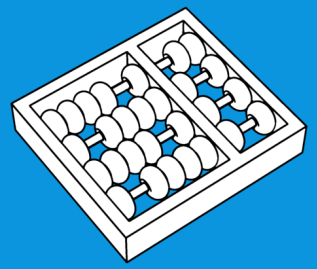
Listas e repetição

MC102 - Algoritmos e Programação de Computadores

6

03/25



Santiago Valdés Ravelo
https://ic.unicamp.br/~santiago/
ravelo@unicamp.br

“Um programa sem um laço e sem uma variável estruturada não vale a pena ser escrito.”

Alan J Perlis.

Dúvidas da aula anterior

Dúvidas selecionadas

- O que é mais importante, a boa estruturação do código ou a efetividade dele?

Dúvidas selecionadas

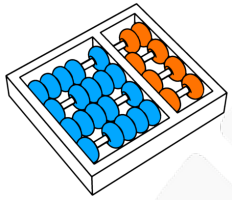
- O que é mais importante, a boa estruturação do código ou a efetividade dele?
- Por que quando eu digo i = i+1 ele entende que a cada i eu devo somar?

Dúvidas selecionadas

- O que é mais importante, a boa estruturação do código ou a efetividade dele?
- Por que quando eu digo i = i+1 ele entende que a cada i eu devo somar?
- Existe i++ em python?

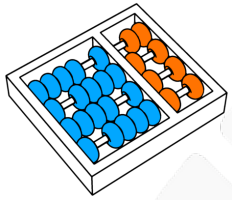
Dúvidas selecionadas

- O que é mais importante, a boa estruturação do código ou a efetividade dele?
- Por que quando eu digo i = i+1 ele entende que a cada i eu devo somar?
- Existe i++ em python?
- Como faço para determinar as condições de entrada e saída do while?



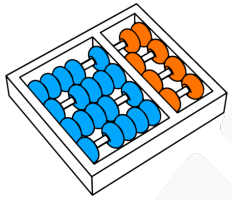
Dúvidas selecionadas

- O que é mais importante, a boa estruturação do código ou a efetividade dele?
- Por que quando eu digo i = i+1 ele entende que a cada i eu devo somar?
- Existe i++ em python?
- Como faço para determinar as condições de entrada e saída do while?
- É possível fazer uma repetição sem definir a variável iterativa antes?



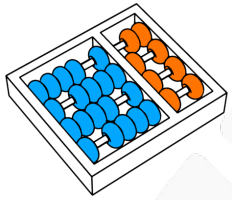
Dúvidas selecionadas

- O que é mais importante, a boa estruturação do código ou a efetividade dele?
- Por que quando eu digo i = i+1 ele entende que a cada i eu devo somar?
- Existe i++ em python?
- Como faço para determinar as condições de entrada e saída do while?
- É possível fazer uma repetição sem definir a variável iterativa antes?
- O teste de mesa é realmente útil no dia a dia? Por que eu usaria o teste de mesa ao invés de um debugger?



Dúvidas selecionadas

- O que é mais importante, a boa estruturação do código ou a efetividade dele?
- Por que quando eu digo i = i+1 ele entende que a cada i eu devo somar?
- Existe i++ em python?
- Como faço para determinar as condições de entrada e saída do while?
- É possível fazer uma repetição sem definir a variável iterativa antes?
- O teste de mesa é realmente útil no dia a dia? Por que eu usaria o teste de mesa ao invés de um debugger?
- Qual a diferença entre a, b, c = input(). split() e a, b, c = map(int, input(). split())

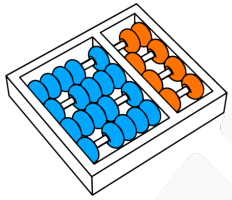


Dúvidas selecionadas

- O que é mais importante, a boa estruturação do código ou a efetividade dele?
- Por que quando eu digo i = i+1 ele entende que a cada i eu devo somar?
- Existe i++ em python?
- Como faço para determinar as condições de entrada e saída do while?
- É possível fazer uma repetição sem definir a variável iterativa antes?
- O teste de mesa é realmente útil no dia a dia? Por que eu usaria o teste de mesa ao invés de um debugger?
- Qual a diferença entre a, b, c = input(). split() e a, b, c = map(int, input(). split())
- Para o split separar os valores em diferentes variáveis eles precisam ser enviados com espaço?

Dúvidas selecionadas

- O que é mais importante, a boa estruturação do código ou a efetividade dele?
- Por que quando eu digo i = i+1 ele entende que a cada i eu devo somar?
- Existe i++ em python?
- Como faço para determinar as condições de entrada e saída do while?
- É possível fazer uma repetição sem definir a variável iterativa antes?
- O teste de mesa é realmente útil no dia a dia? Por que eu usaria o teste de mesa ao invés de um debugger?
- Qual a diferença entre a, b, c = input(). split() e a, b, c = map(int, input(). split())
- Para o split separar os valores em diferentes variáveis eles precisam ser enviados com espaço?
- O que acontece caso a condição do While sempre ser True?



Dúvidas selecionadas

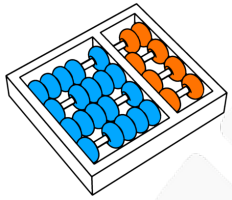
- O que é mais importante, a boa estruturação do código ou a efetividade dele?
- Por que quando eu digo i = i+1 ele entende que a cada i eu devo somar?
- Existe i++ em python?
- Como faço para determinar as condições de entrada e saída do while?
- É possível fazer uma repetição sem definir a variável iterativa antes?
- O teste de mesa é realmente útil no dia a dia? Por que eu usaria o teste de mesa ao invés de um debugger?
- Qual a diferença entre a, b, c = input(). split() e a, b, c = map(int, input(). split())
- Para o split separar os valores em diferentes variáveis eles precisam ser enviados com espaço?
- O que acontece caso a condição do While sempre ser True?
- É possível rodar dois while simultaneamente?

Dúvidas selecionadas

- O que é mais importante, a boa estruturação do código ou a efetividade dele?
- Por que quando eu digo i = i+1 ele entende que a cada i eu devo somar?
- Existe i++ em python?
- Como faço para determinar as condições de entrada e saída do while?
- É possível fazer uma repetição sem definir a variável iterativa antes?
- O teste de mesa é realmente útil no dia a dia? Por que eu usaria o teste de mesa ao invés de um debugger?
- Qual a diferença entre a, b, c = input(). split() e a, b, c = map(int, input(). split())
- Para o split separar os valores em diferentes variáveis eles precisam ser enviados com espaço?
- O que acontece caso a condição do While sempre ser True?
- É possível rodar dois while simultaneamente?
- Não sei se entendi muito bem a razão de qualquer divisor possível de um número ser menor que a raiz do número.

