

# FUNÇÕES

MC102 - Algoritmos e  
Programação de  
Computadores

Santiago Valdés Ravelo  
[https://ic.unicamp.br/~santiago/  
ravelo@unicamp.br](https://ic.unicamp.br/~santiago/ravelo@unicamp.br)

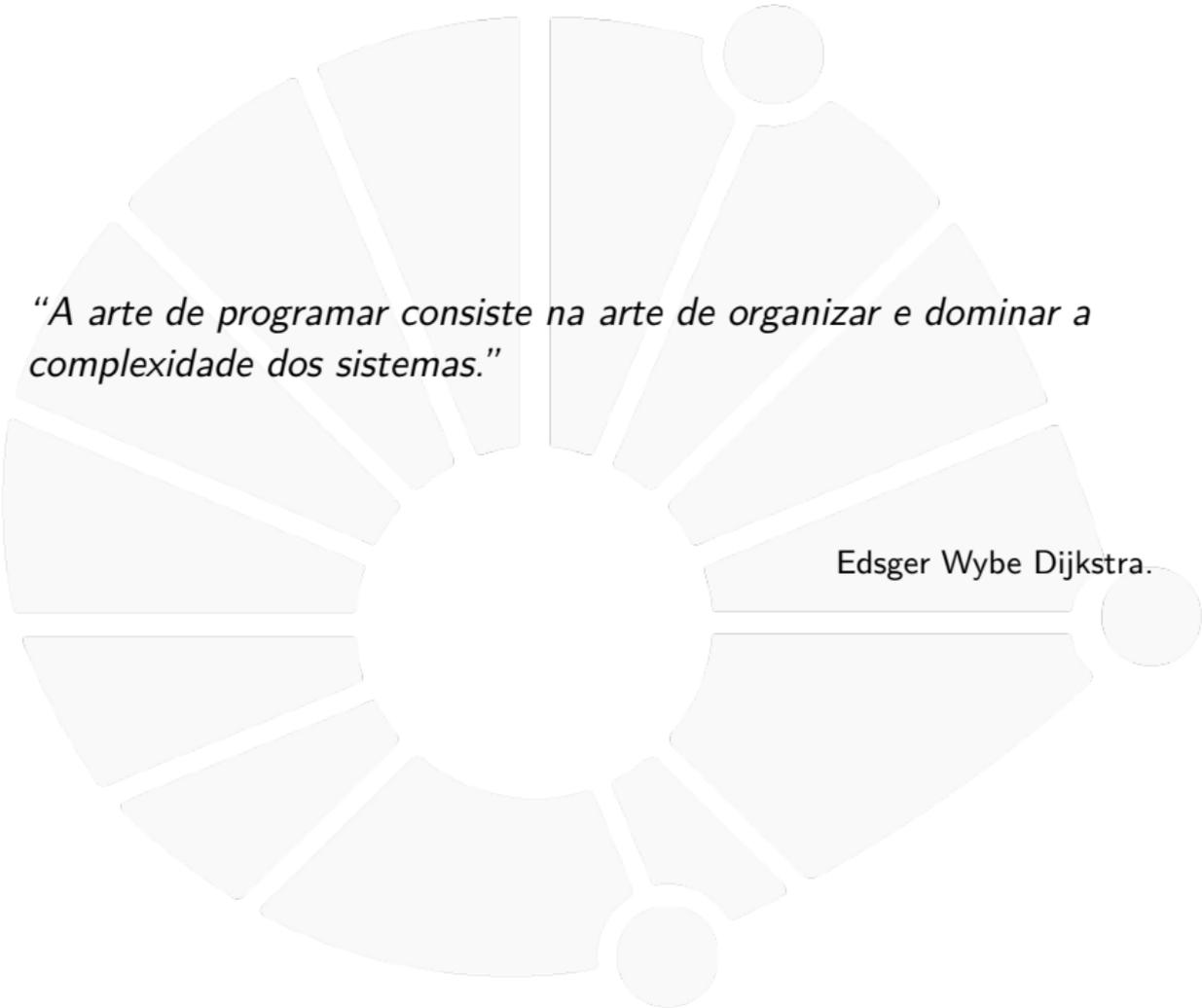
04/25

9



UNICAMP





*“A arte de programar consiste na arte de organizar e dominar a complexidade dos sistemas.”*

Edsger Wybe Dijkstra.



# DÚVIDAS DA AULA ANTERIOR



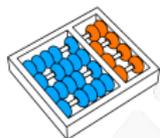
## Dúvidas selecionadas

- ▶ Posso fazer algo tipo, sendo  $v$  uma lista e  $x$  um inteiro  $v[x+=2]$  para aumentar  $x$  em 2 e depois visitar a posição?



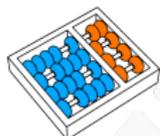
## Dúvidas selecionadas

- ▶ Posso fazer algo tipo, sendo  $v$  uma lista e  $x$  um inteiro  $v[x+=2]$  para aumentar  $x$  em 2 e depois visitar a posição?
- ▶ Como funciona a função `sort`?



## Dúvidas selecionadas

- ▶ Posso fazer algo tipo, sendo  $v$  uma lista e  $x$  um inteiro  $v[x+=2]$  para aumentar  $x$  em 2 e depois visitar a posição?
- ▶ Como funciona a função `sort`?
- ▶ Poderia explicar novamente sobre o problema de inserir uma lista em outra, deixando em ordem, sem usar o `.sort`?



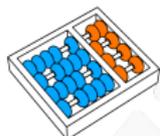
## Dúvidas selecionadas

- ▶ Posso fazer algo tipo, sendo  $v$  uma lista e  $x$  um inteiro  $v[x+=2]$  para aumentar  $x$  em 2 e depois visitar a posição?
- ▶ Como funciona a função `sort`?
- ▶ Poderia explicar novamente sobre o problema de inserir uma lista em outra, deixando em ordem, sem usar o `.sort`?
- ▶ É seguro alterar uma lista enquanto eu percorro ela dentro de um `for`? Eu devo criar uma cópia para isso ou posso fazer na lista principal?



## Dúvidas selecionadas

- ▶ Posso fazer algo tipo, sendo  $v$  uma lista e  $x$  um inteiro  $v[x+=2]$  para aumentar  $x$  em 2 e depois visitar a posição?
- ▶ Como funciona a função `sort`?
- ▶ Poderia explicar novamente sobre o problema de inserir uma lista em outra, deixando em ordem, sem usar o `.sort`?
- ▶ É seguro alterar uma lista enquanto eu percorro ela dentro de um `for`? Eu devo criar uma cópia para isso ou posso fazer na lista principal?
- ▶ Não compreendi totalmente aquele algoritmo de achar os 3 números  $q$  somando dão 0.



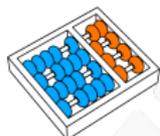
### Dúvidas selecionadas

- ▶ Posso fazer algo tipo, sendo  $v$  uma lista e  $x$  um inteiro  $v[x+=2]$  para aumentar  $x$  em 2 e depois visitar a posição?
- ▶ Como funciona a função `sort`?
- ▶ Poderia explicar novamente sobre o problema de inserir uma lista em outra, deixando em ordem, sem usar o `.sort`?
- ▶ É seguro alterar uma lista enquanto eu percorro ela dentro de um `for`? Eu devo criar uma cópia para isso ou posso fazer na lista principal?
- ▶ Não compreendi totalmente aquele algoritmo de achar os 3 números  $q$  somando dão 0.
- ▶ Há alguma diferença de performance entre usar um `for` normal e usar a instância da lista? Por exemplo: `lista[p * p:2:n]` = alguma coisa.



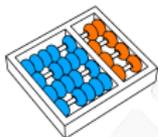
### Dúvidas selecionadas

- ▶ Posso fazer algo tipo, sendo  $v$  uma lista e  $x$  um inteiro  $v[x+=2]$  para aumentar  $x$  em 2 e depois visitar a posição?
- ▶ Como funciona a função `sort`?
- ▶ Poderia explicar novamente sobre o problema de inserir uma lista em outra, deixando em ordem, sem usar o `.sort`?
- ▶ É seguro alterar uma lista enquanto eu percorro ela dentro de um `for`? Eu devo criar uma cópia para isso ou posso fazer na lista principal?
- ▶ Não compreendi totalmente aquele algoritmo de achar os 3 números  $q$  somando dão 0.
- ▶ Há alguma diferença de performance entre usar um `for` normal e usar a instância da lista? Por exemplo: `lista[p * p:2:n]` = alguma coisa.
- ▶ A dinâmica de coding dojo geralmente acontece em que lugares e contextos?



