Home

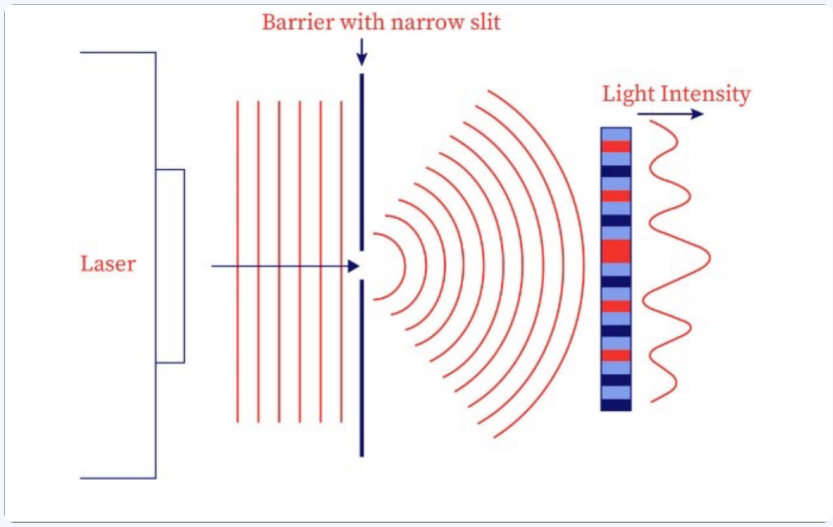
Guião Prático 2

João Capucho

```
# As seguintes librarias serão necessárias para resolver o
import numpy as np # O numpy permite efetuar cálculos de f
                   # sobre vários números (arrays)
import matplotlib.pyplot as plt # O matplotlib e o seu módu
                                # uma interface simples pa
```

Exercício 1

Numa experiência de difração de um feixe de luz por uma fenda única foram medidos 7 pares de valores (na tabela) da distância da fonte de luz ao alvo, L, e a distância entre máximos luminosos consecutivos (entre a mancha vermelha central e as outras manchas vermelhas) da figura de difração, X,



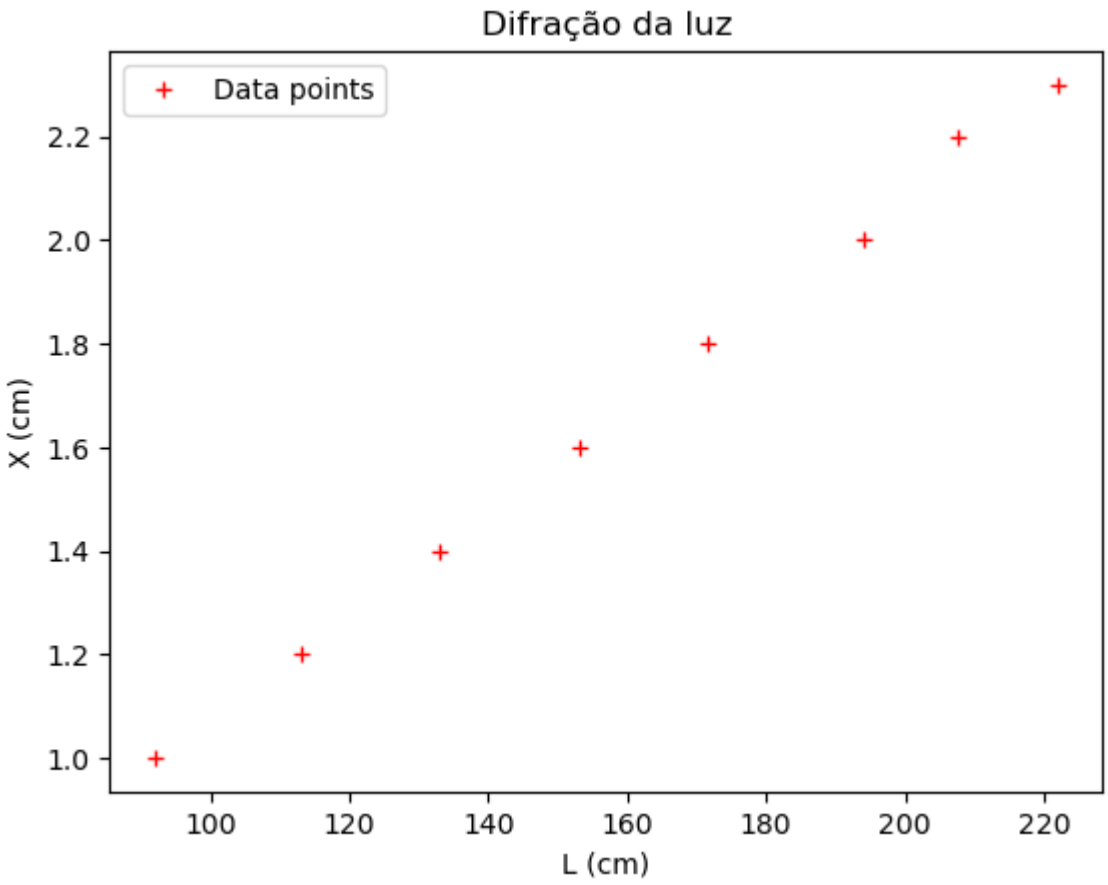
L (cm)	X (cm)
222.0	2.3
207.5	2.2
194.0	2.0
171.5	1.8
153.0	1.6
133.0	1.4
113.0	1.2
92.0	1.0

Escreva um programa em python que calcule as quantidades anteriores (valores da regressão linear).

a) Comece por representar os dados experimentais num gráfico.

```
# Convertemos os dados experimentais em dois arrays do num
# Aqui L vai ser as nossas abcissas pois é variável indepe
# que X é a variável de dependente de L (pois X é consequê
# manipulamos L)
x = np.array([222.0, 207.5, 194.0, 171.5, 153.0, 133.0, 11
y = np.array([2.3, 2.2, 2.0, 1.8, 1.6, 1.4, 1.2, 1.0]) # X

# Agora adicionamos os pontos a um novo gráfico
#
# O "r+" diz ao pyplot para desenhar os nossos valores a v
# e representá-los com um sinal de mais (+).
# O label dá um nome a linha que desenhamos, este nome pod
# ser apresentado no gráfico
plt.plot(x, y, "r+", label="Data points")
# Adicionamos as legendas do eixos
plt.xlabel("L (cm)")
plt.ylabel("X (cm)")
# Este comando diz ao pyplot para mostrar as labels da nos
# no canto superior esquerdo
plt.legend(loc="upper left")
# Este comando adiciona um título ao gráfico
plt.title("Difração da luz")
# Apresentamos o gráfico
plt.show()
```



b) Calcular as somas das expressões acima.

```
# np.multiply calcula a multiplicação elemento a elemento (
# retornando um array com os resultados, np.sum soma todos
# de um array e retorna o número que resulta disso.
mul_sum = np.sum(np.multiply(x, y))
x_sum = np.sum(x)
y_sum = np.sum(y)
# np.square calcula o quadrado para cada elemento do array
# um novo array com os resultados.
x2_sum = np.sum(np.square(x))
x_sum2 = np.square(np.sum(x))
```

