Exercicio 1: O menor grau para encontrar um ponto de inflexao é o grau 3, logo seguimos para resolução. Primeiro armazenamos em lista, os valores de f e os valores de t

```
import numpy as np
f = [15, 23, 33, 45, 58, 69, 79, 86]
t = np.linspace(1,8,8)
f,t
```

```
([15, 23, 33, 45, 58, 69, 79, 86], array([1., 2., 3., 4., 5., 6., 7., 8.]))
```

criamos os vetores g0(t),g1(t),g2(t),g3(t) que vao ser as linhas da matriz g abaixo.

```
g = [[1 for i in range(len(t))],[t[i] for i in range(len(t))],[t[i]**2 for i in range(len(
g
```

```
[[1, 1, 1, 1, 1, 1, 1, 1],
 [1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0],
 [1.0, 4.0, 9.0, 16.0, 25.0, 36.0, 49.0, 64.0],
 [1.0, 8.0, 27.0, 64.0, 125.0, 216.0, 343.0, 512.0]]
```

agora precisamos montar um sistema linear normal, criando o vetor fg, em que os elementos sao o produto interno de f com g0,g1,g2 e g3.

```
gf = [np.dot(g[0],f),np.dot(g[1],f),np.dot(g[2],f),np.dot(g[3],f)]
gf
```

```
[408, 2285.0, 14433.0, 97253.0]
```

montamos tambem a matriz A com o formato $\begin{bmatrix} \langle g_0, g_0 \rangle & \langle g_0, g_1 \rangle & \cdots & \langle g_0, g_m \rangle \\ \langle g_1, g_0 \rangle & \langle g_1, g_1 \rangle & \cdots & \langle g_1, g_m \rangle \\ \vdots & \vdots & \vdots & \vdots \\ \langle g_m, g_0 \rangle & \langle g_m, g_1 \rangle & \cdots & \langle g_m, g_m \rangle \end{bmatrix}$

```
import numpy as np
A = [[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0]]
A[0][0] =np.dot(g[0],g[0])
A[0][1] =np.dot(g[0],g[1])
A[0][2] =np.dot(g[0],g[2])
A[0][3] =np.dot(g[0],g[3])
A[1][0] = A[0][1]
A[1][1] = np.dot(g[1],g[1])
A[1][2] = np.dot(g[1],g[2])
A[1][3] = np.dot(g[1],g[3])
A[2][0] = A[0][2]
A[2][1] = A[1][2]
A[2][2] = np.dot(g[2],g[2])
A[2][3] = np.dot(g[2],g[3])
```

```
A[2][3] = np.dot(g[2],g[3])
A[3][0] = A[0][3]
A[3][1] = A[1][3]
A[3][2] = A[2][3]
A[3][3] = np.dot(g[3],g[3])
A
```

```
[[8, 36.0, 204.0, 1296.0],
 [36.0, 204.0, 1296.0, 8772.0],
 [204.0, 1296.0, 8772.0, 61776.0],
 [1296.0, 8772.0, 61776.0, 446964.0]]
```

Usando um comando do numpy, resolvemos o sistema linear tal que Aa = gf

```
a = [0,0,0,0]
a[0],a[1],a[2],a[3] = np.linalg.solve(A,gf)
a
```

```
[11.071428571426456,
 1.7095959595981345,
 2.4512987012981404,
 -0.18686868686864588]
```

Logo temos o polinomio
$$11.071428571426456 + 1.7095959595981345x + 2.4512987012981404x^2 - 0.186868$$

podemos por no grafico esse polinomio junto com os pontos dados no enunciado atraves da biblioteca matplotlib obs: a gnt cria varios pontos de x de 0 ate 8 para ter um grande arsenal de pontos

```
import matplotlib.pyplot as plt

x = np.linspace(0,8,1000)
y = a[0]+a[1]*x+a[2]*x**2+a[3]*x**3
plt.xlabel("t")
plt.ylabel("f")
plt.plot(x,y)
plt.scatter(t,f)
plt.show()
```

