1. **Tabela de logaritmos**
   Escreva um programa que imprima uma tabela de logarítmos na base 10 de 1 até 10 com passos de 0.5. A tabela deverá ter o seguinte formato:

   ```
   x      log(x)
   1.0    0.0000000000
   1.5    0.1760912591
   2.0    0.3010299957
   ...
   ```

   Nota. Devido aos possíveis erros de arredondamento, utilize uma variável inteira n (o dobro de x) para controlar o ciclo do programa.

2. **Método de Newton**
   O *método de Newton* para resolução numérica de uma equação $f(x) = 0$ consiste em calcular aproximações $x_1, x_2, \ldots$ segundo a fórmula

   $$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

   partindo de uma aproximação inicial x_0 "perto" da raiz.

   (a) "Justifique" graficamente a fórmula anterior.

   (b) Escreva uma função `newton(f, df, x0, n)` que efectue `n` iterações do método de Newton com a função dada por `f`, derivada `df` e ponto inicial `x0`.

   (c) Use a função anterior para obter uma aproximação à raiz da equação $\text{sem}(x) = 0$ partindo de $x_0 = 2.5$ com 5 iterações e com 20 iterações.

3. **Coeficiente binomiais**
   O *coeficiente binomial*, também designado por número de *combinações* de n, k a k, é dado por

   $$\binom{n}{k} = \frac{n \times (n-1) \times \cdots \times (n-k+1)}{k \times (k-1) \times \cdots \times 1} = \frac{n!}{k!(n-k)!}$$

   Escreva uma função `binom(n,k)` que calcule o coeficiente binomial para n e k tais que $n, k \geq 0$. Porque é que a função que escreveu dá resultados correctos no caso $k > n$?

4. **Solução aproximada de uma equação**
   Considere a seguinte função

```
def zero(f,a,b,erro):
    while b-a>=erro:
(1)     x=(a+b)/2.0
        if f(x)*f(a)>0:
            # f(a) e f(x) têm o mesmo sinal -> fica [x,b]
            a=x
        else:
            # f(x) e f(b) têm o mesmo sinal -> fica [a,x]
            b=x
    return a
```

   (a) Descreva, usando no máximo 2 linhas de texto, o efeito da função. A descrição deve obviamente mencionar os parâmetros.
   (b) Determine uma expressão do número de vezes que a " partição" é executada. Essa expressão deve ser função de `b` − `a` e de `erro`.

5. **Máximo de uma função**
   Suponha que f(x) tem um máximo em [a,b] e que tem a segunda derivada negativa nesse intervalo.

   (a) Implemente uma função semelhante à do problema anterior para determinar esse máximo, a menos de um erro dado

```
maximo(f,a,b,erro)
```

   Sugestão: "parta" de cada vez o intervalo [a,b] em 3 sub-intervalos de igual comprimento

```
[a,x1], [x1,x2], [x2,b]
```

$$\begin{array}{lcl} f(x1) > f(x2) & \rightarrow & \text{máximo em } \texttt{[a,x2]}\text{; (porquê?)} \\ f(x1) \leq f(x2) & \rightarrow & \text{máximo em } \texttt{[x1,b]}\text{; (porquê?)} \end{array}$$

   (b) Determine uma expressão do número de vezes que o ciclo (por exemplo, a linha (1)) é executado. Essa expressão deve ser função de b-a e de erro.

6. **O maior, onde está?**
   Seja a uma lista de inteiros. Escreva uma função

```
ind_max(a)
```

   que retorna o primeiro índice do maior valor contido em `a`. Por exemplo, a chamada `ind_max([5,-1,8,1,8,1,6])` deve retornar 2, pois o máximo valor na lista é 8 e o menor índice onde ocorre 8 é 2.
   Nota: só é necessário usar um ciclo "while".
   Como alteraria a função que escreveu de forma a calcular o último índice do maior valor contido em `a`? (Para o exemplo dado, a resposta deverá ser 4)