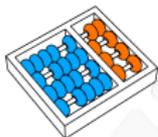
### Dúvidas selecionadas

- ▶ Posso fazer algo tipo, sendo  $v$  uma lista e  $x$  um inteiro  $v[x+=2]$  para aumentar  $x$  em 2 e depois visitar a posição?
- ▶ Como funciona a função `sort`?
- ▶ Poderia explicar novamente sobre o problema de inserir uma lista em outra, deixando em ordem, sem usar o `.sort`?
- ▶ É seguro alterar uma lista enquanto eu percorro ela dentro de um `for`? Eu devo criar uma cópia para isso ou posso fazer na lista principal?
- ▶ Não compreendi totalmente aquele algoritmo de achar os 3 números  $q$  somando dão 0.
- ▶ Há alguma diferença de performance entre usar um `for` normal e usar a instância da lista? Por exemplo: `lista[p * p:2:n] = alguma coisa`.
- ▶ A dinâmica de coding dojo geralmente acontece em que lugares e contextos?
- ▶ Para fazer um `slice` de uma lista que comece no primeiro elemento é preciso deixar um espaço entre o colchete e os dois pontos? Ex. `[ :x]` ou `[:x]`?



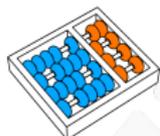
### Dúvidas selecionadas

- ▶ Posso fazer algo tipo, sendo  $v$  uma lista e  $x$  um inteiro  $v[x+=2]$  para aumentar  $x$  em 2 e depois visitar a posição?
- ▶ Como funciona a função `sort`?
- ▶ Poderia explicar novamente sobre o problema de inserir uma lista em outra, deixando em ordem, sem usar o `.sort`?
- ▶ É seguro alterar uma lista enquanto eu percorro ela dentro de um `for`? Eu devo criar uma cópia para isso ou posso fazer na lista principal?
- ▶ Não compreendi totalmente aquele algoritmo de achar os 3 números  $q$  somando dão 0.
- ▶ Há alguma diferença de performance entre usar um `for` normal e usar a instância da lista? Por exemplo: `lista[p * p:2:n] = alguma coisa`.
- ▶ A dinâmica de coding dojo geralmente acontece em que lugares e contextos?
- ▶ Para fazer um slice de uma lista que comece no primeiro elemento é preciso deixar um espaço entre o colchete e os dois pontos? Ex. `[ :x]` ou `[:x]`?
- ▶ Não entendi como foi resolvido o problema da subsequência dentro da sequência.



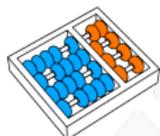
### Dúvidas selecionadas

- ▶ Posso fazer algo tipo, sendo  $v$  uma lista e  $x$  um inteiro  $v[x+=2]$  para aumentar  $x$  em 2 e depois visitar a posição?
- ▶ Como funciona a função `sort`?
- ▶ Poderia explicar novamente sobre o problema de inserir uma lista em outra, deixando em ordem, sem usar o `.sort`?
- ▶ É seguro alterar uma lista enquanto eu percorro ela dentro de um `for`? Eu devo criar uma cópia para isso ou posso fazer na lista principal?
- ▶ Não compreendi totalmente aquele algoritmo de achar os 3 números  $q$  somando dão 0.
- ▶ Há alguma diferença de performance entre usar um `for` normal e usar a instância da lista? Por exemplo: `lista[p * p:2:n] = alguma coisa`.
- ▶ A dinâmica de coding dojo geralmente acontece em que lugares e contextos?
- ▶ Para fazer um `slice` de uma lista que comece no primeiro elemento é preciso deixar um espaço entre o colchete e os dois pontos? Ex. `[ :x]` ou `[:x]`?
- ▶ Não entendi como foi resolvido o problema da subsequência dentro da sequência.
- ▶ Como funciona o `*n` para imprimir uma lista sem colchetes?



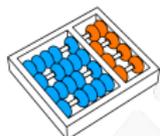
### Dúvidas selecionadas

- ▶ Posso fazer algo tipo, sendo  $v$  uma lista e  $x$  um inteiro  $v[x+=2]$  para aumentar  $x$  em 2 e depois visitar a posição?
- ▶ Como funciona a função `sort`?
- ▶ Poderia explicar novamente sobre o problema de inserir uma lista em outra, deixando em ordem, sem usar o `.sort`?
- ▶ É seguro alterar uma lista enquanto eu percorro ela dentro de um `for`? Eu devo criar uma cópia para isso ou posso fazer na lista principal?
- ▶ Não compreendi totalmente aquele algoritmo de achar os 3 números  $q$  somando dão 0.
- ▶ Há alguma diferença de performance entre usar um `for` normal e usar a instância da lista? Por exemplo: `lista[p * p:2:n] = alguma coisa`.
- ▶ A dinâmica de coding dojo geralmente acontece em que lugares e contextos?
- ▶ Para fazer um `slice` de uma lista que comece no primeiro elemento é preciso deixar um espaço entre o colchete e os dois pontos? Ex. `[ :x]` ou `[:x]`?
- ▶ Não entendi como foi resolvido o problema da subsequência dentro da sequência.
- ▶ Como funciona o `*n` para imprimir uma lista sem colchetes?
- ▶ Porque se não colocar `list` antes de `map(int, input().split())` da erro?



### Dúvidas selecionadas

- ▶ Posso fazer algo tipo, sendo  $v$  uma lista e  $x$  um inteiro  $v[x+=2]$  para aumentar  $x$  em 2 e depois visitar a posição?
- ▶ Como funciona a função `sort`?
- ▶ Poderia explicar novamente sobre o problema de inserir uma lista em outra, deixando em ordem, sem usar o `.sort`?
- ▶ É seguro alterar uma lista enquanto eu percorro ela dentro de um `for`? Eu devo criar uma cópia para isso ou posso fazer na lista principal?
- ▶ Não compreendi totalmente aquele algoritmo de achar os 3 números  $q$  somando dão 0.
- ▶ Há alguma diferença de performance entre usar um `for` normal e usar a instância da lista? Por exemplo: `lista[p * p:2:n] = alguma coisa`.
- ▶ A dinâmica de coding dojo geralmente acontece em que lugares e contextos?
- ▶ Para fazer um `slice` de uma lista que comece no primeiro elemento é preciso deixar um espaço entre o colchete e os dois pontos? Ex. `[ :x]` ou `[:x]`?
- ▶ Não entendi como foi resolvido o problema da subsequência dentro da sequência.
- ▶ Como funciona o `*n` para imprimir uma lista sem colchetes?
- ▶ Porque se não colocar `list` antes de `map(int, input().split())` da erro?
- ▶ Eu posso fazer com que o programa continue executando, mesmo se o usuário entrar com um valor inválido para o meu código? ex: `x = int(input());` o valor de entrada inserido foi um texto.

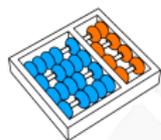


### Dúvidas selecionadas

- ▶ Posso fazer algo tipo, sendo  $v$  uma lista e  $x$  um inteiro  $v[x+=2]$  para aumentar  $x$  em 2 e depois visitar a posição?
- ▶ Como funciona a função `sort`?
- ▶ Poderia explicar novamente sobre o problema de inserir uma lista em outra, deixando em ordem, sem usar o `.sort`?
- ▶ É seguro alterar uma lista enquanto eu percorro ela dentro de um `for`? Eu devo criar uma cópia para isso ou posso fazer na lista principal?
- ▶ Não compreendi totalmente aquele algoritmo de achar os 3 números  $q$  somando dão 0.
- ▶ Há alguma diferença de performance entre usar um `for` normal e usar a instância da lista? Por exemplo: `lista[p * p:2:n] = alguma coisa`.
- ▶ A dinâmica de coding dojo geralmente acontece em que lugares e contextos?
- ▶ Para fazer um `slice` de uma lista que comece no primeiro elemento é preciso deixar um espaço entre o colchete e os dois pontos? Ex. `[ :x]` ou `[:x]`?
- ▶ Não entendi como foi resolvido o problema da subsequência dentro da sequência.
- ▶ Como funciona o `*n` para imprimir uma lista sem colchetes?
- ▶ Porque se não colocar `list` antes de `map(int, input().split())` da erro?
- ▶ Eu posso fazer com que o programa continue executando, mesmo se o usuário entrar com um valor inválido para o meu código? ex: `x = int(input())`; o valor de entrada inserido foi um texto.
- ▶ Não entendi a lógica do 1ro desafio para pegar os números(a posição) pequenos entre 2 adjacentes maiores.

