



ICC901 – Introdução à Programação de Computadores
IECo81 – Introdução à Ciência dos Computadores
IECo37 – Introdução à Programação de Computadores

Aula 05 – Vetores e Strings

Atualização: 6/mar/20



Você tem a liberdade de:



Compartilhar: copiar, distribuir e transmitir esta obra.

Remixar: criar obras derivadas.

Sob as seguintes condições:



Atribuição: você deve creditar a obra da forma especificada pelo autor ou licenciante (mas não de maneira que sugira que estes concedem qualquer aval a você ou ao seu uso da obra).



Uso não comercial: você não pode usar esta obra para fins comerciais.



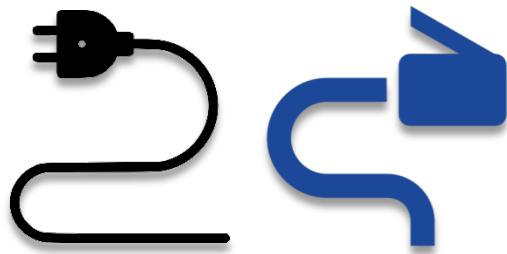
Compartilhamento pela mesma licença: se você alterar, transformar ou criar em cima desta obra, poderá distribuir a obra resultante apenas sob a mesma licença, ou sob uma licença similar à presente.



Conserve o laboratório



Os equipamentos são frágeis:
use-os com cuidado



Não mexa nos cabos



Não consuma alimentos ou bebidas.
Mantenha sua garrafa de água
tampada.

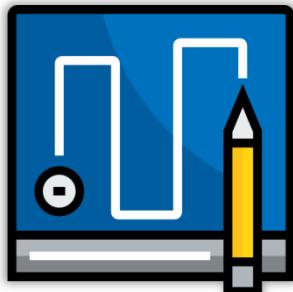
Antes de começar...



Está atento ao **calendário**?



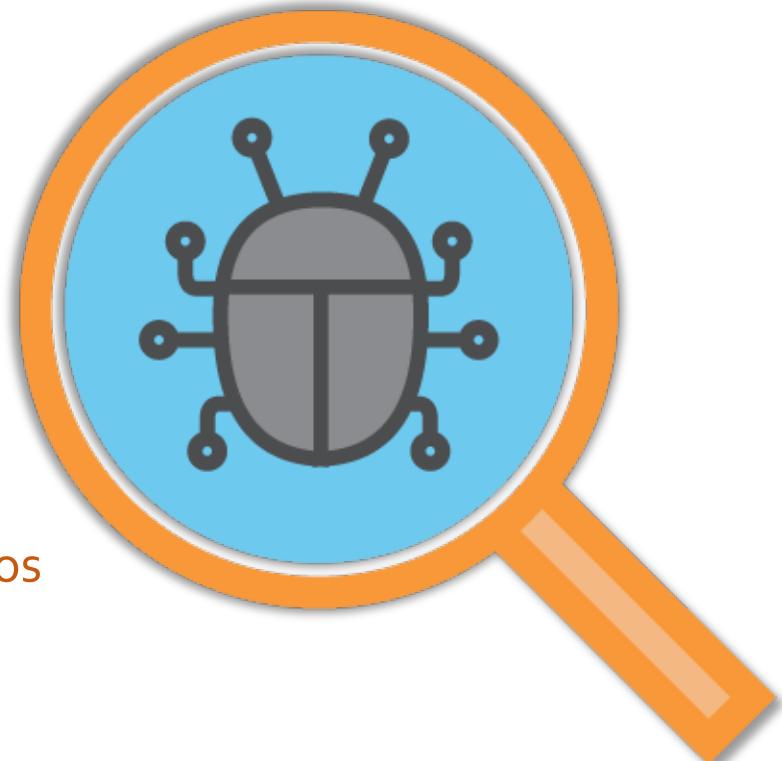
Este módulo tem **Peso 3** na avaliação



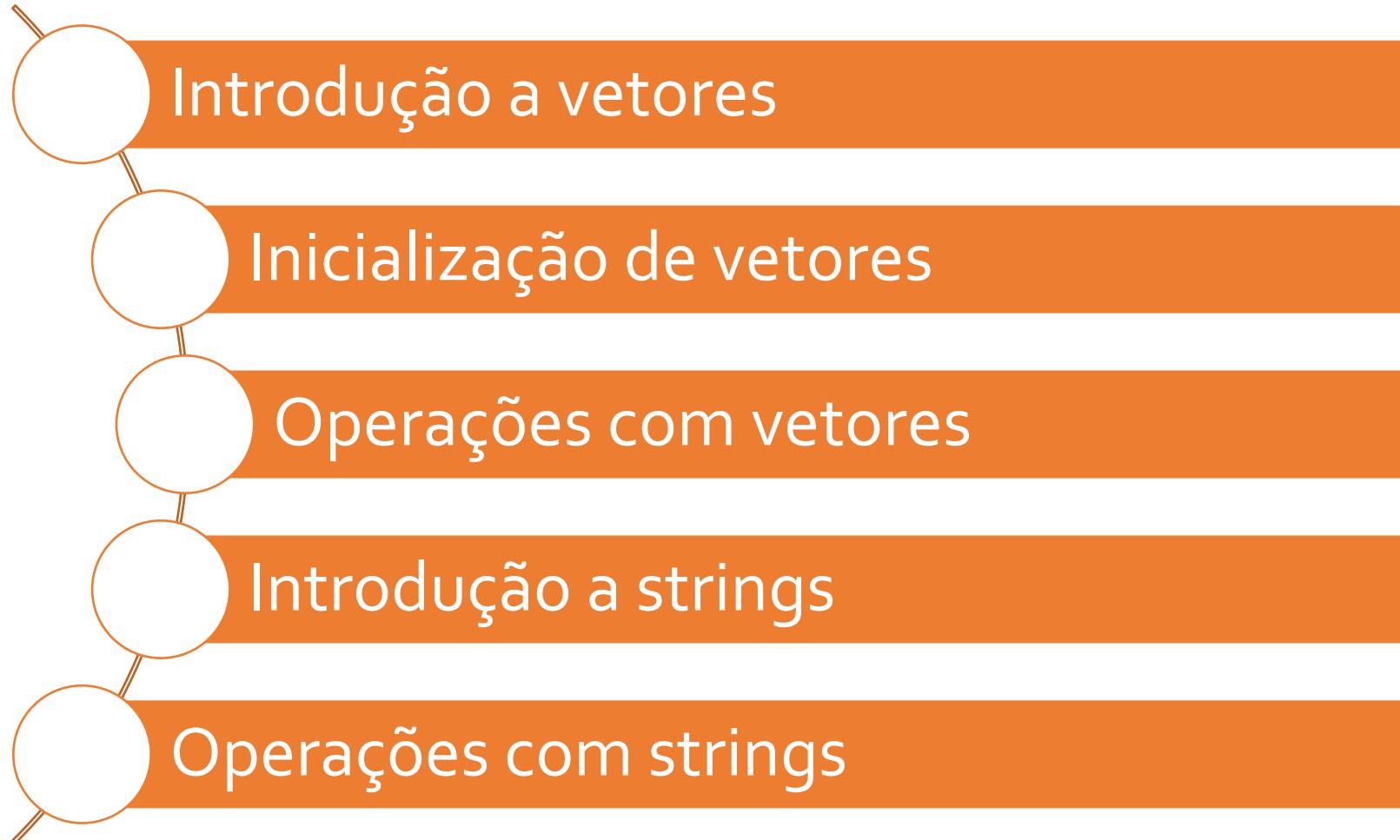
Como está sua **tática** de estudo?

Como lidar com erros de programação?

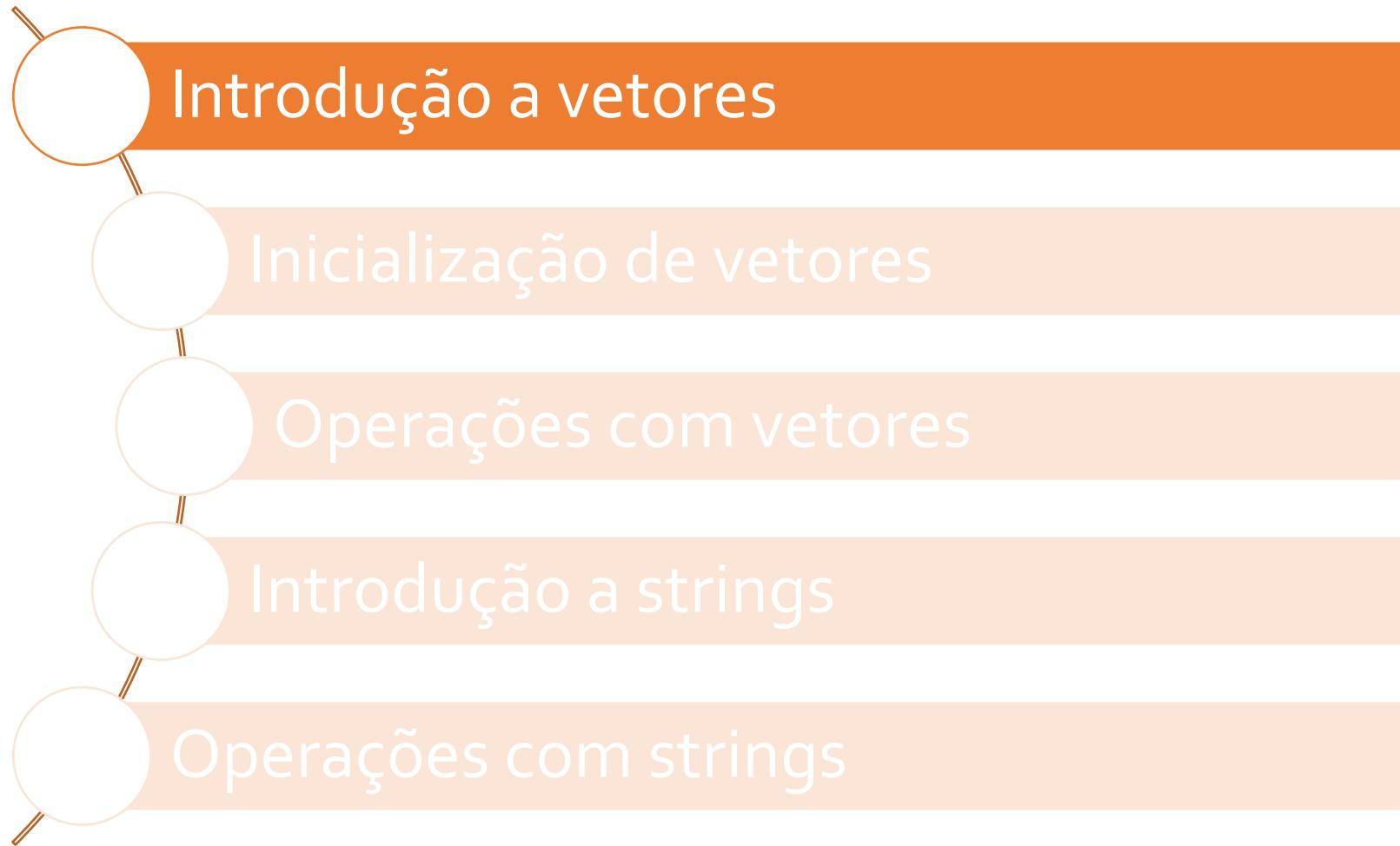
- 🛡️ Não se pergunte: “por que o programa não funciona?”
- 🛡️ Em vez disso, pergunte-se: “por que o programa está funcionando deste jeito?”
- 🛡️ Estratégias:
 - 🛡️ Não se limite ao exemplo: teste vários casos
 - 🛡️ Imprima resultados intermediários
 - 🛡️ Tente explicar o problema para outra pessoa
 - 🛡️ Dê um tempo e tente de novo mais tarde



Conteúdo – Vetores e Strings



Conteúdo – Vetores e Strings



Vetores

- 🛡️ Alguns problemas manipulam **grande quantidade** de valores:
 - 🛡️ Folha de pagamento
 - 🛡️ Notas de uma turma
 - 🛡️ Medições ao longo do tempo



Vetores

- 🛡️ Vetores são um **conjunto de posições da memória:**
 - 🛡️ identificadas por um **único nome**
 - 🛡️ individualizadas por **índices**
 - 🛡️ com conteúdo de um **mesmo tipo** (int, float, string)

Exemplo:

notas	6.1	2.3	9.4	5.1	8.9	9.8	10	7.0	6.3	4.4
índice:	0	1	2	3	4	5	6	7	8	9

```
from numpy import *
```

Para manipular vetores,
utilizamos o módulo **numpy**

Índices referenciam elementos do vetor

- 💡 Cada elemento de um vetor pode ser individualmente identificado por meio da sua **posição**.

notas	6.1	2.3	9.4	5.1	8.9	9.8	10	7.0	6.3	4.4
índice:	0	1	2	3	4	5	6	7	8	9

```
print(notas)
```

Imprime **todos** os elementos do vetor

```
print(notas[0])
```

Imprime apenas o **1º elemento** do vetor

Todo vetor deve ser inicializado

Todo vetor deve ter um nome, que segue as mesmas regras para nomear variáveis

nome = array([v0, v1, v2, ...])

