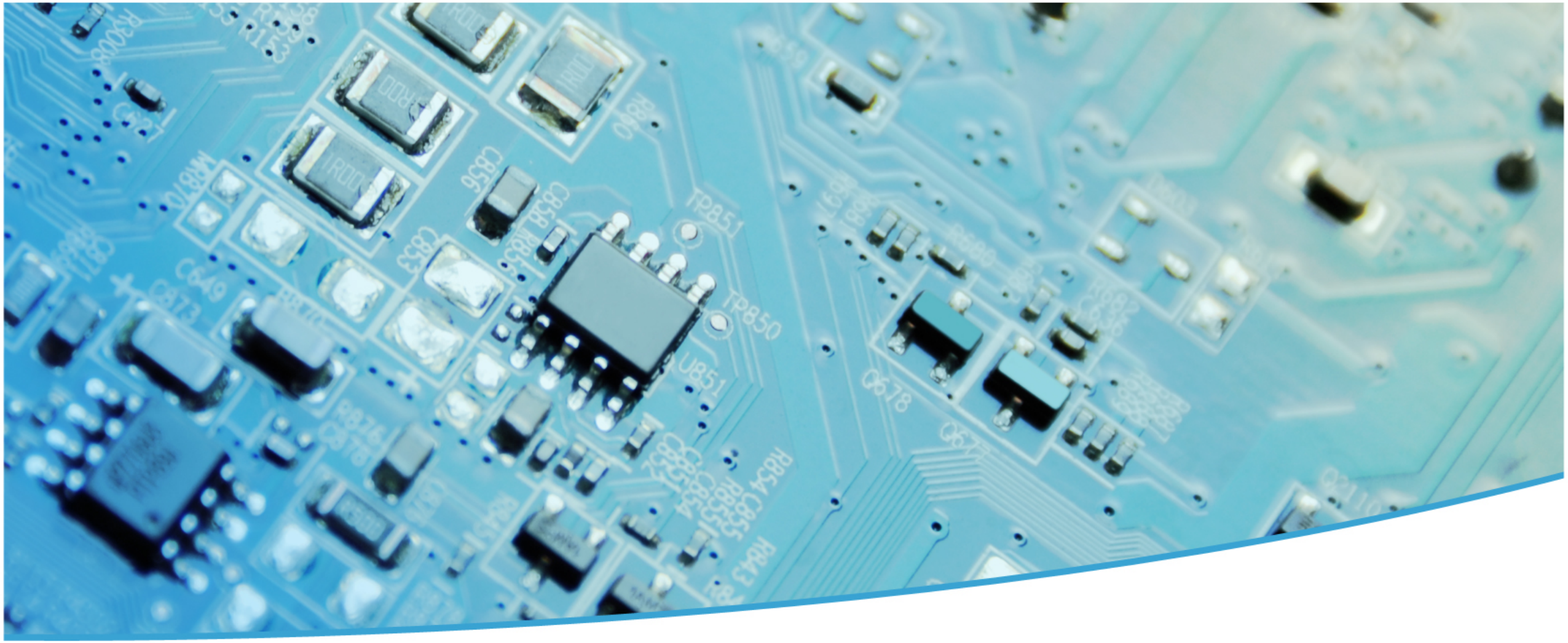
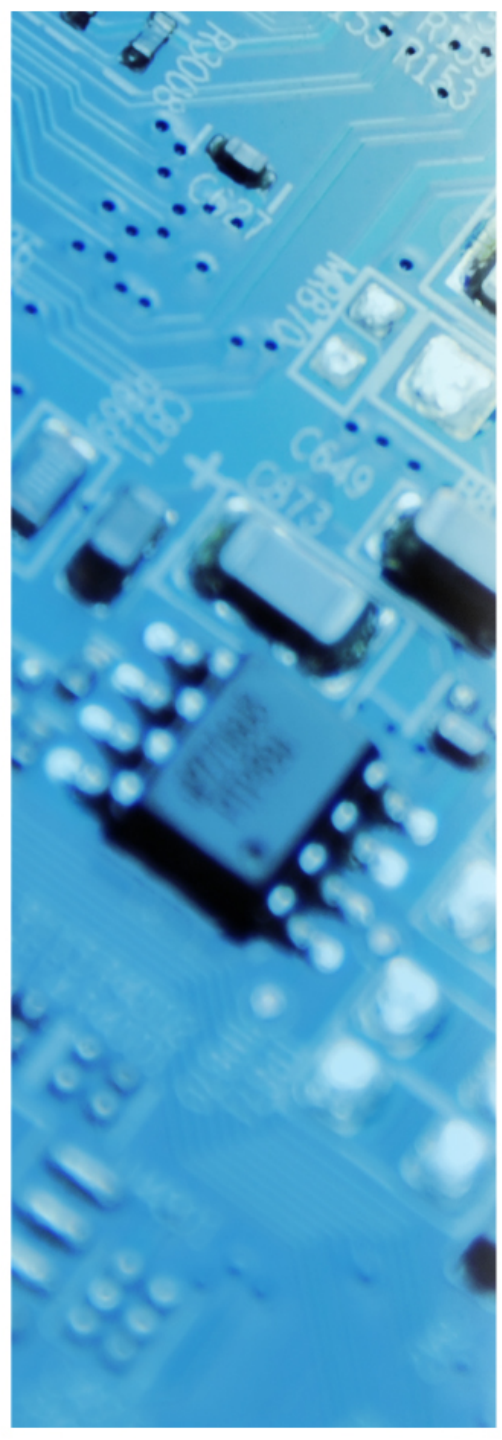
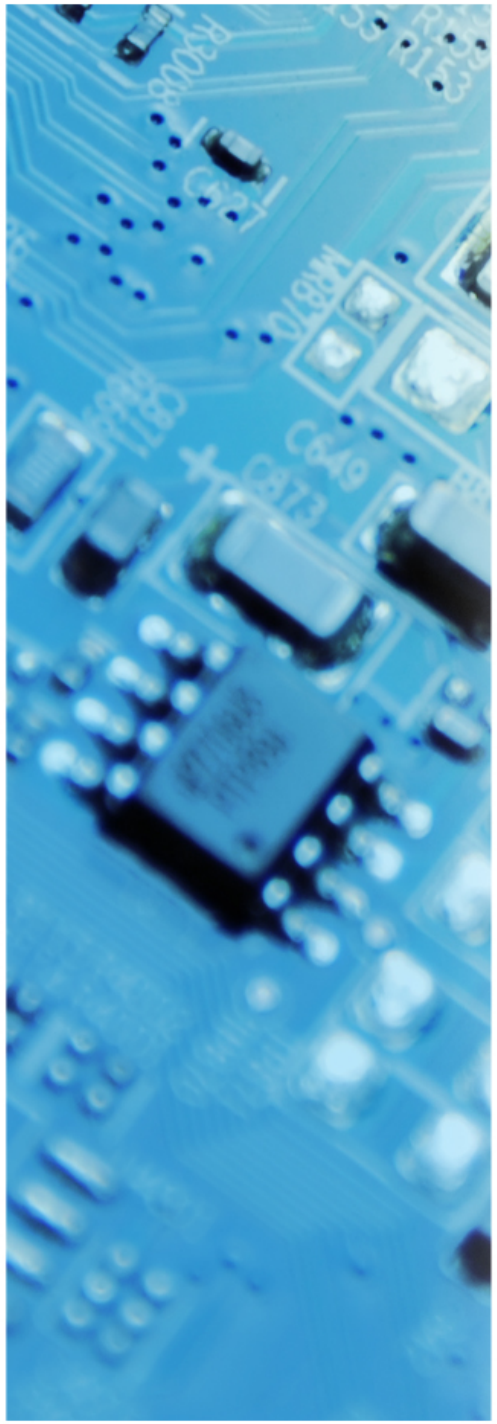