$$\sum_{i=1}^{N} x_i y_i = 2322.4$$

$$\sum_{i=1}^{N} x_i = 1286.0$$

$$\sum_{i=1}^{N} y_i = 13.5$$

$$\sum_{i=1}^{N} x_i^2 = 221719.5$$

$$\left(\sum_{i=1}^{N} x_i\right)^2 = 1653796.0$$

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

```
# np.size dá o número de elementos que um array contém, tar
# utilizar a função len do python aqui (ex. len(x)).
data_points = np.size(x)

# Começamos por calcular o numerador do declive, esta quant
# ser necessária para calcular outros valores por isso guar
# variável de como a não ter de a recalcular.
m_numerator = data_points * mul_sum - x_sum * y_sum
# O denominador do declive também vai ser reutilizado.
x_denom = data_points * x2_sum - x_sum2
# Agora que temos o numerador e o denominador apenas preci
# dividir para obter o declive.
m = m_numerator / x_denom

# Calculamos agora a ordenada na origem.
b = (x2_sum * y_sum - x_sum * mul_sum) / x_denom

# O coeficiente de correlação necessita também de uma quant
# se calcula do mesmo modo que o denominador do declive só
# vez de x.
y2_sum = np.sum(np.square(y))
y_sum2 = np.square(np.sum(y))
y_denom = data_points * y2_sum - y_sum2

# Calculamos agora o coeficiente de determinação.
r2 = m_numerator**2 / (x_denom * y_denom)
```

$$m = 0.01015505$$

$$b = 0.05507544$$

$$r^2 = 0.99845714$$

A seguir calculamos também o valor do erro tanto para o declive como para a ordenada na origem

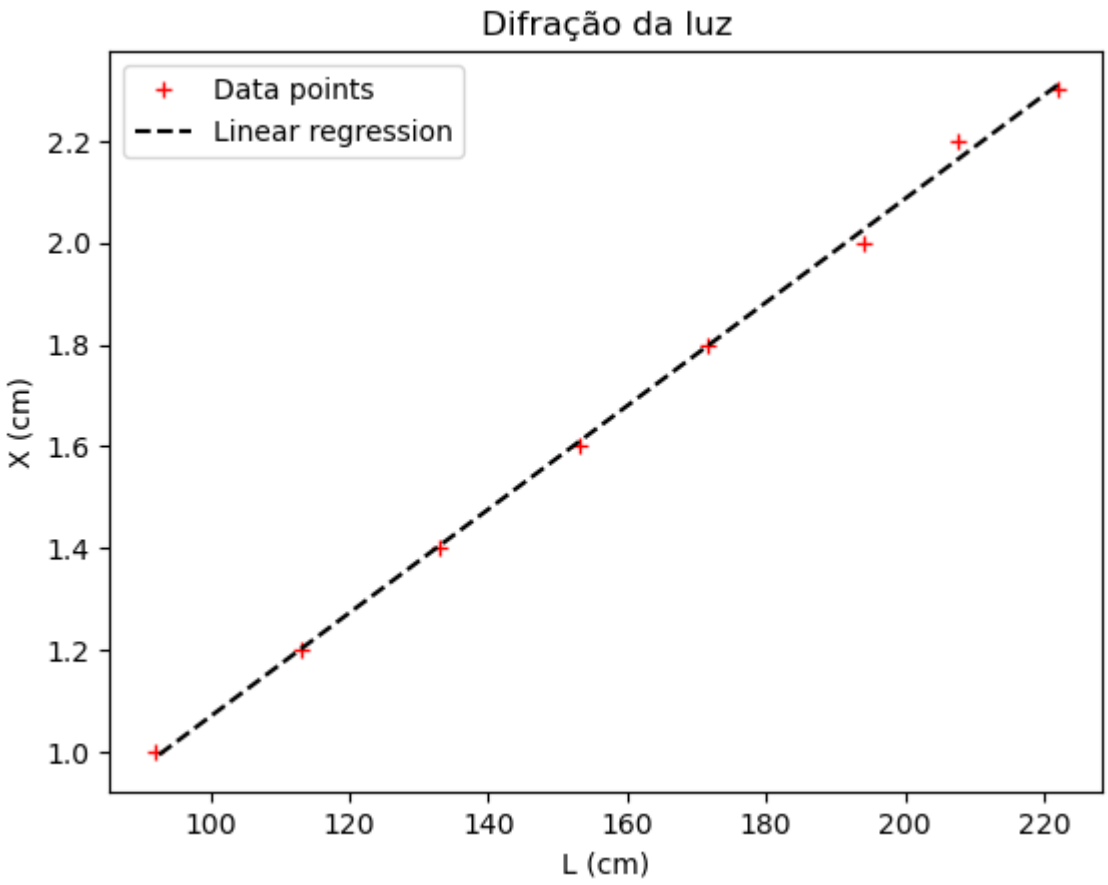
```
# Começamos por calcular o error do declive
# O np.absolute retorna o valor absoluto do número inserid
# calcula e retorna a raiz quadrada do número inserido.
delta_m = np.absolute(m) * np.sqrt((1 / r2 - 1) / (data_po
# Concluímos calculando o erro da ordenada na origem
delta_b = delta_m * np.sqrt(x2_sum / data_points)
```

$$\Delta m = 0.00016297$$

$$\Delta b = 0.02713077$$

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

```
# Voltamos a definir o gráfico como fizemos anteriormente
plt.plot(x, y, "r+", label="Data points")
# Mas agora adicionamos outra linha para a regressão linea
# O formato da reta de regressão linear é mx+b
# "--k" Significa desenhar a tracejado com linha preta
plt.plot(x, m * x + b, "--k", label="Linear regression")
plt.xlabel("L (cm)")
plt.ylabel("X (cm)")
plt.legend(loc="upper left")
plt.title("Difração da luz")
plt.show()
```



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

```
# A reta da regressão linear é definida como
# y = mx+b, para calcular um valor de y para
# um dado x basta substituir na expressão
X = m * 165 + b
```