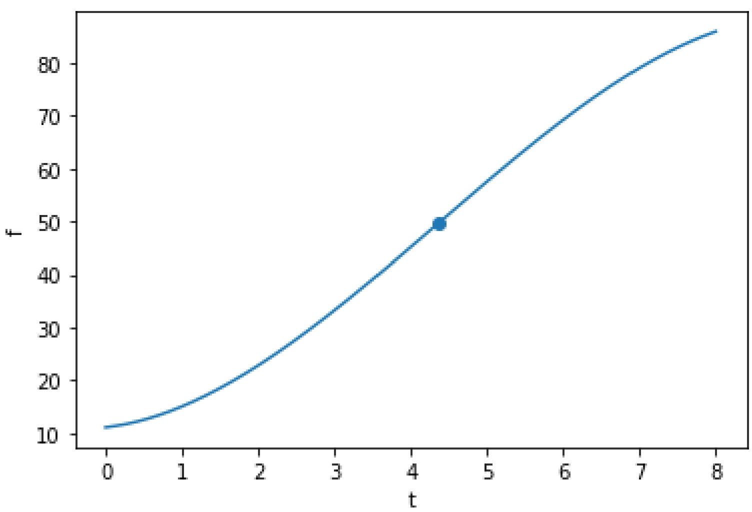
Agora precisamos encontrar o ponto de inflexao da curva, para isso basta igualar a segunda derivada a 0: Como a[0] e a[1] somem na segunda derivada, apenas aplicamos o calculo:

```
m = a[3]*3*2
n = a[2] *2
ponto_de_inflexao_x = -(n/m)
ponto_de_inflexao_y = a[0]+a[1]*ponto_de_inflexao_x+a[2]*ponto_de_inflexao_x**2+a[3]*ponto
print('ponto de inflexão é: (',ponto_de_inflexao_x,',',ponto_de_inflexao_y,')')
```

```
ponto de inflexão é: ( 4.372586872586831 , 49.79188184717652 )
```

podemos botar ele no grafico para uma melhor visualização, em que o ponto marcado é o proprio ponto de inflexao da curva

```
plt.plot(x,y)
plt.xlabel("t")
plt.ylabel("f")
plt.scatter(ponto_de_inflexao_x,ponto_de_inflexao_y)
plt.show()
```



Exercicio 2: Primeiro precisamos manipular a função, de modo que ela se torne uma função linear (a+bx), logo fazemos a seguinte manipulação:
$f(t) = \frac{100}{1+ae^{-\beta t}} \Rightarrow \frac{100}{f(t)} = 1 + ae^{-\beta t} \Rightarrow \frac{100}{f(t)} - 1 = ae^{-\beta t} \Rightarrow ln(\frac{100}{f(t)} - 1) = ln(ae^{-\beta t}) \Rightarrow$
Assim fazemos a transformação $a = e^{Alfa}$, $\beta = -b$ e $F(t) = ln(\frac{100}{f(t)} - 1)$ e obtemos a funcao: $F(t) = Alfa + bx$ Logo vamos para os codigos:

```
import math
import numpy as np
f = [15, 23, 33, 45, 58, 69, 79, 86]
```

```
t = np.linspace(1,8,8)
F = [math.log((100/f[t]) - 1)for t in range(8)]
F
```

```
[1.7346010553881064,
 1.208311205924534,
 0.7081850579244858,
 0.20067069546215124,
 -0.3227733922630512,
 -0.8001193001121132,
 -1.3249254147435983,
 -1.8152899666382487]
```

Os proximos passos sao basicamente iguais ao exercicio passado o tamanho de g é menor porque estamos trabalhando apenas com 2 coeficientes

```
g = [[1 for i in range(len(t))],[t[i] for i in range(len(t))]]
g
```

```
[[1, 1, 1, 1, 1, 1, 1, 1], [1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0]]
```