- Colchetes definem um **conjunto** de valores
- Elemento do vetor
- Vírgulas separam um elemento de outro no conjunto
- Toda função envolve seus argumentos entre **parênteses**
- Função que cria o vetor

Problema 1

:: Notas de uma turma

- As **notas** dos alunos de uma turma estão armazenadas em um vetor.
- Imprima:
 1. As notas de **todos** os alunos.
 2. A nota do **3º** aluno.
 3. O professor esqueceu de dar **1,0 ponto** de participação aos alunos de índice 4 e 7. Some e imprima o **vetor corrigido** da turma.



Problema 1

2 – Identifique as entradas e saídas

	Grandeza	Unidade de medida	Faixa de valores
Entradas	Vetor de notas	---	[[0, 10]]
Saídas	Vetor de notas	---	[[0, 10]]
	Nota do 3º aluno (escalar)	---	[0, 10]
	Vetor corrigido	---	[[0, 10]]

Verifique se as grandezas envolvidas
são **vetores** (coleção de valores) ou
escalares (valor único)

Vamos usar colchetes
duplos para indicar
um vetor

Problema 1

3 – Projete o script

1. As notas de **todos** os alunos:

```
print(notas)
```

2. A nota do **3º** aluno:

```
print(notas[2])
```

3. Vetor corrigido:

```
notas[4] = notas[4] + 1.0
notas[7] = notas[7] + 1.0
print(notas)
```

Problema 1

4 – Codifique em Python

```
from numpy import *

# Carregar vetor
notas = array([6.2, 2.3, 9.4, 5.1,
8.9, 9.8, 10.0, 7.0, 6.3, 4.4])

# Saída 1: vetor notas
print(notas)

# Saída 2: nota do 3º. aluno
print(notas[2])

# Saída 3: vetor corrigido
notas[4] = notas[4] + 1.0
notas[7] = notas[7] + 1.0
print(notas)
```

Não confunda: o comando **notas = notas + 1** somaria um ponto para **toda** a turma, e não apenas para os alunos 4 e 7



Problema 1

5 – Teste o script

Saída 1

- O vetor de saída é **exatamente** igual ao de entrada?

Saída 2

- O valor impresso é realmente a nota do 3º aluno?
- Ou é a nota de um dos vizinhos?

Saída 3

- As posições 4 e 7 do vetor foram alteradas corretamente?
- As outras posições do vetor foram alteradas por engano?

Operações sobre o vetor × operações sobre o elemento

	0	1	2	3
v	11	22	33	44

	0	1	2	3
w	20	30	40	50

$$v = v + 1$$

	0	1	2	3
	12	23	34	45

$$v[2] = v[2] + 1$$

11	22	34	44
----	----	----	----

$$v = v + w$$

31	52	73	94
----	----	----	----

$$v[1] = v[1] + w[1]$$

11	52	33	44
----	----	----	----

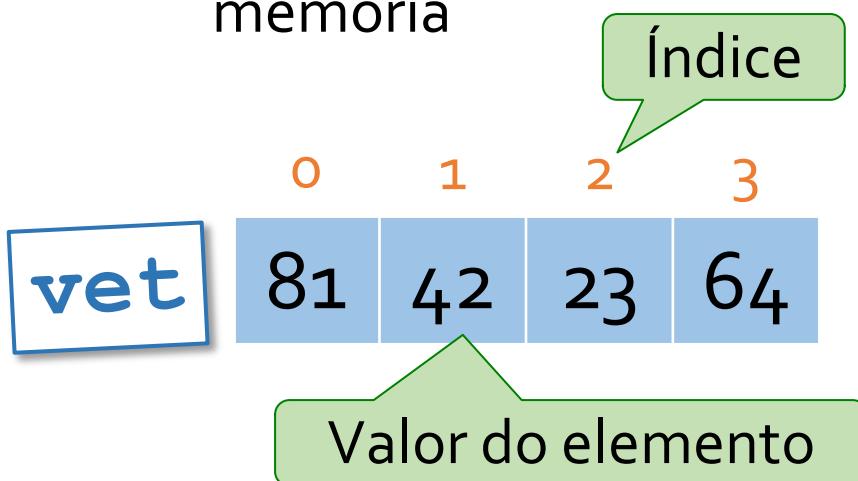
Índice ≠ elemento

🛡 Índice (**i**):

🛡 É uma referência

🛡 Elemento (**vet[i]**):

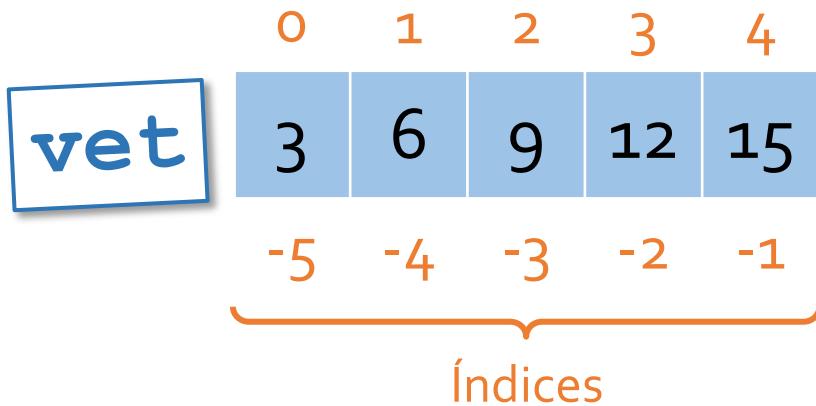
🛡 É o valor guardado em
memória



Índices

:: Limites

- Indices fora dos limites causam erro.
- Indices podem ser negativos, permitindo o acesso na ordem inversa.



A screenshot of a Python 3 terminal window titled 'main.py'. The code defines a numpy array 'vet' with elements [3, 6, 9, 12, 15]. It prints the first and last elements correctly, then attempts to print 'vet[5]', which results in an `IndexError: index 5 is out of bounds for axis 0 with size 5`. The terminal also shows the 'Console' tab selected.

```
Arquivo Editar Buscar Executar Ferramentas
Python 3 main.py
from numpy import *
vet = array([3,6,9,12,15])

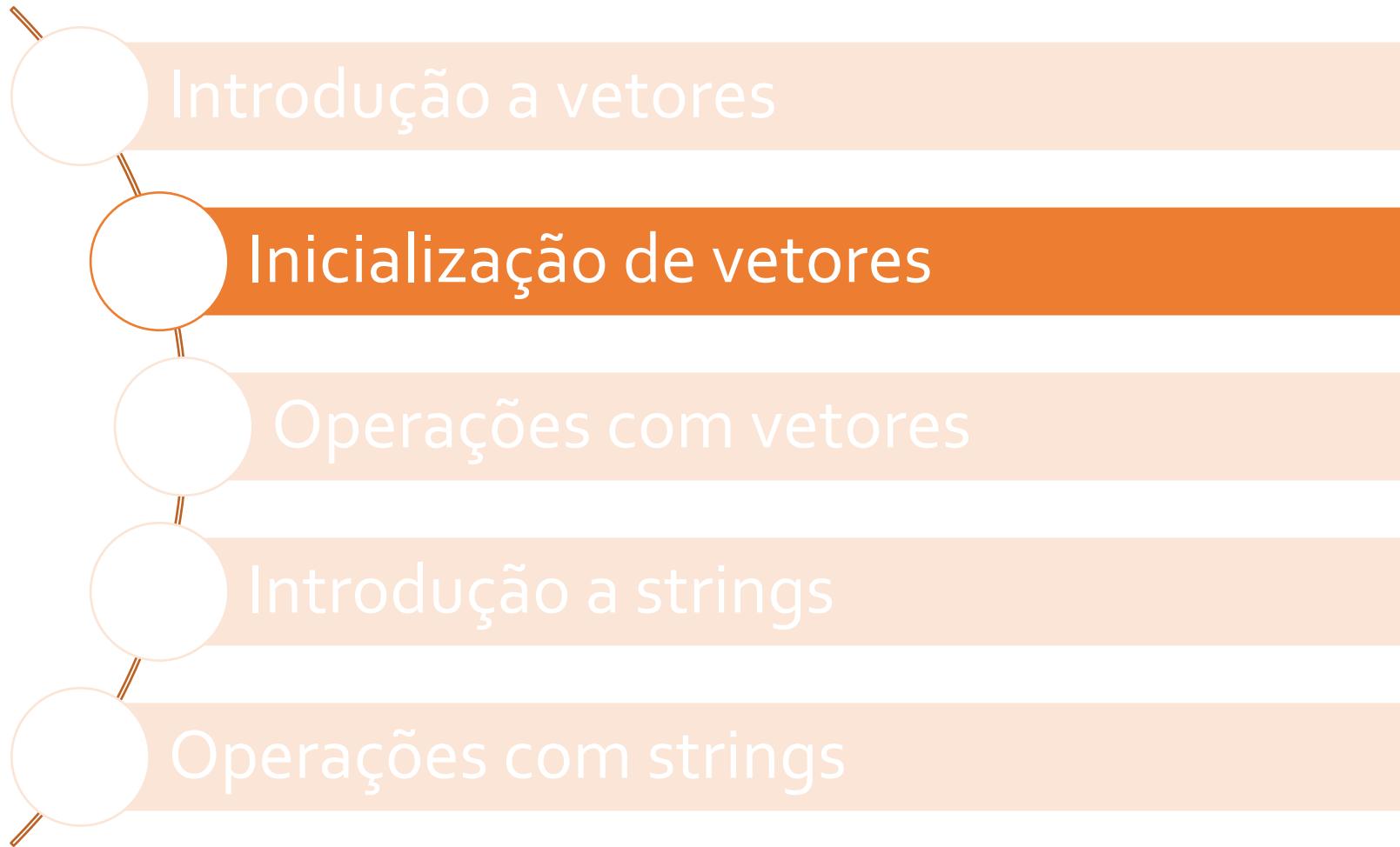
# Primeiro elemento
print(vet[0])
# Último elemento
print(vet[-1])
# ERRO
print(vet[5])
```

Console Shell

3
15
IndexError: index 5 is
out of bounds for axis
0 with size 5

Índice 5 está fora dos limites

Conteúdo – Vetores e Strings



Iniciando vetor no corpo do script

- 🛡 Você pode informar o valor dos elementos do vetor no **corpo do script**.
- 🛡 Útil em pequenos **testes**.

```
Arquivo Editar Buscar Executar Ferramentas
Python 3 main.py
from numpy import *
v1 = array([11, 22, 33])
print(v1)

# Vetor vazio
v2 = array([])
print(v2)
```

Console Shell

```
[11 22 33]
[]
```

Separe os elementos por **vírgula**

Iniciando vetor via teclado

- 💡 Você pode inserir vetor completo via teclado.
- 💡 Será a forma mais comum de entrar com casos de correção no CodeBench.

```
Arquivo Editar Buscar Executar Ferramentas
Python 3 main.py
from numpy import *
v = array(eval(input("v:")))
print(v)
```

- A função **eval** trata colchetes e vírgulas;
- As funções **int** e **float**, não.