Escopo de parâmetros e argumentos
Funções anônimas
Funções de alta ordem



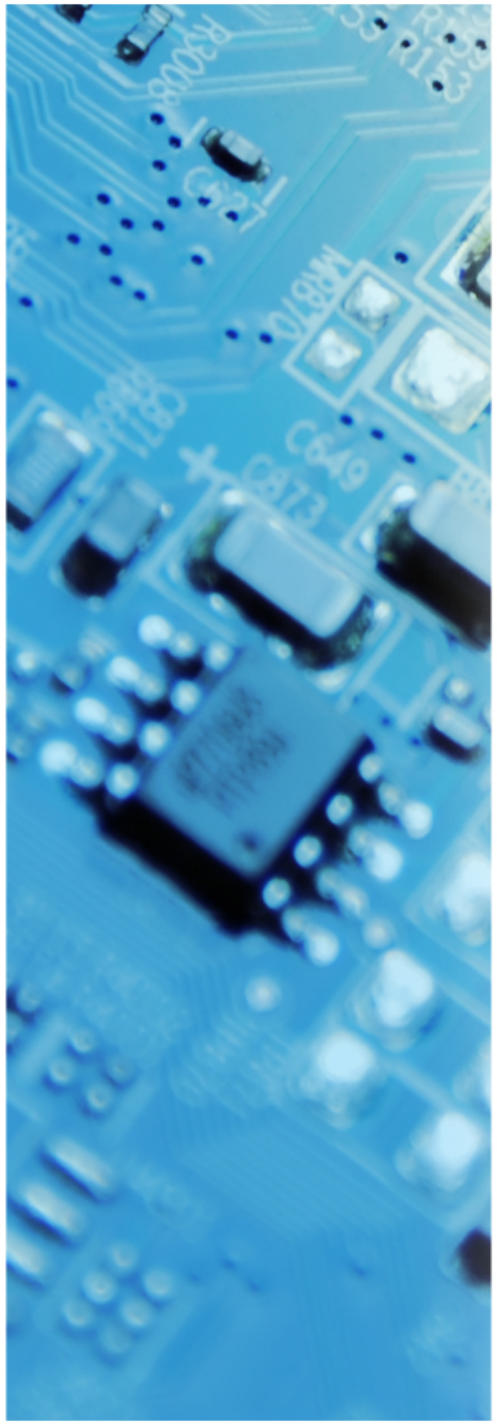
Módulos e pacotes

- Funções permitem dividir uma solução em partes menores.
- Módulos são usados para agrupar funções de acordo com a aplicação ou a natureza da função.
- Pacotes são grupos de módulos, ou seja, repositórios que apresentam diversos módulos.
- Em python, um módulo é basicamente um arquivo .py.



