encontre o valor de X, quando L = 165.0 cm. Use a reta determinada pela regressão linear.

In [6]:
```
print("X(L=165.0 cm^1) = {0:.2f} cm".format(m * 165.0 + b))
print("ΔX = Δm * 165 + Δb = {0:.2f} cm".format(dm * 165.0 + db))
```

```
X(L=165.0 cm^1) = 1.73 cm
ΔX = Δm * 165 + Δb = 0.05 cm
```

f) Afaste da reta encontrada um dos valores medidos de X. Compare o coeficiente de determinação com o valor anterior. Faça um gráfico com os novos pontos experimentais e a nova reta.

In [7]:
```
# Afastei o valor perto do centro, X(153) de 1.6 para 1.8
X_i = np.array([2.3, 2.2, 2.0, 1.8, 1.8, 1.4, 1.2, 1.0])

# Executar novamente a função (calcular melhor reta)
m, b, r2, dm, db = minimos_quadrados(L_i, X_i)

# Definir dois pontos (x1, y1) e (x2, y2) para representar melhor reta
L = np.array([80.0, 230.0])
X = m * L + b

# Representar dados num grafico (usando o matplotlib)
plt.scatter(L_i, X_i)
plt.plot(L, X)
plt.xlabel("Distância da fenda ao alvo, L [cm]")
plt.ylabel("Distância entre máximos, X [cm]")
plt.show()

# Imprimir resultados
print("m = {0:.4f} cm^2".format(m))
print("b = {0:.1f} cm".format(b))
print("r² = {0:.4f}...".format(r2))
print("Δm = {0:.4f} cm^2".format(dm))
print("Δb = {0:.1f} cm".format(db))
```




```
m = 0.0101 cm^2
b = 0.1 cm
r² = 0.9782...
Δm = 0.0006 cm^2
Δb = 0.1 cm
```

O coeficiente de correlação baixa de 0.998... para 0.978..., ou seja, a qualidade do ajuste piora.

O erro $Δb$ aumenta uma ordem de grandeza

Problema cap 1.6

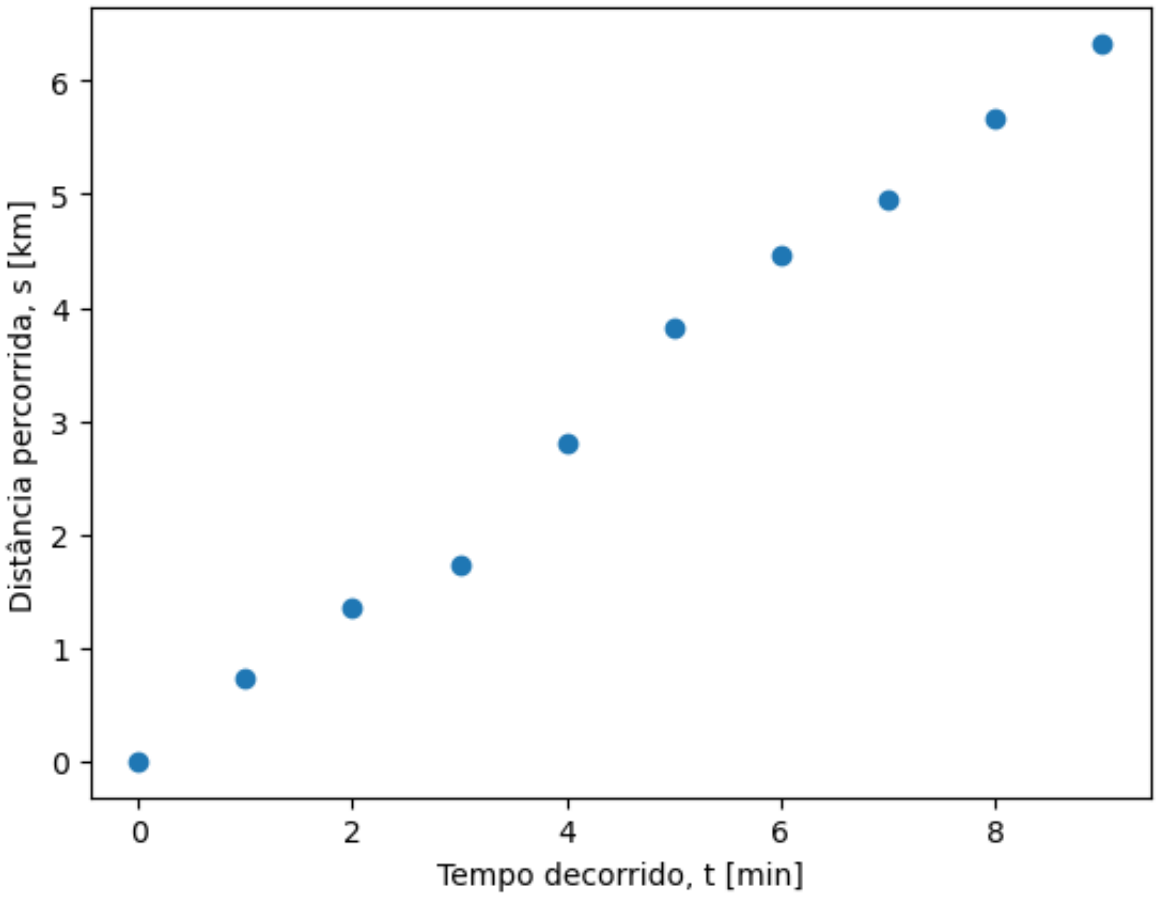
Um ciclista tenta percorrer uma distância de 10 km com uma velocidade constante (uniforme).