Dúvidas selecionadas

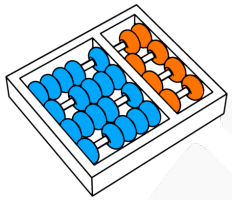
- O que é mais importante, a boa estruturação do código ou a efetividade dele?
- Por que quando eu digo i = i+1 ele entende que a cada i eu devo somar?
- Existe i++ em python?
- Como faço para determinar as condições de entrada e saída do while?
- É possível fazer uma repetição sem definir a variável iterativa antes?
- O teste de mesa é realmente útil no dia a dia? Por que eu usaria o teste de mesa ao invés de um debugger?
- Qual a diferença entre a, b, c = input(). split() e a, b, c = map(int, input(). split())
- Para o split separar os valores em diferentes variáveis eles precisam ser enviados com espaço?
- O que acontece caso a condição do While sempre ser True?
- É possível rodar dois while simultaneamente?
- Não sei se entendi muito bem a razão de qualquer divisor possível de um número ser menor que a raiz do número.
- É possível colocar um else depois de um while para caso a condição do while seja falsa?



Dúvidas selecionadas

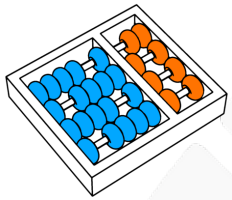
- O que é mais importante, a boa estruturação do código ou a efetividade dele?
- Por que quando eu digo i = i+1 ele entende que a cada i eu devo somar?
- Existe i++ em python?
- Como faço para determinar as condições de entrada e saída do while?
- É possível fazer uma repetição sem definir a variável iterativa antes?
- O teste de mesa é realmente útil no dia a dia? Por que eu usaria o teste de mesa ao invés de um debugger?
- Qual a diferença entre a, b, c = input(). split() e a, b, c = map(int, input(). split())
- Para o split separar os valores em diferentes variáveis eles precisam ser enviados com espaço?
- O que acontece caso a condição do While sempre ser True?
- É possível rodar dois while simultaneamente?
- Não sei se entendi muito bem a razão de qualquer divisor possível de um número ser menor que a raiz do número.
- É possível colocar um else depois de um while para caso a condição do while seja falsa?
- Existe algum lugar que tenha a resolução dos exercícios dos slides postada?

Motivação



Média: média das notas

Faça um programa que lê o número **n** de alunos, as notas dos **n** alunos e informa a média da sala.



Problema: média das notas. Algoritmo

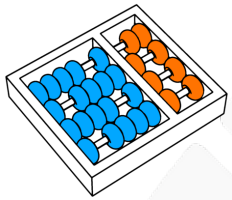
```
leia n
soma ← 0
i ← 1 enquanto i ≤ n
    leia nota
    soma ← soma + nota
    i ← i + 1
escreva soma/n
```

Problema: média das notas. Código

```
1  n = int(input())
2  soma = 0
3  i = 0
4  while i < n:
5      nota = float(input())
6      soma += nota
7      i += 1
8  print('{:.2f}'.format(soma/n))
9  # formata para duas casas decimais
```

Problema: maiores que a média

Faça um programa que lê o número **n** de alunos, as notas dos **n** alunos, informa a média da sala e imprime as notas dos maiores que a média.



Problema: maiores que a média. Algoritmo

```
leia n
soma ← 0
i ← 1
enquanto i ≤ n
    leia nota
    soma ← soma + nota
    i ← i + 1
escreva soma/n
```

ESCREVA AS NOTAS SUPERIORES QUE A MÉDIA!

Pergunta

Como armazenar vários valores?

Armazenando dados

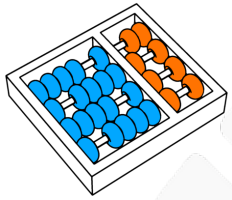
Armazenando dados

O que queremos:

Armazenando dados

O que queremos:

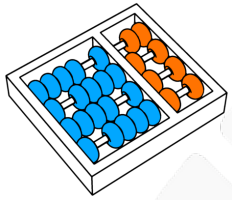
- Ter fácil acesso aos dados.



Armazenando dados

O que queremos:

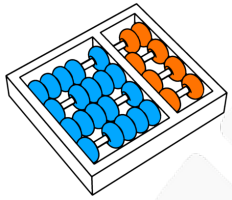
- Ter fácil acesso aos dados.
- Para não ter tantas variáveis.



Armazenando dados

O que queremos:

- Ter fácil acesso aos dados.
- Para não ter tantas variáveis.
- Para ter um código mais simples.

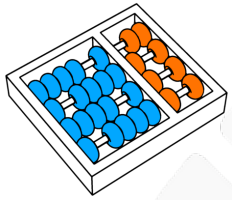


Armazenando dados

O que queremos:

- Ter fácil acesso aos dados.
- Para não ter tantas variáveis.
- Para ter um código mais simples.

Para tanto, usaremos listas!



Armazenando dados

O que queremos:

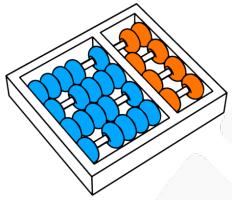
- Ter fácil acesso aos dados.
- Para não ter tantas variáveis.
- Para ter um código mais simples.

Para tanto, usaremos listas!

- E, futuramente, outras formas de acesso.

Listas

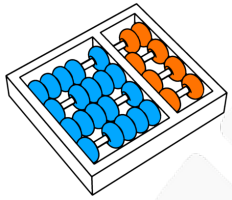
Lista (**list**) é um tipo do Python:



Listas

Lista (**list**) é um tipo do Python:

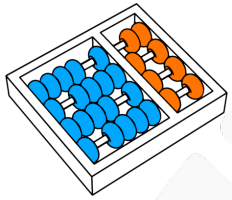
- Permite armazenar uma grande quantidade de dados.



Listas

Lista (**list**) é um tipo do Python:

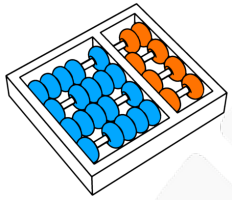
- Permite armazenar uma grande quantidade de dados.
  - Tanto quanto você queira...



Listas

Lista (**list**) é um tipo do Python:

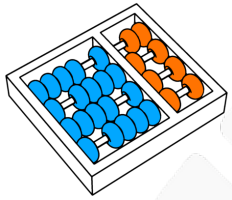
- Permite armazenar uma grande quantidade de dados.
  - Tanto quanto você queira...
  - Claro, até o limite de memória do computador.



Listas

Lista (**list**) é um tipo do Python:

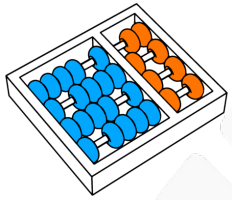
- Permite armazenar uma grande quantidade de dados.
  - Tanto quanto você queira...
  - Claro, até o limite de memória do computador.
- Permite acessar os dados usando um ***ÍNDICE***:



Listas

Lista (**list**) é um tipo do Python:

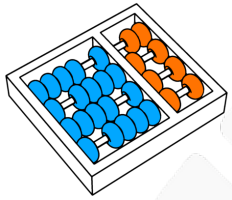
- Permite armazenar uma grande quantidade de dados.
  - Tanto quanto você queira...
  - Claro, até o limite de memória do computador.
- Permite acessar os dados usando um ***ÍNDICE***:
  - Ex: **lista[0]**, **lista[1]**, ...