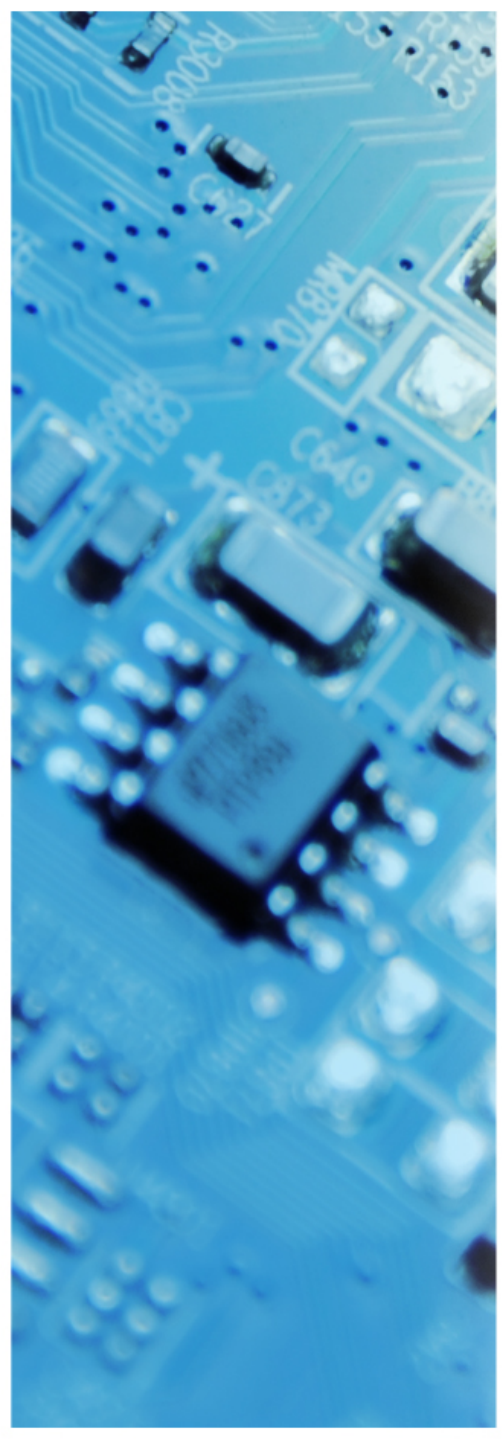
Módulos e pacotes

- A diferença para um código-fonte é que em um módulo são inseridas diversas funções e estas estarão disponíveis quando o módulo é importada.
- A instrução import permite utilizar diferentes módulos em uma aplicação.
- A forma como ocorre a importação vai indicar se a referência para a função inclui o nome do módulo ou não.
- A importação influi no escopo dos objetos e funções disponíveis na rotina que importa o módulo.



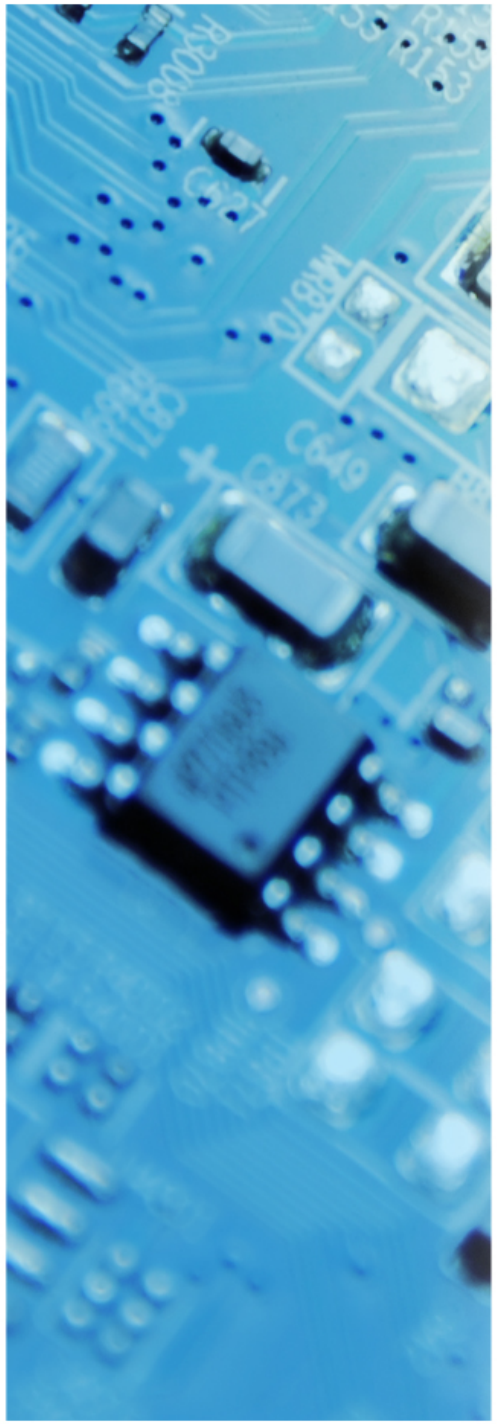
Módulos e pacotes

- Pacote em Python é um diretório que pode conter diversos módulos.
- O nome do pacote é o nome do diretório.
- No diretório deve existir um arquivo chamado __init__.py.
- O arquivo de inicialização pode estar vazio, o que já é suficiente para definir o diretório como um pacote.
- É possível também indicar neste arquivo algumas informações, como, por exemplo, quais objetos e funções serão importadas pela instrução from Modulo import *



