

# CK0211 - Fundamentos de Programação: Introdução a Vetores

Emanuele Santos

Bibliografia: Ascencio, Cap. 6

# Objetivos

- Apresentar o conceito de Vetor em algoritmos
- Introduzir o conceito de listas em Python



# VETOR EM ALGORITMOS

# Introdução

- Frequentemente, necessitaremos trabalhar com grandes coleções de dados e precisaremos guardar esses dados em memória
  - Criar uma variável para cada item de dado é impraticável
- Utilizaremos variáveis compostas, chamadas vetores

# Definição de Vetor

- Vetor é conhecido como variável composta homogênea unidimensional

# Definição de Vetor

- Vetor é conhecido como **variável composta** homogênea unidimensional

conjunto de variáveis com  
mesmo nome

# Definição de Vetor

- Vetor é conhecido como variável composta **homogênea** unidimensional

conjunto de variáveis com  
mesmo nome

do mesmo tipo

# Definição de Vetor

- Vetor é conhecido como variável composta homogênea **unidimensional**

conjunto de variáveis com  
mesmo nome

acessadas por um  
único índice

do mesmo tipo

# Definição de Vetor

- Vetor é conhecido como variável composta homogênea unidimensional
- Como as variáveis no vetor têm o mesmo nome, o que as distingue é um índice que referencia sua localização dentro da estrutura

# Declaração de vetor

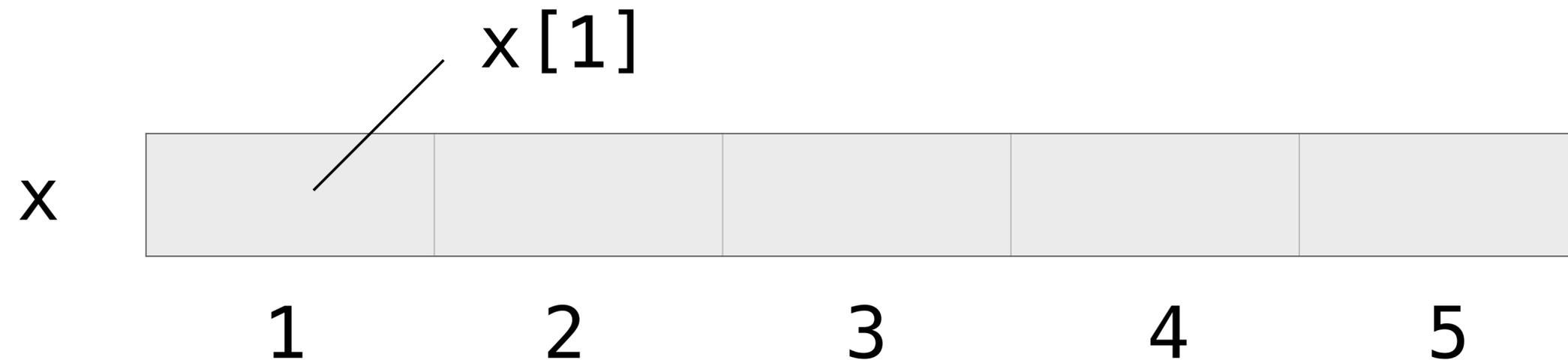
```
DECLARE nome[tamanho] TIPO
```

- onde:
  - nome é o nome da variável do tipo vetor;
  - tamanho é a quantidade de variáveis que vão compor o vetor;
  - tipo é o tipo básico de cada elemento do vetor

# Exemplos de vetor

**DECLARE  $x[5]$  NUMÉRICO**

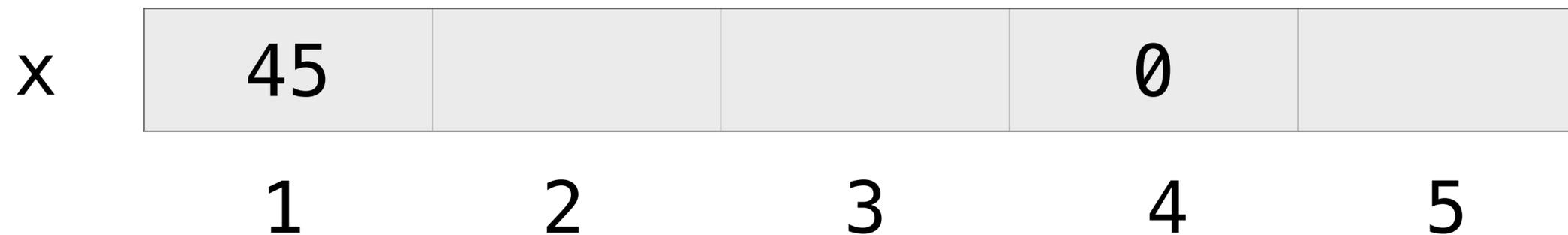
- $x$  é um vetor com 5 posições



# Atribuindo valores ao vetor

```
x[1] ← 45  
x[4] ← 0
```

- Atribuimos valores às posições do vetor, por isso precisamos identificar o nome e a posição (índice)