Listas

Lista (**list**) é um tipo do Python:

- Permite armazenar uma grande quantidade de dados.
  - Tanto quanto você queira...
  - Claro, até o limite de memória do computador.
- Permite acessar os dados usando um ***ÍNDICE***:
  - Ex: **lista[0]**, **lista[1]**, ...
  - Tanto para escrita quanto para leitura.



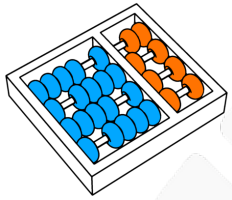
Listas

Lista (**list**) é um tipo do Python:

- Permite armazenar uma grande quantidade de dados.
  - Tanto quanto você queira...
  - Claro, até o limite de memória do computador.
- Permite acessar os dados usando um ***ÍNDICE***:
  - Ex: **lista[0]**, **lista[1]**, ...
  - Tanto para escrita quanto para leitura.
  - O índice começa em zero.

Criando Listas e Adicionando Itens

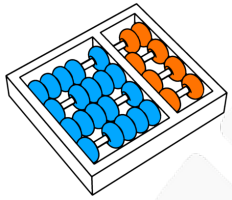
Criando uma lista vazia (sem nada armazenado):



Criando Listas e Adicionando Itens

Criando uma lista vazia (sem nada armazenado):

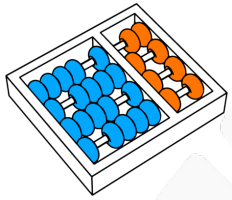
- **lista = []** (mais usado)



Criando Listas e Adicionando Itens

Criando uma lista vazia (sem nada armazenado):

- **lista = []** (mais usado)
- **lista = list()**

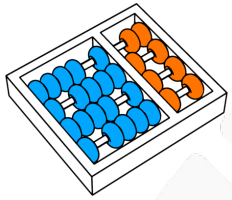


Criando Listas e Adicionando Itens

Criando uma lista vazia (sem nada armazenado):

- **lista = []** (mais usado)
- **lista = list()**

Criando uma lista com conteúdo:



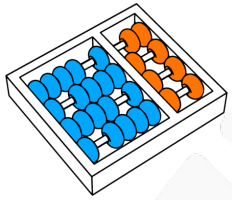
Criando Listas e Adicionando Itens

Criando uma lista vazia (sem nada armazenado):

- **lista = []** (mais usado)
- **lista = list()**

Criando uma lista com conteúdo:

- **lista = [1, 7, 2, 2, 15]**.



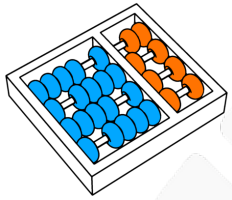
Criando Listas e Adicionando Itens

Criando uma lista vazia (sem nada armazenado):

- **lista = []** (mais usado)
- **lista = list()**

Criando uma lista com conteúdo:

- **lista = [1, 7, 2, 2, 15]**.
- **lista = ["ana", "joão", "pedro"]**.



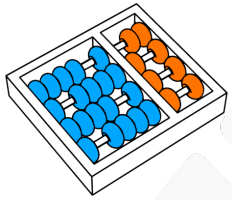
Criando Listas e Adicionando Itens

Criando uma lista vazia (sem nada armazenado):

- **lista = []** (mais usado)
- **lista = list()**

Criando uma lista com conteúdo:

- **lista = [1, 7, 2, 2, 15]**.
- **lista = ["ana", "joão", "pedro"]**.
- **lista = [1.3, 7.5, -2.1]**.



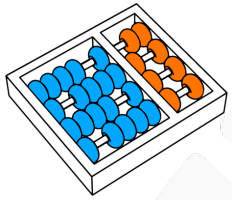
Criando Listas e Adicionando Itens

Criando uma lista vazia (sem nada armazenado):

- **lista = []** (mais usado)
- **lista = list()**

Criando uma lista com conteúdo:

- **lista = [1, 7, 2, 2, 15]**.
- **lista = ["ana", "joão", "pedro"]**.
- **lista = [1.3, 7.5, -2.1]**.
- **lista = [x, y, z]** ⟵ Não é uma lista de variáveis!



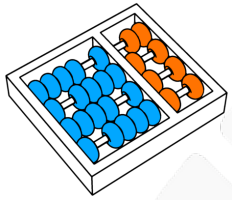
Criando Listas e Adicionando Itens

Criando uma lista vazia (sem nada armazenado):

- **lista = []** (mais usado)
- **lista = list()**

Criando uma lista com conteúdo:

- **lista = [1, 7, 2, 2, 15]**.
- **lista = ["ana", "joão", "pedro"]**.
- **lista = [1.3, 7.5, -2.1]**.
- **lista = [x, y, z]** ⟵ Não é uma lista de variáveis!
- **lista = [1, "mc102", 3.7]**.



Criando Listas e Adicionando Itens

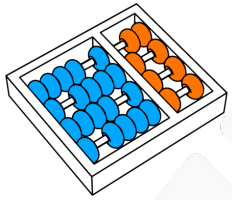
Criando uma lista vazia (sem nada armazenado):

- **lista = []** (mais usado)
- **lista = list()**

Criando uma lista com conteúdo:

- **lista = [1, 7, 2, 2, 15]**.
- **lista = ["ana", "joão", "pedro"]**.
- **lista = [1.3, 7.5, -2.1]**.
- **lista = [x, y, z]** ⟵ Não é uma lista de variáveis!
- **lista = [1, "mc102", 3.7]**.

lista.append(x):



Criando Listas e Adicionando Itens

Criando uma lista vazia (sem nada armazenado):

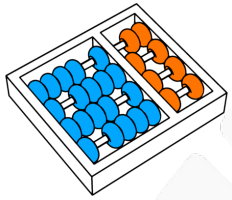
- **lista = []** (mais usado)
- **lista = list()**

Criando uma lista com conteúdo:

- **lista = [1, 7, 2, 2, 15]**.
- **lista = ["ana", "joão", "pedro"]**.
- **lista = [1.3, 7.5, -2.1]**.
- **lista = [x, y, z]** ⟵ Não é uma lista de variáveis!
- **lista = [1, "mc102", 3.7]**.

lista.append(x):

- Insere o valor **x** no final da lista.



Criando Listas e Adicionando Itens

Criando uma lista vazia (sem nada armazenado):

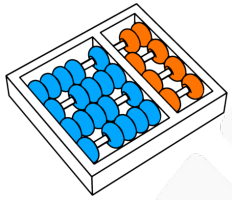
- **lista = []** (mais usado)
- **lista = list()**

Criando uma lista com conteúdo:

- **lista = [1, 7, 2, 2, 15]**.
- **lista = ["ana", "joão", "pedro"]**.
- **lista = [1.3, 7.5, -2.1]**.
- **lista = [x, y, z]** ⟵ Não é uma lista de variáveis!
- **lista = [1, "mc102", 3.7]**.