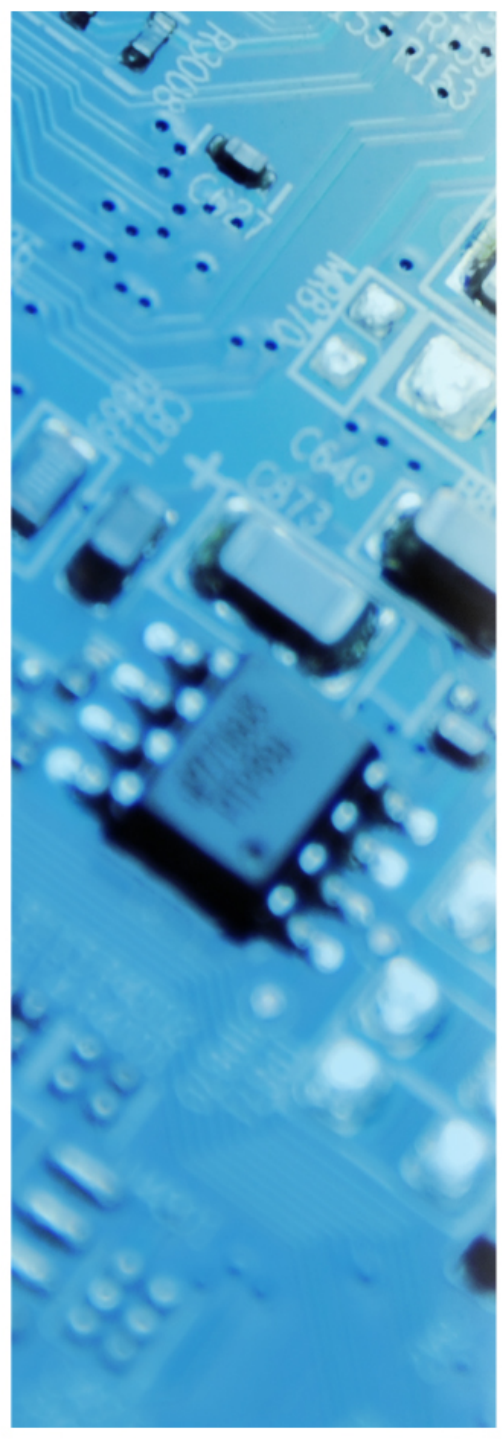
Parâmetros e argumentos

- É possível definir funções com um número indefinido de parâmetros/argumentos.
- Neste caso os parâmetros são recebidos em uma tupla.
- A lista é definida por *args.
- O uso de for...loop permite iterar pelos argumentos.
- O parâmetro *args armazena os parâmetros posicionais, não nomeados.



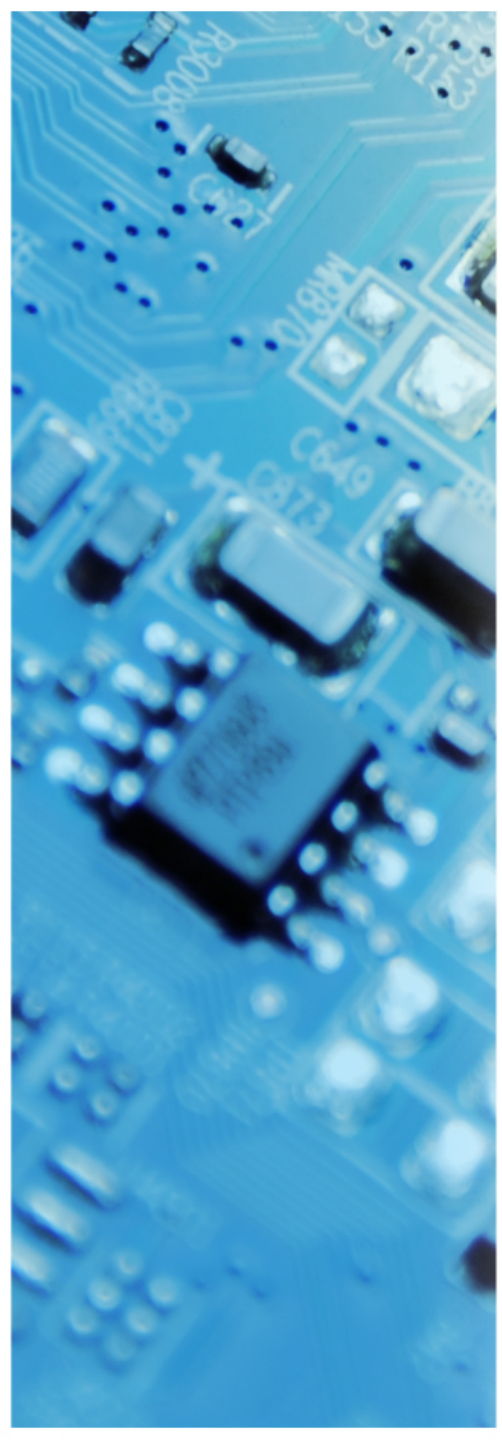
Parâmetros e argumentos

- Outra alternativa é utilizar um número indeterminado de argumentos nomeados.
- Argumentos nomeados são recebidos na função em um dicionário.
- O parâmetro **kwargs é definido no cabeçalho da função.
- Ao iterar sobre kwargs são obtidos o nome e o valor de cada parâmetro.
- Uma mesma função pode definir em seu caeçalho *args e **kwargs.
- Argumentos nomeados devem vir sempre depois dos posicionais, quando ambos forem usados.