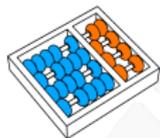


# SUBPROGRAMAS



## Motivação

- ▶ No desenvolvimento de software um problema recorrente é o de como programar sistemas complexos.



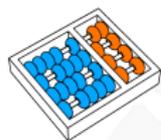
## Motivação

- ▶ No desenvolvimento de software um problema recorrente é o de como programar sistemas complexos.
- ▶ Grandes projetos envolvem centenas de programadores que trabalham em milhões de linhas de códigos.



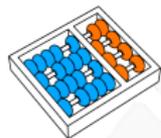
## Motivação

- ▶ No desenvolvimento de software um problema recorrente é o de como programar sistemas complexos.
- ▶ Grandes projetos envolvem centenas de programadores que trabalham em milhões de linhas de códigos.
- ▶ É importante **MODULARIZAR**, organizar o código em base na tarefa que executa, de forma que se torne **REUTILIZÁVEL** e mais fácil de **DEPURAR** e **GERENCIAR**.



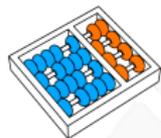
## Programação estruturada

- ▶ Desenvolvimento de algoritmos por fases ou refinamentos sucessivos.



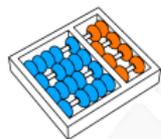
## Programação estruturada

- ▶ Desenvolvimento de algoritmos por fases ou refinamentos sucessivos.
- ▶ Uso de um número muito limitado de estruturas de controle.



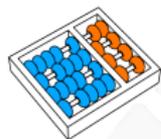
## Programação estruturada

- ▶ Desenvolvimento de algoritmos por fases ou refinamentos sucessivos.
- ▶ Uso de um número muito limitado de estruturas de controle.
- ▶ Decomposição do algoritmo em módulos: **DIVIDE ET IMPERA.**



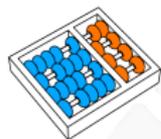
## Objetivos principais

- ▶ Evitar repetições de sequências de comandos.



## Objetivos principais

- ▶ Evitar repetições de sequências de comandos.
- ▶ Dividir e estruturar um programa em partes fechadas e logicamente coerentes.



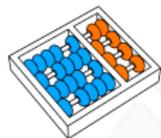
## O que é um subprograma?

- ▶ Programas independentes.



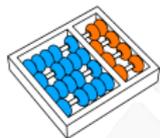
## O que é um subprograma?

- ▶ Programas independentes.
- ▶ Executados unicamente quando chamados por outro programa.



## O que é um subprograma?

- ▶ Programas independentes.
- ▶ Executados unicamente quando chamados por outro programa.
- ▶ Devem executar uma tarefa específica, muito bem identificada.



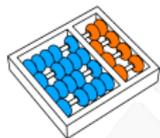
## O que é um subprograma?

- ▶ Programas independentes.
- ▶ Executados unicamente quando chamados por outro programa.
- ▶ Devem executar uma tarefa específica, muito bem identificada.
- ▶ Podem ser testados separadamente.



## O que é um subprograma?

- ▶ Programas independentes.
- ▶ Executados unicamente quando chamados por outro programa.
- ▶ Devem executar uma tarefa específica, muito bem identificada.
- ▶ Podem ser testados separadamente.
- ▶ É possível criar bibliotecas com subprogramas.



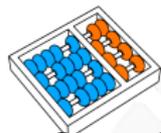
## O que é um subprograma?

- ▶ Programas independentes.
- ▶ Executados unicamente quando chamados por outro programa.
- ▶ Devem executar uma tarefa específica, muito bem identificada.
- ▶ Podem ser testados separadamente.
- ▶ É possível criar bibliotecas com subprogramas.

Como desenvolver? Usando funções.

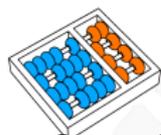


# INTRODUÇÃO



## Função na Matemática

O que é uma função na matemática?



## Função na Matemática

O que é uma função na matemática?

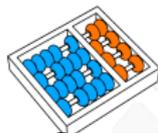
▶ Ex:  $f(x) = x$ ,  $f(x) = 2x + 3$ ,  $f(x) = \sqrt{x}$ , ...



## Função na Matemática

O que é uma função na matemática?

- ▶ Ex:  $f(x) = x$ ,  $f(x) = 2x + 3$ ,  $f(x) = \sqrt{x}$ , ...
- ▶ É uma relação entre dois conjuntos  $X$  e  $Y$ .



## Função na Matemática

O que é uma função na matemática?

- ▶ Ex:  $f(x) = x$ ,  $f(x) = 2x + 3$ ,  $f(x) = \sqrt{x}$ , ...
- ▶ É uma relação entre dois conjuntos  $X$  e  $Y$ .
- ▶ Para cada  $x \in X$ , temos um único  $y \in Y$  relacionado ( $f(x) = y$ ).



## Função na Matemática

O que é uma função na matemática?

- ▶ Ex:  $f(x) = x$ ,  $f(x) = 2x + 3$ ,  $f(x) = \sqrt{x}$ , ...
- ▶ É uma relação entre dois conjuntos  $X$  e  $Y$ .
- ▶ Para cada  $x \in X$ , temos um único  $y \in Y$  relacionado ( $f(x) = y$ ).
- ▶ Escrevemos  $f: X \rightarrow Y$ .



## Função na Matemática

O que é uma função na matemática?

- ▶ Ex:  $f(x) = x$ ,  $f(x) = 2x + 3$ ,  $f(x) = \sqrt{x}$ , ...
- ▶ É uma relação entre dois conjuntos  $X$  e  $Y$ .
- ▶ Para cada  $x \in X$ , temos um único  $y \in Y$  relacionado ( $f(x) = y$ ).
- ▶ Escrevemos  $f: X \rightarrow Y$ .
- ▶ Note que  $x$  não precisa ser um único valor:



## Função na Matemática

O que é uma função na matemática?

- ▶ Ex:  $f(x) = x$ ,  $f(x) = 2x + 3$ ,  $f(x) = \sqrt{x}$ , ...
- ▶ É uma relação entre dois conjuntos  $X$  e  $Y$ .
- ▶ Para cada  $x \in X$ , temos um único  $y \in Y$  relacionado ( $f(x) = y$ ).
- ▶ Escrevemos  $f: X \rightarrow Y$ .
- ▶ Note que  $x$  não precisa ser um único valor:
  - ▶ Ex:  $f(x_1, x_2) = x_1 + x_2$ ,  $f(a, b, c) = a \cdot b \cdot c$ , ...



## Função na Matemática

O que é uma função na matemática?

- ▶ Ex:  $f(x) = x$ ,  $f(x) = 2x + 3$ ,  $f(x) = \sqrt{x}$ , ...
- ▶ É uma relação entre dois conjuntos  $X$  e  $Y$ .
- ▶ Para cada  $x \in X$ , temos um único  $y \in Y$  relacionado ( $f(x) = y$ ).
- ▶ Escrevemos  $f: X \rightarrow Y$ .
- ▶ Note que  $x$  não precisa ser um único valor:
  - ▶ Ex:  $f(x_1, x_2) = x_1 + x_2$ ,  $f(a, b, c) = a \cdot b \cdot c$ , ...
  - ▶ Isso é,  $X$  pode ser o produto cartesiano de outros conjuntos.



## Função na Matemática

O que é uma função na matemática?

- ▶ Ex:  $f(x) = x$ ,  $f(x) = 2x + 3$ ,  $f(x) = \sqrt{x}$ , ...
- ▶ É uma relação entre dois conjuntos  $X$  e  $Y$ .
- ▶ Para cada  $x \in X$ , temos um único  $y \in Y$  relacionado ( $f(x) = y$ ).
- ▶ Escrevemos  $f: X \rightarrow Y$ .
- ▶ Note que  $x$  não precisa ser um único valor:
  - ▶ Ex:  $f(x_1, x_2) = x_1 + x_2$ ,  $f(a, b, c) = a \cdot b \cdot c$ , ...
  - ▶ Isso é,  $X$  pode ser o produto cartesiano de outros conjuntos.
- ▶ Note que  $y$  também não precisa ser um único valor:



## Função na Matemática

O que é uma função na matemática?