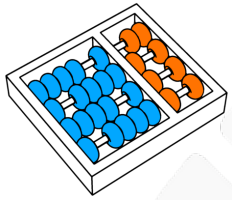
lista.append(x):

- Insere o valor **x** no final da lista.
  - **x** pode ser variável, constante ou uma expressão.



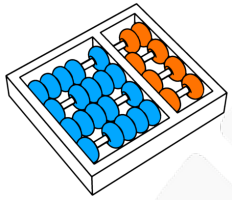
Exemplo

```
>>> l = []
```



Exemplo

```
>>> l = []
>>> l
```



Exemplo

```
>>> l = []
>>> l
[]
```

Exemplo

```
>>> l = []
>>> l
[]
>>> type(l)
```

Exemplo

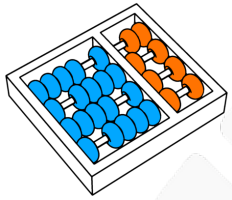
```
>>> l = []
>>> l
[]
>>> type(l)
<class 'list'>
```

Exemplo

```
>>> l = []
>>> l
[]
>>> type(l)
<class 'list'>
>>> l.append("Ana")
```

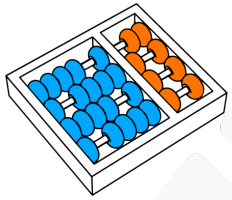
Exemplo

```
>>> l = []
>>> l
[]
>>> type(l)
<class 'list'>
>>> l.append("Ana")
>>> l.append("Beto")
```



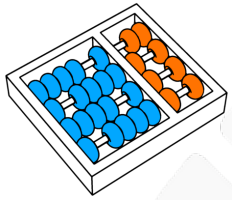
Exemplo

```
>>> l = []
>>> l
[]
>>> type(l)
<class 'list'>
>>> l.append("Ana")
>>> l.append("Beto")
>>> l.append("Carlos")
```



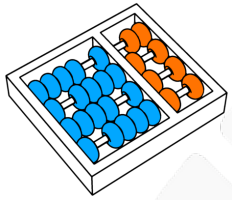
Exemplo

```
1 >>> l = []
2 >>> l
3 []
4 >>> type(l)
5 <class 'list'>
6 >>> l.append("Ana")
7 >>> l.append("Beto")
8 >>> l.append("Carlos")
9 >>> l
```



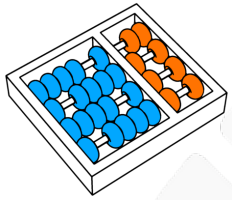
Exemplo

```
>>> l = []
>>> l
[]
>>> type(l)
<class 'list'>
>>> l.append("Ana")
>>> l.append("Beto")
>>> l.append("Carlos")
>>> l
['Ana', 'Beto', 'Carlos']
```



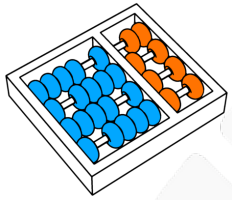
Exemplo

```
>>> l = []
>>> l
[]
>>> type(l)
<class 'list'>
>>> l.append("Ana")
>>> l.append("Beto")
>>> l.append("Carlos")
>>> l
['Ana', 'Beto', 'Carlos']
>>> l[0]
```



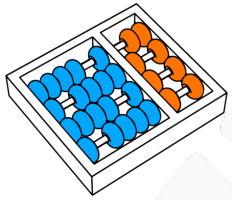
Exemplo

```
>>> l = []
>>> l
[]
>>> type(l)
<class 'list'>
>>> l.append("Ana")
>>> l.append("Beto")
>>> l.append("Carlos")
>>> l
['Ana', 'Beto', 'Carlos']
>>> l[0]
'Ana'
```



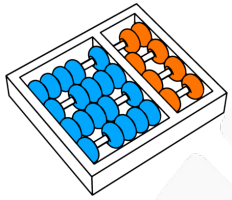
Exemplo

```
>>> l = []
>>> l
[]
>>> type(l)
<class 'list'>
>>> l.append("Ana")
>>> l.append("Beto")
>>> l.append("Carlos")
>>> l
['Ana', 'Beto', 'Carlos']
>>> l[0]
'Ana'
>>> l[1]
```



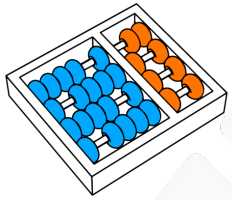
Exemplo

```
>>> l = []
>>> l
[]
>>> type(l)
<class 'list'>
>>> l.append("Ana")
>>> l.append("Beto")
>>> l.append("Carlos")
>>> l
['Ana', 'Beto', 'Carlos']
>>> l[0]
'Ana'
>>> l[1]
'Beto'
```



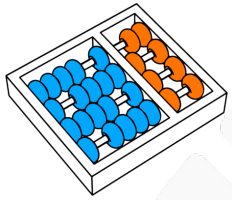
Exemplo

```
>>> l = []
>>> l
[]
>>> type(l)
<class 'list'>
>>> l.append("Ana")
>>> l.append("Beto")
>>> l.append("Carlos")
>>> l
['Ana', 'Beto', 'Carlos']
>>> l[0]
'Ana'
>>> l[1]
'Beto'
>>> l[2]
```



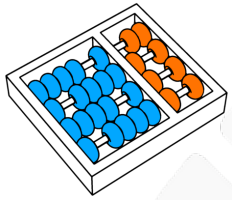
Exemplo

```
>>> l = []
>>> l
[]
>>> type(l)
<class 'list'>
>>> l.append("Ana")
>>> l.append("Beto")
>>> l.append("Carlos")
>>> l
['Ana', 'Beto', 'Carlos']
>>> l[0]
'Ana'
>>> l[1]
'Beto'
>>> l[2]
'Carlos'
```



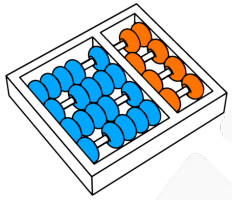
Exemplo

```
>>> l = []
>>> l
[]
>>> type(l)
<class 'list'>
>>> l.append("Ana")
>>> l.append("Beto")
>>> l.append("Carlos")
>>> l
['Ana', 'Beto', 'Carlos']
>>> l[0]
'Ana'
>>> l[1]
'Beto'
>>> l[2]
'Carlos'
>>> l[3]
```



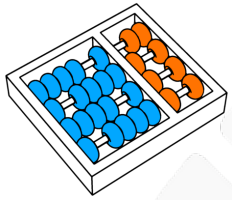
Exemplo

```
>>> l = []
>>> l
[]
>>> type(l)
<class 'list'>
>>> l.append("Ana")
>>> l.append("Beto")
>>> l.append("Carlos")
>>> l
['Ana', 'Beto', 'Carlos']
>>> l[0]
'Ana'
>>> l[1]
'Beto'
>>> l[2]
'Carlos'
>>> l[3]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IndexError: list index out of range
```