Durante os primeiros 9 minutos, a cada minuto que passa, o seu treinador e regista os valores da distância percorrida em km,

```
0.00, 0.735, 1.363, 1.739, 2.805, 3.814, 4.458, 4.955,
5.666, 6.329
```

a) Apresente estas medições num gráfico. A analisar o gráfico, a relação entre o tempo e a distância percorrida é linear?

In [8]:
```
# Definir cadeias de valores {t_i} e {s_i}
t_i = np.arange(0.0,10.0) # tempo decorrido [min]
s_i = np.array([0.00, 0.735, 1.363, 1.739, 2.805, 3.814, 4.458, 4.955, 5.

# Representar dados num grafico (usando o matplotlib)
plt.scatter(t_i, s_i)
plt.xlabel("Tempo decorrido, t [min]")
plt.ylabel("Distância percorrida, s [km]")
plt.show()
```



Sim. É aproximadamente linear. O ciclista mostra uma pequena quebra de ritmo aos 3 minutos, seguida de recuperação e novamente uma ligeira quebra de ritmo aos 7 minutos.

b) Encontre o declive, a ordenada na origem, os erros respetivos e o coeficiente de determinação (correlação).

In [9]:
```
# Executar novamente a função (calcular melhor reta)
m, b, r2, dm, db = minimos_quadrados(t_i, s_i)


# Imprimir resultados
print("m = {0:.2f} km/min".format(m))
print("b = {0:.1f} km".format(b))
print("r² = {0:.4f}...".format(r2))
print("Δm = {0:.2f} km/min".format(dm))
print("Δb = {0:.1f} km".format(db))
```

```
m = 0.72 km/min
b = -0.0 km
r² = 0.9938...
Δm = 0.02 km/min
Δb = 0.1 km
```

Sendo $r^2=0.99384\ldots$, podemos afirmar que os dados seguem uma relação aproximadamente linear.

Sendo a velocidade $s = s_0 + vt$, claramente verificamos que $v = m$.
Considerando, que

- $r^2 \sim 1$;
- O erro na velocidade $\Delta v = \Delta m = 0.02~\textrm{km/min}$;
- A variância da velocidade $\sigma_v = \Delta v \sqrt{N}=0.06~\textrm{km/min}$,

concluimos que o ciclista **não** conseguiu mater a velocidade constante.

c) Qual a velocidade média do ciclista?