No input, separe os elementos por **vírgula**

```
v: [11, 22, 33]
[11 22 33]
```

Na saída, o Python remove as vírgulas

Iniciando vetor :: zeros, uns, sequência

```
Arquivo Editar Buscar Executar Ferramentas
Python 3 main.py
from numpy import *
v0 = zeros(5, dtype=int)
v1 = ones(4, dtype=int)
print(v0)
print(v1)
```

The code editor shows two examples of creating NumPy arrays. The first example creates a vector of zeros with 5 elements, and the second creates a vector of ones with 4 elements. The number of elements (5 and 4) is highlighted with a green oval.

Quantidade de elementos no vetor

Se não informado, cria vetor de floats

```
Console Shell
[0 0 0 0 0]
[1 1 1 1]
```

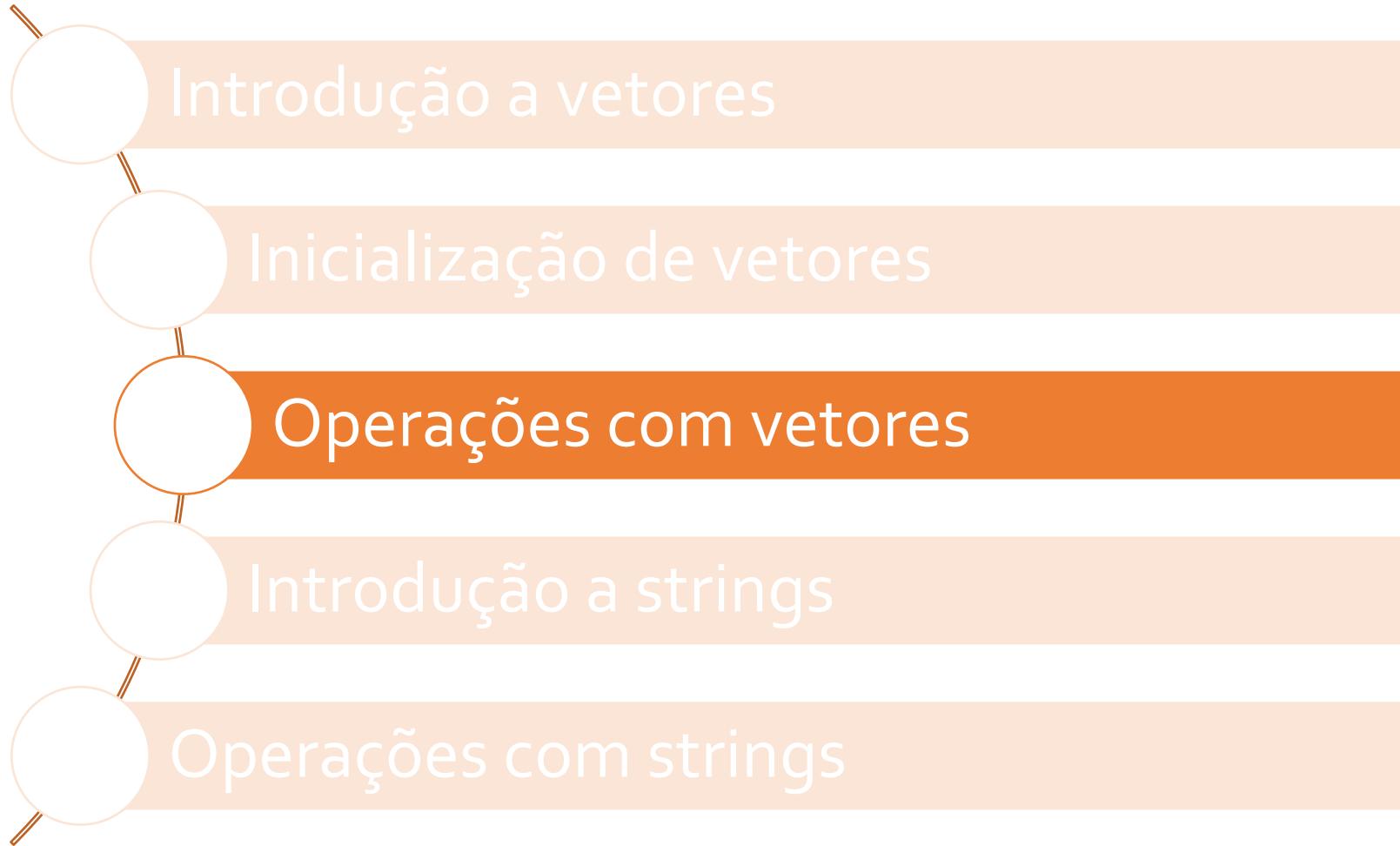
```
Arquivo Editar Buscar Executar Ferramentas
Python 3 main.py
from numpy import *
v = arange(5)
print(v)
```

The code editor shows a third example of creating a vector using the arange function, which generates a sequence from 0 to 4.

```
Console Shell
[0 1 2 3 4]
```

Sequência de zero a N-1

Conteúdo – Vetores e Strings



Operações com Vetores

:: Mínimo, Máximo, Tamanho, Soma

min (vet)

Menor elemento

max (vet)

Maior elemento

size (vet)

Quantidade de elementos

sum (vet)

Soma dos elementos

Problema 2

:: Estatísticas da turma

- 🛡️ Leia um vetor contendo as **notas** de uma turma.
- 🛡️ Determine:
 1. Qual a **maior** nota?
 2. Qual a **menor** nota?
 3. **Quantos** alunos tem na turma?
 4. Qual a **média** da turma?



Problema 2

2 – Identifique as entradas e saídas

	Grandeza	Unidade de medida	Faixa de valores
Entradas	Vetor de notas	---	[[0, 10]]
Saídas	Maior nota (escalar)	---	[0, 10]
	Menor nota (escalar)	---	[0, 10]
	Nº de alunos (escalar)	---	> 0
	Média das notas (escalar)	---	[0, 10]

Problema 2

3 – Projete o script

Qual a maior nota?

max(notas)

Qual a menor nota?

min(notas)

Quantos alunos tem na turma?

size(notas)

Qual a média da turma?

sum(notas) / size(notas)

Problema 2

4 – Codifique em Python

Não se esqueça
de mandar
imprimir

```
from numpy import *

# Leitura de dados
notas = array(eval(input("Notas: ")))

# Maior nota
print(max(notas))

# Menor nota
print(min(notas))

# Qtde. de alunos
print(size(notas))

# Media de desempenho
print(sum(notas) / size(notas))
```

Problema 2

5 – Teste o script



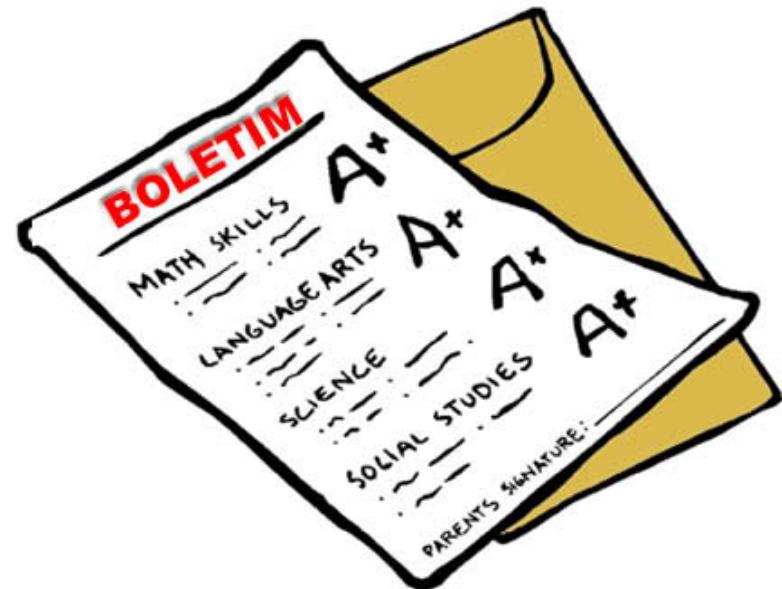
Notas: [1, 2, 3, 4, 5]

Use valores fáceis, que você
conheça a resposta **antes** de
executar o script

Problema 3

:: Média ponderada

- ◆ Coeficiente de rendimento (CR) é a **média ponderada** entre as notas de cada disciplina e os respectivos números de créditos.
- ◆ Escreva um script que leia dois vetores distintos:
 - ◆ **notas** de um aluno; e
 - ◆ **créditos** das disciplinas correspondentes.
- ◆ Determine o **CR** do aluno, com 03 casas de precisão.



Problema 3

2 – Identifique as entradas e saídas

	Grandeza	Unidade de medida	Faixa de valores
Entradas	Vetor de notas	---	[[0, 10]]
	Vetor de créditos	---	[[0, 10]]
Saídas	Coeficiente (escalar)	---	[0, 10]

Problema 3

3 – Projete o script

- 💡 Ler vetores **notas** e **creditos**

$$\text{coeficiente} = \frac{n_0 \cdot c_0 + n_1 \cdot c_1 + n_2 \cdot c_2 + \dots}{c_0 + c_1 + c_2 + \dots}$$

- 💡 Duas variáveis acumuladoras:

numerador → Acumula produtos nota * crédito

denominador → Acumula soma dos créditos

- 💡 Qual a condição para repetição?
 - 💡 Enquanto o contador não atingir o número de notas (tamanho dos vetores)

Problema 3

4 – Codifique em Python

```
from numpy import *
# Leitura dos vetores
notas = array(eval(input("Informe as notas: ")))
cred  = array(eval(input("Informe os creditos: ")))

i = 0          # Variavel contadora
numerador = 0    # Acumula produto notas * creditos
denominador = 0    # Acumula os creditos

while (i < size(notas)) : Ou size(cred)
    numerador = numerador + notas[i] * cred[i]
    denominador = denominador + cred[i]
    i = i + 1

coeficiente = numerador / denominador
print(round(coeficiente, 3))
```

Problema 3

5 – Teste o script



Notas: [1, 2, 3, 4]



Créditos: [4, 3, 2, 1]

Selecionando partes de um vetor

Código	Objetivo
<code>vec[i]</code>	Seleciona o elemento de índice <code>i</code> do vetor <code>vec</code>
<code>vec[i:j]</code>	Seleciona os elementos do vetor <code>vec</code> cujos índices estão compreendidos entre <code>i</code> e <code>j-1</code>
<code>vec[i:]</code>	Seleciona os elementos do vetor <code>vec</code> do índice <code>i</code> até o final do vetor
<code>vec[:i]</code>	Seleciona os elementos do vetor <code>vec</code> desde o primeiro até o de índice <code>i-1</code>

Problema 4

:: Votos válidos

- Leia um vetor contendo:
 - Quantidade de votos em cada candidato de uma eleição
 - Qtde. de votos nulos
 - Qtde. de votos brancos
 - Nº de eleitores ausentes

- Determine:
 - Quantos eleitores presentes?
 - Quantos votos válidos?
 - Qual o percentual de votos válidos do vencedor?