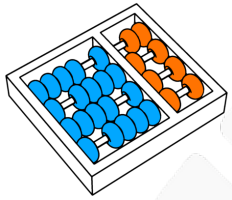
Exemplo

```
>>> l = []
>>> l
[]
>>> type(l)
<class 'list'>
>>> l.append("Ana")
>>> l.append("Beto")
>>> l.append("Carlos")
>>> l
['Ana', 'Beto', 'Carlos']
>>> l[0]
'Ana'
>>> l[1]
'Beto'
>>> l[2]
'Carlos'
>>> l[3]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IndexError: list index out of range
>>> l[1] = 'Roberto'
```



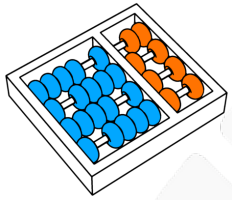
Exemplo

```
>>> l = []
>>> l
[]
>>> type(l)
<class 'list'>
>>> l.append("Ana")
>>> l.append("Beto")
>>> l.append("Carlos")
>>> l
['Ana', 'Beto', 'Carlos']
>>> l[0]
'Ana'
>>> l[1]
'Beto'
>>> l[2]
'Carlos'
>>> l[3]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IndexError: list index out of range
>>> l[1] = 'Roberto'
>>> l
```



Exemplo

```
>>> l = []
>>> l
[]
>>> type(l)
<class 'list'>
>>> l.append("Ana")
>>> l.append("Beto")
>>> l.append("Carlos")
>>> l
['Ana', 'Beto', 'Carlos']
>>> l[0]
'Ana'
>>> l[1]
'Beto'
>>> l[2]
'Carlos'
>>> l[3]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IndexError: list index out of range
>>> l[1] = 'Roberto'
>>> l
['Ana', 'Roberto', 'Carlos']
```

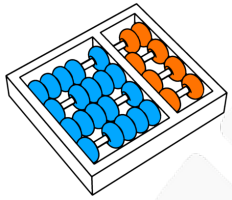


Maiores que a média

```
1  n = int(input())
2  soma = 0
3  i = 0
4  notas = []
5  while i < n:
6      nota = float(input())
7      soma += nota
8      notas.append(nota)
9      i += 1
10 media = soma / n
11 print('{:.2f}'.format(media))
12 i = 0
13 while i < n:
14     if notas[i] > media:
15         print('{:.2f}'.format(notas[i]))
16     i += 1
```

Lendo uma lista de nomes

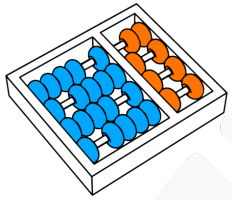
Se tivermos a quantidade **n** de nomes, podemos fazer:



Lendo uma lista de nomes

Se tivermos a quantidade **n** de nomes, podemos fazer:

```
1  n = int(input("Entre com a quantidade de nomes: "))
2  l = []
3  i = 0
4  while i < n:
5      nome = input("Entre com o nome: ")
6      l.append(nome)
7      i += 1
8  print("Nomes digitados:")
9  i = 0
10 while i < n:
11     print(l[i])
12     i += 1
```

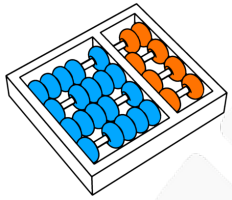


Lendo uma lista de nomes

Se tivermos a quantidade **n** de nomes, podemos fazer:

```
1  n = int(input("Entre com a quantidade de nomes: "))
2  l = []
3  i = 0
4  while i < n:
5      nome = input("Entre com o nome: ")
6      l.append(nome)
7      i += 1
8  print("Nomes digitados:")
9  i = 0
10 while i < n:
11     print(l[i])
12     i += 1
```

Exercício: faça uma versão onde a quantidade de nomes não é conhecida de antemão.



Lendo uma lista de nomes

Se tivermos a quantidade **n** de nomes, podemos fazer:

```
1  n = int(input("Entre com a quantidade de nomes: "))
2  l = []
3  i = 0
4  while i < n:
5      nome = input("Entre com o nome: ")
6      l.append(nome)
7      i += 1
8  print("Nomes digitados:")
9  i = 0
10 while i < n:
11     print(l[i])
12     i += 1
```

Exercício: faça uma versão onde a quantidade de nomes não é conhecida de antemão.

- Considere que o usuário pressiona enter quando não quer dar mais nomes.

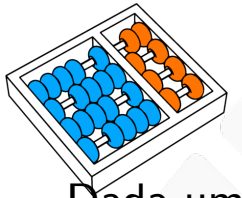
Elemento está na lista?

Dada uma sequência de inteiros e um inteiro k queremos saber se k está na sequência.

Elemento está na lista?

Dada uma sequência de inteiros e um inteiro k queremos saber se k está na sequência.

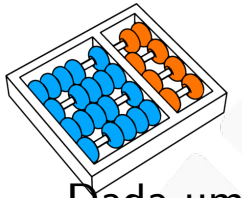
- Se desse k primeiro, não precisava armazenar a sequência.



Elemento está na lista?

Dada uma sequência de inteiros e um inteiro k queremos saber se k está na sequência.

- Se desse k primeiro, não precisava armazenar a sequência.
- Como não sabemos o k antes, precisamos guardá-la!

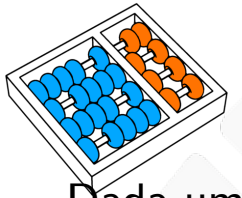


Elemento está na lista?

Dada uma sequência de inteiros e um inteiro k queremos saber se k está na sequência.

- Se desse k primeiro, não precisava armazenar a sequência.
- Como não sabemos o k antes, precisamos guardá-la!

```
1  n = int(input("Número de elementos: "))
2  l = []
3  i = 0
4  while i < n:
5      l.append(int(input("Entre com o número: ")))
6      i = i + 1
7  k = int(input("k: "))
8  i = 0
9  encontrou = False
10 while i < n:
11     if l[i] == k:
12         encontrou = True
13     i += 1
14 print(encontrou)
```



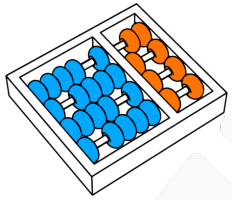
Elemento está na lista?

Dada uma sequência de inteiros e um inteiro k queremos saber se k está na sequência.

- Se desse k primeiro, não precisava armazenar a sequência.
- Como não sabemos o k antes, precisamos guardá-la!

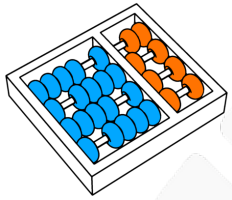
```
1  n = int(input("Número de elementos: "))
2  l = []
3  i = 0
4  while i < n:
5      l.append(int(input("Entre com o número: ")))
6      i = i + 1
7  k = int(input("k: "))
8  i = 0
9  encontrou = False
10 while i < n:
11     if l[i] == k:
12         encontrou = True
13     i += 1
14 print(encontrou)
```

É muito comum perguntar se um elemento está em uma lista, para isso Python tem o comando **in**.