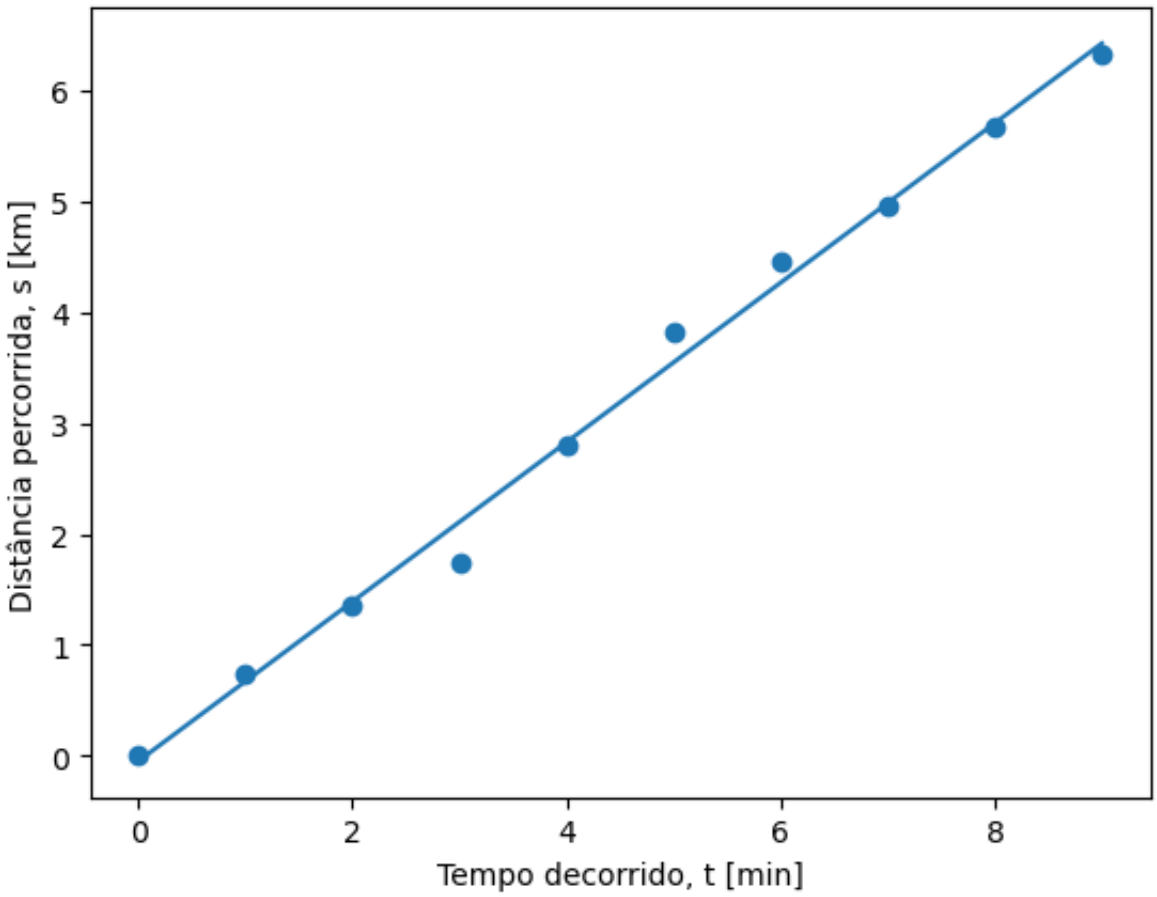
A velocidade média do ciclista é:

In [10]:
```
print("v = {0:.2f} km/min".format(m))
print("Δv = Δm = {0:.2f} km/min".format(dm))
```

```
v = 0.72 km/min
Δv = Δm = 0.02 km/min
```

d) Use a função polyfit dos pacote numpy ou do pacote pylab para encontrar a reta que mais se aproxima das medições. O declive e a ordenada na origem concordam com os valores calculados na alínea b)?

In [11]:
```
# Função polyfit, encontra os coeficientes a = [a_0, a_1, a_2,..., a_n] d
# s(t) = a_0 * t^n + ... + a_n-2 * t^2 + a_n-1 * t^1 + a_n
# que melhor se ajusta a um conjunto de pontos {(T_i, S_i)}
a = np.polyfit(t_i, s_i, 1)

# Gerar valores do tempo e espaço percorrido para representação da reta
t = np.linspace(0, 9.0, 2)
s = a[0] * t + a[1]

# Representar dados num grafico (usando o matplotlib)
plt.scatter(t_i, s_i)
plt.plot(t, s)
plt.xlabel("Tempo decorrido, t [min]")
plt.ylabel("Distância percorrida, s [km]")
plt.show()

# Imprimir resultados
print("a[0] = ", a[0], "km/min")
print("a[1] = ", a[1], "km")
print("m = ", m, "km/min")
print("b = ", b, "km")
```




```
a[0] =  0.7188121212121215 km/min
a[1] =  -0.048254545454546265 km
m =  0.7188121212121212 km/min
b =  -0.048254545454544835 km
```

Os valores obtidos pela função `np.polyfit` são identicos aos obtidos pelo método dos mínimos quadrados.

e) Apresente a velocidade em km/hora.

min = 1/60 h

km/min = 60 km/h

In [12]:
```
print("v = ", m, "km/min = ", m * 60, "km/h")
```

```
v =  0.7188121212121212 km/min =  43.128727272727275 km/h
```