Problema 4

2 – Identifique as entradas e saídas

	Grandeza	Unidade de medida	Faixa de valores
Entradas	Vetor de votos	---	$[[\geq 0]]$
Saídas	Nº de eleitores presentes (escalar)	---	≥ 0
	Nº de votos válidos (escalar)	---	≥ 0
	% de votos válidos do candidato vencedor (escalar)	---	≥ 0

Problema 4

3 – Projete o script

Eleitores presentes?

sum(votos [:-1])

Seleciona até o elemento anterior ao último (-1)

Votos válidos?

sum(votos [:-3])

Exclui os três últimos (-3) elementos do vetor

Percentual de votos válidos do vencedor?

max(votos [:-3]) / sum(votos [:-3]) * 100

Evita erros devido a uma situação inválida ser mais numerosa

Problema 4

4 – Codifique em Python

```
from numpy import *

# Leitura de dados
votos = array(eval(input("Qtde de votos: ")))

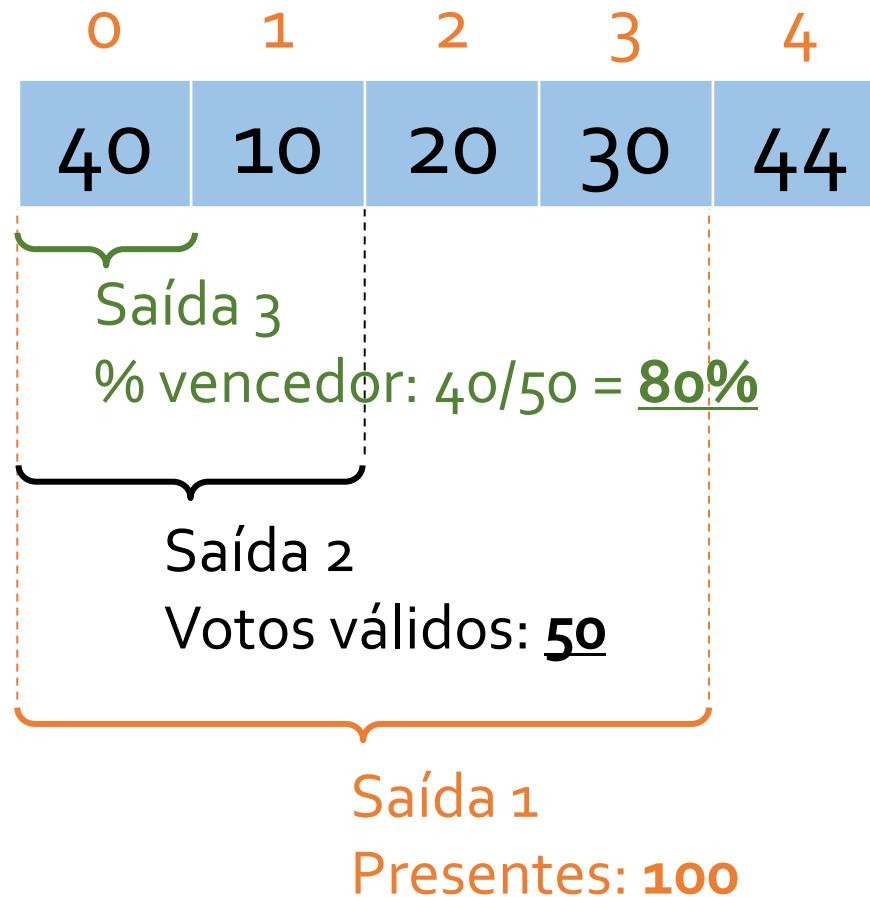
# Eleitores presentes
print(sum(votos[:-1]))

# Votos validos
print(sum(votos[:-3]))

# % de votos do vencedor
venc = max(votos[:-3])/sum(votos[:-3]) * 100
print(round(venc, 2))
```

Problema 4

5 – Teste o script



Problema 5

:: Qual o nome do vencedor?

- 🛡 Leia dois vetores:
 - 🛡 Nomes de candidatos
 - 🛡 Quantidade de votos

- 🛡 Determine:
 - 🛡 Qual o nome do candidato **mais** votado?



Problema 5

2 – Identifique as entradas e saídas

	Grandeza	Unidade de medida	Faixa de valores
Entradas	Vetor de nomes	---	[[String]]
Saídas	Vetor de qtde. de votos	---	[[≥ 0]]
	Nome do candidato vencedor (string)	---	String

Problema 5

3 – Projete o script (1/2)

- 🛡️ Vetores de entrada:

nomes → Vetor com os **nomes** dos candidatos

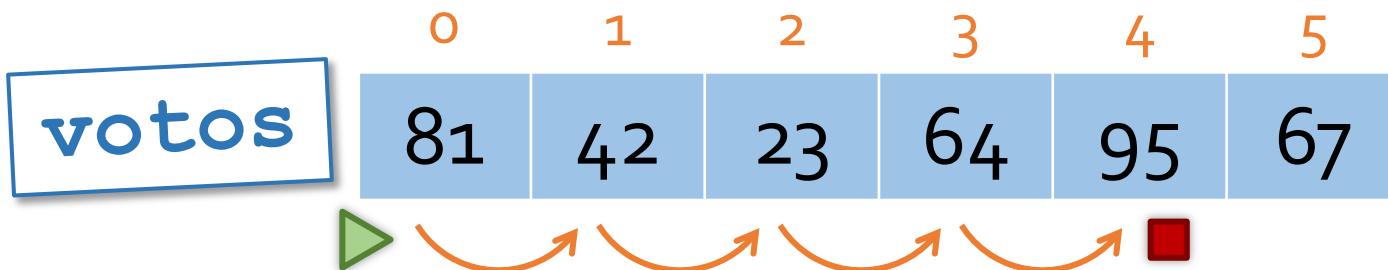
votos → Vetor com os **votos** recebidos

- 🛡️ Achar o nº de votos do candidato mais votado é fácil:
 - 🛡️ Função **max**
- 🛡️ Para saber o **nome** do candidato mais votado, precisamos:
 - 🛡️ Buscar o **índice** do maior elemento do vetor “votos”
 - 🛡️ Imprimir o **valor correspondente** no vetor “nomes”

Problema 5

3 – Projete o script (2/2)

- Para buscar o índice do maior elemento do vetor “votos”, devemos **percorrer** o vetor:



Contador:

- Valor inicial:** zero (primeiro índice do vetor)
- Condição do laço:** não encontrar o valor máximo
- Incremento:** 1

Problema 5

4 – Codifique em Python

```
from numpy import *

# Leitura de dados
nomes = array(eval(input("Nomes dos candidatos: ")))
votos = array(eval(input("Qtde de votos: ")))

# Primeiro indice
i = 0

while (votos[i] != max(votos)):
    i = i + 1

print(nomes[i])
```

Quando a condição for violada, então o valor máximo foi encontrado

Problema 5

5 – Teste o script

nomes	0	1	2	
votos	11	22	33	Carol
	33	22	11	Ana
	22	33	11	Bento

Problema 6

:: Problema da Busca

- 🛡 Escreva um programa que leia:
 - 🛡 Um vetor v
 - 🛡 Um valor x
- 🛡 Encontre um índice k tal que $v[k] == x$.



Problema 6

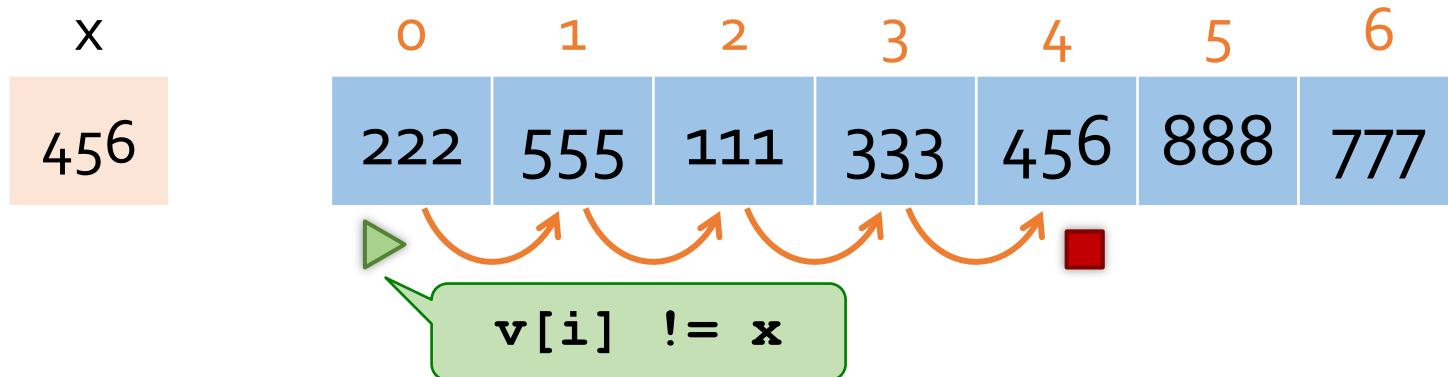
2 – Identifique as entradas e saídas

	Grandeza	Unidade de medida	Faixa de valores
Entradas	Vetor v	---	[[0, 10]]
	Valor x (escalar)	---	[0, 10]
Saídas	Posição (escalar)	---	{ ≥ 0 , mensagem}

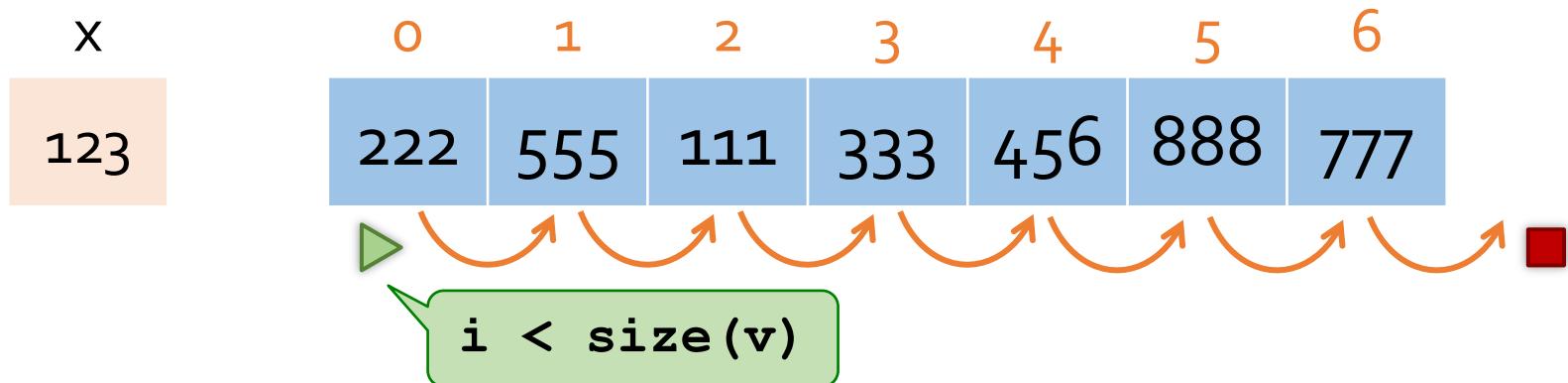
Problema 6

3 – Projete o script

- Caso 1: valor x está no vetor



- Caso 2: valor x não está no vetor



Problema 6

4 – Codifique em Python

```
from numpy import *

# Leitura de dados
v = array(eval(input("Vetor: ")))
x = int(input("Valor a ser encontrado: "))

i = 0

while (i < size(v) and v[i] != x):
    i = i + 1

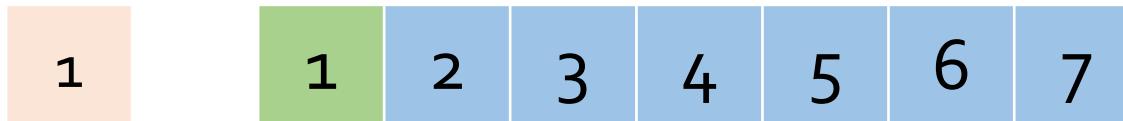
if (i < size(v)):
    print(i)
else:
    print("Valor não encontrado")
```

