



UniEVANGÉLICA
UNIVERSIDADE EVANGÉLICA DE GOIÁS

Engenharia de Software

Algoritmos de Programação

Aula 16: Strings

Professor: Dr. Henrique Valle de Lima
henrique.lima@unievangelica.edu.br







▶ Uma string é uma sequência de letras (chamadas de caracteres).

▶ Em Python, as strings começam e terminam com aspas simples ou duplas.

```
>>> "foo"
```

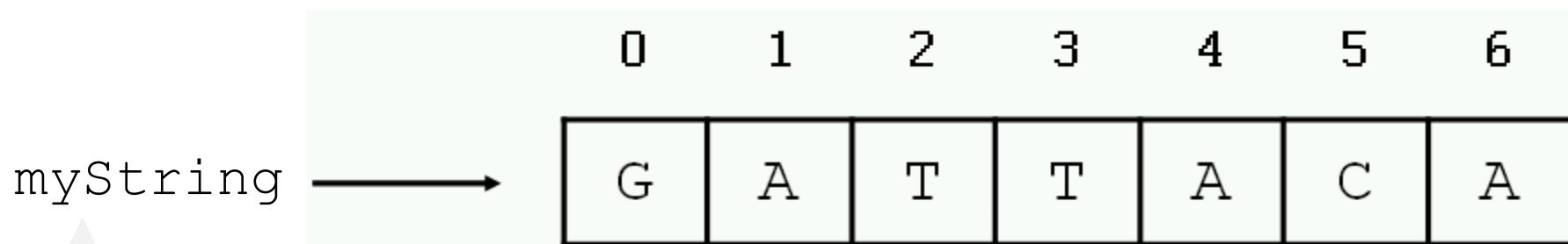
```
>>> 'foo'
```

```
'foo'
```

```
'foo'
```

▶ Cada string é armazenada na memória do computador como uma lista de caracteres.

```
>>> myString = "GATTACA"
```



▶ Você pode acessar caracteres individuais usando índices entre colchetes.

```
>>> myString = "GATTACA"
```

```
>>> myString[0]
```

```
'G'
```

```
>>> myString[1]
```

```
'A'
```

```
>>> myString[-1]
```

```
'A'
```

```
>>> myString[-2]
```

```
'C'
```

```
>>> myString[7]
```

```
Traceback (most recent call last):
```

```
  File "<stdin>", line 1, in ?
```

```
IndexError: string index out of range
```

ACESSANDO SUBSTRINGS

```
>>> myString = "GATTACA"
```

```
>>> myString[1:3]
```

```
'AT'
```

```
>>> myString[:3]
```

```
'GAT'
```

```
>>> myString[4:]
```

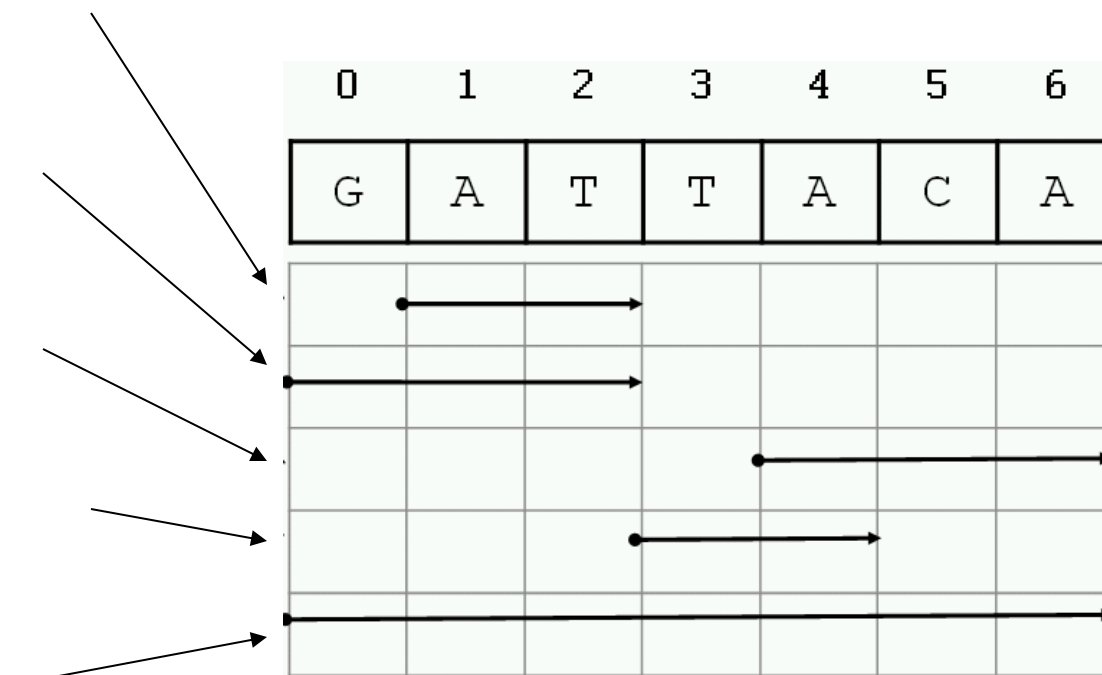
```
'ACA'
```

```
>>> myString[3:5]
```

```
'TA'
```

```
>>> myString[:]
```

```
'GATTACA'
```