- ▶ Ex:  $f(x) = x$ ,  $f(x) = 2x + 3$ ,  $f(x) = \sqrt{x}$ , ...
- ▶ É uma relação entre dois conjuntos  $X$  e  $Y$ .
- ▶ Para cada  $x \in X$ , temos um único  $y \in Y$  relacionado ( $f(x) = y$ ).
- ▶ Escrevemos  $f: X \rightarrow Y$ .
- ▶ Note que  $x$  não precisa ser um único valor:
  - ▶ Ex:  $f(x_1, x_2) = x_1 + x_2$ ,  $f(a, b, c) = a \cdot b \cdot c$ , ...
  - ▶ Isso é,  $X$  pode ser o produto cartesiano de outros conjuntos.
- ▶ Note que  $y$  também não precisa ser um único valor:
  - ▶ Ex:  $f(x_1, x_2) = (2x_1, 3x_2)$ ,  $f(x) = (x, x^2, x^3)$ , ...



## Função na Matemática

O que é uma função na matemática?

- ▶ Ex:  $f(x) = x$ ,  $f(x) = 2x + 3$ ,  $f(x) = \sqrt{x}$ , ...
- ▶ É uma relação entre dois conjuntos  $X$  e  $Y$ .
- ▶ Para cada  $x \in X$ , temos um único  $y \in Y$  relacionado ( $f(x) = y$ ).
- ▶ Escrevemos  $f: X \rightarrow Y$ .
- ▶ Note que  $x$  não precisa ser um único valor:
  - ▶ Ex:  $f(x_1, x_2) = x_1 + x_2$ ,  $f(a, b, c) = a \cdot b \cdot c$ , ...
  - ▶ Isso é,  $X$  pode ser o produto cartesiano de outros conjuntos.
- ▶ Note que  $y$  também não precisa ser um único valor:
  - ▶ Ex:  $f(x_1, x_2) = (2x_1, 3x_2)$ ,  $f(x) = (x, x^2, x^3)$ , ...
  - ▶ Isso é,  $Y$  pode ser o produto cartesiano de outros conjuntos.



## Função na Matemática

O que é uma função na matemática?

- ▶ Ex:  $f(x) = x$ ,  $f(x) = 2x + 3$ ,  $f(x) = \sqrt{x}$ , ...
- ▶ É uma relação entre dois conjuntos  $X$  e  $Y$ .
- ▶ Para cada  $x \in X$ , temos um único  $y \in Y$  relacionado ( $f(x) = y$ ).
- ▶ Escrevemos  $f: X \rightarrow Y$ .
- ▶ Note que  $x$  não precisa ser um único valor:
  - ▶ Ex:  $f(x_1, x_2) = x_1 + x_2$ ,  $f(a, b, c) = a \cdot b \cdot c$ , ...
  - ▶ Isso é,  $X$  pode ser o produto cartesiano de outros conjuntos.
- ▶ Note que  $y$  também não precisa ser um único valor:
  - ▶ Ex:  $f(x_1, x_2) = (2x_1, 3x_2)$ ,  $f(x) = (x, x^2, x^3)$ , ...
  - ▶ Isso é,  $Y$  pode ser o produto cartesiano de outros conjuntos.
  - ▶ Mas, no final das contas o resultado é um **único** vetor...



## Função na Matemática

O que é uma função na matemática?

- ▶ Ex:  $f(x) = x$ ,  $f(x) = 2x + 3$ ,  $f(x) = \sqrt{x}$ , ...
- ▶ É uma relação entre dois conjuntos  $X$  e  $Y$ .
- ▶ Para cada  $x \in X$ , temos um único  $y \in Y$  relacionado ( $f(x) = y$ ).
- ▶ Escrevemos  $f: X \rightarrow Y$ .
- ▶ Note que  $x$  não precisa ser um único valor:
  - ▶ Ex:  $f(x_1, x_2) = x_1 + x_2$ ,  $f(a, b, c) = a \cdot b \cdot c$ , ...
  - ▶ Isso é,  $X$  pode ser o produto cartesiano de outros conjuntos.
- ▶ Note que  $y$  também não precisa ser um único valor:
  - ▶ Ex:  $f(x_1, x_2) = (2x_1, 3x_2)$ ,  $f(x) = (x, x^2, x^3)$ , ...
  - ▶ Isso é,  $Y$  pode ser o produto cartesiano de outros conjuntos.
  - ▶ Mas, no final das contas o resultado é um **único** vetor...

Informalmente,  $f$  nos diz como calcular  $y = f(x)$  a partir de  $x$ :



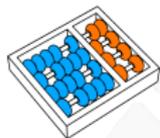
## Função na Matemática

O que é uma função na matemática?

- ▶ Ex:  $f(x) = x$ ,  $f(x) = 2x + 3$ ,  $f(x) = \sqrt{x}$ , ...
- ▶ É uma relação entre dois conjuntos  $X$  e  $Y$ .
- ▶ Para cada  $x \in X$ , temos um único  $y \in Y$  relacionado ( $f(x) = y$ ).
- ▶ Escrevemos  $f: X \rightarrow Y$ .
- ▶ Note que  $x$  não precisa ser um único valor:
  - ▶ Ex:  $f(x_1, x_2) = x_1 + x_2$ ,  $f(a, b, c) = a \cdot b \cdot c$ , ...
  - ▶ Isso é,  $X$  pode ser o produto cartesiano de outros conjuntos.
- ▶ Note que  $y$  também não precisa ser um único valor:
  - ▶ Ex:  $f(x_1, x_2) = (2x_1, 3x_2)$ ,  $f(x) = (x, x^2, x^3)$ , ...
  - ▶ Isso é,  $Y$  pode ser o produto cartesiano de outros conjuntos.
  - ▶ Mas, no final das contas o resultado é um **único** vetor...

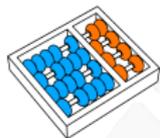
Informalmente,  $f$  nos diz como calcular  $y = f(x)$  a partir de  $x$ :

- ▶ Como obter uma saída a partir de uma entrada.



## Função na Programação

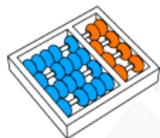
Na programação, o conceito de função é bem parecido:



## Função na Programação

Na programação, o conceito de função é bem parecido:

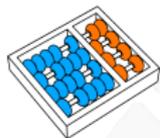
- ▶ Temos um conjunto de dados de entrada.



## Função na Programação

Na programação, o conceito de função é bem parecido:

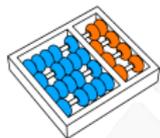
- ▶ Temos um conjunto de dados de entrada.
  - ▶ Mesmo papel do  $X$  na função  $f: X \rightarrow Y$ .



## Função na Programação

Na programação, o conceito de função é bem parecido:

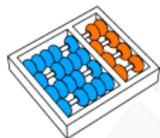
- ▶ Temos um conjunto de dados de entrada.
  - ▶ Mesmo papel do  $X$  na função  $f: X \rightarrow Y$ .
  - ▶ São chamados de **parâmetros** da função.



## Função na Programação

Na programação, o conceito de função é bem parecido:

- ▶ Temos um conjunto de dados de entrada.
  - ▶ Mesmo papel do  $X$  na função  $f: X \rightarrow Y$ .
  - ▶ São chamados de **parâmetros** da função.
- ▶ Temos as instruções de como calcular uma resposta.