Se o contador for menor que o tamanho do vetor, então o elemento foi achado

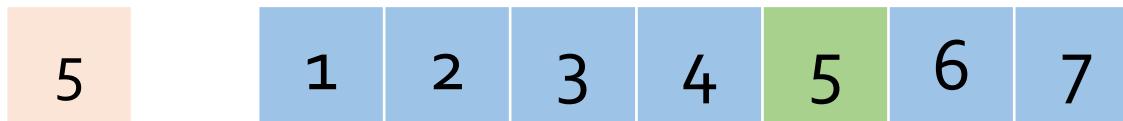
Problema 6

5 – Teste o script

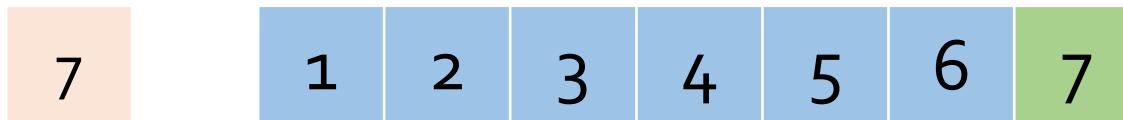
- ◆ **Teste 1:** valor **x** está no **início** do vetor



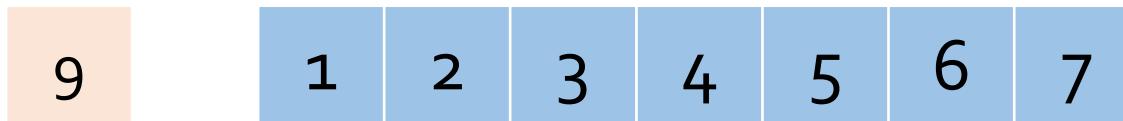
- ◆ **Teste 2:** valor **x** está no **meio** do vetor



- ◆ **Teste 3:** valor **x** está no **final** do vetor



- ◆ **Teste 4:** valor **x** **não** está no vetor

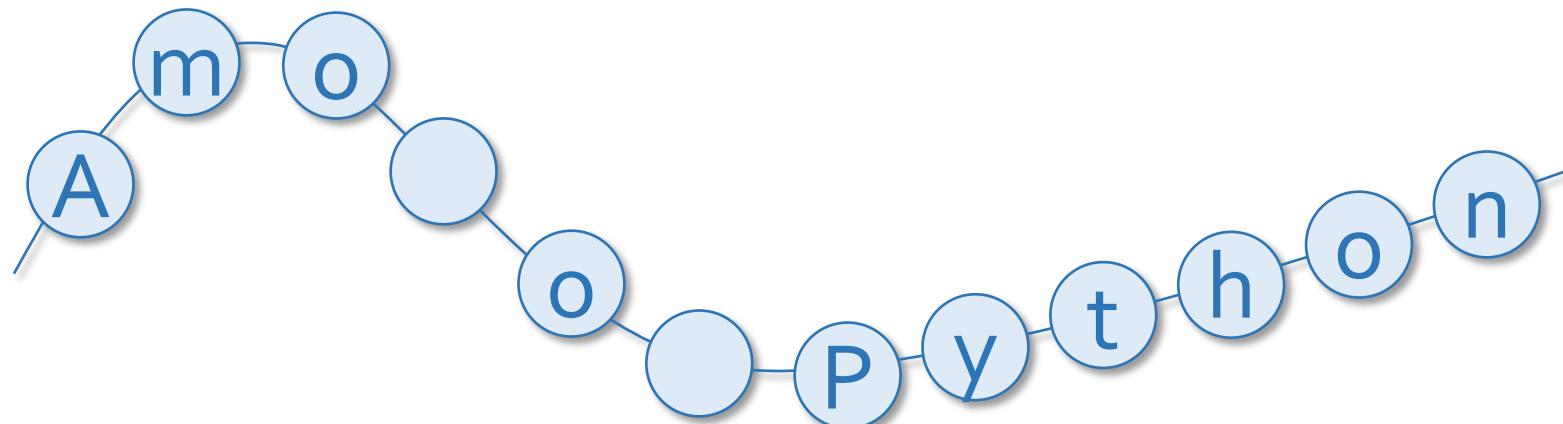


Conteúdo – Vetores e Strings



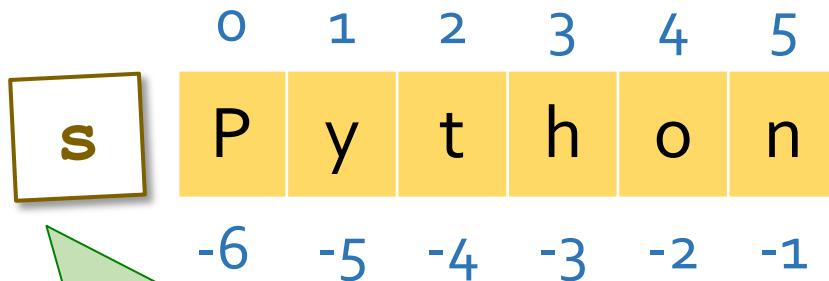
Strings

- Uma **string** (corda, em inglês) é uma sequência de caracteres.
- Não requer o módulo **numpy**.



Acesso aos caracteres

💡 Cada caractere é indexado pela sua posição, **semelhante** a um vetor.



Não use **str** como identificador,
pois é nome de função

The screenshot shows a Python 3 code editor with a file named "main.py". The code prints the first and last characters of the string "Python", and then attempts to print the 6th character, which results in an error. The error message is "TypeError: 'type' object is not subscriptable".

```
Arquivo Editar Buscar Executar Ferramentas
Python 3 main.py
s = "Python"
# Primeiro caractere
print(s[0])
# Ultimo caractere
print(s[5])
print(s[-1])
# ERRO
print(s[6])
```

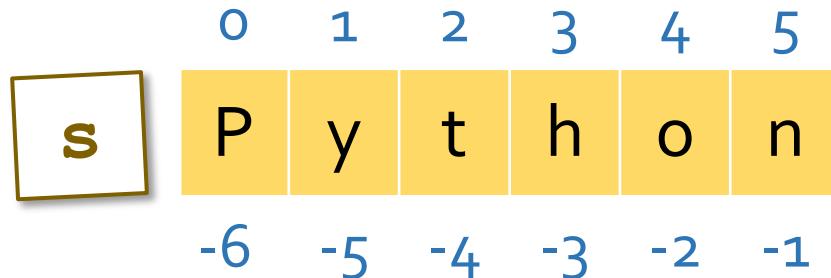
Console Shell

```
P
n
n
TypeError: 'type' object
is not subscriptable
```

Erro relativo aos índices

Seleção de partes da string

Além de caracteres individuais, você também pode selecionar **substrings**, ou seja, “fatias” da string.



A screenshot of a Python 3 IDE interface. The menu bar includes Arquivo, Editar, Buscar, Executar, and Ferramentas. The tab bar shows Python 3 and main.py. The code editor contains four print statements:

```
print(s[0:2])  
print(s[:2])  
print(s[3:6])  
print(s[2:])
```

The console output window shows the results of each print statement:

```
Py  
Py  
hon  
thon
```

Strings são imutáveis

:: strings x vetores

```
Arquivo Editar Buscar Executar Ferramentas
Python 3 main.py

s = "Tolkien"
s[0] = 'R'
print(s)

Console Shell

TypeError: 'str' object does not support item assignment
```

```
Arquivo Editar Buscar Executar Ferramentas
Python 3 main.py

from numpy import *
vet = array(['T', 'o', 'l', 'k', 'i', 'e', 'n'])
vet[0] = 'R'
print(vet)

Console Shell

['R' 'o' 'l' 'k' 'i' 'e' 'n']
```

Não é possível alterar
um caractere da string

Strings são imutáveis

:: Crie uma nova string

```
Arquivo Editar Buscar Executar Ferramentas
Python 3 main.py

s = "Tolkien"

new = 'R' + s[1:]

print(new)

Console Shell

Rolkien
```

Troca a primeira expressão pela segunda

```
Arquivo Editar Buscar Executar Ferramentas
Python 3 main.py

s = "Tolkien"

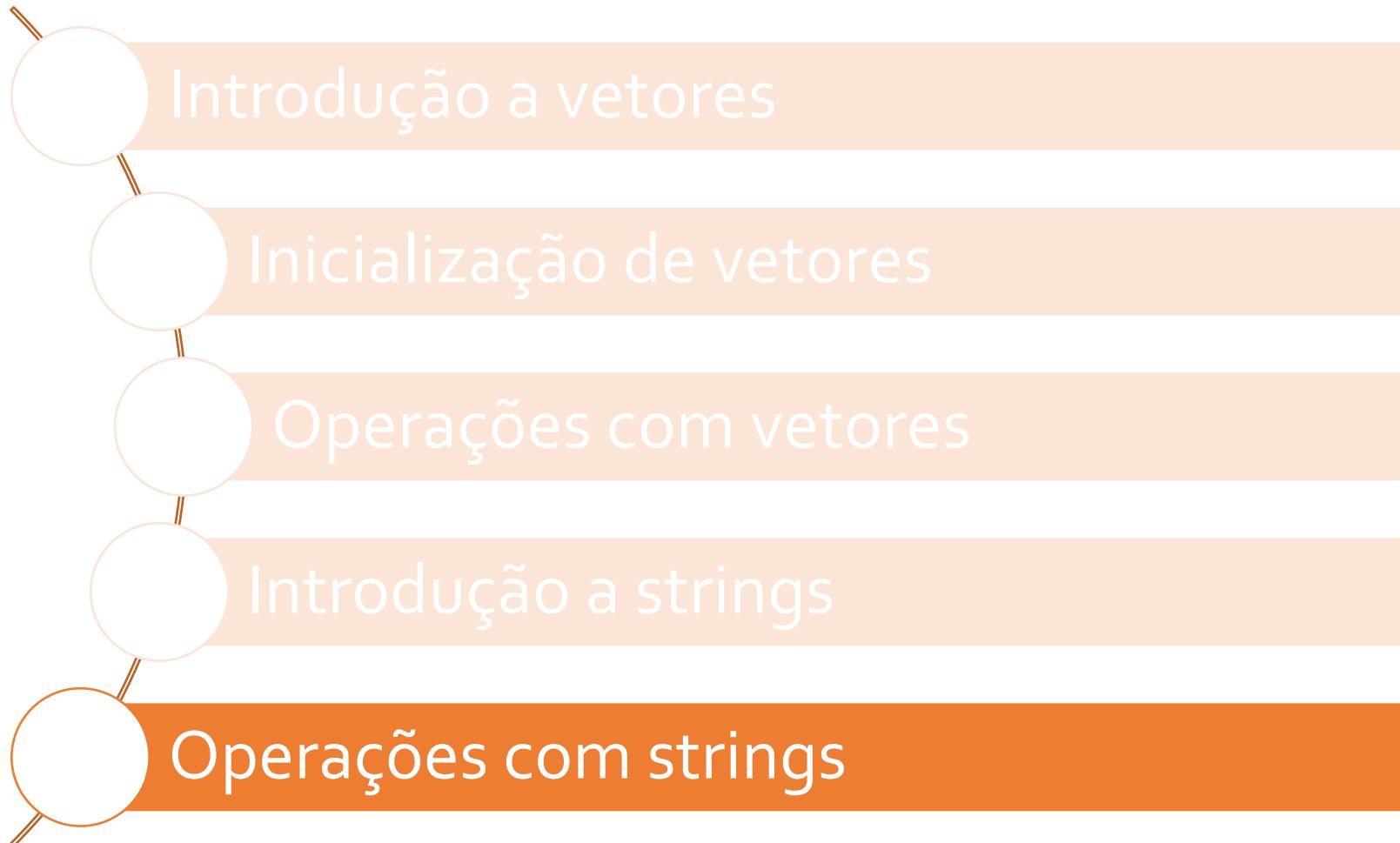
new = s.replace('T', 'R')

print(new)

Console Shell

Rolkien
```