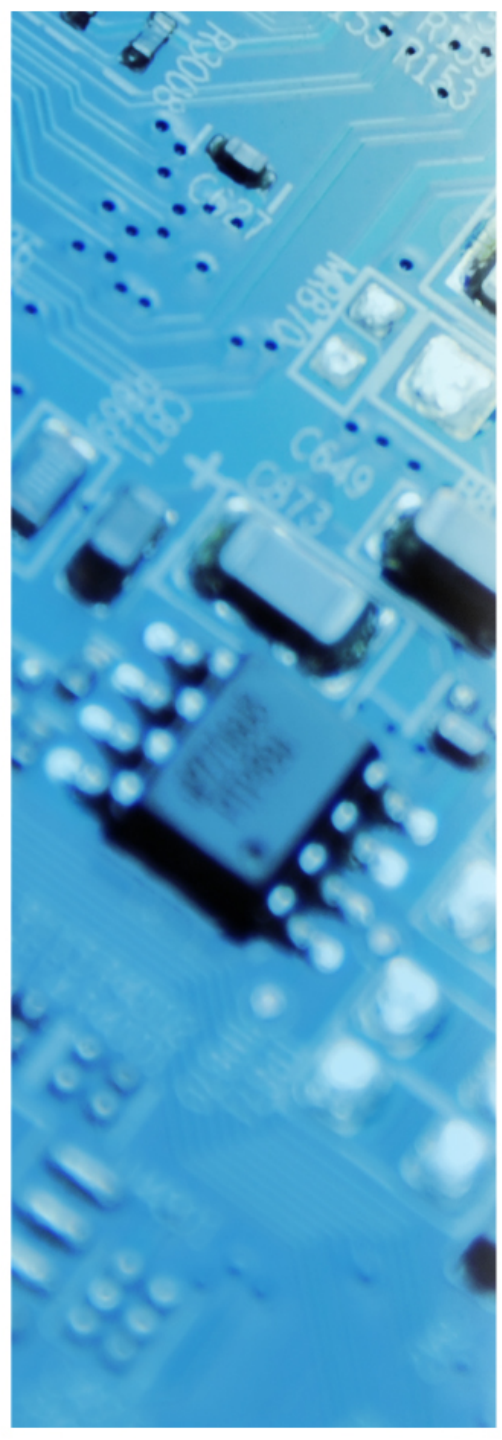
Escopo e parâmetros

- O valor de um parâmetro é usado para referenciar um objeto local, que é usado dentro da função.
- O escopo é o espaço no qual os nomes ou referências são válidos.
- Variáveis locais são acessíveis somente dentro da função ou da rotina na qual foram definidas.
- Por padrão uma referência é válida dentro do local (função, rotina) em que é criada.



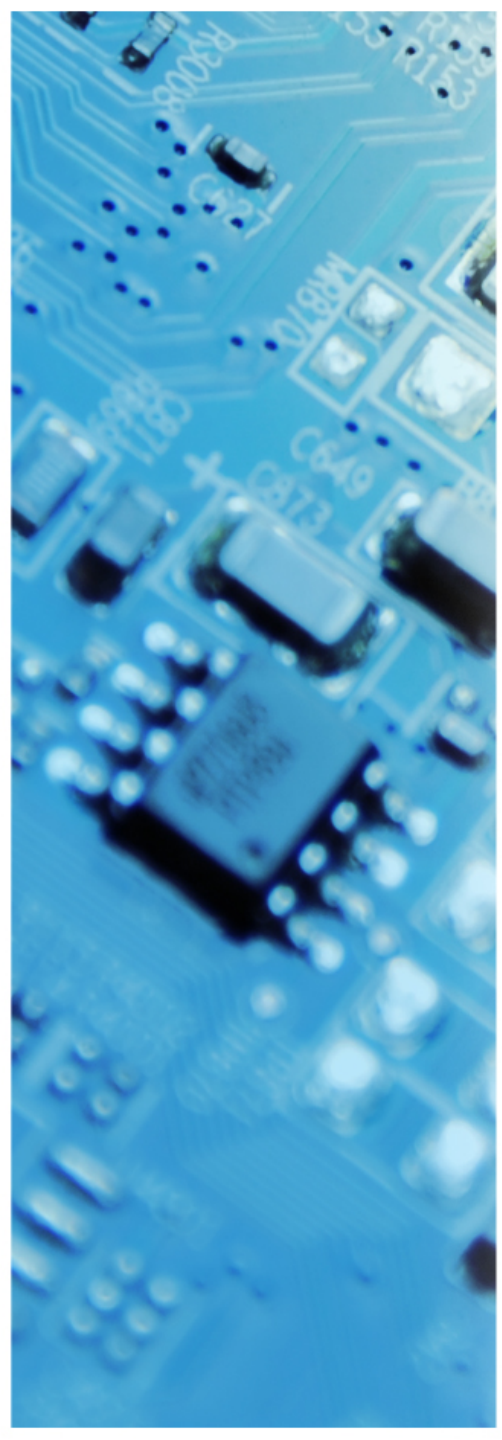
Escopo e parâmetros

- Objetos criados em um módulo são válidos dentro do módulo e das funções de forma global.
- Objetos criados dentro de funções não podem ser acessados fora destas.
- Dentro de uma função é possível utilizar o modificador *global* para alterar o valor de uma variável de módulo.
- Objetos de um módulo são acessíveis externamente pelo import.