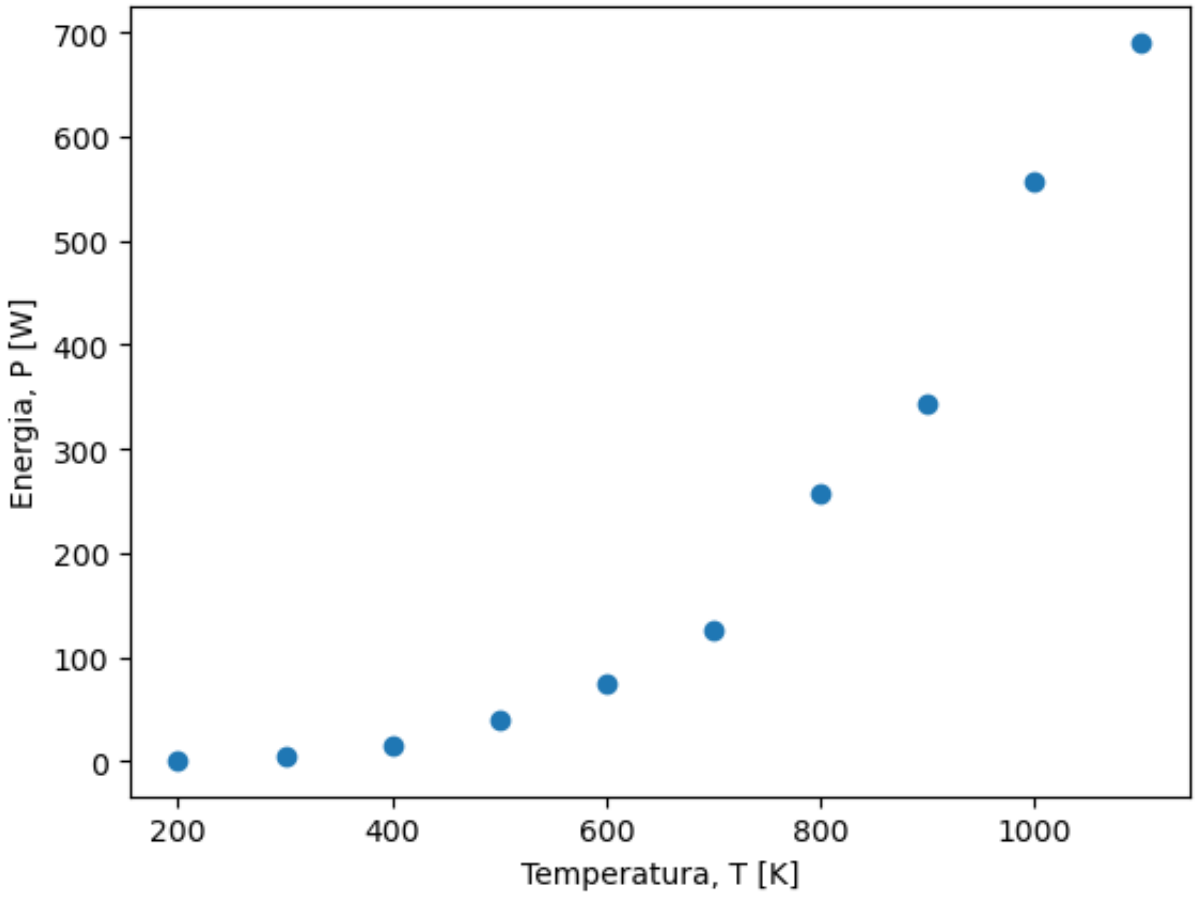
Problema cap 1.8

Foi medida a energia por segundo (potência) emitida por um corpo negro (corpo que absorve toda a energia que incide nele) com uma área superficial $A=100~\textrm{cm}^2$ em função da temperatura absoluta, T, e registada na seguinte tabela,

Grandeza	Medições
$T~(K)$	200.0, 300.0, 400.0, 500.0, 600.0, 700.0, 800.0, 900.0, 1000.0, 1100.0
$P~(W)$	0.6950, 4.363, 15.53, 38.74, 75.08, 125.2, 257.9, 344.1, 557.4, 690.7

a) Apresente estas medições num gráfico. A analisar o gráfico, a relação entre a energia emitida e a temperatura é linear?