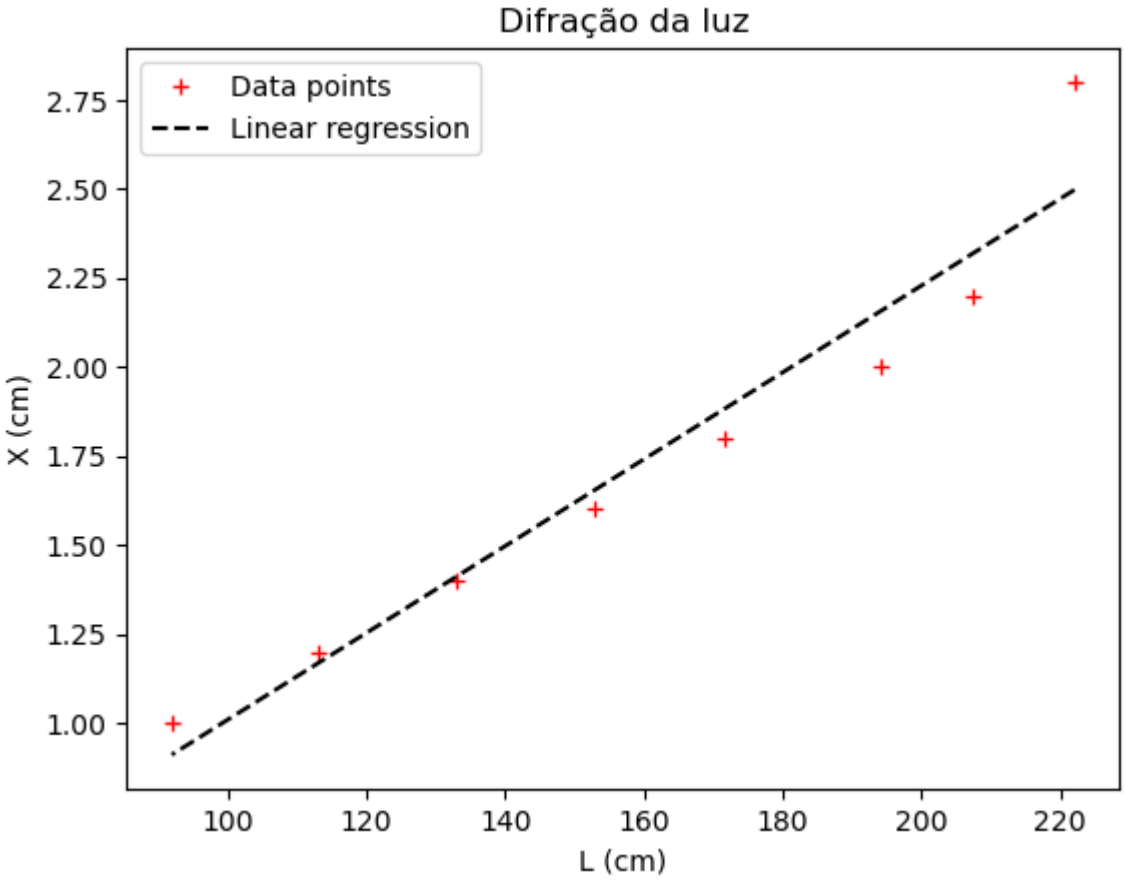
$$X = 1.73\text{cm}$$

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

```
# Os mesmos valores de y exceto o primeiro que foi alterad
y = np.array([2.8, 2.2, 2.0, 1.8, 1.6, 1.4, 1.2, 1.0]) # X

# Voltamos a calcular a regressão linear

# Voltamos a desenhar o gráfico
```



$$r^2 = 0.99845714$$

$$r^2 = 0.93735475 \, (\text{novo})$$

Como podemos observar a nova reta apresenta um coeficiente de correlação (r^2) menor do que o da primeira reta, o que resulta na reta ficar mais distante de alguns pontos experimentais.

Exercício 2

Um ciclista tenta percorrer a velocidade constante (uniforme) uma distância de 10 km. O seu treinador nos primeiros 9 minutos e a cada minuto mede a distância percorrida, e regista os valores em km:

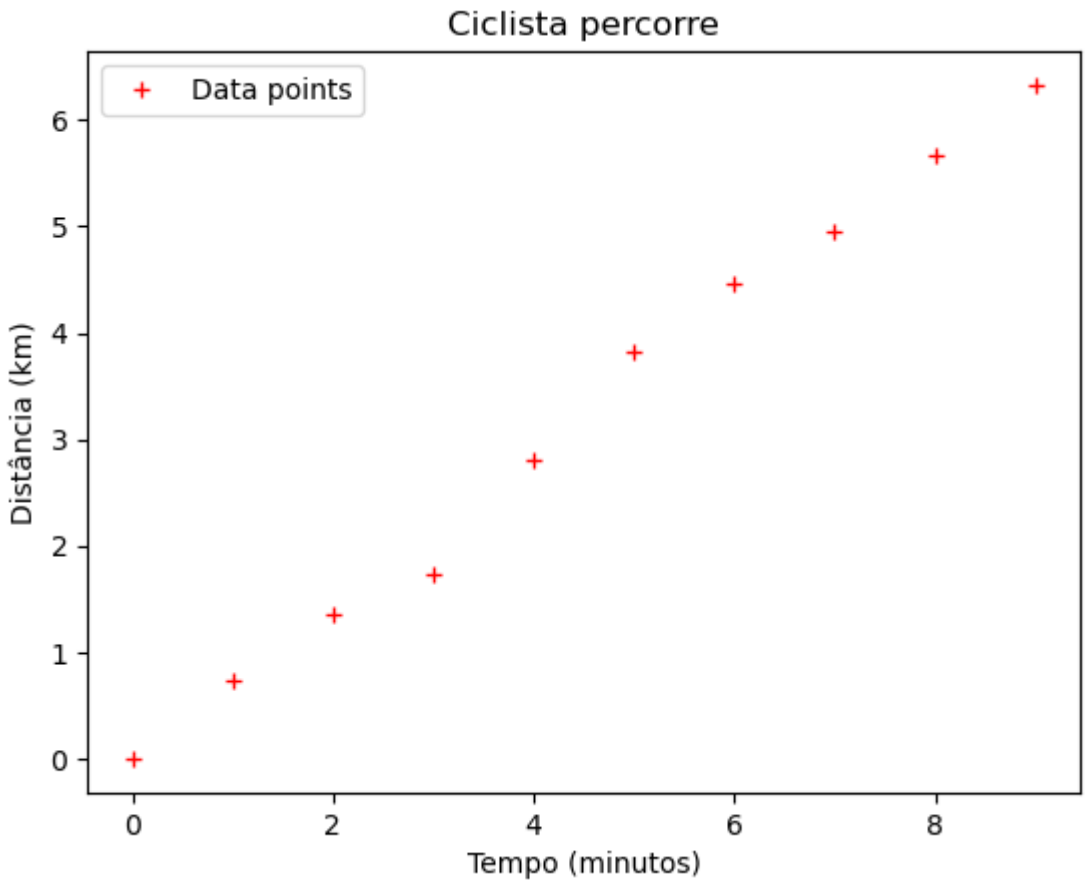
0.00	0.735	1.363	1.739	2.805	3.814	4.458	4.955	5.666	6.329

Antes de mais vamos passar estes dados para python

```
# np.arange(start, end, step é semelhante a função range d
# esta função cria um array com o primeiro valor igual a s
# os restantes ate end (exclusivo) sendo o valor anterior
x = np.arange(0, 10, 1) # [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
y = np.array([0.00, 0.735, 1.363, 1.739, 2.805, 3.814, 4.4
```

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

```
plt.plot(x, y, "r+", label="Data points")
plt.xlabel("Tempo (minutos)")
plt.ylabel("Distância (km)")
plt.legend(loc="upper left")
plt.title("Ciclista percorre")
plt.show()
```



A relação entre o tempo e a distância percorrida aparenta ser linear

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

É uma relação linear bem aproximada? O ciclista conseguiu manter a mesma velocidade uniforme durante o percurso?

```
# Calculamos a regressão linear
```

$$m = 0.71881212$$

$$b = -0.04825455$$

$$r^2 = 0.99384774$$

Como o coeficiente de determinação é tão alto podemos assumir que é uma relação linear bem aproximada e que o ciclista consegui manter uma velocidade mais ao menos uniforme durante todo o percurso.

c) Qual a velocidade média do ciclista?