## Função na Programação

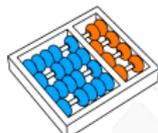
Na programação, o conceito de função é bem parecido:

- ▶ Temos um conjunto de dados de entrada.
  - ▶ Mesmo papel do  $X$  na função  $f: X \rightarrow Y$ .
  - ▶ São chamados de **parâmetros** da função.
- ▶ Temos as instruções de como calcular uma resposta.
- ▶ A resposta calculada (saída) é o nosso “ $y = f(x)$ ”.



$$f(x) = x^2$$

## Exemplos com Pseudocódigo



$$f(x) = x^2$$

## Exemplos com Pseudocódigo

---

**Algoritmo:** QUADRADO( $x$ )

---

1 devolva  $x \cdot x$

---



$$f(x) = x^2$$

---

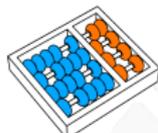
**Algoritmo:** QUADRADO( $x$ )

---

1 devolva  $x \cdot x$

---

$$f(x) = |x|$$



$$f(x) = x^2$$

## Exemplos com Pseudocódigo

---

### Algoritmo: QUADRADO( $x$ )

---

1 devolva  $x \cdot x$

---

$$f(x) = |x|$$

---

### Algoritmo: ABSOLUTO( $x$ )

---

1 se  $x \geq 0$

2   └ devolva  $x$

3 senão

4   └ devolva  $-x$

---



$$f(x) = x^2$$

## Exemplos com Pseudocódigo

---

### Algoritmo: QUADRADO( $x$ )

---

1 devolva  $x \cdot x$

---

$$f(x) = |x|$$

---

### Algoritmo: ABSOLUTO( $x$ )

---

1 se  $x \geq 0$

2   └ devolva  $x$

3 senão

4   └ devolva  $-x$

---

$$f(x) = \text{soma dos dígitos na base 10 de } x$$



$$f(x) = x^2$$

## Exemplos com Pseudocódigo

---

### Algoritmo: QUADRADO( $x$ )

---

1 devolva  $x \cdot x$

---

$$f(x) = |x|$$

---

### Algoritmo: ABSOLUTO( $x$ )

---

1 se  $x \geq 0$   
 2     └ devolva  $x$   
 3 senão  
 4     └ devolva  $-x$

---

$$f(x) = \text{soma dos dígitos na base 10 de } x$$

---

### Algoritmo: SOMADOSDIGITOS( $x$ )

---

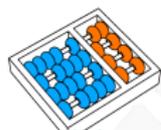
1  $soma \leftarrow 0$   
 2 enquanto  $x > 0$   
 3     └  $soma \leftarrow soma + x \% 10$   
 4     └  $x \leftarrow \lfloor x / 10 \rfloor$   
 5 devolva  $soma$

---



$$f(x) = x^2$$

## Exemplos com Python



$$f(x) = x^2$$

```
1 def quadrado(x):  
2     return x * x      # ou x ** 2
```



## Exemplos com Python

$$f(x) = x^2$$

```
1 def quadrado(x):  
2     return x * x    # ou x ** 2
```

$$f(x) = |x|$$



## Exemplos com Python

$$f(x) = x^2$$

```
1 def quadrado(x):  
2     return x * x     # ou x ** 2
```

$$f(x) = |x|$$

```
1 def absoluto(x):  
2     if x >= 0:      # funções podem ter if/elif/else  
3         return x  
4     else:  
5         return -x
```



## Exemplos com Python

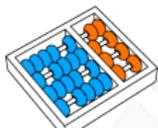
$$f(x) = x^2$$

```
1 def quadrado(x):  
2     return x * x    # ou x ** 2
```

$$f(x) = |x|$$

```
1 def absoluto(x):  
2     if x >= 0:      # funções podem ter if/elif/else  
3         return x  
4     else:  
5         return -x
```

$f(x)$  = soma dos dígitos na base 10 de  $x$



## Exemplos com Python

$$f(x) = x^2$$

```
1 def quadrado(x):  
2     return x * x      # ou x ** 2
```

$$f(x) = |x|$$

```
1 def absoluto(x):  
2     if x >= 0:        # funções podem ter if/elif/else  
3         return x  
4     else:  
5         return -x
```

$$f(x) = \text{soma dos dígitos na base 10 de } x$$

```
1 def soma_dos_digitos(x):  
2     soma = 0  
3     while x > 0:      # funções podem ter while/for  
4         soma += x % 10  
5         x //= 10  
6     return soma
```



## Usando funções no Python

**Definindo** a função, i.e., informando o Python que:



## Usando funções no Python

**Definindo** a função, i.e., informando o Python que:

- ▶ A função existe.



## Usando funções no Python

**Definindo** a função, i.e., informando o Python que:

- ▶ A função existe.
- ▶ Qual o seu nome.



## Usando funções no Python

**Definindo** a função, i.e., informando o Python que:

- ▶ A função existe.
- ▶ Qual o seu nome.
- ▶ Quais são seus parâmetros.



## Usando funções no Python

**Definindo** a função, i.e., informando o Python que:

- ▶ A função existe.
- ▶ Qual o seu nome.
- ▶ Quais são seus parâmetros.

```
1 def nome_da_funcao(parametro_1, parametro_2, ..., parametro_n):  
2     # instruções para computar o resultado  
3     return resultado
```



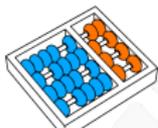
## Usando funções no Python

**Definindo** a função, i.e., informando o Python que:

- ▶ A função existe.
- ▶ Qual o seu nome.
- ▶ Quais são seus parâmetros.

```
1 def nome_da_funcao(parametro_1, parametro_2, ..., parametro_n):  
2     # instruções para computar o resultado  
3     return resultado
```

**Chamando** a função, i.e., pedindo que seja executada:



## Usando funções no Python

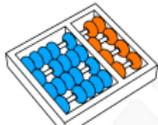
**Definindo** a função, i.e., informando o Python que:

- ▶ A função existe.
- ▶ Qual o seu nome.
- ▶ Quais são seus parâmetros.

```
1 def nome_da_funcao(parametro_1, parametro_2, ..., parametro_n):  
2     # instruções para computar o resultado  
3     return resultado
```

**Chamando** a função, i.e., pedindo que seja executada:

- ▶ A execução segue para as instruções da função.



## Usando funções no Python

**Definindo** a função, i.e., informando o Python que:

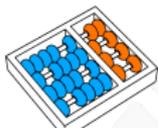
- ▶ A função existe.
- ▶ Qual o seu nome.
- ▶ Quais são seus parâmetros.

```
1 def nome_da_funcao(parametro_1, parametro_2, ..., parametro_n):  
2     # instruções para computar o resultado  
3     return resultado
```

**Chamando** a função, i.e., pedindo que seja executada:

- ▶ A execução segue para as instruções da função.
- ▶ E depois **retorna** para onde estava.

```
1 calculado = nome_da_funcao(valor_1, valor_2, ..., valor_n)
```



## Usando funções no Python

**Definindo** a função, i.e., informando o Python que:

- ▶ A função existe.
- ▶ Qual o seu nome.
- ▶ Quais são seus parâmetros.

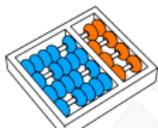
```
1 def nome_da_funcao(parametro_1, parametro_2, ..., parametro_n):  
2     # instruções para computar o resultado  
3     return resultado
```

**Chamando** a função, i.e., pedindo que seja executada:

- ▶ A execução segue para as instruções da função.
- ▶ E depois **retorna** para onde estava.

```
1 calculado = nome_da_funcao(valor_1, valor_2, ..., valor_n)
```

Observações:



## Usando funções no Python

**Definindo** a função, i.e., informando o Python que:

- ▶ A função existe.
- ▶ Qual o seu nome.
- ▶ Quais são seus parâmetros.

```
1 def nome_da_funcao(parametro_1, parametro_2, ..., parametro_n):  
2     # instruções para computar o resultado  
3     return resultado
```

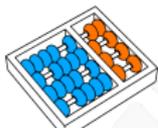
**Chamando** a função, i.e., pedindo que seja executada:

- ▶ A execução segue para as instruções da função.
- ▶ E depois **retorna** para onde estava.

```
1 calculado = nome_da_funcao(valor_1, valor_2, ..., valor_n)
```

**Observações:**

- ▶ O valor passado para o parâmetro pode vir de uma constante, variável, ou expressão.



## Usando funções no Python

**Definindo** a função, i.e., informando o Python que:

- ▶ A função existe.
- ▶ Qual o seu nome.
- ▶ Quais são seus parâmetros.

```
1 def nome_da_funcao(parametro_1, parametro_2, ..., parametro_n):  
2     # instruções para computar o resultado  
3     return resultado
```

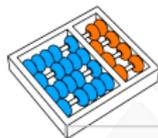
**Chamando** a função, i.e., pedindo que seja executada:

- ▶ A execução segue para as instruções da função.
- ▶ E depois **retorna** para onde estava.

```
1 calculado = nome_da_funcao(valor_1, valor_2, ..., valor_n)
```

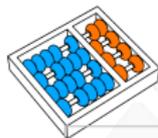
Observações:

- ▶ O valor passado para o parâmetro pode vir de uma constante, variável, ou expressão.
- ▶ A ordem dos valores é importante!