Conteúdo – Vetores e Strings



Operações com strings

:: Tamanho, Conversão, Concatenação

`len(str1)`

Quantidade de caracteres

`str(num)`

Converte número em string

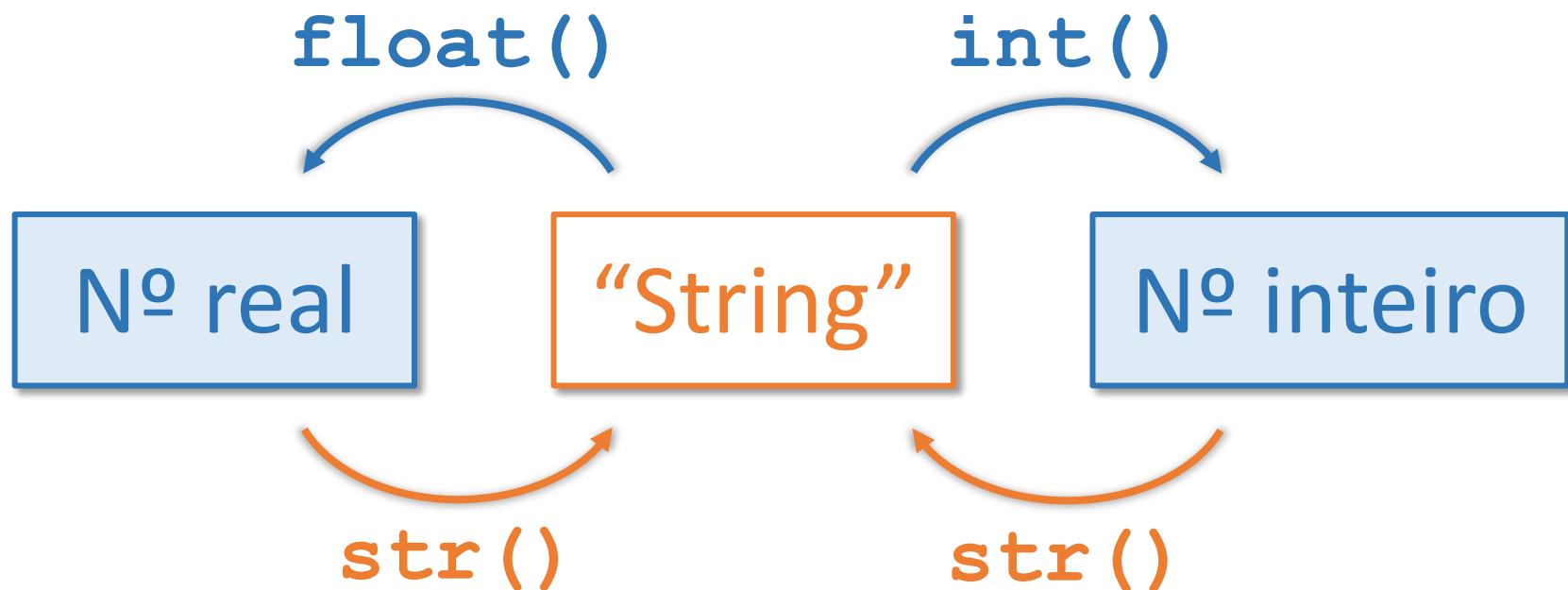
`s1 + s2`

Concatena as strings **s1** e **s2**

`s1 * n`

Replica **n** vezes a string **s1**

Conversão de strings para números e vice-versa



Problema 7

:: Copiando o DNA

- 🛡 O DNA é formado por uma longa sequência de bases nitrogenadas, de quatro tipos: Adenina (**A**), Citosina (**C**), Guanina (**G**), Timina (**T**).
- 🛡 Durante a duplicação de uma molécula de DNA:
 - 🛡 A se une com T
 - 🛡 C se une com G
 - 🛡 Vice-versa
- 🛡 Leia uma sequência de DNA e determine a **sequência-cópia** correspondente.



Problema 7

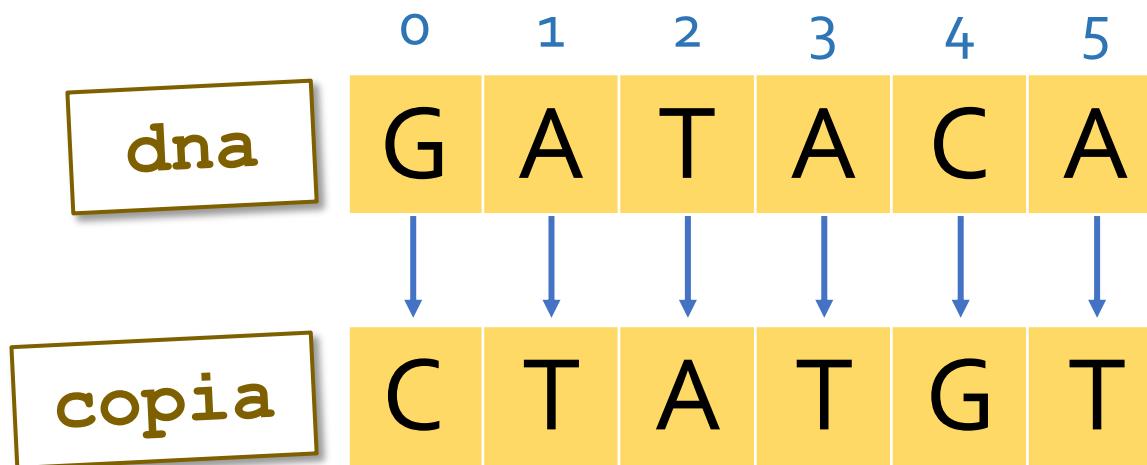
2 – Identifique as entradas e saídas

	Grandeza	Unidade de medida	Faixa de valores
Entradas	Sequência de DNA (string)	---	{"A", "C", "G", "T"}
Saídas	Sequência de DNA (string)	---	{"A", "C", "G", "T"}

Problema 7

3 – Projete o script

-)Vetores são criados com tamanho fixo, que **não pode ser mudado** ao longo do programa.
- Strings podem mudar de tamanho ao longo do programa.
 - Basta usar a concatenação (+)



Problema 7

4 – Codifique em Python

```
dna = input("Sequencia DNA: ").upper()  
  
# Cria string vazia  
copia = ""  
  
i = 0      # inicia contador  
while i < len(dna):  
    if dna[i] == 'A':  
        copia = copia + 'T'  
    elif dna[i] == 'C':  
        copia = copia + 'G'  
    elif dna[i] == 'G':  
        copia = copia + 'C'  
    elif dna[i] == 'T':  
        copia = copia + 'A'  
    i = i + 1    # incrementa contador  
  
print(copia)
```

Previne entradas digitadas em minúsculas

Concatena símbolos à string do DNA

Problema 7

5 – Teste o script

ACGT

Um de cada



Saída = TGCA

ggggg

Tudo 'G'



Saída = CCCCC

Problema 8

:: De trás pra frente

- 🛡 Elabore um programa que leia uma string.
- 🛡 Como saída, imprima a string na **ordem inversa**.



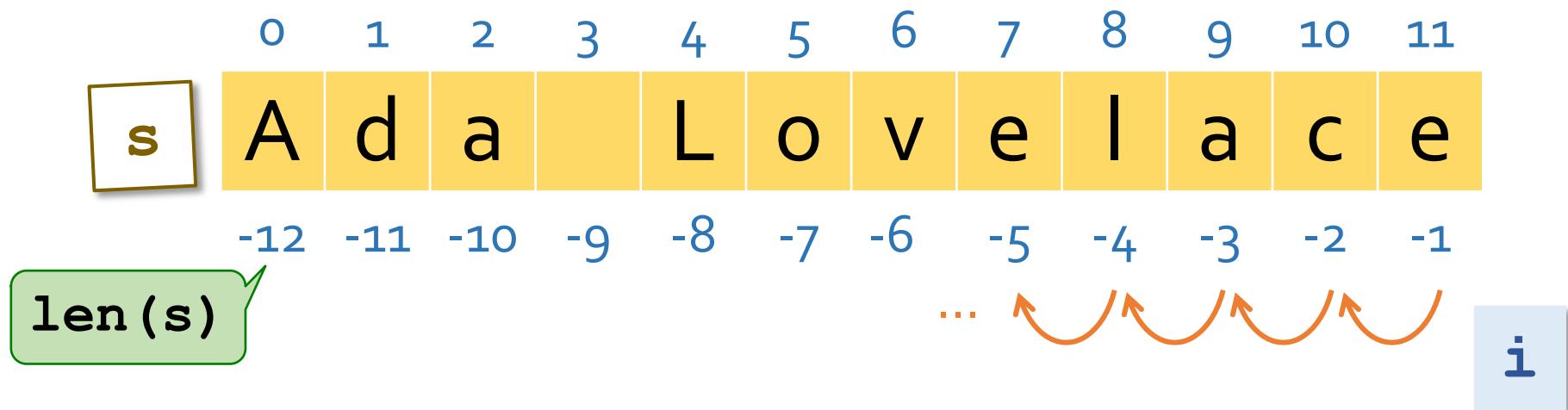
Problema 8

2 – Identifique as entradas e saídas

	Grandeza	Unidade de medida	Faixa de valores
Entradas	Frase (string)	---	Caracteres
Saídas	Frase invertida (string)	---	Caracteres

Problema 8

3 – Projete o script



Contador (`i`):

- **Valor inicial:** -1 (último índice da string)
- **Valor final:** -`len(s)`
- **Incremento:** -1 (decremento)

Problema 8

4 – Codifique em Python

```
s = input("Digite uma string: ")

# String vazia
inv = ""

# Contador começa no fim da string
i = -1
while i >= -len(s):
    inv = inv + s[i]
    i = i - 1
print(inv)
```

Desta vez, decremente
o contador

Problema 8

5 – Teste o script

0 1 2 3 4 5 6 7 8 9 10

s	A	I	a	n		T	U	r	i	n	g
---	---	---	---	---	--	---	---	---	---	---	---

0 1 2 3 4 5 6 7 8 9 10 11

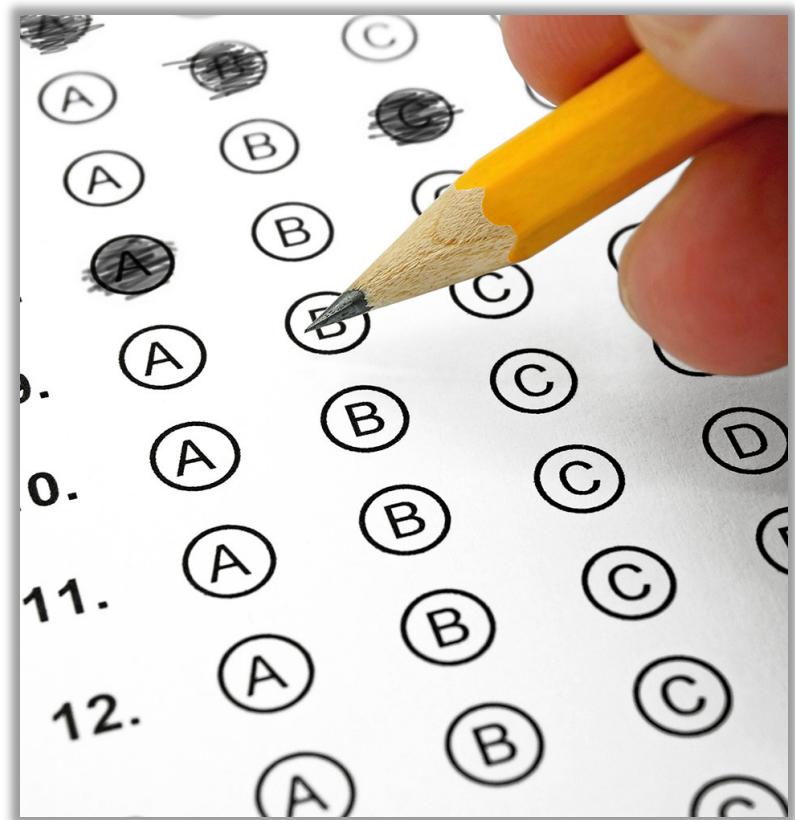
s	G	e	o	r	g	e		B	o	o	l	e
---	---	---	---	---	---	---	--	---	---	---	---	---

Problema 9

:: Gabarito de prova

- 🛡️ Leia duas strings:
 - 🛡️ **Gabarito** de uma prova de múltipla escolha
 - 🛡️ **Respostas** marcadas pelo aluno.