Se assumirmos uma velocidade uniforme (o que fazemos quando calculamos uma velocidade média) então a posição do ciclista é dada pela função $y = vt$, onde v é a velocidade média e t é o tempo, logo a velocidade média do ciclista pode ser extraída do declive da reta de regressão linear, sendo assim a sua velocidade média $\approx 0.71881212 \text{ km/min}$

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)?

```
# np.polyfit(x, y, deg) tenta calcular o polinómio de grau
# que mais se aproxima dos dados experimentais passados.
# Como no enunciado é pedido para "encontrar a reta que ma
# aproxima" sabemos que o grau do polinómio é 1.
(m, b) = np.polyfit(x, y, 1)
```

$$m = 0.71881212$$

$$b = -0.04825455$$

Os valores são exatamente iguais aos valores que calculamos anteriormente (esperado).

e) Apresente a velocidade em km/hora.

$$0.71881212\ \ (\mathrm{km/min}) * 60 = 43.1287272\ \ (\mathrm{km/h})$$

Exercício 3

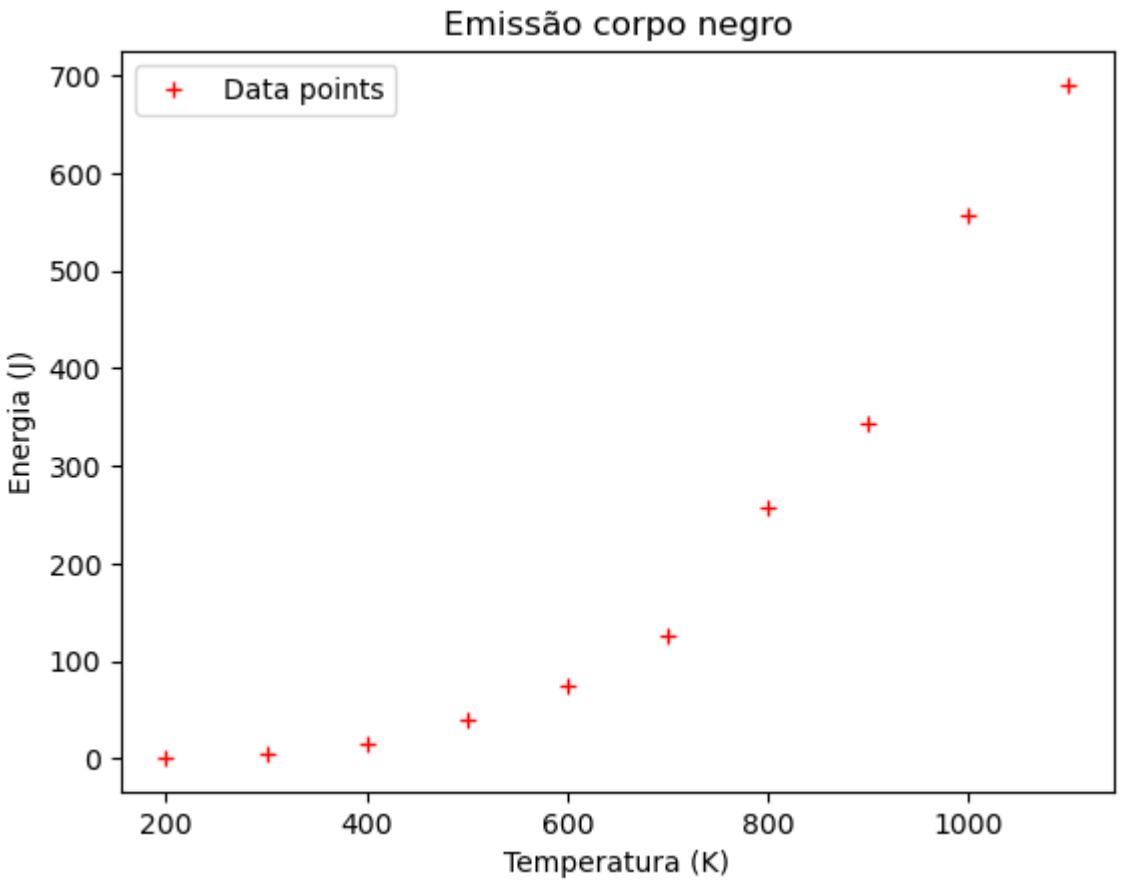
Foi medida a energia por segundo (potência) emitida por um corpo negro (corpo que absorve toda a energia que incide nele) de área $100\ \mathrm{cm}^2$ em função da temperatura absoluta, T, e registada na seguinte tabela

T(K)	200	300	400	500	600	700	800	900	1000	1100
E(J)	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?

```
x = np.array([200, 300, 400, 500, 600, 700, 800, 900, 1000
y = np.array(
    [0.6950, 4.363, 15.53, 38.74, 75.08, 125.2, 257.9, 344
)
```

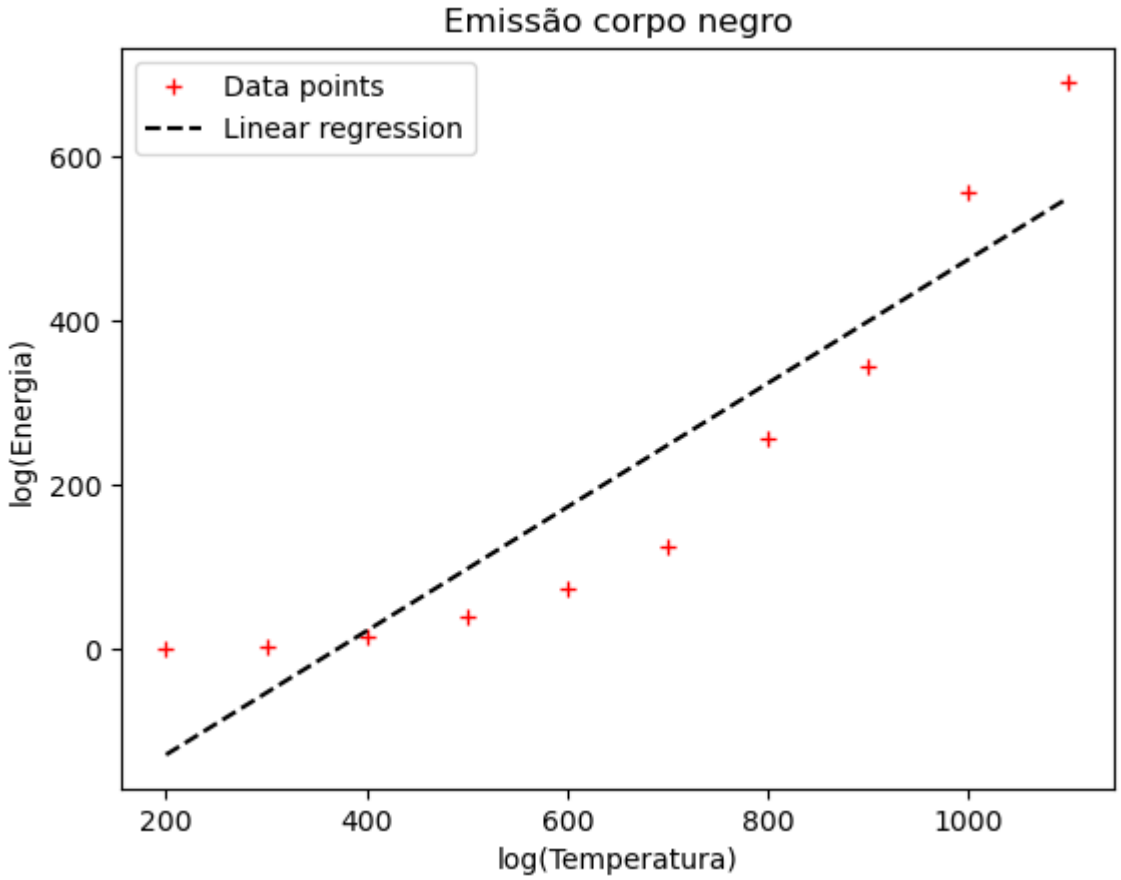
```
plt.plot(x, y, "r+", label="Data points")
plt.xlabel("Temperatura (K)")
plt.ylabel("Energia (J)")
plt.legend(loc="upper left")
plt.title("Emissão corpo negro")
plt.show()
```



Para determinarmos se a relação entre a energia emitida e a temperatura é linear, podemos calcular o coeficiente de determinação da regressão linear.