Agora criamos o vetor GF para montar um sistema linear

```
gF = [np.dot(g[0],F),np.dot(g[1],F)]
gF
```

```
[-0.4113400590577334, -23.132918975439875]
```

montamos tambem a matriz A com o formato $\begin{bmatrix} \langle g_0, g_0 \rangle & \langle g_0, g_1 \rangle & \cdots & \langle g_0, g_m \rangle \\ \langle g_1, g_0 \rangle & \langle g_1, g_1 \rangle & \cdots & \langle g_1, g_m \rangle \\ \vdots & \vdots & \vdots & \vdots \\ \langle g_m, g_0 \rangle & \langle g_m, g_1 \rangle & \cdots & \langle g_m, g_m \rangle \end{bmatrix}$

```
import numpy as np
A = [[0,0],[0,0]]
A[0][0] =np.dot(g[0],g[0])
A[0][1] =np.dot(g[0],g[1])
A[1][0] = A[0][1]
A[1][1] = np.dot(g[1],g[1])
A
```

```
[[8, 36.0], [36.0, 204.0]]
```

Usando um comando do numpy, resolvemos o sistema linear tal que Ac = gF em que c[0] e c[1] sao Alfa e b respectivamente

```
c = np.linalg.solve(A,gF)
```

```
Alfa = c[0]
b = c[1]
Alfa, b
```

```
(2.228784854369221, -0.5067116359447639)
```

substituindo agora nas variáveis, temos

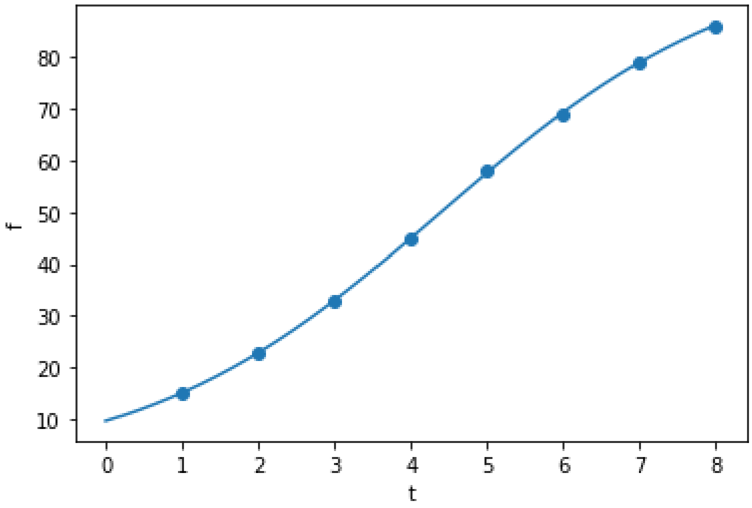
```
import math
a = math.e**Alfa
Beta = -b
a, Beta
```

```
(9.288572251063083, 0.5067116359447639)
```