# Preenchendo um vetor

```
PARA  $i \leftarrow 1$  ATÉ 5 FAÇA  
INÍCIO  
  ESCREVA "Digite o ",  $i$ , "º número: "  
  LEIA  $x[i]$   
FIM
```

- A maneira mais comum para se preencher um vetor é utilizando a estrutura **PARA**
- A variável  $i$  irá assumir todos os valores possíveis e válidos para as posições do vetor  $x$
- Em cada iteração, será utilizada uma posição diferente do vetor

# Simulação

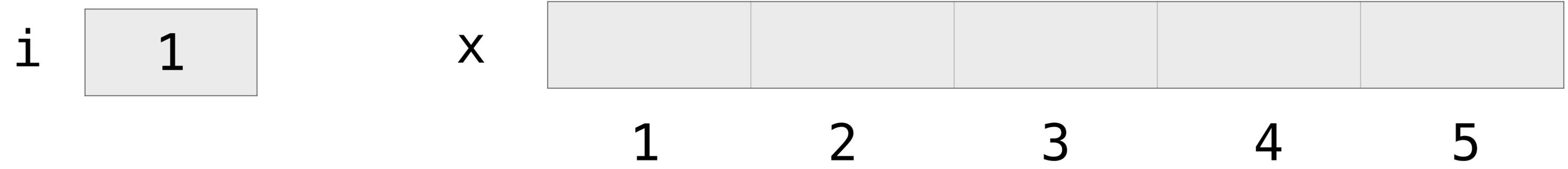
Tela:

Digite o 1º número:

```

PARA i ← 1 ATÉ 5 FAÇA
INÍCIO
  ESCREVA "Digite o ", i, "º número: "
  LEIA x[i]
FIM
  
```

Memória:



# Simulação

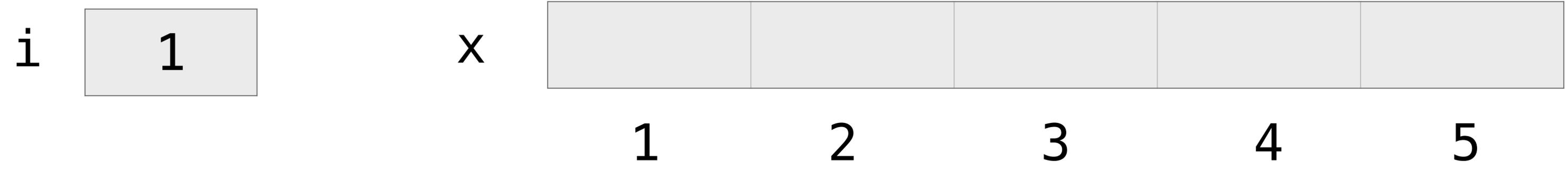
```

PARA  $i \leftarrow 1$  ATÉ 5 FAÇA
INÍCIO
    ESCREVA "Digite o ",  $i$ , "º número: "
    LEIA  $x[i]$ 
FIM
    
```

**Tela:**

**Digite o 1º número: 10**

**Memória:**



# Simulação

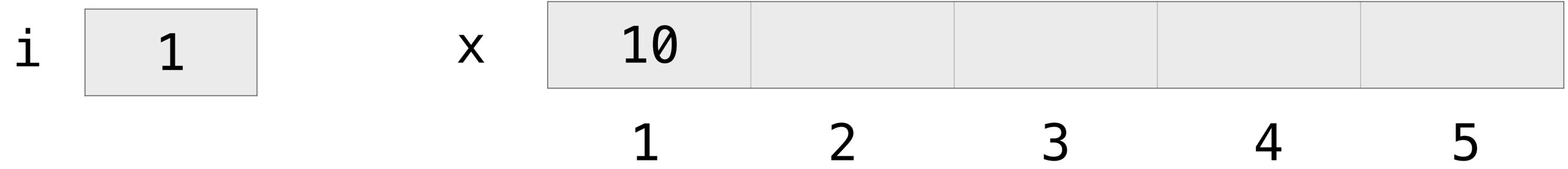
```

PARA  $i \leftarrow 1$  ATÉ 5 FAÇA
INÍCIO
    ESCREVA "Digite o ",  $i$ , "º número: "
    LEIA  $x[i]$ 
FIM
    
```