$$r^2 = 0.85047828$$

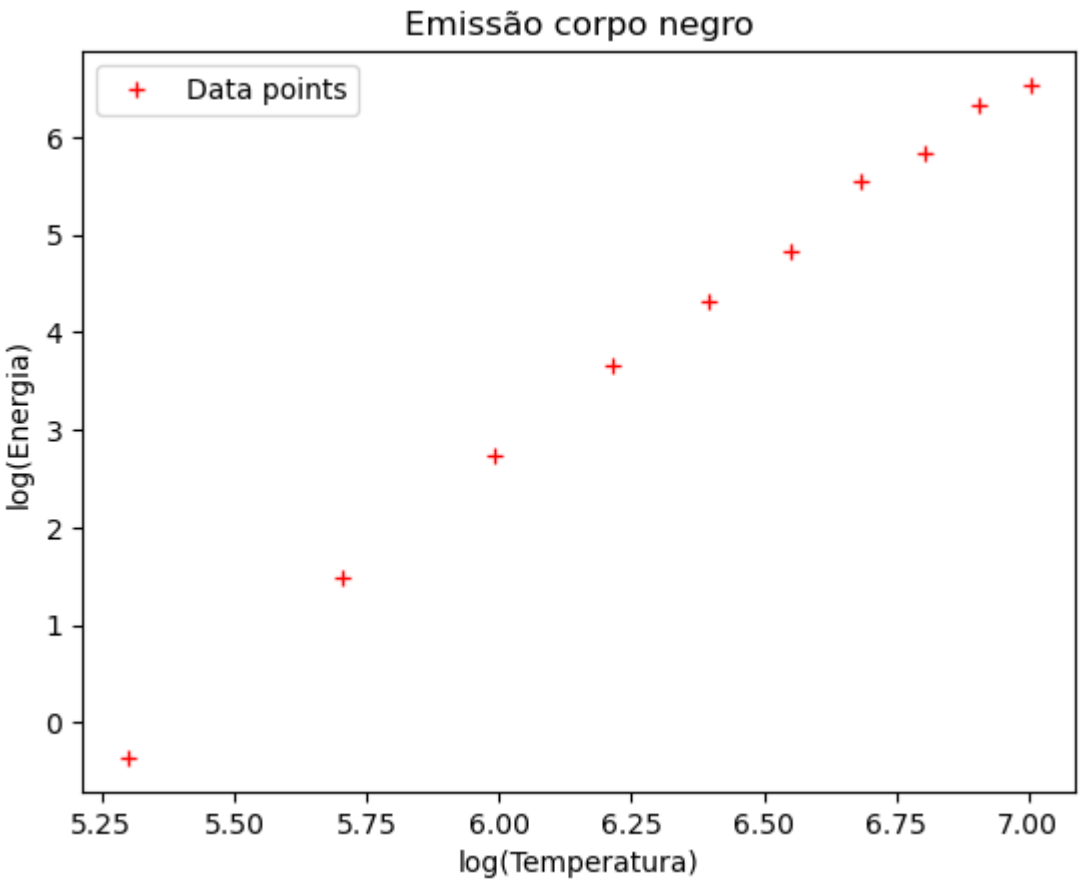
O valor do coeficiente indica que a relação não aparente ser linear, podemos verificar isto observando o gráfico da regressão linear e notando que a reta diverge em muito dos pontos.



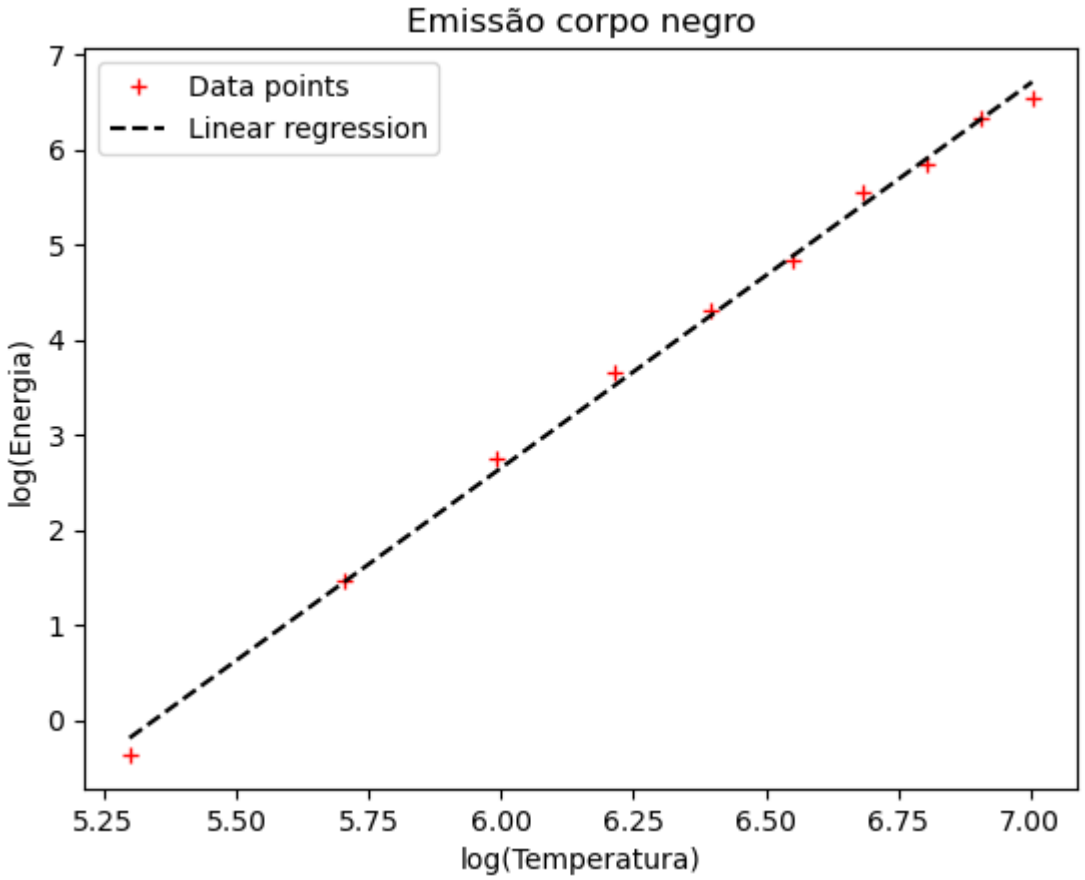
b) Apresente as medições num gráfico log-log. Qual a dependência entre as quantidade energia emitida e a temperatura?

```
logx = np.log(x)
logy = np.log(y)

plt.plot(logx, logy, "r+", label="Data points")
plt.xlabel("log(Temperatura)")
plt.ylabel("log(Energia)")
plt.legend(loc="upper left")
plt.title("Emissão corpo negro")
plt.show()
```



A relação entre os logaritmos dos valores experimentais aparenta ser linear, podemos verificar isto olhando para o gráfico da regressão linear e o seu coeficiente de determinação.



$$r^2 = 0.99730522$$

Verifica-se que o gráfico log-log apresenta uma relação linear, logo o gráfico original terá uma relação de potência $y = cx^n$, estes valores podem ser calculados da seguinte forma:

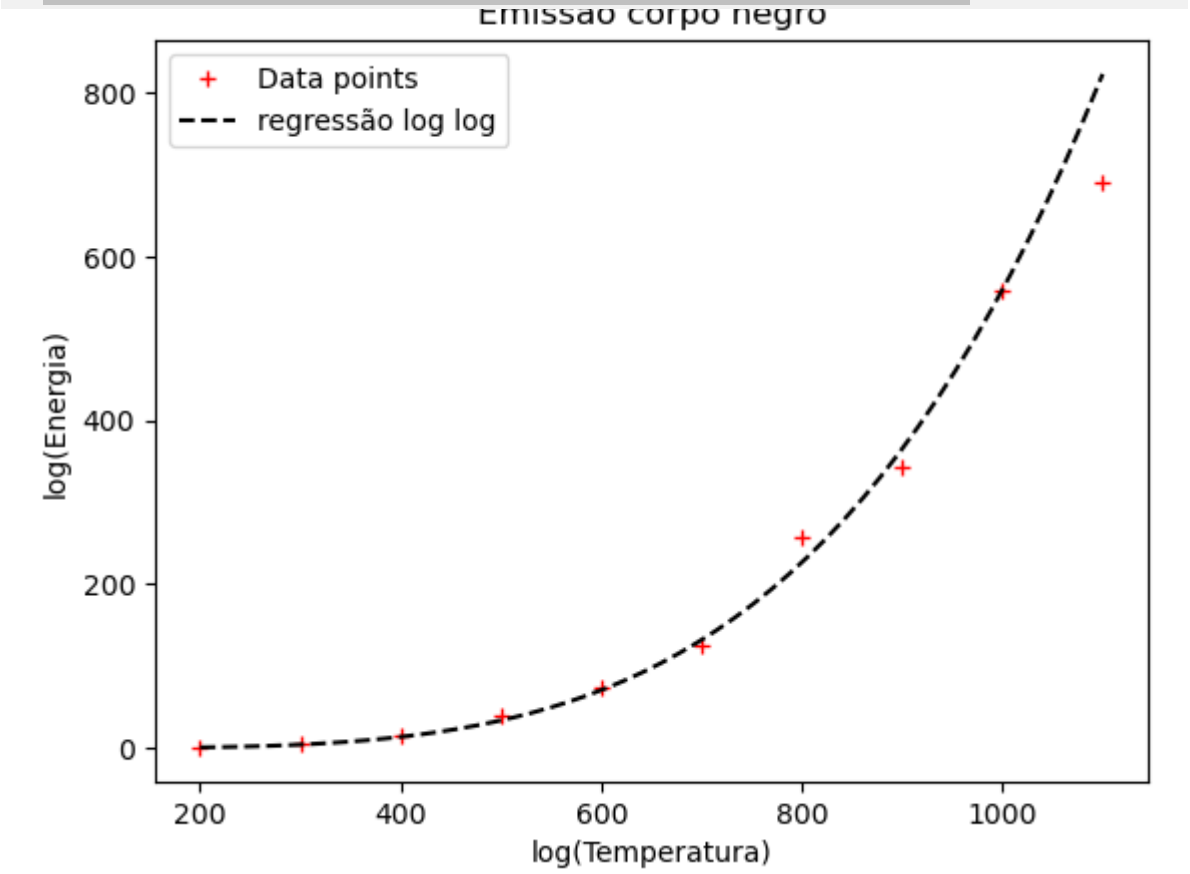
$$c = e^b \quad n = m$$

Onde b e m são a ordenada na origem e o declive da regressão linear do gráfico log-log.

```
# np.exp calcula o exponencial do valor (e**b) passado
c = np.exp(b)
n = m

plt.plot(x, y, "r+", label="Data points")
# np.linspace(min, max, [points]) cria um array com `point
# elementos que estão espaçados igualmente entre si desde
# `min` até `max`.
# Utilizamos isto porque o pyplot interpola linearmente os
# valores para fazer os gráficos, no entanto dados os pouc
# dados que temos, isto faria com que o gráfico da potênci
```

```
# ficasse com muitos segmentos visíveis.
x_fitted = np.linspace(np.min(x), np.max(x))
plt.plot(x_fitted, c * x_fitted**n, "--k", label="regressã
plt.xlabel("log(Temperatura)")
plt.ylabel("log(Energia)")
plt.legend(loc="upper left")
plt.title("Emissão corpo negro")
plt.show()
```



Logo a dependência entre a energia emitida e a temperatura é a seguinte (onde x é a temperatura e y é a energia emitida):

$$y = cx^n = 0.0000000004x^{4.04826952}$$

Exercício 4

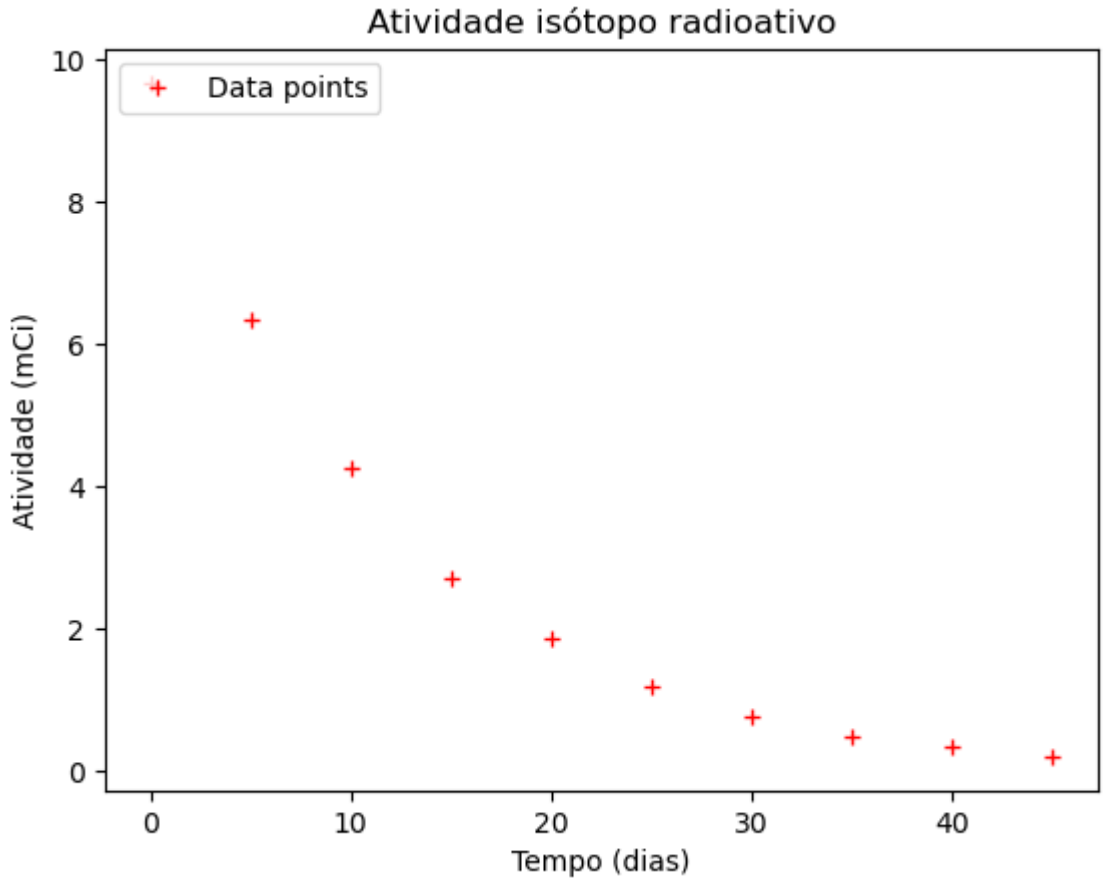
Foi medida a atividade de uma amostra do isótopo radioativo $^{131}\mathrm{I}$ tem de 5 em 5 dias. Os valores medidos da atividade com o tempo são, em mCi:

9.676	6.355	4.261	2.729	1.862	1.184	0.7680	0.4883	0.3461	0.2119

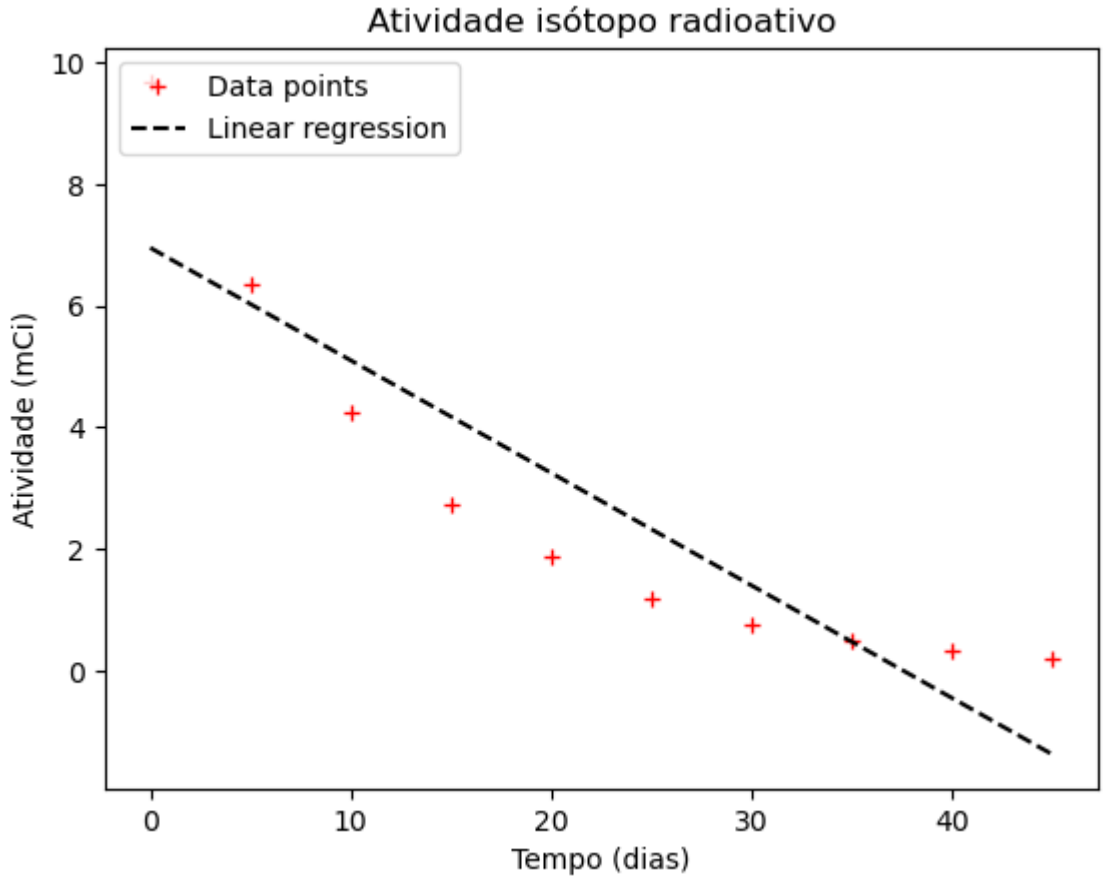
a) Apresente estas medições num gráfico. A analisar o gráfico, a relação entre a atividade e o tempo é linear?

```
t = np.arange(0, 50, 5)
y = np.array(
    [9.676, 6.355, 4.261, 2.729, 1.862, 1.184, 0.7680, 0.4
)
```

```
plt.plot(t, y, "r+", label="Data points")
plt.xlabel("Tempo (dias)")
plt.ylabel("Atividade (mCi)")
plt.legend(loc="upper left")
plt.title("Atividade isótopo radioativo")
plt.show()
```



Calculando agora a regressão linear e o seu coeficiente de determinação.



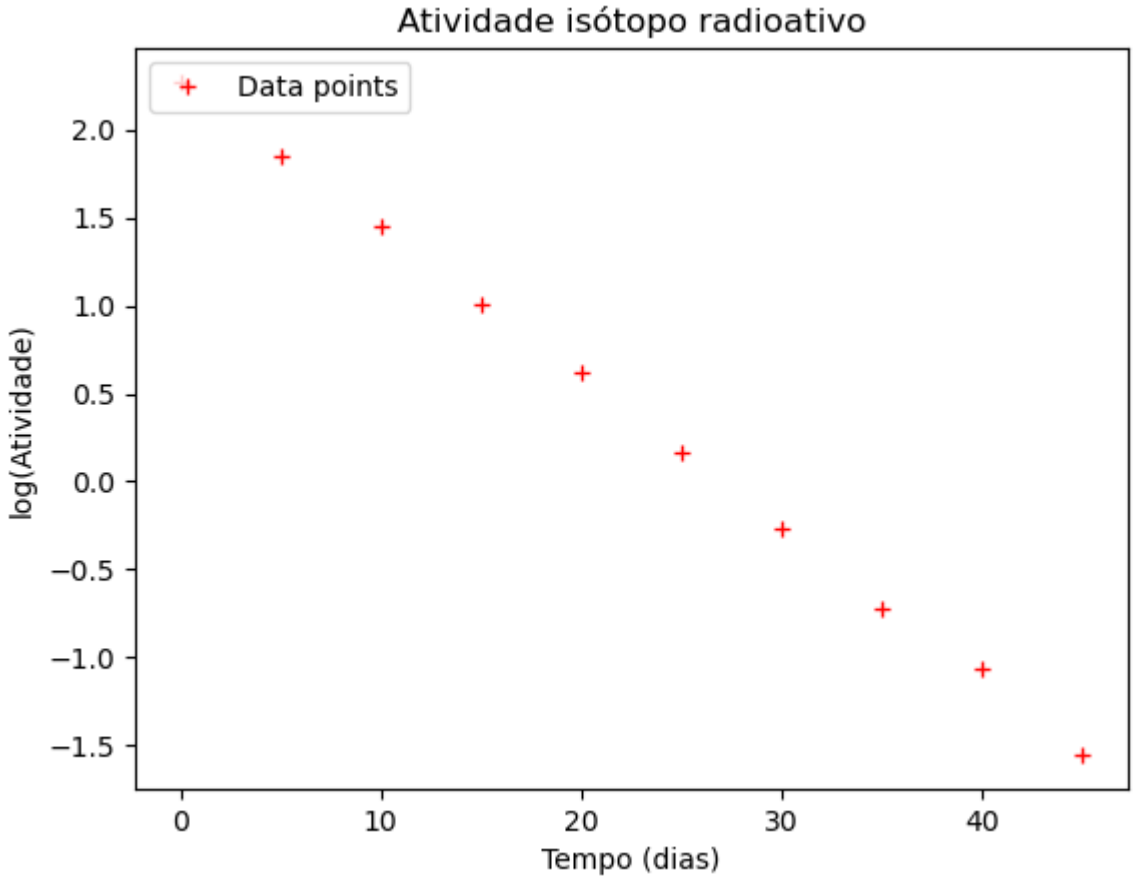
$$r^2 = 0.80493604$$

Como podemos ver a relação entre a atividade e o tempo é não linear.

