1. Ao contrário...
   Escrever uma função `inverte(a)` que inverte a ordem dos elementos da lista `a`; a lista `a` é alterada (a função não é pura) e devolvida. Por exemplo:

   ```
   inverte([2,3,2,8,9,5])  ->  [5,9,8,2,3,2]
   ```

   (a) O método a utilizar deve basear-se em trocas de elementos da lista. Por exemplo, o efeito da primeira troca `a[0] <-> a[5]` é

   ```
   [2,3,2,8,9,5]  ->  [5,3,2,8,9,2]
    ^           ^
   ```

   Depois deverão ser trocados os elementos de índices 3 e 9, etc... Determine previamente:
   - Qual é o índice `j` tal que `a[i]` troca `a[j]`? (`j` é função de `i` e de `len(a)`).
   - Até que valor de `i` se devem efectuar trocas `a[i] <-> a[j]`? Esse limite é função de `len(a)`.

   (b) Implemente uma função pura com o mesmo objectivo, usando um método baseado na concatenação de listas (operador `+`); a uma lista inicialmente vazia `r`, concatenam-se sucessivamente as listas

   $$[\texttt{a}[0]], [\texttt{a}[1]], ...[\texttt{a}[\texttt{len(a)} - 1]]$$

   à esquerda de r. Por exemplo, para `a=[2,9,8,2,3,5]`, obtemos as listas

   ```
   1) r=[],              2) r=[2]+[]   =[2],
   3) r=[9]+[2]=[9,2],   4) r=[8]+[9,2]=[8,9,2], etc.
   ```

2. Está bem ordenada?
   Implemente uma função `ordenacao(a,b)` que verifica se a lista `b` resulta da ordenação da lista `a`. Por exemplo:

   ```
   ordenacao([2,1,3,1],[1,1,2,3])    ->  True
   ordenacao([2,1,3,1],[1,2,1,3])    ->  False
   ordenacao([2,1,3,1],[1,2,2,3])    ->  False
   ```

   As condições que tem que verificar são:

   (i) `b[i]` $\leq$ `b[i + 1]`, para $i = 0,...$ `len(b)-2`.

   (ii) `len(a)=len(b)`.

   (iii) Para cada `i`, se `a[i]` ocorre `k` vezes na lista `a`, também tem que ocorrer `k` vezes na lista `b`; no exemplo, 1 ocorre 2 vezes em `a` e em `b`.

   Nota. (ii) e (iii) implicam que: para cada `i`, se `b[i]` ocorre `k` vezes na lista `b`, também ocorre `k` vezes na lista `a`. No seu programa teste estas condições, uma a uma; se alguma condição falhar pode retornar imediatamente "False".
   Sugestão. Use (e defina) as funções auxiliares testei(b), testeii(a,b) e testeiii(a,b).

3. (Problema já proposto numa folha prática)
   Um polinómio em x de grau n, $a_0 + a_1x + \ldots + a_nx^n$, é representado pela lista dos seus coeficientes $[a_0, a_1, ..., a_n]$ sendo a_n diferente de 0 (se o polinómio for 0, a lista é [ ]). Escreva uma função (pura) `somap(p,q)` que calcula o polinómio $p + q$ na forma indicada. Exemplos:

```
[1,2,3] + [4,5]      -> [5,7,3]
[1,2,3] + [0,-2,-3]  -> [1]
[1,2,3] + [-1,-2,-3] -> []
```

   Note que a lista representativa de um polinómio não pode terminar em 0. Use uma função auxilir `remove0(li)` para remover os 0's finais da lista `li`; não pode usar métodos associados a listas, para além de `len` e `append`.

4. Reduzir o contraste de uma imagem, um problema resolvido!
   Uma imagem é caracterizada por um quadrado de "pixels", tendo cada célula (cada pixel) um valor inteiro compreendido emtre 0 e 255. Uma imagem pode representar-se por uma lista de listas, como

```
im = [[0,  0,  2,  90],
      [10, 20, 80, 200],
      [5,  70, 90, 250],
      [12, 90, 100,190]]
```

   Escreva uma função

```
reduz(im) -> outra "imagem"  (im não é alterado)
```

   que reduza o "contraste" da imagem, no sentido de substituir cada elemento " interno" (isto é, que não esteja em nenhum dos lados do quadrado) de `im` pela média desse elemento e dos 4 elementos adjacentes (acima, abaixo, à esquerda e à direita). Por exemplo, o elemento 20 do exemplo é substituído por $(20+0+70+10+80)/5 = 36$. Para o exemplo dado a função retorna

```
[[0,  0,  2,  90],
 [10, 36, 78, 200],
 [5,  55, 118,250],
 [12, 90, 100,190]]
```

   Uma solução do problema é:

```
def reduz(im):
    from copy import *
    r=deepcopy(im)  # para se obter uma nova cópia de im
    n=len(im)
    i=1
    while i <= n-2:
        j=1
        while j<= n-2:
            r[i][j]=(im[i][j]  +im[i][j-1]+im[i][j+1]+
                     im[i-1][j]+im[i+1][j])/5
            j=j+1
        i=i+1
    return r
```