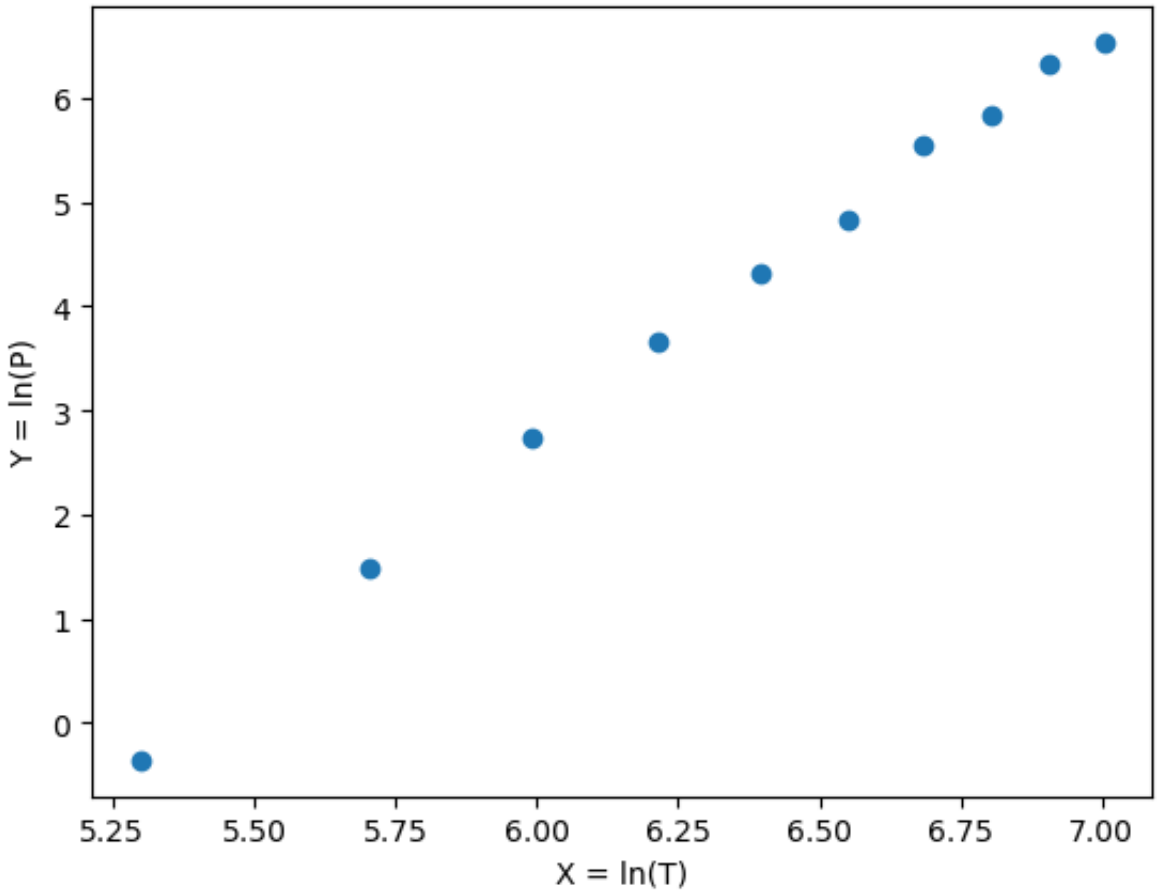
In [13]:
```
# Temperatura medida [K]
T_i = np.array([200.0, 300.0, 400.0, 500.0, 600.0, 700.0, 800.0, 900.0, 1

# Energia medida [W]
P_i = np.array([0.6950, 4.363, 15.53, 38.74, 75.08, 125.2, 257.9, 344.1,

# Representar dados num grafico (usando o matplotlib)
plt.scatter(T_i, P_i)
plt.xlabel("Temperatura, T [K]")
plt.ylabel("Energia, P [W]")
plt.show()
```



Claramente, a relação entre a potência dissipada e a temperatura do corpo negro **não é linear**.

b) Apresente as medições num gráfico log-log.

In [14]:
```
# Atenção à diferença entre o logaritmo natural "np.log" e o logarítmo de
X_i = np.log(T_i)
Y_i = np.log(P_i)

# Representar dados num grafico (usando o matplotlib)
plt.scatter(X_i, Y_i)
plt.xlabel("X = ln(T)")
plt.ylabel("Y = ln(P)")
plt.show()
```



c) Qual a dependência entre a potência emitida e a temperatura?

Pista: encontre uma lei de potência ou uma lei exponencial que se ajuste aos dados

De facto, segundo a lei de Stefan–Boltzmann, a potência dissipada por um corpo negro é proporcional a T^4.