Comando

```
1 n = int(input("Número de elementos: "))
2 l = []
3 i = 0
4 while i < n:
5     l.append(int(input("Entre com o número: ")))
6     i += 1
7 k = int(input("k: "))
8 print(k in l)
```



Comando

```
1 n = int(input("Número de elementos: "))
2 l = []
3 i = 0
4 while i < n:
5     l.append(int(input("Entre com o número: ")))
6     i += 1
7 k = int(input("k: "))
8 print(k in l)
```

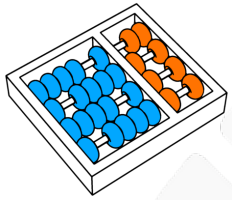
k in l é:

Comando

```
1 n = int(input("Número de elementos: "))
2 l = []
3 i = 0
4 while i < n:
5     l.append(int(input("Entre com o número: ")))
6     i += 1
7 k = int(input("k: "))
8 print(k in l)
```

k in l é:

- **True** se **k** está em **l**.



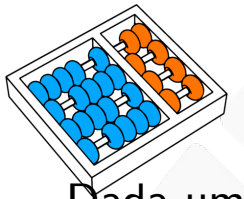
Comando

```
1 n = int(input("Número de elementos: "))
2 l = []
3 i = 0
4 while i < n:
5     l.append(int(input("Entre com o número: ")))
6     i += 1
7 k = int(input("k: "))
8 print(k in l)
```

k in l é:

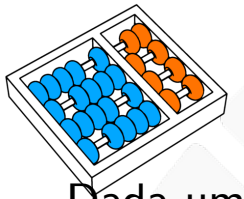
- **True** se **k** está em **l**.
- **False** se **k** não está em **l**.

Outra forma de repetir



Contando as repetições

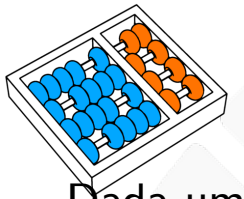
Dada uma sequência de inteiros e um inteiro k queremos saber quantas vezes k está na sequência:



Contando as repetições

Dada uma sequência de inteiros e um inteiro k queremos saber quantas vezes k está na sequência:

- Poderíamos contar usando **while**.

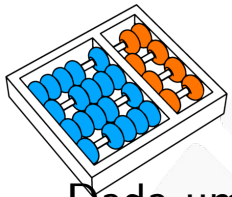


Contando as repetições

Dada uma sequência de inteiros e um inteiro k queremos saber quantas vezes k está na sequência:

- Poderíamos contar usando **while**.

É tão comum percorrer uma lista, que temos algo mais simples:



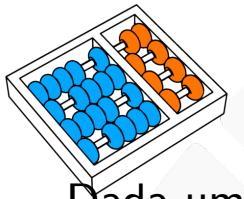
Contando as repetições

Dada uma sequência de inteiros e um inteiro k queremos saber quantas vezes k está na sequência:

- Poderíamos contar usando **while**.

É tão comum percorrer uma lista, que temos algo mais simples:

```
1  n = int(input("Número de elementos: "))
2  l = []
3  i = 0
4  while i < n:
5      l.append(int(input("Entre com o número: ")))
6      i += 1
7  k = int(input("k: "))
8  conta = 0
9  for x in l: # para cada elemento x da lista l faça
10     if x == k:
11         conta += 1
12 print(conta)
```



Contando as repetições

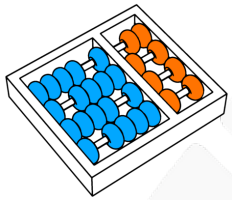
Dada uma sequência de inteiros e um inteiro k queremos saber quantas vezes k está na sequência:

- Poderíamos contar usando **while**.

É tão comum percorrer uma lista, que temos algo mais simples:

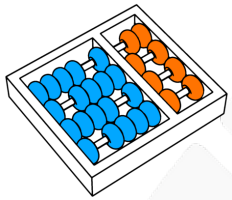
```
1  n = int(input("Número de elementos: "))
2  l = []
3  i = 0
4  while i < n:
5      l.append(int(input("Entre com o número: ")))
6      i += 1
7  k = int(input("k: "))
8  conta = 0
9  for x in l: # para cada elemento x da lista l faça
10     if x == k:
11         conta += 1
12 print(conta)
```

Note que o significado de **in** nesse caso é levemente diferente...



Padrão de contagem

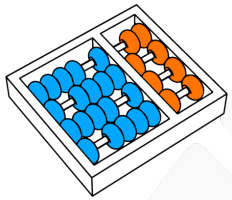
É comum contarmos de um número até outro em um laço.



Padrão de contagem

É comum contarmos de um número até outro em um laço.

```
1  i = 0
2  while i < n:
3      l.append(int(input("Entre com o número: ")))
4      i += 1
```

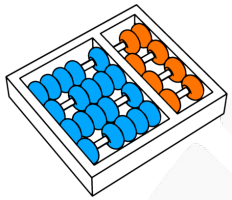


Padrão de contagem

É comum contarmos de um número até outro em um laço.

```
1 i = 0
2 while i < n:
3     l.append(int(input("Entre com o número: ")))
4     i += 1
```

Há uma estrutura que fica sempre aparecendo:



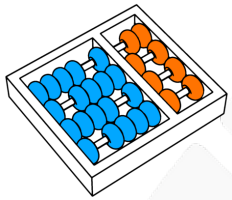
Padrão de contagem

É comum contarmos de um número até outro em um laço.

```
1 i = 0
2 while i < n:
3     l.append(int(input("Entre com o número: ")))
4     i += 1
```

Há uma estrutura que fica sempre aparecendo:

- Inicializa a variável com zero (linha 1).



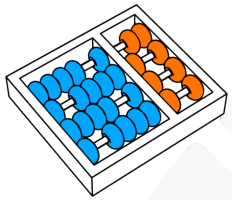
Padrão de contagem

É comum contarmos de um número até outro em um laço.

```
1 i = 0
2 while i < n:
3     l.append(int(input("Entre com o número: ")))
4     i += 1
```

Há uma estrutura que fica sempre aparecendo:

- Inicializa a variável com zero (linha 1).
- Executa um laço até chegar em um valor (linha 2).



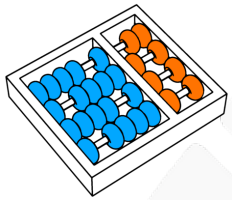
Padrão de contagem

É comum contarmos de um número até outro em um laço.

```
1 i = 0
2 while i < n:
3     l.append(int(input("Entre com o número: ")))
4     i += 1
```

Há uma estrutura que fica sempre aparecendo:

- Inicializa a variável com zero (linha 1).
- Executa um laço até chegar em um valor (linha 2).
- Faz alguma coisa (linha 3).



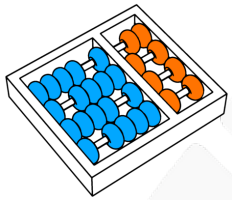
Padrão de contagem

É comum contarmos de um número até outro em um laço.

```
1 i = 0
2 while i < n:
3     l.append(int(input("Entre com o número: ")))
4     i += 1
```

Há uma estrutura que fica sempre aparecendo:

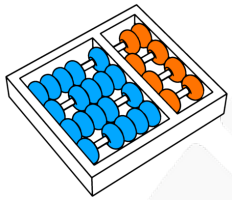
- Inicializa a variável com zero (linha 1).
- Executa um laço até chegar em um valor (linha 2).
- Faz alguma coisa (linha 3).
- Soma um na variável ao final do laço (linha 4).