## Exemplo de um código completo

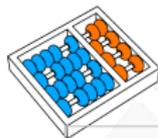
```
1 def le_lista(n):
2     lista = []
3     for i in range(n):
4         lista.append(float(input()))
5     return lista
6 def soma_valores(lista):
7     soma = 0
8     for x in lista :
9         soma += x
10    return soma
11 n = int(input())
12 lista = le_lista(n)
13 print(soma_valores(lista))
```



## Exemplo de um código completo

```
1 def le_lista(n):
2     lista = []
3     for i in range(n):
4         lista.append(float(input()))
5     return lista
6 def soma_valores(lista):
7     soma = 0
8     for x in lista :
9         soma += x
10    return soma
11 n = int(input())
12 lista = le_lista(n)
13 print(soma_valores(lista))
```

Nas linhas:

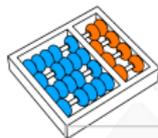


## Exemplo de um código completo

```
1 def le_lista(n):
2     lista = []
3     for i in range(n):
4         lista.append(float(input()))
5     return lista
6 def soma_valores(lista):
7     soma = 0
8     for x in lista :
9         soma += x
10    return soma
11 n = int(input())
12 lista = le_lista(n)
13 print(soma_valores(lista))
```

Nas linhas:

- ▶ 1 a 5 definimos uma função chamada `le_lista`.

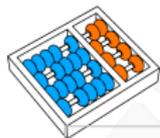


## Exemplo de um código completo

```
1 def le_lista(n):
2     lista = []
3     for i in range(n):
4         lista.append(float(input()))
5     return lista
6 def soma_valores(lista):
7     soma = 0
8     for x in lista :
9         soma += x
10    return soma
11 n = int(input())
12 lista = le_lista(n)
13 print(soma_valores(lista))
```

Nas linhas:

- ▶ 1 a 5 definimos uma função chamada `le_lista`.
- ▶ 6 a 10 definimos uma função chamada `soma_valores`.



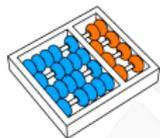
## Exemplo de um código completo

```
1 def le_lista(n):
2     lista = []
3     for i in range(n):
4         lista.append(float(input()))
5     return lista
6 def soma_valores(lista):
7     soma = 0
8     for x in lista :
9         soma += x
10    return soma
11 n = int(input())
12 lista = le_lista(n)
13 print(soma_valores(lista))
```

Nas linhas:

- ▶ 1 a 5 definimos uma função chamada `le_lista`.
- ▶ 6 a 10 definimos uma função chamada `soma_valores`.
- ▶ 11 a 13 chamamos essas funções para somar os números.

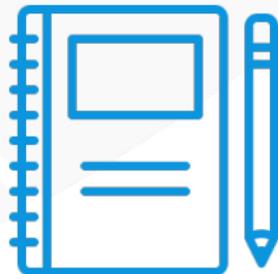
Vamos simular esse código!

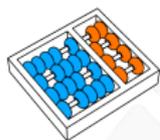


## Listas e repetição



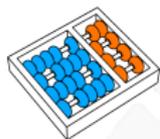
**Vamos fazer alguns exercícios?**





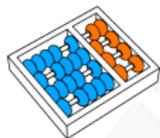
## Exercícios

1. Faça uma função que acha o maior entre dois números.



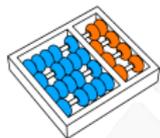
## Exercícios

1. Faça uma função que acha o maior entre dois números.
2. Faça uma função que verifica se um número é primo.



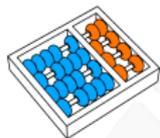
## Exercícios

1. Faça uma função que acha o maior entre dois números.
2. Faça uma função que verifica se um número é primo.
3. Faça uma função que recebe uma lista e devolve uma nova lista invertida.



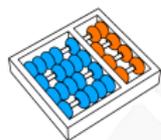
## Exercícios

1. Faça uma função que acha o maior entre dois números.
2. Faça uma função que verifica se um número é primo.
3. Faça uma função que recebe uma lista e devolve uma nova lista invertida.
4. Faça uma função que devolve a lista dos divisores de um número.



## Por que usar funções?

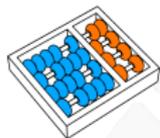
Usamos funções para:



## Por que usar funções?

Usamos funções para:

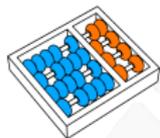
- ▶ Evitar repetição de código.



## Por que usar funções?

Usamos funções para:

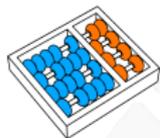
- ▶ Evitar repetição de código.
- ▶ Reutilizar o código de outras pessoas.



## Por que usar funções?

Usamos funções para:

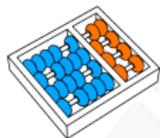
- ▶ Evitar repetição de código.
- ▶ Reutilizar o código de outras pessoas.
- ▶ Permitir que outros reutilizem o nosso código.



## Por que usar funções?

Usamos funções para:

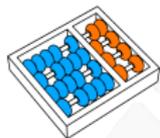
- ▶ Evitar repetição de código.
- ▶ Reutilizar o código de outras pessoas.
- ▶ Permitir que outros reutilizem o nosso código.
- ▶ Deixar o programa mais fácil de entender.



## Por que usar funções?

Usamos funções para:

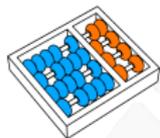
- ▶ Evitar repetição de código.
- ▶ Reutilizar o código de outras pessoas.
- ▶ Permitir que outros reutilizem o nosso código.
- ▶ Deixar o programa mais fácil de entender.
- ▶ Deixar o programa mais fácil de *debuggar*.



## Por que usar funções?

Usamos funções para:

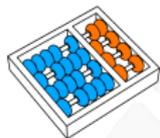
- ▶ Evitar repetição de código.
- ▶ Reutilizar o código de outras pessoas.
- ▶ Permitir que outros reutilizem o nosso código.
- ▶ Deixar o programa mais fácil de entender.
- ▶ Deixar o programa mais fácil de *debuggar*.
- ▶ Criar conjuntos de funções (bibliotecas) úteis.



## Por que usar funções?

Usamos funções para:

- ▶ Evitar repetição de código.
- ▶ Reutilizar o código de outras pessoas.
- ▶ Permitir que outros reutilizem o nosso código.
- ▶ Deixar o programa mais fácil de entender.
- ▶ Deixar o programa mais fácil de *debuggar*.
- ▶ Criar conjuntos de funções (bibliotecas) úteis.
- ▶ Entre muitas outras coisas.

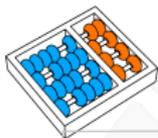


## Por que usar funções?

Usamos funções para:

- ▶ Evitar repetição de código.
- ▶ Reutilizar o código de outras pessoas.
- ▶ Permitir que outros reutilizem o nosso código.
- ▶ Deixar o programa mais fácil de entender.
- ▶ Deixar o programa mais fácil de *debuggar*.
- ▶ Criar conjuntos de funções (bibliotecas) úteis.
- ▶ Entre muitas outras coisas.

O uso de funções é parte fundamental da programação!



## Exemplo

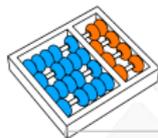
```
1 def primo(p):
2     k = 2
3     while k * k <= p :
4         if p % k == 0:
5             return False
6         k = k + 1
7     return p > 1
8 p = int(input("Entre com p: "))
9 q = int(input("Entre com q : "))
10 if primo(p) and primo(q):
11     print("Ambos são primos")
12 else:
13     print("Pelo menos um deles não é primo")
```



## Exemplo

```
1 def primo(p):
2     k = 2
3     while k * k <= p :
4         if p % k == 0:
5             return False
6         k = k + 1
7     return p > 1
8 p = int(input("Entre com p: "))
9 q = int(input("Entre com q : "))
10 if primo(p) and primo(q):
11     print("Ambos são primos")
12 else:
13     print("Pelo menos um deles não é primo")
```

Vantagens:

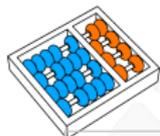


## Exemplo

```
1 def primo(p):
2     k = 2
3     while k * k <= p :
4         if p % k == 0:
5             return False
6         k = k + 1
7     return p > 1
8 p = int(input("Entre com p: "))
9 q = int(input("Entre com q : "))
10 if primo(p) and primo(q):
11     print("Ambos são primos")
12 else:
13     print("Pelo menos um deles não é primo")
```

Vantagens:

- ▶ Podemos chamar a função várias vezes.