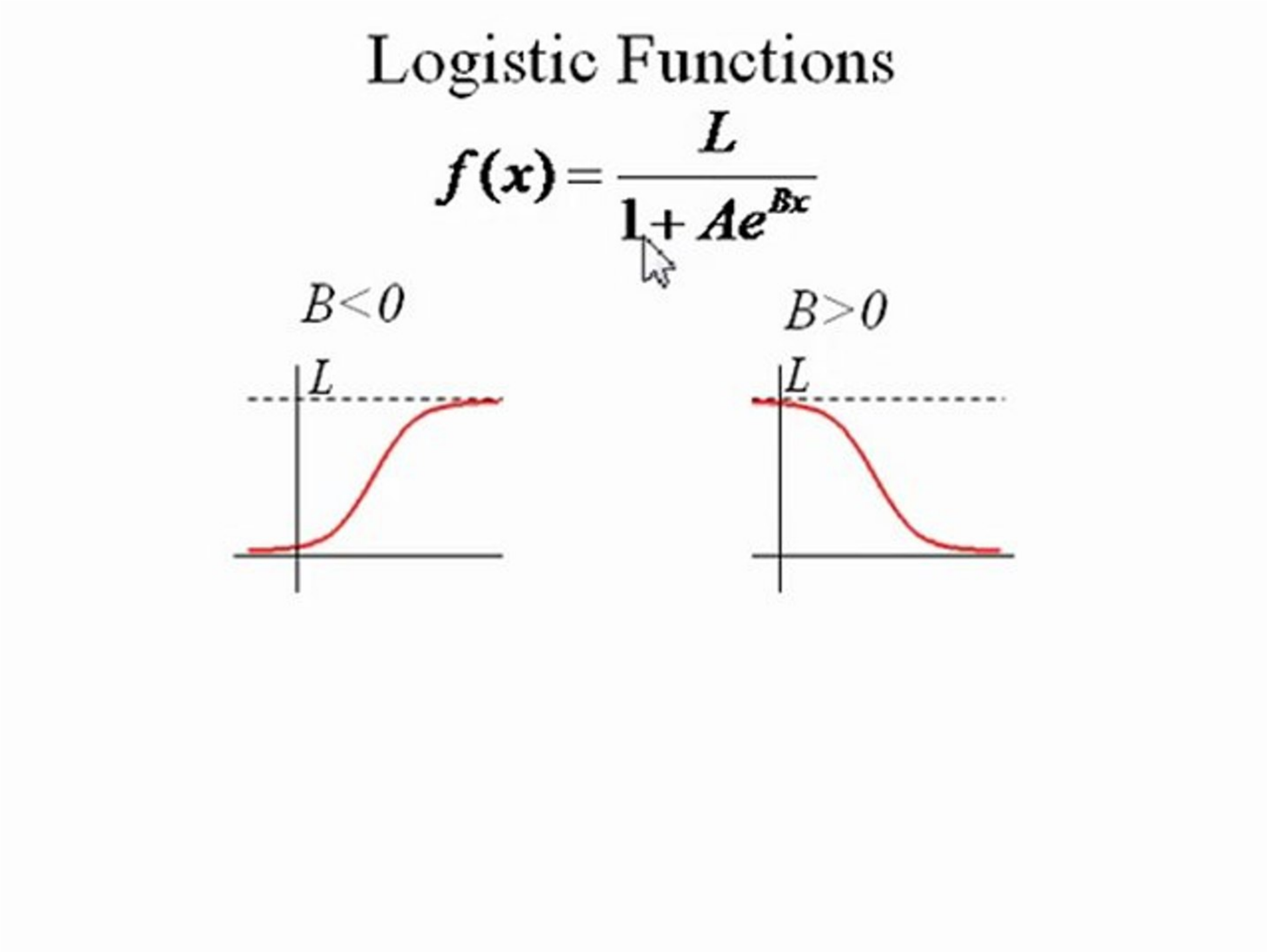
Agora plotamos a curva que a $f(t) = \frac{100}{1+9.288572251063083e^{-0.5067116359447639t}}$ faz

```
import matplotlib.pyplot as plt

x  = np.linspace(0,8,1000)
f_aprox = 100/(1+(9.288572251063083*(math.e**(-0.5067116359447639*x))))
plt.xlabel("t")
plt.ylabel("f")
plt.plot(x,f_aprox)
plt.scatter(t,f)
plt.show()
```



Precisamos agora encontrar o ponto de inflexao, como se trata de uma funcao logistica, isto é



temos a propriedade desta funcao, em que o ponto de inflexão é $\left(\frac{lnA}{B}, \frac{L}{2}\right)$ Em nosso caso, temos: A = a B = Beta e L = 100.

```
ponto_de_inflexao2 = [math.log(a)/Beta, 100/2]
ponto_de_inflexao2
```

```
[4.398527083779419, 50.0]
```

Percebemos pontos de inflexao muito proximos ao comparar o primeiro ex e o segundo ex. Como podemos ver ao deixar um ao lado do outro

```
ponto_de_inflexao_x,ponto_de_inflexao2[0],ponto_de_inflexao_y,ponto_de_inflexao2[1]
```

```
(4.372586872586831, 4.398527083779419, 49.79188184717652, 50.0)
```

podemos botar no grafico o ponto para melhor visualizacao tambem

```
plt.plot(x,f_aprox)
```

```
plt.scatter(ponto_de_inflexao2[0],ponto_de_inflexao2[1])
plt.show()
```

Por ultimo, botaremos as duas curvas uma ao lado da outra para percebermos que as duas curvas sao extremamente semelhantes

```
plt.plot(x,f_aprox)
plt.scatter(t,f)
plt.show()
plt.plot(x,y)
plt.scatter(t,f)
plt.show()
```