Assumindo uma relação $P = c\, T^n$,

$P = c \, T^n \Leftrightarrow \ln P =\ln c + n \ln T$

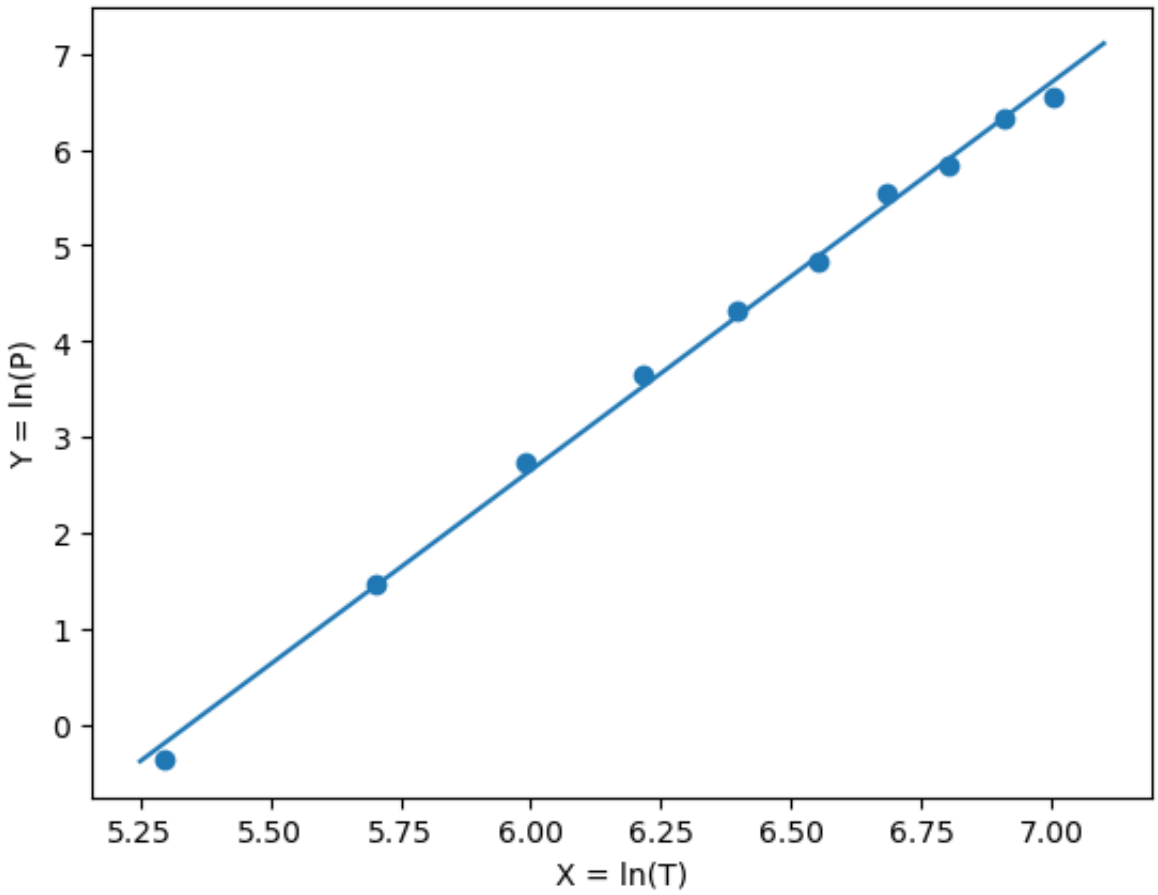
$X = \ln T$

$Y = \ln P$

$m = n$

$b = \ln c$

In [15]:
```
# Função polyfit de primeira ordem, encontra os coeficientes a = [a_0, a_
# y(x) = a_0 * x + a_1
# que melhor se ajusta a um conjunto de pontos {(X_i, Y_i)}
a = np.polyfit(X_i, Y_i, 1)

# Gerar valores do tempo e espaço percorrido para representação da reta
x = np.linspace(5.25, 7.10, 2)
y = a[0] * x + a[1]

# Representar dados num grafico (usando o matplotlib)
plt.scatter(X_i, Y_i)
plt.plot(x, y)
plt.xlabel("X = ln(T)")
plt.ylabel("Y = ln(P)")
plt.show()

print("n = ", a[0])
print("c = ", np.exp(a[1]), "W/K^n")
```




```
n =  4.048269520127869
c =  4.0095023319733347e-10 W/K^n
```