- 🛡️ Determine:
 - 🛡️ **Quantidade** de questões na prova.
 - 🛡️ Quantidade de questões marcadas **erradas**.
 - 🛡️ Percentual de **erro**.



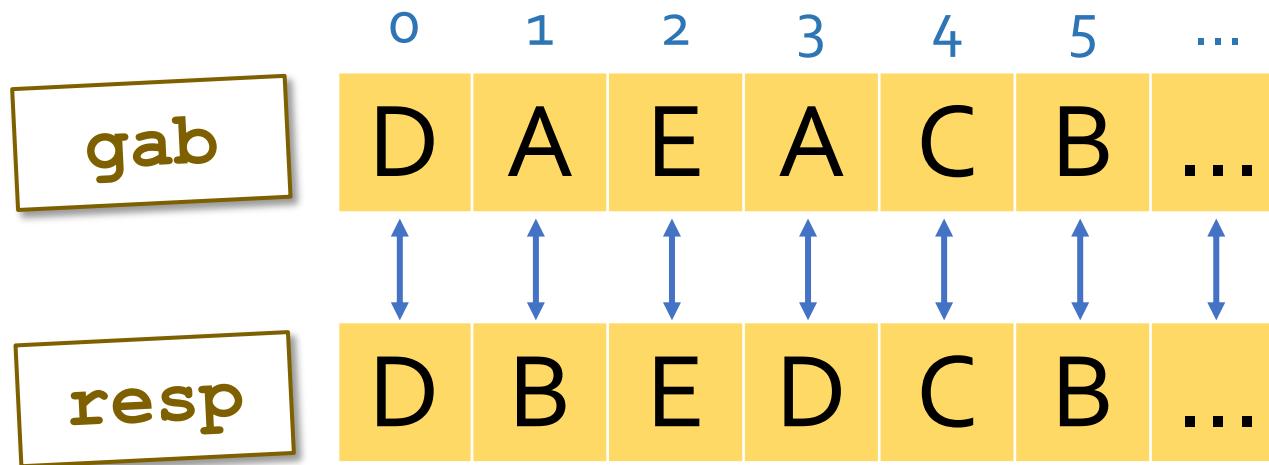
Problema 9

2 – Identifique as entradas e saídas

	Grandeza	Unidade de medida	Faixa de valores
Entradas	Gabarito (string)	---	{"A", "B", "C", "D", "E"}
	Respostas (string)	---	{"A", "B", "C", "D", "E"}
Saídas	Qtde. total de questões (escalar)	---	≥ 0
	Qtde. de erros (escalar)	---	≥ 0
	% de erro (escalar)	---	≥ 0.0

Problema 9

3 – Projete o script



Duas variáveis de controle:

i

Contador que percorre cada caractere das strings

erros

Acumulador do nº de erros cometidos

Problema 9

4 – Codifique em Python

```
gab = input("Gabarito: ").upper()
resp = input("Respostas: ").upper()

# inicia contadores
i = 0          # caracteres na string
erros = 0      # questoes marcadas erradas

while i < len(gab): ou i < len(resp):
    if gab[i] != resp[i]:
        erros = erros + 1
    i = i + 1

print(len(gab)) # total de questoes
print(erros)    # qtde de questoes erradas
# Percentual de erros:
print(100 * erros/len(gab))
```

Previne entradas digitadas em minúsculas

Problema 9

5 – Teste o script

ABCDE
ABCDE

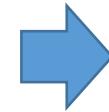
Tudo certo



5
0
0.0

ABCD
BCDE

Tudo errado



4
4
100.0

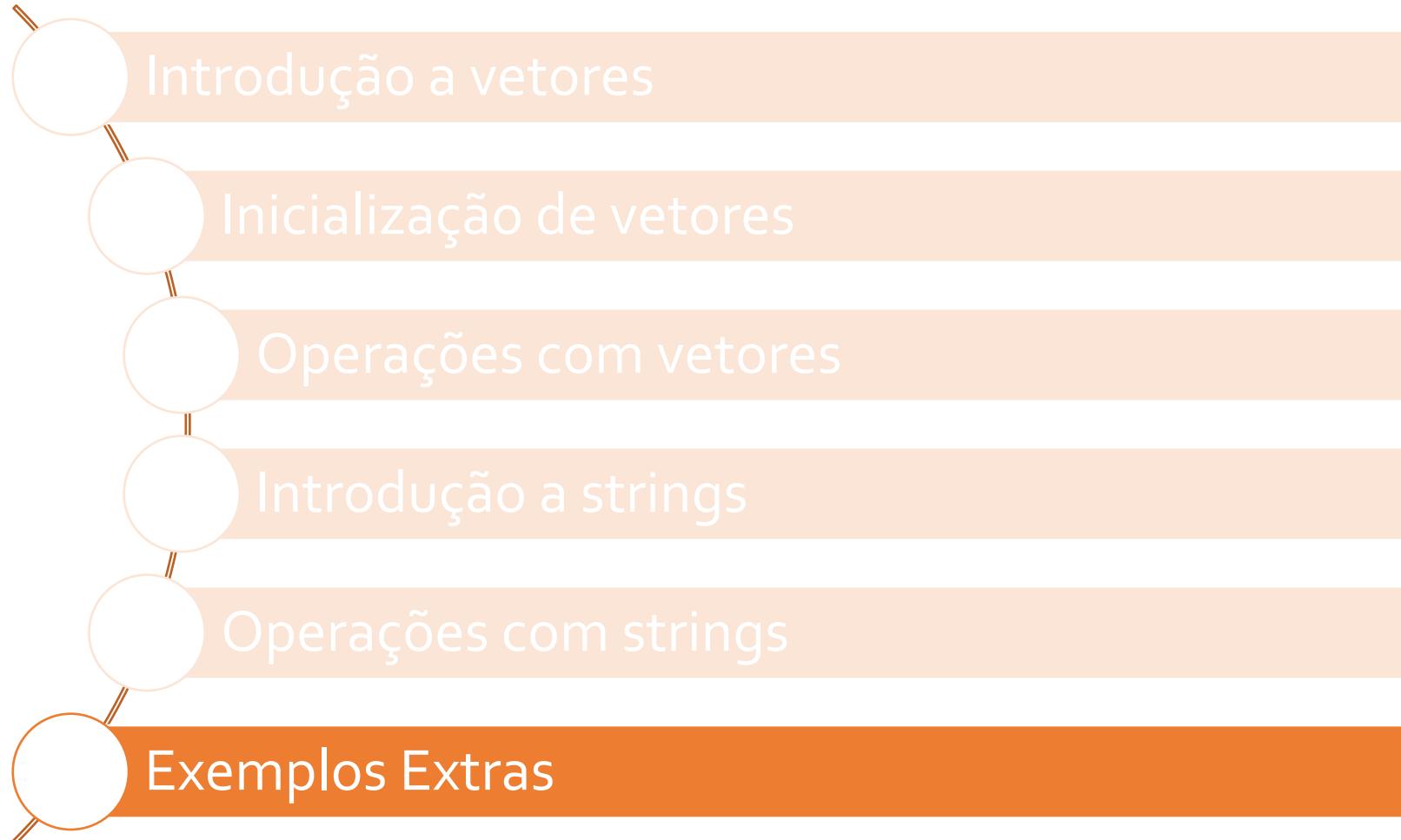
ABCD
ABDE

Misturado



4
2
50.0

Conteúdo – Vetores e Strings



Exemplo Extra 1

:: Dispensa da prova final

- 💡 Leia um vetor contendo as **médias parciais** de uma turma.
- 💡 Determine quantos alunos estão **dispensados** da prova final, ou seja, têm média parcial **maior** ou igual a 8,0.



Exemplo Extra 1

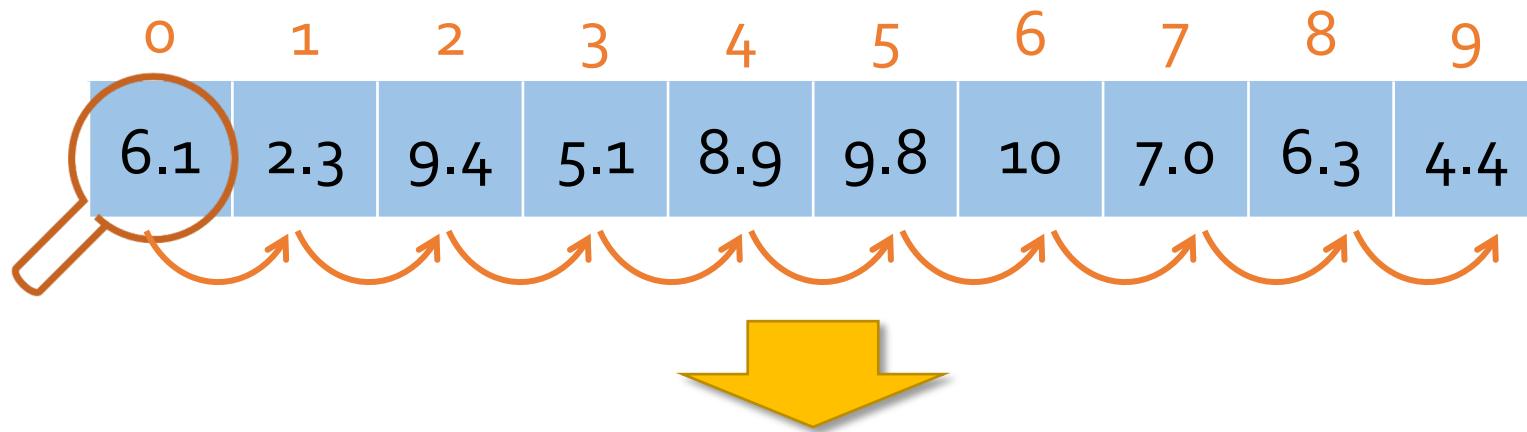
2 – Identifique as entradas e saídas

	Grandeza	Unidade de medida	Faixa de valores
Entradas	Vetor de médias parciais	---	$[[0, 10]]$
Saídas	Nº de alunos dispensados (escalar)	---	≥ 0

Exemplo Extra 1

3 – Projete o script

- 💡 Precisamos **percorrer** o vetor, analisando cada elemento individualmente.



laço **while**

i → Controla qual o índice do elemento pesquisado

cont → Conta quantos alunos estão dispensados

Exemplo Extra 1

4 – Codifique em Python

```
from numpy import *

# Leitura das notas
notas = array(eval(input("Notas: ")))

i = 0          # Variavel contadora
cont = 0        # Acumula notas acima de 8

# Percorre vetor, indice por indice
while (i < size(notas)):
    # Verifica se aluno estah dispensado
    if notas[i] >= 8:
        cont = cont + 1
    i = i + 1

print(cont)
```

Incrementa somente se nota for maior ou igual a 8

Exemplo Extra 1

5 – Teste o script



Notas: [1, 2, 3]

Ninguém está dispensado



Notas: [8, 9, 10]

Todos estão dispensados



Notas: [7, 8, 9]

Misturado

Exemplo Extra 2

:: Datas

- 💡 Escreva um programa que leia uma data no formato “dd/mm/aaaa”.
- 💡 Reescreva a data no seguinte formato:

Dia-Mês-Ano

- 💡 Regras:
 - 💡 **Dia**: se menor que 10, não pode ter zero à esquerda
 - 💡 **Mês**: string de exatamente três letras
 - 💡 **Ano**: quatro dígitos, igual ao original.