**Tela:**

Digite o 1º número: 10

**Memória:**



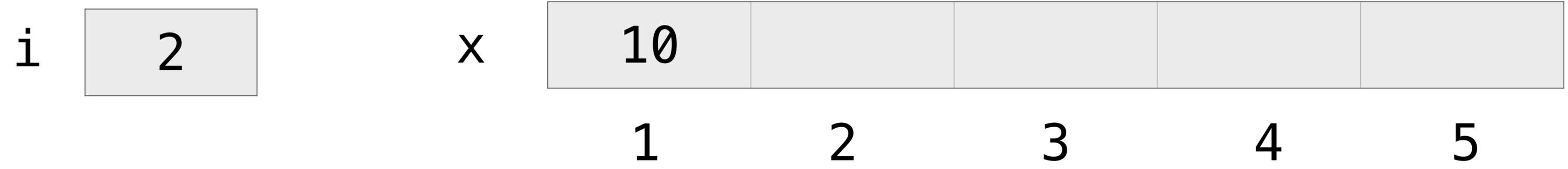
# Simulação

Tela:

```
Digite o 1º número: 10
Digite o 2º número:
```

```
PARA i ← 1 ATÉ 5 FAÇA
INÍCIO
  ESCREVA "Digite o ", i, "º número: "
  LEIA x[i]
FIM
```

Memória:



# Simulação

```

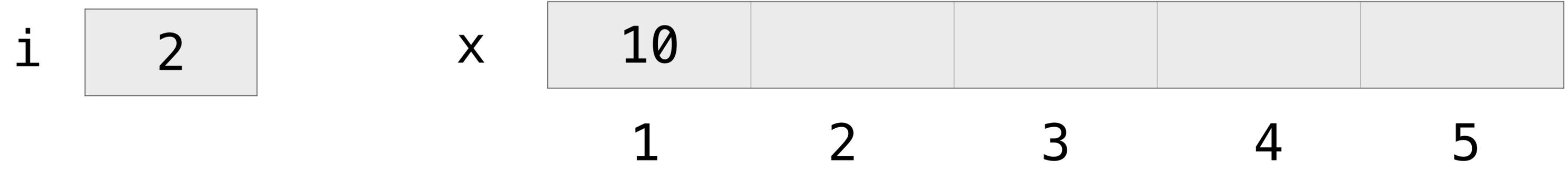
PARA  $i \leftarrow 1$  ATÉ 5 FAÇA
INÍCIO
    ESCREVA "Digite o ",  $i$ , "º número: "
    LEIA  $x[i]$ 
FIM
    
```

## Tela:

```

Digite o 1º número: 10
Digite o 2º número: 3
    
```

## Memória:



# Simulação

```

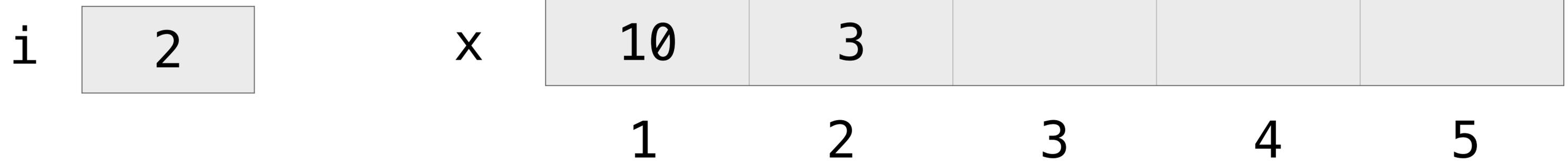
PARA  $i \leftarrow 1$  ATÉ 5 FAÇA
INÍCIO
    ESCREVA "Digite o ",  $i$ , "º número: "
    LEIA  $x[i]$ 
FIM
    
```