Problema cap 1.9

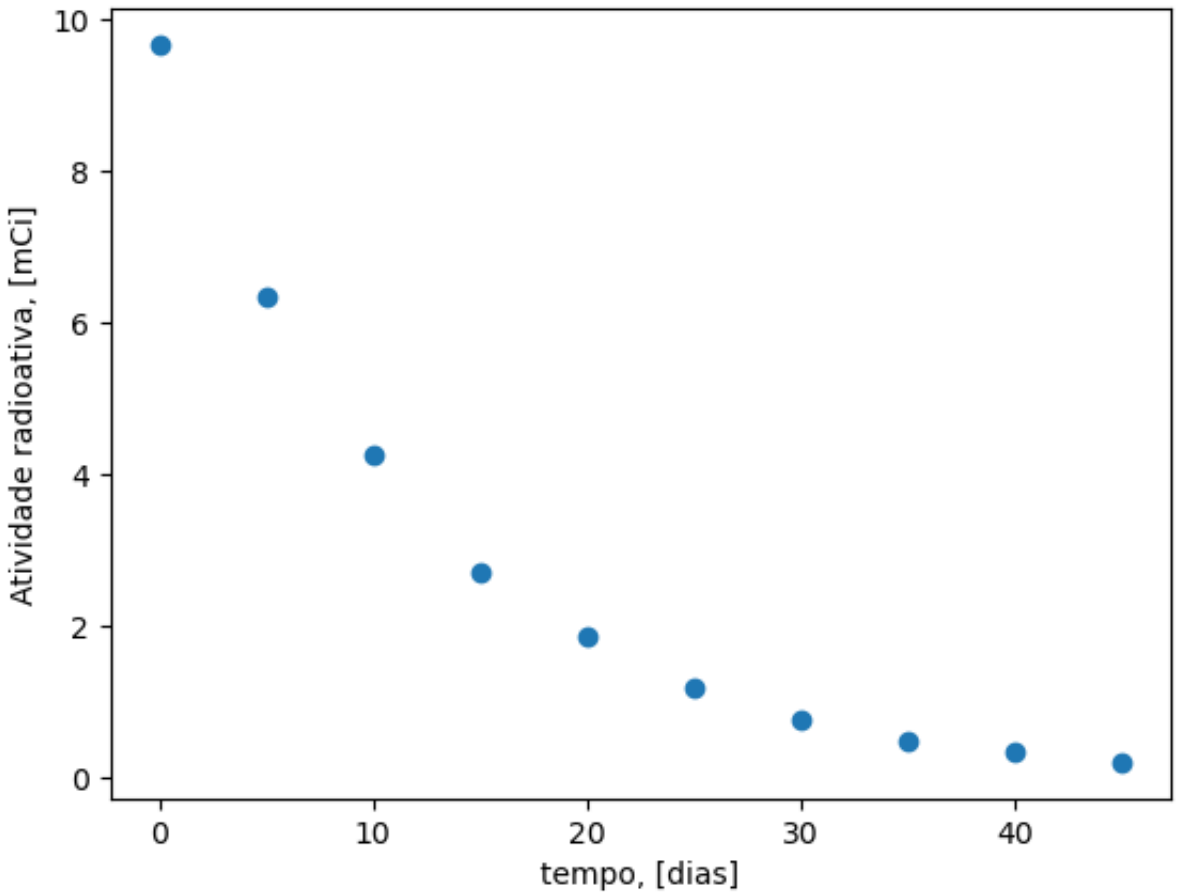
Foi medida a atividade de uma amostra do isótopo radioativo $^{131}\textrm{I}$. Foram registadas 10 medições separadas por intervalos de 5 dias. Os valores medidos são, em mCi (milicurie):

```
9.676, 6.355, 4.261, 2.729, 1.862, 1.184, 0.7680, 0.4883,
0.3461, 0.2119
```

A metado do tempo de vida deste isótopo radioativo é $\tau_{1/2}=8.0197~\textrm{dias}$

a) Apresente estas medições num gráfico. A analisar o gráfico, a relação entre a atividade e o tempo é linear?

In [16]:
```
# Tempo [dias]
t_i = np.array([0.0, 5.0, 10.0, 15.0, 20.0, 25.0, 30.0, 35.0, 40.0, 45.0]

# Atividade radioativa [mCi]
A_i = np.array([9.676 , 6.355, 4.261, 2.729, 1.862, 1.184, 0.7680, 0.4883

# Representar dados num grafico (usando o matplotlib)
plt.scatter(t_i, A_i)
plt.xlabel("tempo, [dias]")
plt.ylabel("Atividade radioativa, [mCi]")
plt.show()
```