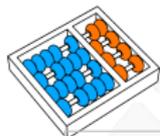


## Exemplo

```
1 def primo(p):
2     k = 2
3     while k * k <= p :
4         if p % k == 0:
5             return False
6         k = k + 1
7     return p > 1
8 p = int(input("Entre com p: "))
9 q = int(input("Entre com q: "))
10 if primo(p) and primo(q):
11     print("Ambos são primos")
12 else:
13     print("Pelo menos um deles não é primo")
```

### Vantagens:

- ▶ Podemos chamar a função várias vezes.
- ▶ Outra pessoa poderia usar a função **primo**.

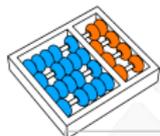


## Exemplo

```
1 def primo(p):
2     k = 2
3     while k * k <= p :
4         if p % k == 0:
5             return False
6         k = k + 1
7     return p > 1
8 p = int(input("Entre com p: "))
9 q = int(input("Entre com q: "))
10 if primo(p) and primo(q):
11     print("Ambos são primos")
12 else:
13     print("Pelo menos um deles não é primo")
```

### Vantagens:

- ▶ Podemos chamar a função várias vezes.
- ▶ Outra pessoa poderia usar a função `primo`.
- ▶ Outra pessoa poderia implementar a função `primo`.



## Exemplo

```
1 def primo(p):
2     k = 2
3     while k * k <= p :
4         if p % k == 0:
5             return False
6         k = k + 1
7     return p > 1
8 p = int(input("Entre com p: "))
9 q = int(input("Entre com q: "))
10 if primo(p) and primo(q):
11     print("Ambos são primos")
12 else:
13     print("Pelo menos um deles não é primo")
```

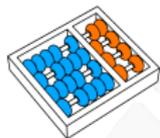
## Vantagens:

- ▶ Podemos chamar a função várias vezes.
- ▶ Outra pessoa poderia usar a função `primo`.
- ▶ Outra pessoa poderia implementar a função `primo`.
- ▶ O código é mais “fácil” de ler.



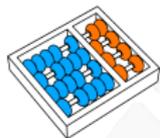
SEM RETORNO E SEM  
PARÂMETROS

Sem retorno e sem parâmetros



## Funções que “não” devolvem valor

Uma função não precisa ter o comando `return`.

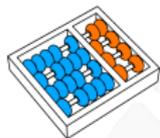


## Funções que “não” devolvem valor

Uma função não precisa ter o comando `return`.

Ex:

```
1 def imprime(lista):  
2     for x in lista:  
3         print(x)
```



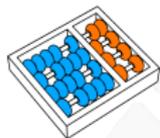
## Funções que “não” devolvem valor

Uma função não precisa ter o comando `return`.

Ex:

```
1 def imprime(lista):  
2     for x in lista:  
3         print(x)
```

Essa função não precisa devolver nada...



## Funções que “não” devolvem valor

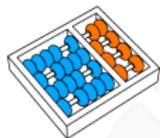
Uma função não precisa ter o comando `return`.

Ex:

```
1 def imprime(lista):  
2     for x in lista:  
3         print(x)
```

Essa função não precisa devolver nada...

Mas, não é bem assim...



## Funções que “não” devolvem valor

Uma função não precisa ter o comando `return`.

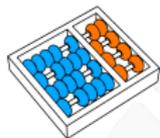
Ex:

```
1 def imprime(lista):  
2     for x in lista:  
3         print(x)
```

Essa função não precisa devolver nada...

Mas, não é bem assim...

```
1 def imprime(lista):  
2     for x in lista:  
3         print(x)  
4  
5 valor = imprime([1, 2, 3, 4])  
6 print(valor)
```



## Funções que “não” devolvem valor

Uma função não precisa ter o comando `return`.

Ex:

```
1 def imprime(lista):  
2     for x in lista:  
3         print(x)
```

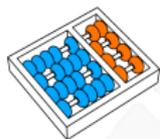
Essa função não precisa devolver nada...

Mas, não é bem assim...

```
1 def imprime(lista):  
2     for x in lista:  
3         print(x)  
4  
5 valor = imprime([1, 2, 3, 4])  
6 print(valor)
```

Será impresso, em cada linha, `1`, `2`, `3`, `4` e `None`.

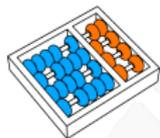
Sem retorno e sem parâmetros



## O tipo `NoneType`

O tipo `NoneType` tem um único valor, o `None`.

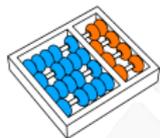
Sem retorno e sem parâmetros



## O tipo `NoneType`

O tipo `NoneType` tem um único valor, o `None`.

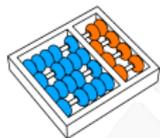
- ▶ O `None` representa o nada...



## O tipo **NoneType**

O tipo **NoneType** tem um único valor, o **None**.

- ▶ O **None** representa o nada...
- ▶ A ideia é que não é um número, não é uma string, etc.

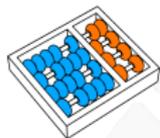


## O tipo `NoneType`

O tipo `NoneType` tem um único valor, o `None`.

- ▶ O `None` representa o nada...
- ▶ A ideia é que não é um número, não é uma string, etc.

É algo bastante comum em linguagens de programação.



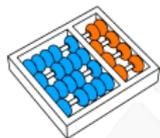
## O tipo `NoneType`

O tipo `NoneType` tem um único valor, o `None`.

- ▶ O `None` representa o nada...
- ▶ A ideia é que não é um número, não é uma string, etc.

É algo bastante comum em linguagens de programação.

- ▶ Em outras linguagens pode chamar: `NULL`, `nil`, entre outros.



## O tipo `NoneType`

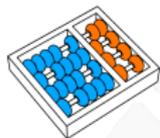
O tipo `NoneType` tem um único valor, o `None`.

- ▶ O `None` representa o nada...
- ▶ A ideia é que não é um número, não é uma string, etc.

É algo bastante comum em linguagens de programação.

- ▶ Em outras linguagens pode chamar: `NULL`, `nil`, entre outros.

Toda função de Python devolve algum valor:



## O tipo `NoneType`

O tipo `NoneType` tem um único valor, o `None`.

- ▶ O `None` representa o nada...
- ▶ A ideia é que não é um número, não é uma string, etc.

É algo bastante comum em linguagens de programação.

- ▶ Em outras linguagens pode chamar: `NULL`, `nil`, entre outros.

Toda função de Python devolve algum valor:

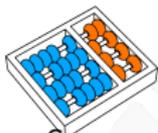
- ▶ Nem que esse valor seja `None` é devolvido.

Sem retorno e sem parâmetros



Um cuidado!