## Tela:

```

Digite o 1º número: 10
Digite o 2º número: 3
    
```

## Memória:



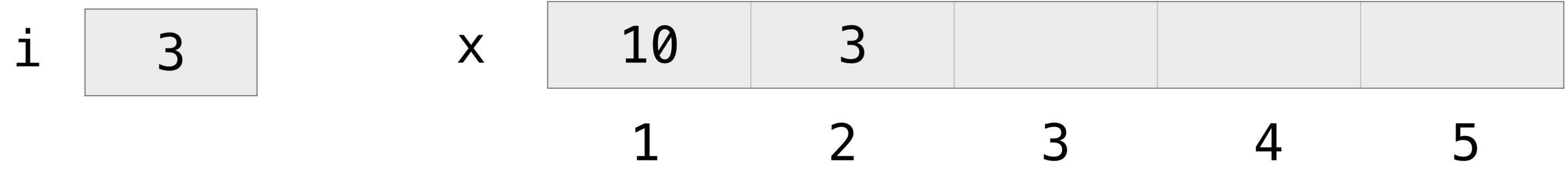
# Simulação

Tela:

```
Digite o 1º número: 10
Digite o 2º número: 3
Digite o 3º número:
```

```
PARA i ← 1 ATÉ 5 FAÇA
INÍCIO
  ESCREVA "Digite o ", i, "º número: "
  LEIA x[i]
FIM
```

Memória:



# Simulação

```

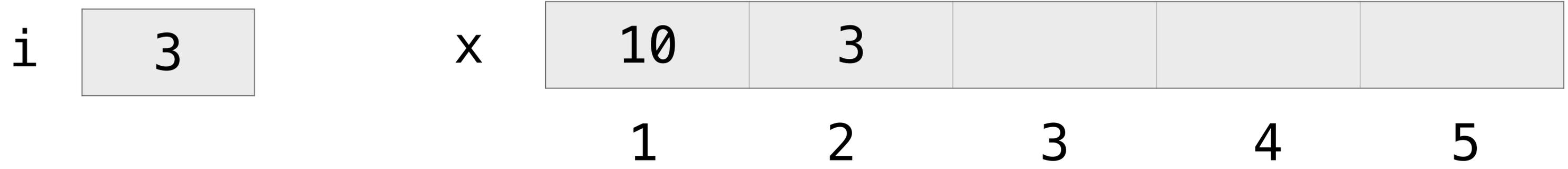
PARA  $i \leftarrow 1$  ATÉ 5 FAÇA
INÍCIO
    ESCREVA "Digite o ",  $i$ , "º número: "
    LEIA  $x[i]$ 
FIM
    
```

## Tela:

```

Digite o 1º número: 10
Digite o 2º número: 3
Digite o 3º número: 28
    
```

## Memória:



# Simulação

```

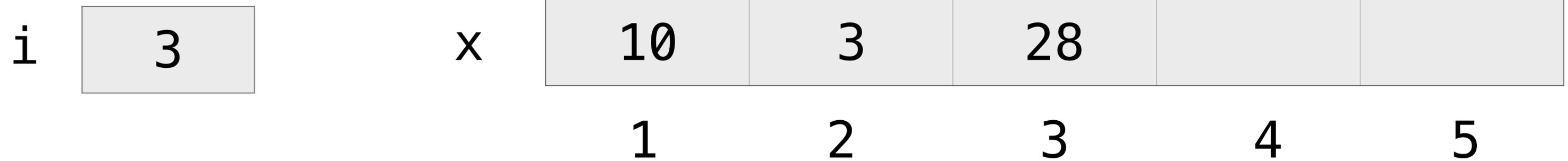
PARA  $i \leftarrow 1$  ATÉ 5 FAÇA
INÍCIO
    ESCREVA "Digite o ",  $i$ , "º número: "
    LEIA  $x[i]$ 
FIM
    
```

## Tela:

```

Digite o 1º número: 10
Digite o 2º número: 3
Digite o 3º número: 28
    
```

## Memória:



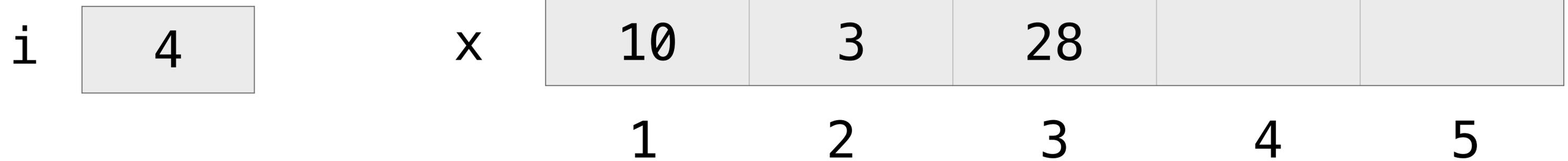
# Simulação

## Tela:

```

Digite o 1º número: 10
Digite o 2º número: 3
Digite o 3º número: 28
Digite o 4º número:
    
```

## Memória:



```

PARA  $i \leftarrow 1$  ATÉ 5 FAÇA
INÍCIO
  ESCREVA "Digite o ",  $i$ , "º número: "
  LEIA  $x[i]$ 
FIM
    
```