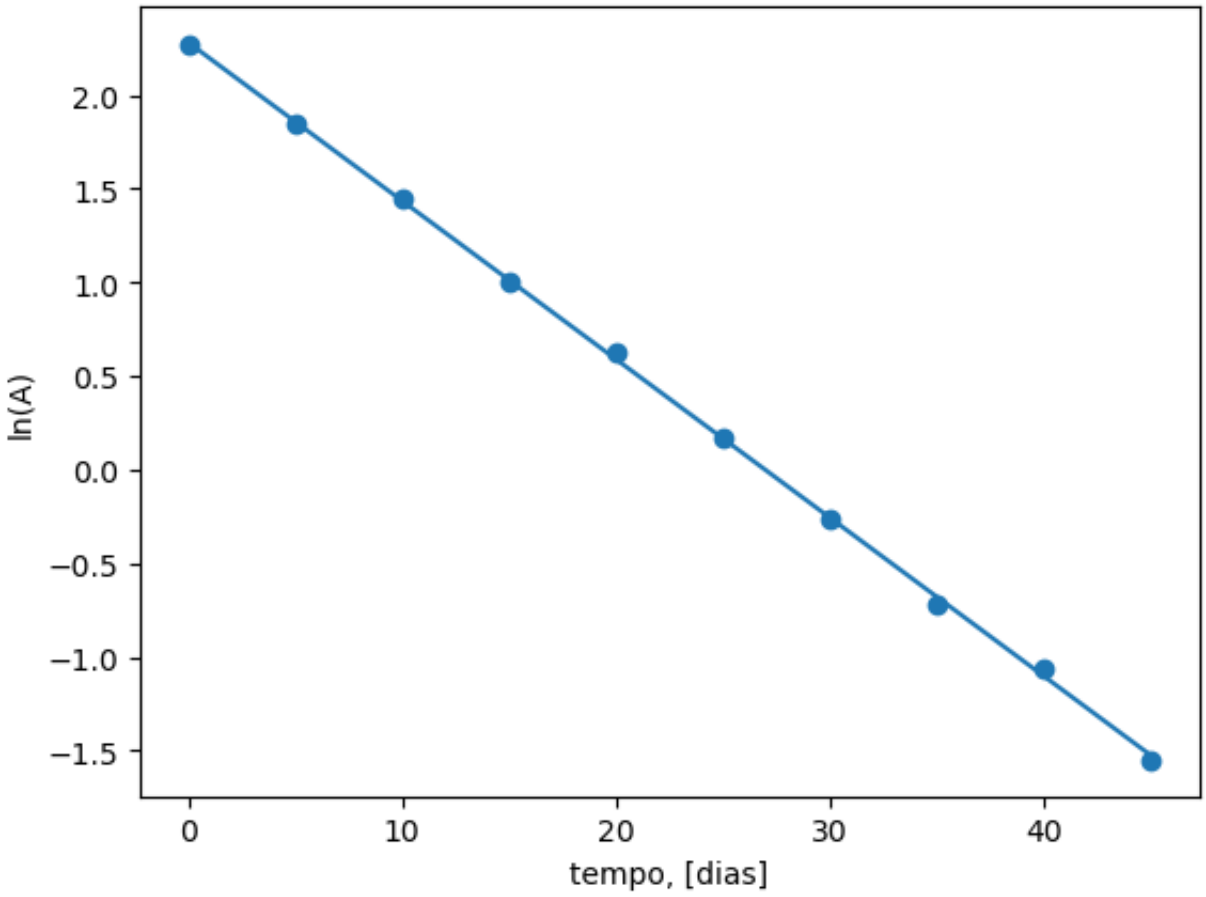


Claramente, a relação entre a atividade radioativa e o tempo decorrido **não é linear**.

b) Apresente as medições num gráfico semilog. Como depende a atividade com o tempo?

A unidade curie indica 3.7×10^{-10} desintegrações nucleares por segundo.

In [17]:
```
# Atenção à diferença entre o logaritmo natural "np.log" e o logarítmo de
X_i = t_i
Y_i = np.log(A_i)

# Função polyfit de primeira ordem, encontra os coeficientes a = [a_0, a_
# y(x) = a_0 * x + a_1
# que melhor se ajusta a um conjunto de pontos {(X_i, Y_i)}
a = np.polyfit(X_i, Y_i, 1)

# Gerar valores do tempo e espaço percorrido para representação da reta
x = np.linspace(0.0, 45.0, 2)
y = a[0] * x + a[1]

# Representar dados num grafico (usando o matplotlib)
plt.scatter(X_i, Y_i)
plt.plot(x, y)
plt.xlabel("tempo, [dias]")
plt.ylabel("ln(A)")
plt.show()
```



De facto, a relação entre a atividade radioativa e o tempo decorrido é dada por $A(t) = A_0\, \mathrm{e}^{-t/\tau}$.

$\log A = \log A_0 - t/\tau$

$Y_i = \log A_i$

$X_i = t_i$

$m = -1/\tau$

$b = \log A_0$

In [18]:
```
print("tau = -1/m = ", -1 / a[0], "dias")
```

```
tau = -1/m =  11.810754501499657 dias
```

the half-life is related to the decay constant as follows: set N = N0/2 and t = T1/2 to obtain

A metade do tempo de vida $\tau_{1/2}$ relaciona-se com a constante de decaimento τ da seguite forma: Definindo um decaimento da radioatividade para metade $A=A0/2$ quando $t=\tau_{1/2}$, obtemos:

$\tau_{1/2} = \tau \ln 2$

In [19]:
```
print("tau_1/2 = ", -1 / a[0] * np.log(2), "dias")
```

```
tau_1/2 =  8.18659118300017 dias
```

O que é próximo do valor de referência $\tau_{1/2}=8.0197~\textrm{dias}$