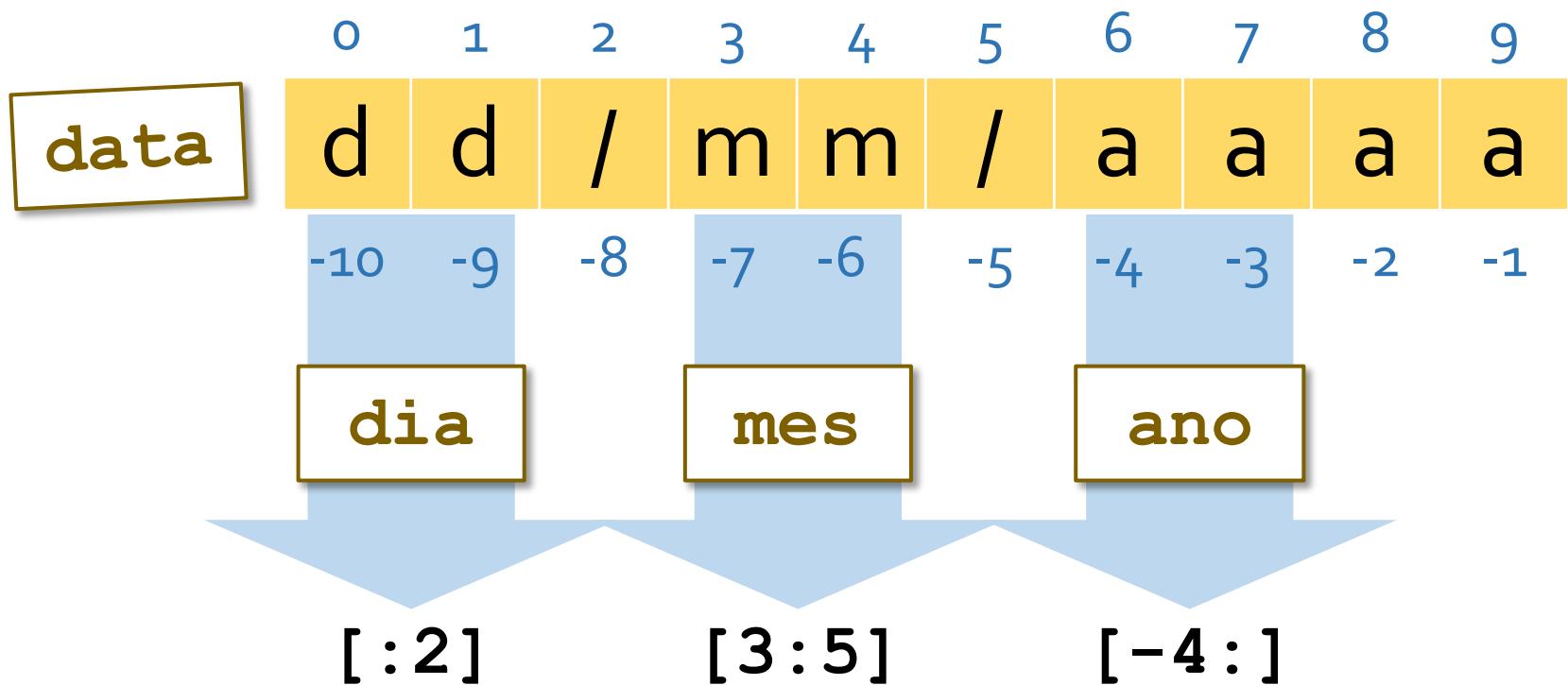
Exemplo Extra 2

2 – Identifique as entradas e saídas

	Grandeza	Unidade de medida	Faixa de valores
Entradas	Data (dd/mm/aaaa)	---	Caracteres
Saídas	Data (d-mmm-aaaa)	---	Caracteres

Exemplo Extra 2

3 – Projete o script – fatiando a string



Exemplo Extra 2

3 – Projete o script – achando o mês

- Para achar o mês, não é prático criar uma lista de doze casos **if-elif**.
- Em vez disso, guardamos a abreviatura dos meses em uma string:

```
'JanFevMarAbrMaiJunJulAgoSetOutNovDez'  
↑      ↑  
      fim = ini + 3  
  
      ↓  
ini = 3 * (mes - 1)
```

Exemplo Extra 2

4 – Codifique em Python

```
nome_meses = 'JanFevMarAbrMaiJunJulAgoSetOutNovDez'

data = input("Informe a data (dd/mm/aaaa): ")

dia = int(data[:2])
mes = int(data[3:5])           Transformar a string em inteiro  
remove o zero à esquerda
ano = data[-4:]

# Índices do fatiamento
ini = 3 * (mes - 1)
fim = ini + 3

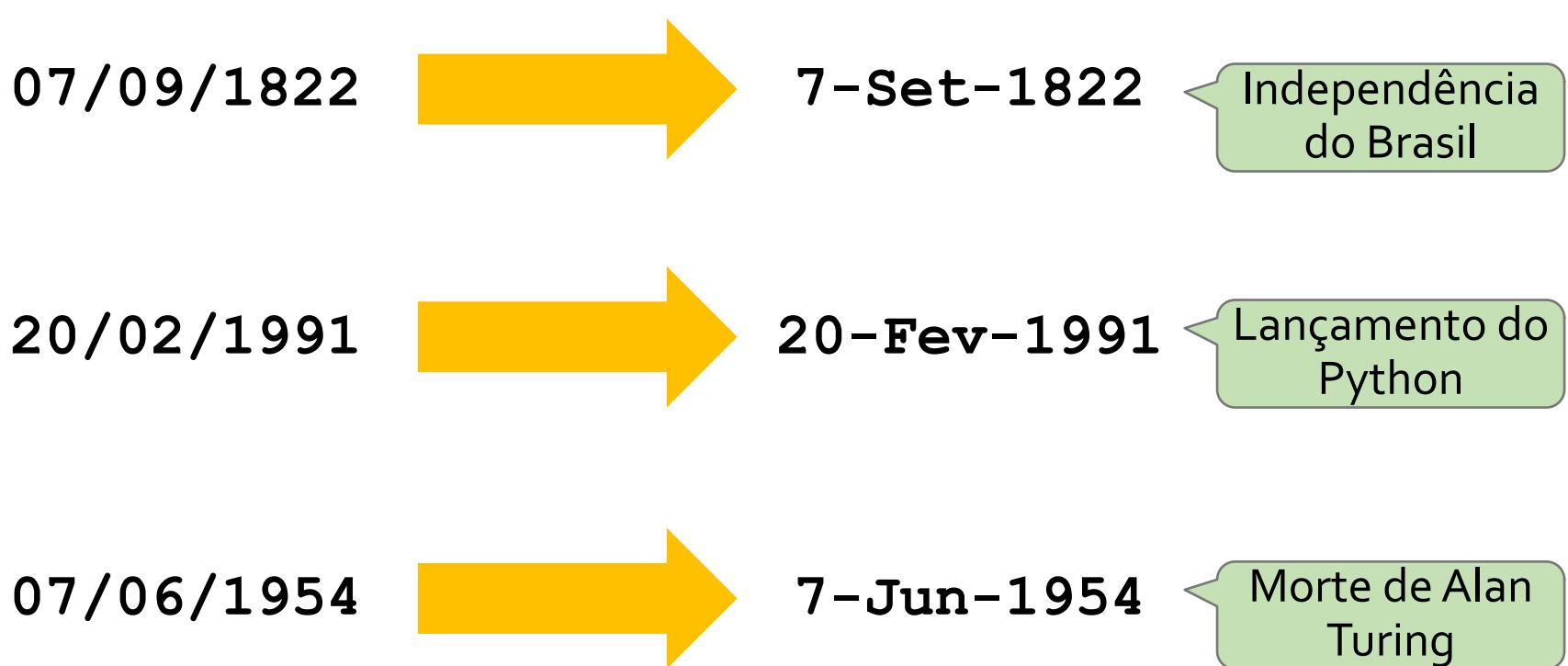
# Mes abreviado
mes_abrev = nome_meses[ini : fim]           Concatenação

nova_data = str(dia) + '-' + mes_abrev + '-' + ano

print(nova_data)
```

Exemplo Extra 2

5 – Teste o script



Exemplo Extra 3

:: Total de Danos

- 🛡 Um ataque a uma Aranha Dwarven causa um dano de:
 - 🛡 **20 pontos de força** se for na cabeça;
 - 🛡 **8 pontos de força** se for em outro lugar;
- 🛡 Escreva um programa que leia uma string, onde:
 - 🛡 **C** representa um ataque na cabeça;
 - 🛡 **O** representa um ataque em outro lugar.
- 🛡 Como saída, imprima o **dano total**.



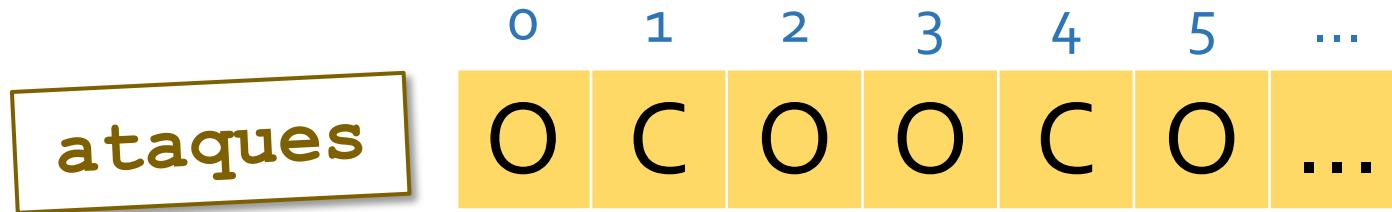
Exemplo Extra 3

2 – Identifique as entradas e saídas

	Grandeza	Unidade de medida	Faixa de valores
Entradas	Ataques (string)	---	{ "C", "O" }
Saídas	Total de Danos (escalar)	---	≥ 0

Exemplo Extra 3

3 – Projete o script



Duas variáveis de controle:

i

Contador que percorre cada caractere da string

danos

Acumulador da quantidade de dano sofrido

Exemplo Extra 3

4 – Codifique em Python

```
# Leitura do vetor ataques
ataques = input("Informe os ataques: ").upper()

i = 0                      # Variavel contadora
danos = 0                   # Acumula o dano total

while (i < len(ataques)):
    if (ataques[i] == "C"):
        danos = danos + 20
    elif (ataques[i] == "O"):
        danos = danos + 8
    i = i + 1

print(danos)
```

Exemplo Extra 3

5 – Teste o script

CCC

Tudo 'C'



Saída = $3 * 20 = 60$

OOO

Tudo 'O'



Saída = $3 * 8 = 24$

OCCO

2 de cada



Saída = $2 * 28 = 56$

Referências bibliográficas

-  MENEZES, Nilo Ney Coutinho (2014). *Introdução à Programação com Python*, 2 ed. Editora Novatec.
-  HETLAND, Magnus Lie (2008). *Beginning Python: From Novice to Professional*. Springer eBooks, 2^a edição.
Disponível em: <http://dx.doi.org/10.1007/978-1-4302-0634-7>.
-  HORSTMANN, Cay & NECAISE, Rance D. (2013). *Python for Everyone*. John Wiley & Sons.

Dúvidas?