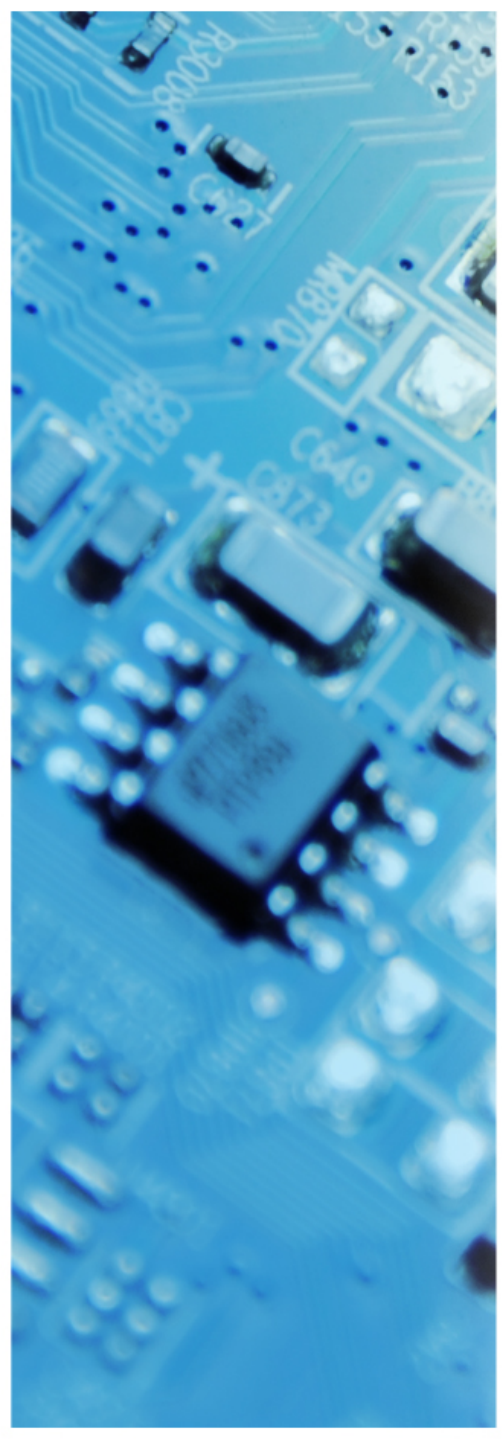
Escopo e parâmetros

- Python utiliza uma regra denominada LEGB.
- A regra indica que referências são buscadas localmente, nas funções envolventes, globalmente e por fim em ambiente interno.
- O ambiente interno diz respeito a a módulos pré-definidos pela linguagem (___builtin__).
- Uma atribuição por padrão altera uma referência local.
- Funções criadas dentro de outras funções podem ser retornadas por estas.



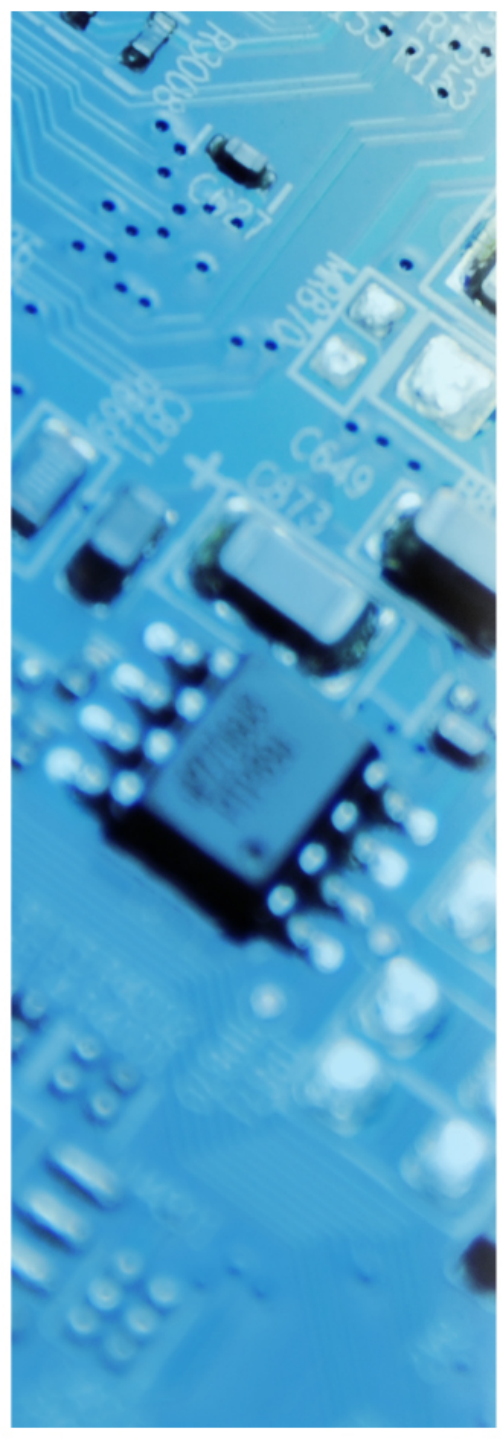
Escopo e parâmetros

- Quando o argumento é um objeto mutável, como uma lista, a alteração do mesmo dentro de uma função modifica o objeto original.
- Se for atribuída uma nova lista à variável a lista original não se altera.
- Esta situação assemelha-se à passagem de parâmetros por referência utilizada em outras linguagens.
- É possível considerar que nestes casos a função recebe um ponteiro para o objeto mutável.