# Simulação

```

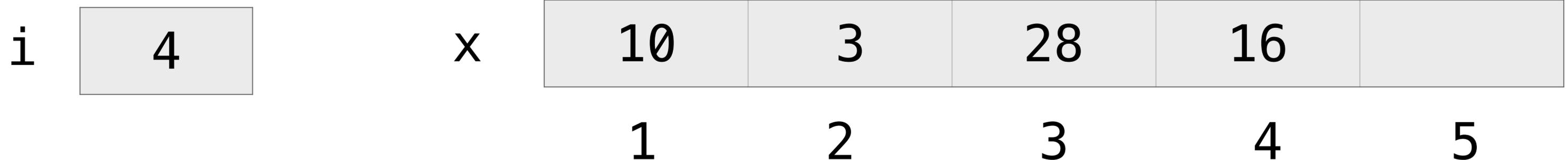
PARA  $i \leftarrow 1$  ATÉ 5 FAÇA
INÍCIO
    ESCREVA "Digite o ",  $i$ , "º número: "
    LEIA  $x[i]$ 
FIM
    
```

## Tela:

```

Digite o 1º número: 10
Digite o 2º número: 3
Digite o 3º número: 28
Digite o 4º número: 16
    
```

## Memória:

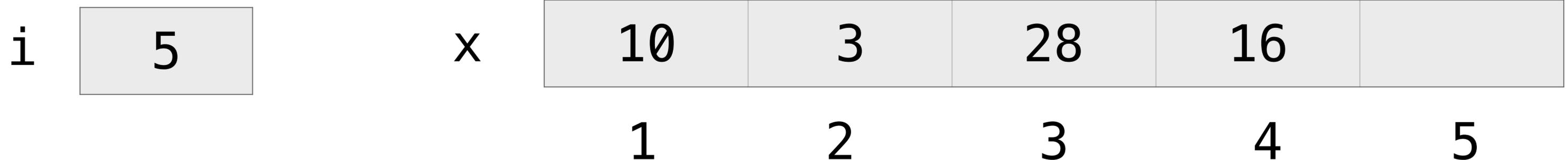


# Simulação

## Tela:

```
Digite o 1º número: 10
Digite o 2º número: 3
Digite o 3º número: 28
Digite o 4º número: 16
Digite o 5º número:
```

## Memória:



```
PARA  $i \leftarrow 1$  ATÉ 5 FAÇA
INÍCIO
  ESCREVA "Digite o ",  $i$ , "º número: "
  LEIA  $x[i]$ 
FIM
```

# Simulação

```

PARA  $i \leftarrow 1$  ATÉ 5 FAÇA
INÍCIO
  ESCREVA "Digite o ",  $i$ , "º número: "
  LEIA  $x[i]$ 
FIM
  
```

## Tela:

```

Digite o 1º número: 10
Digite o 2º número: 3
Digite o 3º número: 28
Digite o 4º número: 16
Digite o 5º número: 8
  
```

## Memória:

$i$	5	$x$	10	3	28	16	8
			1	2	3	4	5

# Mostrando os elementos do vetor

```
PARA  $i \leftarrow 1$  ATÉ 5 FAÇA  
INÍCIO  
  ESCREVA "Este é o ",  $i$ , "º número do vetor:",  $x[i]$   
FIM
```

- A maneira mais comum para se percorrer um vetor também é utilizando a estrutura **PARA**
- A variável  $i$  irá assumir todos os valores possíveis e válidos para as posições do vetor  $x$
- Em cada iteração, será mostrado o conteúdo de uma posição diferente do vetor



# LISTAS EM PYTHON

# Implementação de vetores

- A maneira da qual um vetor é implementado em um programa varia de linguagem para linguagem
- Conceitualmente, vetores são variáveis compostas homogêneas de tamanho fixo
  - Uma vez que um vetor é declarado com um determinado tipo e tamanho ele não pode ser alterado ao longo da execução do programa
- Normalmente são chamados de **array** e os índices podem iniciar em 0 (C/C++/Java)