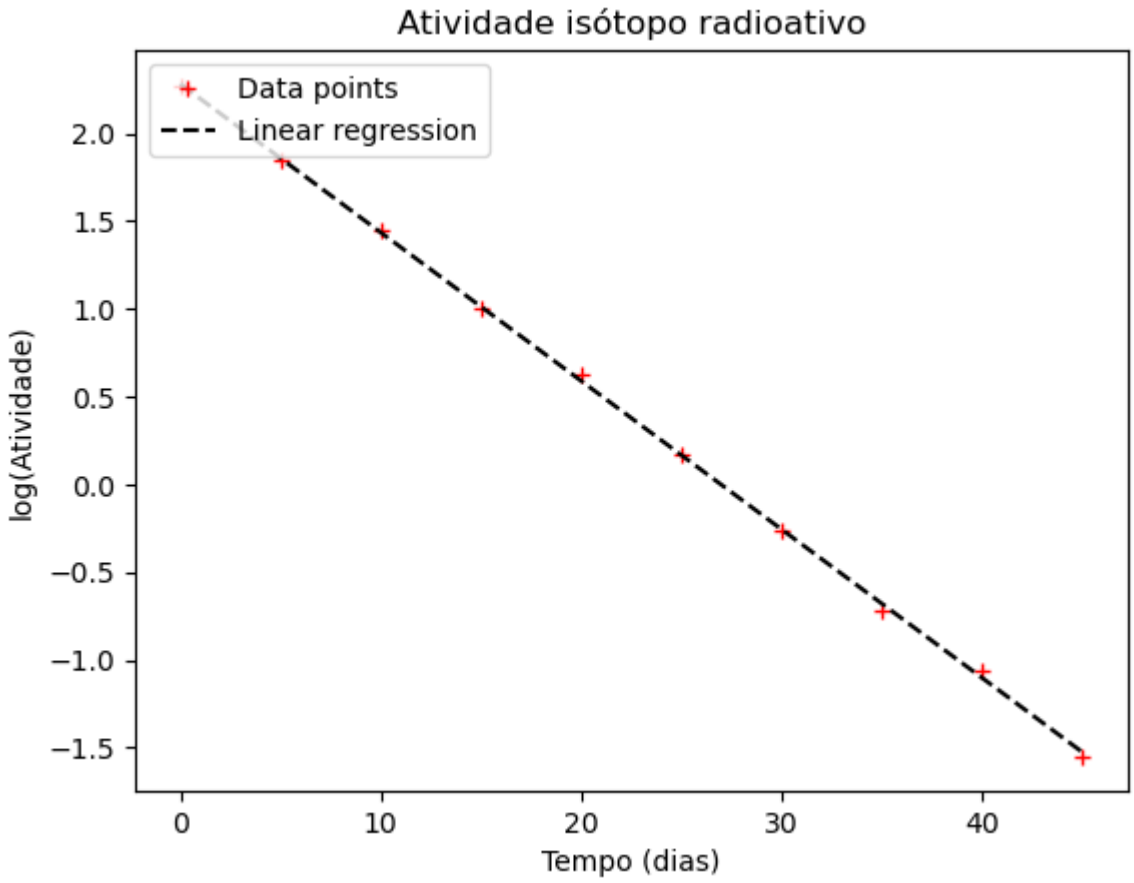
b) Apresente as medições num gráfico semilog. Como depende a atividade com o tempo?

```
logy = np.log(y)

plt.plot(t, logy, "r+", label="Data points")
plt.xlabel("Tempo (dias)")
plt.ylabel("log(Atividade)")
plt.legend(loc="upper left")
plt.title("Atividade isótopo radioativo")
plt.show()
```



O gráfico semilog parece estabelecer uma relação linear entre os seus eixos, mas podemos confirmar isto calculando a regressão linear e o seu coeficiente de determinação.



$$r^2 = 0.99963718$$

Verifica-se que o gráfico semilog apresenta uma relação linear, logo o gráfico original terá uma relação exponencial do tipo $y = y_0 e^{\lambda t}$, estes valores podem ser calculados da seguinte forma:

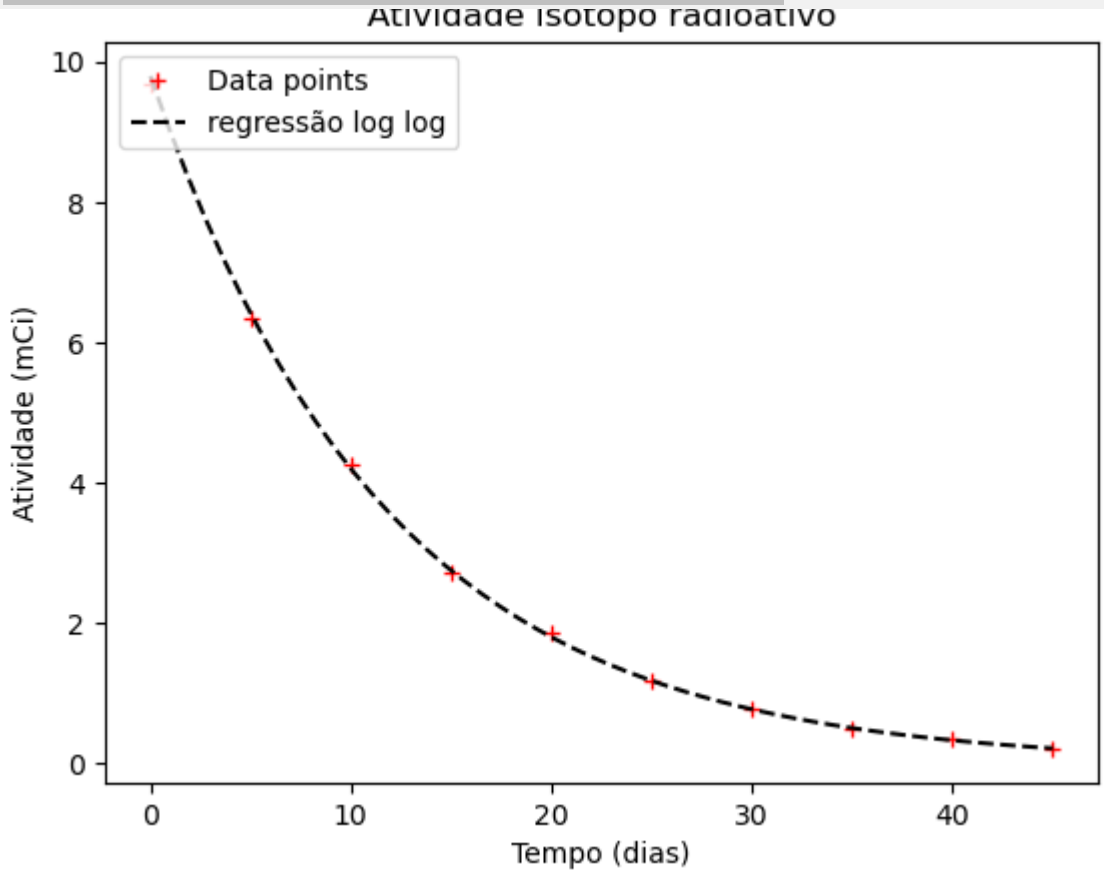
$$y_0 = e^b \quad \lambda = m$$

Onde b e m são a ordenada na origem e o declive da regressão linear do gráfico semilog.

```
y_0 = np.exp(b)
lam = m

plt.plot(t, y, "r+", label="Data points")
t_fitted = np.linspace(np.min(t), np.max(t))
plt.plot(t_fitted, y_0 * np.exp(lam * t_fitted), "--k", la
plt.xlabel("Tempo (dias)")
plt.ylabel("Atividade (mCi)")
plt.legend(loc="upper left")
```

```
plt.title("Atividade isótopo radioativo")
plt.show()
```



Logo a dependência entre a atividade e o tempo é a seguinte (onde t é o tempo e y é a atividade):

$$y = y_0 e^{\lambda t} = 9.79611052 e^{-0.08466860 t}$$