Reescrevendo usando for e range

Ao invés de:

```
1  i = 0
2  while i < n:
3      l.append(int(input("Entre com o número: ")))
4      i += 1
```



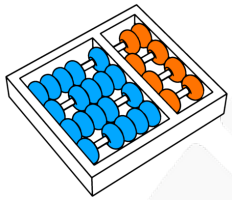
Reescrevendo usando for e range

Ao invés de:

```
1  i = 0
2  while i < n:
3      l.append(int(input("Entre com o número: ")))
4      i += 1
```

Podemos escrever:

```
1  for i in range(n):
2      l.append(int(input("Entre com o número: ")))
```



Reescrevendo usando for e range

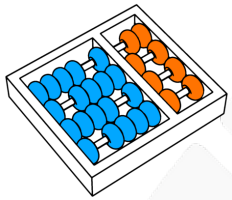
Ao invés de:

```
1  i = 0
2  while i < n:
3      l.append(int(input("Entre com o número: ")))
4      i += 1
```

Podemos escrever:

```
1  for i in range(n):
2      l.append(int(input("Entre com o número: ")))
```

O **range(n)** é como se fosse a lista **[0, 1, ..., n - 1]**.



Reescrevendo usando for e range

Ao invés de:

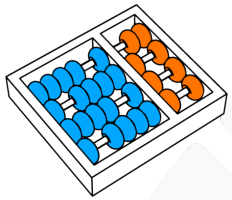
```
1  i = 0
2  while i < n:
3      l.append(int(input("Entre com o número: ")))
4      i += 1
```

Podemos escrever:

```
1  for i in range(n):
2      l.append(int(input("Entre com o número: ")))
```

O **range(n)** é como se fosse a lista **[0, 1, ..., n - 1]**.

- Mas não é uma lista.



Reescrevendo usando for e range

Ao invés de:

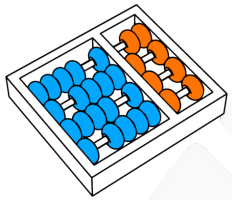
```
1  i = 0
2  while i < n:
3      l.append(int(input("Entre com o número: ")))
4      i += 1
```

Podemos escrever:

```
1  for i in range(n):
2      l.append(int(input("Entre com o número: ")))
```

O **range(n)** é como se fosse a lista **[0, 1, ..., n - 1]**.

- Mas não é uma lista.
- Seu tipo é **range**.



Reescrevendo usando for e range

Ao invés de:

```
1  i = 0
2  while i < n:
3      l.append(int(input("Entre com o número: ")))
4      i += 1
```

Podemos escrever:

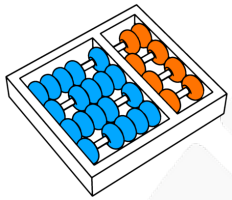
```
1  for i in range(n):
2      l.append(int(input("Entre com o número: ")))
```

O **range(n)** é como se fosse a lista **[0, 1, ..., n - 1]**.

- Mas não é uma lista.
- Seu tipo é **range**.
- Pode ser convertido em lista usando **list(range(n))**.

Revisitando soluções

Maiores que a média:



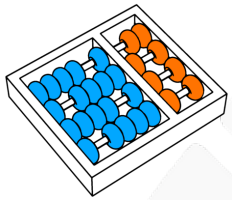
Revisitando soluções

Maiores que a média:

```
1  n = int(input())
2  soma = 0
3  notas = []
4  for i in range(n):
5      nota = float(input())
6      soma += nota
7      notas.append(nota)
8  media = soma / n
9  print('{:.2f}'.format(media))
10 for nota in notas:
11     if nota > media:
12         print('{:.2f}'.format(nota))
```

Revisitando soluções

Imprimindo os próximos k números:



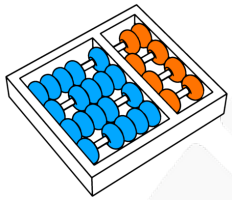
Revisitando soluções

Imprimindo os próximos k números:

```
1 n = int(input("Entre com n: "))
2 k = int(input("Entre com k: "))
3 for i in range(k):
4     print(n + i)
```

Revisitando soluções

Soma da Progressão Aritmética:



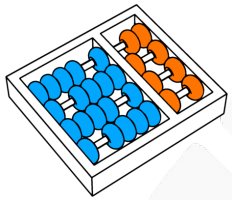
Revisitando soluções

Soma da Progressão Aritmética:

```
1  a_1 = int(input("Entre com a_1: "))
2  n = int(input("Entre com n: "))
3  r = int(input("Entre com r: "))
4  esperado = a_1*n + n * (n - 1) * r / 2
5  soma = 0
6  atual = a_1
7  for i in range(n):
8      soma += atual
9      atual += r
10 if soma != esperado:
11     print("Fórmula incorreta!")
12     print("Esperado: ", esperado)
13     print("Obtido: ", soma)
14 else:
15     print("Fórmula correta!")
```

Revisitando soluções

Lendo e imprimindo uma lista:



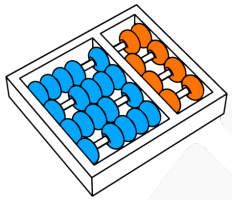
Revisitando soluções

Lendo e imprimindo uma lista:

```
1 n = int(input("Entre com a quantidade de nomes: "))
2 l = []
3 for i in range(n):
4     nome = input("Entre com o nome: ")
5     l.append(nome)
6 print("Nomes digitados:")
7 for x in l:
8     print(x)
```

Versões do range

Temos três versões:



Versões do range

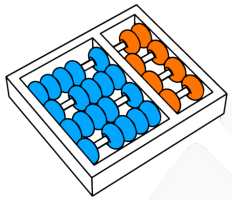
Temos três versões:

- **range(fim)**:

Versões do range

Temos três versões:

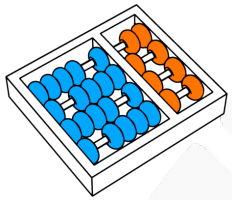
- **range(fim)**:
  - Intervalo de **0** (incluído) a **fim** (excluído).



Versões do range

Temos três versões:

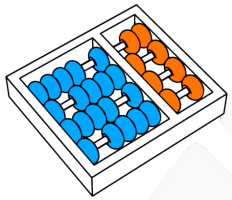
- **range(fim)**:
  - Intervalo de **0** (incluído) a **fim** (excluído).
  - Ex: **range(10)** é $0, 1, \ldots, 9$.