Se executarmos esse código:



## Um cuidado!

Se executarmos esse código:

```
1 n = int(input())
2 lista = le_lista(n)
3 print(soma_valores(lista))
4 def le_lista(n):
5     lista = []
6     for i in range(n):
7         lista.append(float(input()))
8     return lista
9 def soma_valores(lista):
10    soma = 0
11    for x in lista :
12        soma += x
13    return soma
```

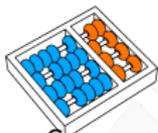


## Um cuidado!

Se executarmos esse código:

```
1 n = int(input())
2 lista = le_lista(n)
3 print(soma_valores(lista))
4 def le_lista(n):
5     lista = []
6     for i in range(n):
7         lista.append(float(input()))
8     return lista
9 def soma_valores(lista):
10    soma = 0
11    for x in lista :
12        soma += x
13    return soma
```

Temos o seguinte erro após digitar o valor de **n**:



## Um cuidado!

Se executarmos esse código:

```
1 n = int(input())
2 lista = le_lista(n)
3 print(soma_valores(lista))
4 def le_lista(n):
5     lista = []
6     for i in range(n):
7         lista.append(float(input()))
8     return lista
9 def soma_valores(lista):
10    soma = 0
11    for x in lista :
12        soma += x
13    return soma
```

Temos o seguinte erro após digitar o valor de **n**:

```
1 Traceback (most recent call last):
2 File " cuidado.py " , line 2 , in <module>
3 lista = le_lista(n)
4 NameError : name le_lista is not defined
```



## Um cuidado!

Se executarmos esse código:

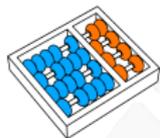
```
1 n = int(input())
2 lista = le_lista(n)
3 print(soma_valores(lista))
4 def le_lista(n):
5     lista = []
6     for i in range(n):
7         lista.append(float(input()))
8     return lista
9 def soma_valores(lista):
10    soma = 0
11    for x in lista :
12        soma += x
13    return soma
```

Temos o seguinte erro após digitar o valor de **n**:

```
1 Traceback (most recent call last):
2 File " cuidado.py " , line 2 , in <module>
3 lista = le_lista(n)
4 NameError : name le_lista is not defined
```

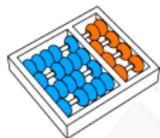
A função ainda não havia sido definida!

Sem retorno e sem parâmetros



## Outras informações

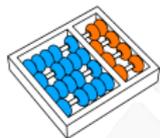
Uma função pode ter zero parâmetros:



## Outras informações

Uma função pode ter zero parâmetros:

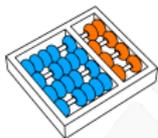
- ▶ Não recebe nada de entrada, mas tem saída...



## Outras informações

Uma função pode ter zero parâmetros:

- ▶ Não recebe nada de entrada, mas tem saída...
- ▶ Ex: função que lê um número.

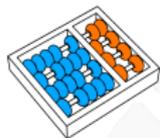


## Outras informações

Uma função pode ter zero parâmetros:

- ▶ Não recebe nada de entrada, mas tem saída...
- ▶ Ex: função que lê um número.

Uma função pode chamar outras funções:



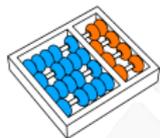
## Outras informações

Uma função pode ter zero parâmetros:

- ▶ Não recebe nada de entrada, mas tem saída...
- ▶ Ex: função que lê um número.

Uma função pode chamar outras funções:

- ▶ Na verdade, pode (inclusive) chamar a si mesmo!



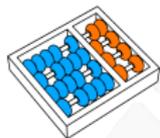
## Outras informações

Uma função pode ter zero parâmetros:

- ▶ Não recebe nada de entrada, mas tem saída...
- ▶ Ex: função que lê um número.

Uma função pode chamar outras funções:

- ▶ Na verdade, pode (inclusive) chamar a si mesmo!
- ▶ Veremos mais sobre isso no futuro!



## Outras informações

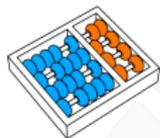
Uma função pode ter zero parâmetros:

- ▶ Não recebe nada de entrada, mas tem saída...
- ▶ Ex: função que lê um número.

Uma função pode chamar outras funções:

- ▶ Na verdade, pode (inclusive) chamar a si mesmo!
- ▶ Veremos mais sobre isso no futuro!

É importante escolher bons nomes para as funções:



## Outras informações

Uma função pode ter zero parâmetros:

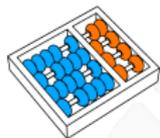
- ▶ Não recebe nada de entrada, mas tem saída...
- ▶ Ex: função que lê um número.

Uma função pode chamar outras funções:

- ▶ Na verdade, pode (inclusive) chamar a si mesmo!
- ▶ Veremos mais sobre isso no futuro!

É importante escolher bons nomes para as funções:

- ▶ Um nome que descreva bem o que ela faz.



## Outras informações

Uma função pode ter zero parâmetros:

- ▶ Não recebe nada de entrada, mas tem saída...
- ▶ Ex: função que lê um número.

Uma função pode chamar outras funções:

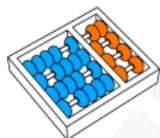
- ▶ Na verdade, pode (inclusive) chamar a si mesmo!
- ▶ Veremos mais sobre isso no futuro!

É importante escolher bons nomes para as funções:

- ▶ Um nome que descreva bem o que ela faz.
- ▶ Mas tente criar um nome razoavelmente curto.



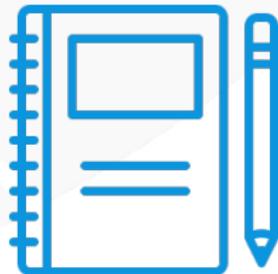
# EXERCÍCIOS

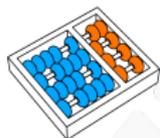


## Listas e repetição



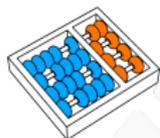
**Vamos fazer alguns exercícios?**





## Exercício 1

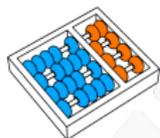
Um número  $n$  é **perfeito** se  $n$  é a soma dos seus divisores próprios.



## Exercício 1

Um número  $n$  é **perfeito** se  $n$  é a soma dos seus divisores próprios.

▶ Ex:  $6 = 1 + 2 + 3$ .

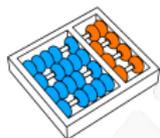


## Exercício 1

Um número  $n$  é **perfeito** se  $n$  é a soma dos seus divisores próprios.

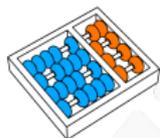
▶ Ex:  $6 = 1 + 2 + 3$ .

Faça uma função que, dado  $n$ , decide se  $n$  é perfeito ou não.



## Exercício 2

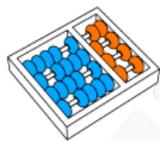
Dois números são ***amigáveis*** se pelo menos um deles for igual à soma dos divisores próprios do outro.



## Exercício 2

Dois números são **amigáveis** se pelo menos um deles for igual à soma dos divisores próprios do outro.

▶ Ex:  $4 = 1 + 3$  é amigável com 9.

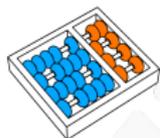


## Exercício 2

Dois números são **amigáveis** se pelo menos um deles for igual à soma dos divisores próprios do outro.

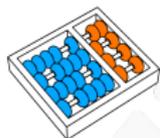
▶ Ex:  $4 = 1 + 3$  é amigável com 9.

Faça uma função que, dados dois números  $m$  e  $n$ , decide se são amigáveis.



### Exercício 3

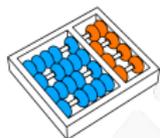
Dado um número inteiro, seu **valor invertido** é ele lido de trás para frente.



### Exercício 3

Dado um número inteiro, seu **valor invertido** é ele lido de trás para frente.

- ▶ Ex: O valor invertido de 1234 é 4321.

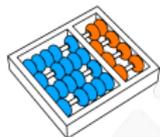


### Exercício 3

Dado um número inteiro, seu **valor invertido** é ele lido de trás para frente.

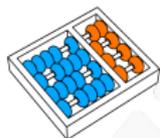
▶ Ex: O valor invertido de 1234 é 4321.

Faça uma função que, dado um número  $n$ , devolve seu valor invertido.



## Exercício 4

Faça uma função que, dado um número inteiro  $n$ , desenhe um triângulo invertido na tela de base  $n$ , usando caracteres '\*'.

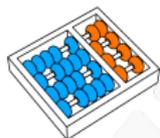


## Exercício 4

Faça uma função que, dado um número inteiro  $n$ , desenhe um triângulo invertido na tela de base  $n$ , usando caracteres '\*'.

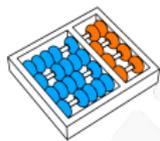
- ▶ Se a base for 5, imprima:





## Exercício 5

Um número é triangular se ele é igual ao produto de três inteiros consecutivos.



## Exercício 5

Um número é triangular se ele é igual ao produto de três inteiros consecutivos.

Faça uma função que, dado um número  $n$ , decide se ele é triangular ou não.

# FUNÇÕES

MC102 - Algoritmos e  
Programação de  
Computadores

Santiago Valdés Ravelo  
[https://ic.unicamp.br/~santiago/  
ravelo@unicamp.br](https://ic.unicamp.br/~santiago/ravelo@unicamp.br)

04/25

9



UNICAMP