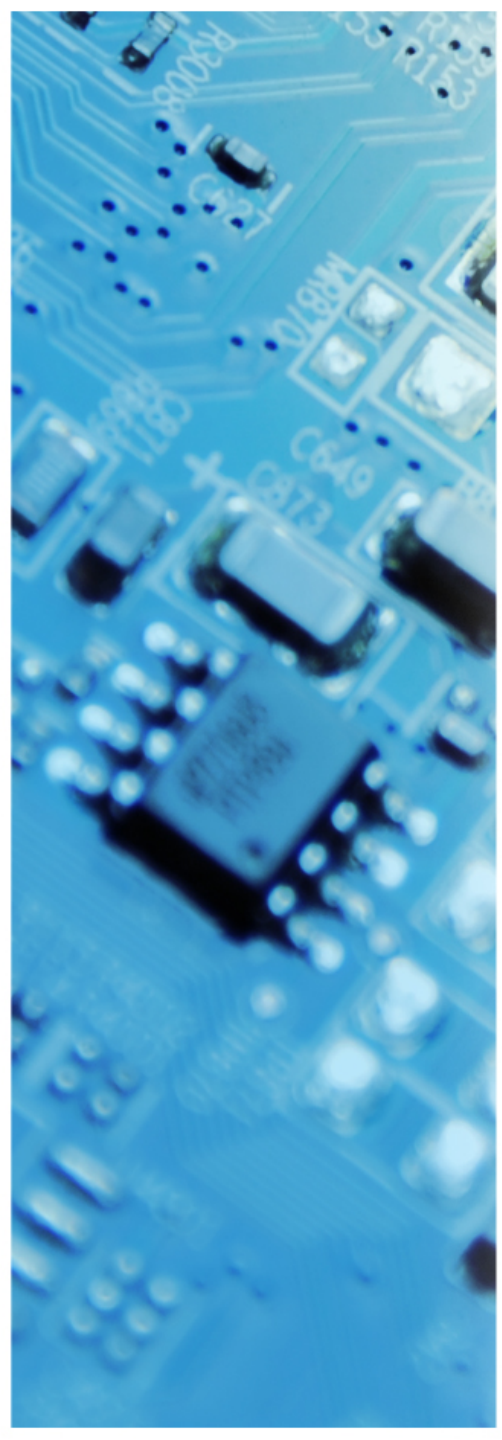
Retornando funções

- Uma função pode retornar uma referência para uma função.
- Desta forma é possível chamar uma função que irá criar outras e retornar a referência para uma delas.
- Funções são objetos, portanto, a referência retornada aponta para um objeto existente.
- Ao retornar para quem fez a chamada da função, o objeto fica disponível para ser utilizado



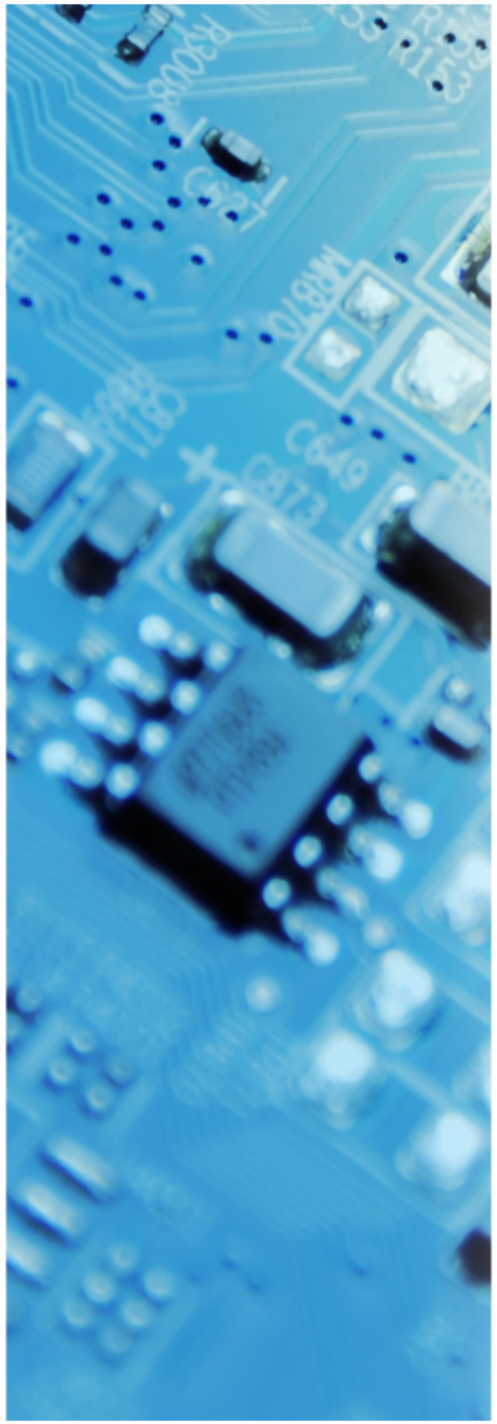
Funções anônimas

- Python possui um recurso para criar uma função sem utilizar def.
- A expressão cria e retorna uma função.
- Como funções são objetos, uma referência para uma função pode ser referenciada por uma variável.
- Estas expressões são chamadas LAMBDA.
- Definição de uma função lambda:
  - *lambda arg1, arg2...arg N: expressão com argumentos*