Versões do range

Temos três versões:

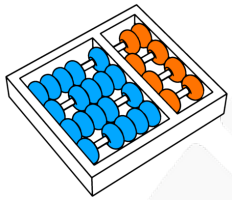
- **range(fim)**:
  - Intervalo de **0** (incluído) a **fim** (excluído).
  - Ex: **range(10)** é $0, 1, \ldots, 9$.
- **range(inicio, fim)**:



Versões do range

Temos três versões:

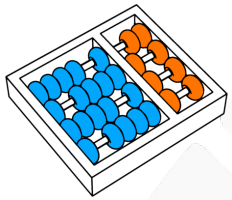
- **range(fim)**:
  - Intervalo de **0** (incluído) a **fim** (excluído).
  - Ex: **range(10)** é $0, 1, \ldots, 9$.
- **range(inicio, fim)**:
  - Intervalo de **inicio** (incluído) a **fim** (excluído).



Versões do range

Temos três versões:

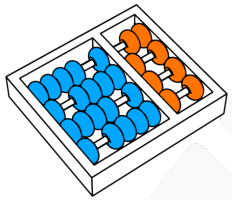
- **range(fim)**:
  - Intervalo de **0** (incluído) a **fim** (excluído).
  - Ex: **range(10)** é $0, 1, \ldots, 9$.
- **range(inicio, fim)**:
  - Intervalo de **inicio** (incluído) a **fim** (excluído).
  - Ex: **range(2, 10)** é $2, 3, \ldots, 9$.



Versões do range

Temos três versões:

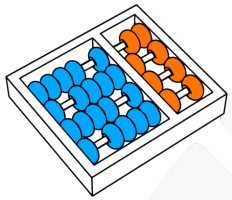
- **range(fim)**:
  - Intervalo de **0** (incluído) a **fim** (excluído).
  - Ex: **range(10)** é $0, 1, \ldots, 9$.
- **range(inicio, fim)**:
  - Intervalo de **inicio** (incluído) a **fim** (excluído).
  - Ex: **range(2, 10)** é $2, 3, \ldots, 9$.
- **range(inicio, fim, passo)**:



Versões do range

Temos três versões:

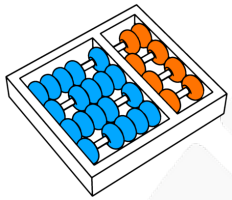
- **range(fim)**:
  - Intervalo de **0** (incluído) a **fim** (excluído).
  - Ex: **range(10)** é $0, 1, \ldots, 9$.
- **range(inicio, fim)**:
  - Intervalo de **inicio** (incluído) a **fim** (excluído).
  - Ex: **range(2, 10)** é $2, 3, \ldots, 9$.
- **range(inicio, fim, passo)**:
  - Intervalo de **inicio** (incluído) a **fim** (excluído), pulando de **passo** em **passo**.



Versões do range

Temos três versões:

- **range(fim)**:
  - Intervalo de **0** (incluído) a **fim** (excluído).
  - Ex: **range(10)** é $0, 1, \ldots, 9$.
- **range(inicio, fim)**:
  - Intervalo de **inicio** (incluído) a **fim** (excluído).
  - Ex: **range(2, 10)** é $2, 3, \ldots, 9$.
- **range(inicio, fim, passo)**:
  - Intervalo de **inicio** (incluído) a **fim** (excluído), pulando de **passo** em **passo**.
  - Ex: **range(3, 10, 2)** é $3, 5, 7, 9$.



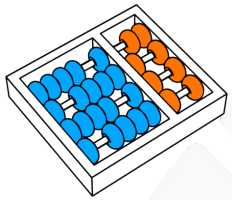
Versões do range

Temos três versões:

- **range(fim)**:
  - Intervalo de **0** (incluído) a **fim** (excluído).
  - Ex: **range(10)** é $0, 1, \ldots, 9$.
- **range(inicio, fim)**:
  - Intervalo de **inicio** (incluído) a **fim** (excluído).
  - Ex: **range(2, 10)** é $2, 3, \ldots, 9$.
- **range(inicio, fim, passo)**:
  - Intervalo de **inicio** (incluído) a **fim** (excluído), pulando de **passo** em **passo**.
  - Ex: **range(3, 10, 2)** é $3, 5, 7, 9$.
  - Ex: **range(10, 3, -2)** é $10, 8, 6, 4$.

Revisitando soluções

Primo



Revisitando soluções

Primo

```
1 p = int(input("Entre com p: "))
2 primo = True
3 for k in range(2, p):
4     if p % k == 0:
5         primo = False
6         break  # para a execução do laço
7 print(p >= 2 and primo)
```

Revisitando soluções

Primo

```
1 p = int(input("Entre com p: "))
2 primo = True
3 for k in range(2, p):
4     if p % k == 0:
5         primo = False
6         break  # para a execução do laço
7 print(p >= 2 and primo)
```

Note que agora estamos indo até $p - 1$ e não $\sqrt{p}$...

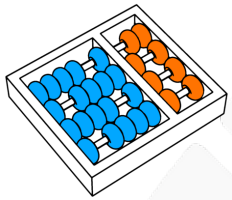
Revisitando soluções

Primo

```
1 p = int(input("Entre com p: "))
2 primo = True
3 for k in range(2, p):
4     if p % k == 0:
5         primo = False
6         break  # para a execução do laço
7 print(p >= 2 and primo)
```

Note que agora estamos indo até $p - 1$ e não $\sqrt{p}$...

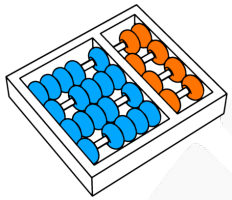
- Daria para calcular $\sqrt{p}$ antes.



Sempre dá para usar?

Como usar **for ... in range(...)** nesse código?

```
1 n = int(input("Entre com o n: "))
2 d = 2
3 while n != 1:
4     if n % d == 0:
5         n //= d
6         print(d)
7     else:
8         d += 1
9 print(lista)
```



Sempre dá para usar?

Como usar **for ... in range(...)** nesse código?

```
1 n = int(input("Entre com o n: "))
2 d = 2
3 while n != 1:
4     if n % d == 0:
5         n //= d
6         print(d)
7     else:
8         d += 1
9 print(lista)
```

Esse é um **while** que foge do padrão!

Exercícios

Listas e repetição

Vamos fazer alguns exercícios?

Exercício 1

(a) Dado n, imprima todos os números ímpares menores do que n.

Exercício 1

(a) Dado n, imprima todos os números ímpares menores do que n.

(b) Imprima os n primeiros números naturais em ordem inversa.

Exercício 1

(a) Dado n, imprima todos os números ímpares menores do que n.

(b) Imprima os n primeiros números naturais em ordem inversa.

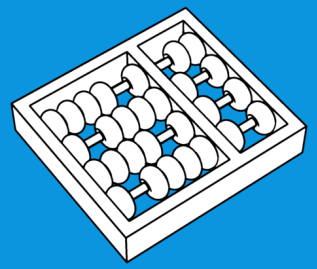
(c) Dado um inteiro n seguido por n inteiros, imprima o menor valor e o maior valor dos n inteiros, seguido pela quantidade de vezes que esses valores se repetem.

Listas e repetição

MC102 - Algoritmos e Programação de Computadores

6

03/25



Santiago Valdés Ravelo
https://ic.unicamp.br/~santiago/
ravelo@unicamp.br