# Vetores em Python

- O tipo vetor propriamente dito não existe em Python\*
- Python possui um tipo chamado **list** (lista) que é bem mais **flexível** que um vetor e pode ser utilizado nas mesmas situações que exigiriam um vetor

*\* Existe uma classe chamada array, mas não é tão utilizada como a lista*

# Listas em Python

- Listas possuem tamanho variável
  - Elas podem aumentar ou diminuir de tamanho ao longo do tempo
- São heterogêneas (podem armazenar elementos de diferentes tipos e tamanhos)
- Os índices iniciam em **0**

*\* Existe uma classe chamada array, mas não é tão utilizada como a lista*

# Criando listas

```
# criando uma lista com 5 inteiros  
x = [10, 3, 28, 16, 8]
```

x	10	3	28	16	8
	0	1	2	3	4

# Acessando elementos da lista

```
# criando uma lista com 5 inteiros  
x = [10, 3, 28, 16, 8]  
x[0] # 10  
x[1] # 3  
x[4] # 8  
x[5] # ?
```

x	10	3	28	16	8
	0	1	2	3	4

# Acessando elementos da lista

```
# criando uma lista com 5 inteiros
x = [10, 3, 28, 16, 8]
x[0] # 10
x[1] # 3
x[4] # 8
x[5]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IndexError: list index out of range
```

# Listas em Python

- Os índices iniciam em **0**
- E terminam em **tamanho - 1**
- Uma lista de tamanho 5 terá o último índice igual a 4

# Acessando elementos da lista

```
# quando não se sabe o tamanho de uma lista,  
# também se usa o índice -1 para acessar o  
# último elemento  
x = [10, 3, 28, 16, 8]  
x[-1] # 8  
x[-2] # ?
```

# Acessando elementos da lista

```
# quando não se sabe o tamanho de uma lista,  
# também se usa o índice -1 para acessar o  
# último elemento  
x = [10, 3, 28, 16, 8]  
x[-1] # 8  
x[-2] # 16
```

# Simulando a criação de um vetor com um determinado tamanho

- Suponha que queiramos criar uma lista de tamanho 50

*$x = [0] * 50$  #  $x$  terá 50 posições preenchidas com 0*

- Ou uma lista  $y$  de tamanho 100

*$y = [0] * 100$  #  $y$  terá 100 posições preenchidas com 0*

# Preenchendo uma lista

- Em algoritmos, nós preenchemos um vetor assim:

```
DECLARE x[5] NUMÉRICO  
PARA i ← 1 ATÉ 5 FAÇA  
INÍCIO  
    ESCREVA “Digite o ”, i, “º número: ”  
    LEIA x[i]  
FIM
```

- Em Python:

```
x = [0] * 5 # x = [0, 0, 0, 0, 0]  
for i in range(0,5):  
    x[i] = int(input(“Digite o %dº número: ” % (i+1)))
```

# Exercício

- Na pasta fup, crie um programa (`notas_lista.py`) que cria uma lista com 10 posições e peça para o usuário entrar com as 10 notas. As notas podem ser números reais

# Mostrando os elementos de uma lista

- Em algoritmos, nós mostramos os elementos do vetor assim:

```
PARA  $i \leftarrow 1$  ATÉ 5 FAÇA  
INÍCIO  
  ESCREVA “Este é o ”,  $i$ , “º número do vetor:”,  $x[i]$   
FIM
```

- Em Python:

```
for  $i$  in range(0,5):  
  print(“Este é o ”,  $i+1$ , “º número do vetor:”,  $x[i]$ )
```

# Mostrando os elementos de uma lista

- Se não for preciso mostrar o índice do vetor, uma maneira mais concisa de mostrar os elementos de uma lista é:

```
for e in x:  
    print(e)
```

# Exercício

- Modifique o programa (`notas_lista.py`) de modo a mostrar as notas digitadas ao final do programa (tente usar os dois modos)

# Calculando o tamanho de uma lista

- Podemos usar a função `len` para calcular o tamanho de uma lista

```
x = [0] * 5  
len(x) # 5
```

- E assim podemos usá-la dentro da função `range`

```
for i in range(0, len(x)):  
    print("Este é o ", i+1, "º número do vetor:", x[i])
```

# Exercício

- Modifique o programa (`notas_lista.py`) tal que depois de receber as notas dos alunos, calcular a média da turma, e mostrar a posição na lista e a nota dos alunos que ficaram de AF ( $4 \leq \text{nota} < 7$ ).