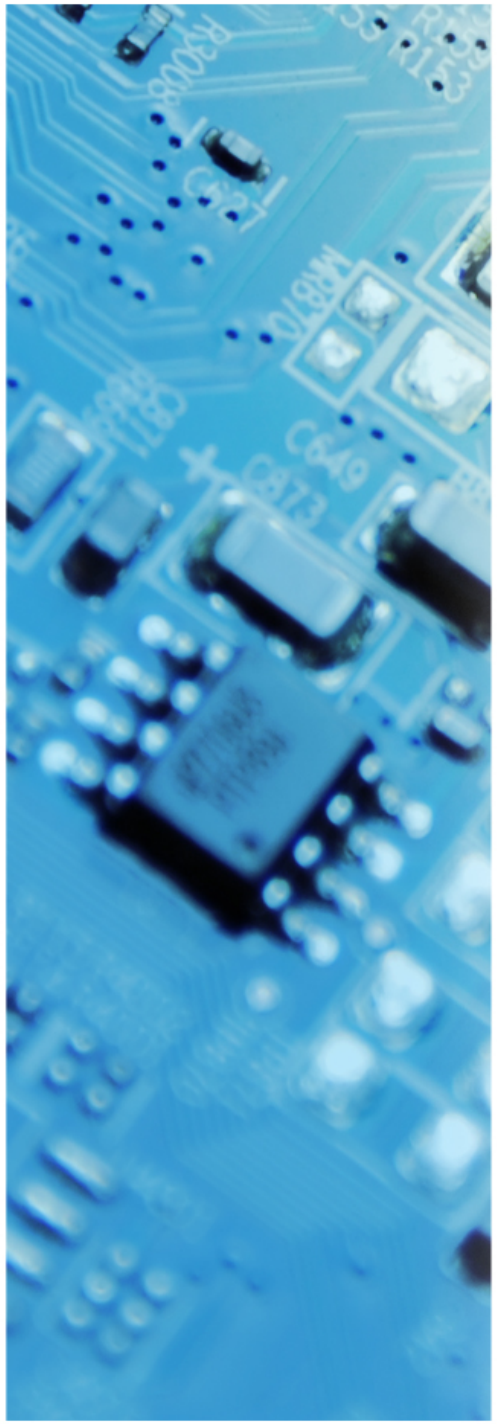
Funções anônimas

- O objeto que é retornado por lambda é do mesmo tipo de def.
- Lambda é uma expressão e não um bloco de instruções, por isso pode aparecer em diferentes locais, de diferentes formas.
- A principal limitação é que funções nomeadas não podem possuir diversas instruções.
- O uso de funções anônimas normalmente é mais restrito do que funções criadas por def.



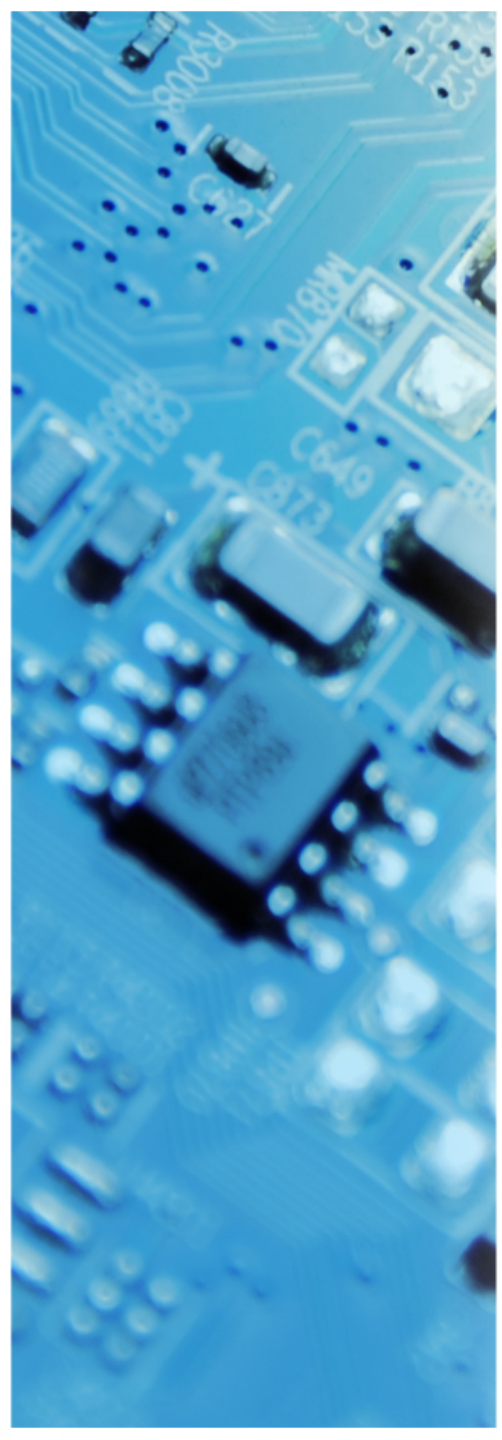
Desafio

- Versão 1:
  - Crie uma calculadora em Python.A calculadora deve solicitar ao usuário a operação e os valores envolvidos.
  - A operação pode ser soma, subtração, divisão, multiplicação ou potenciação.
  - Os valores são os operadores para execução da operação.
  - Criar uma ou mais funções para receber os dados.
  - Outra função deve receber os operadores e a operação e deve retornar o resultado.
  - Nesta função, devem ser usadas funções anônimas para que o cálculo seja feito e o resultado seja retornado.
  - As funções devem ser armazenadas em uma variável e ao final, esta será executada.



Desafio

- Versão 2:
  - A função que executa as operações deve mudar e retornar uma função, que é responsável pelo cálculo.
  - Nesta versão a função recebe somente a operação e cria as funções considerando que podem ser recebidos diversos valores, por exemplo (10+20+30+...).
  - Assim, é preciso criar a função usando um número indeterminado de parâmetros posicionais (*args).
  - No programa principal a função será executada com os parâmetros correspondentes, ex. Fun(10, 20, 30,..).
  - A função que recebe a operação e os valores também deve ser ajustada para permitir que o usuário informe mais operadores, além da operação.



Desafio opcional

- Converter o desafio desenvolvido na aula 2 em uma aplicação com funções.
- O arquivo .py principal só deve conter as chamadas para as funções que devem ser inseridos em outros arquivos fonte (módulos).
- Os módulos devem ser importados no programa principal para que seja possível chamar as funções que serão executadas.
- Se possível utilize uma função anônima em alguma parte da solução.