MAIS FUNCIONALIDADES DA STRING

```
>>> len("GATTACA")
```

← Tamanho

```
7
```

```
>>> "GAT" + "TACA"
```

← Concatenação

```
'GATTACA'
```

```
>>> "A" * 10
```

← Repetir

```
'AAAAAAAAAA'
```

```
>>> "GAT" in "GATTACA"
```

← Teste de Substring

```
True
```

```
>>> "AGT" in "GATTACA"
```

```
False
```


STRINGS SÃO IMUTÁVEIS

🔑 Strings não podem ser modificadas; em vez disso, crie um novo.

```
>>> s = "GATTACA"
```

```
>>> s[3] = "C"
```

```
Traceback (most recent call last):
```

```
  File "<stdin>", line 1, in ?
```

```
TypeError: object doesn't support item assignment
```

```
>>> s = s[:3] + "C" + s[4:]
```

```
>>> s
```

```
'GATCACA'
```

```
>>> s = s.replace("G", "U")
```

```
>>> s
```

```
'UATCACA'
```

NÍMR



*Até a prátia é amada!
E você ai heim?
Sem ninguém!*

Ana Maria Braga



- ▶ Faça um programa que leia uma string e a imprima.
- ▶ Crie um programa que imprima o comprimento de uma string.
- ▶ Faça um programa para converter uma letra maiúscula em letra minúscula. Use a tabela ASCII para resolver o problema.
- ▶ Crie um programa que compara duas strings.
- ▶ Faça um programa que leia um nome e imprima as 4 primeiras letras do nome.
- ▶ Digite um nome, calcule e retorne quantas letras tem esse nome.
- ▶ Ler nome, sexo e idade. Se sexo for feminino e idade menor que 25, imprime o nome da pessoa e a palavra “ACEITA”, caso contrário imprimir “NÃO ACEITA”.

- ▶ Python possui uma estrutura similar a vetores denominada **listas**.
- ▶ **Lista** é um conjunto ordenado de valores, onde cada valor é identificado por um índice ;
- ▶ Os valores na lista são denominados **elementos**.
- ▶ Podemos denominar uma lista de **agregado homogêneo** unidimensional se todos os elementos são do **mesmo tipo**.

- Definir um vetor “nota” de tamanho 5 de tipo inteiro. `nota = [60, 95, 80, 50, 98]`

Índice	Valor
0	60
1	95
2	80
3	50
4	98

<code>nota[0]</code>
<code>nota[1]</code>
<code>nota[2]</code>
<code>nota[3]</code>
<code>nota[4]</code>

0	1	2	3	4
60	95	80	50	98

- Os valores 60, 95, 80, 50, 98 correspondem as notas, isto é, aos elementos do vetor.
- Os valores 0, 1, 2, 3, 4 correspondem aos índices.

Manipulando Elementos e Índices

▶ A manipulação do vetor depende da manipulação dos índices. Por exemplo, imprimir a quarta nota do vetor **nota**:

▶ `print (nota [3])`

0	1	2	3	4
60	95	80	50	98

```
>>> nota = [60,95,80,50,98]
>>> nota
[60, 95, 80, 50, 98]
>>> nota[3]
50
>>> |
```

▶ **nota** é a variável.

▶ 3 é o índice.

▶ [] é o operador de elemento.

▶ Lê-se: Acessando a variável **nota** na posição de **índice 3**.

▶ Acessar um índice inválido.

▶ `print (nota [5])` #Não existe o índice 5

0	1	2	3	4
60	95	80	50	98

▶ Não colocar o índice.

▶ `print (nota)` # Qual Posição???

▶ Não foi especificado o índice ;

▶ Em Python, imprime a lista inteira.

```
>>> nota[5]
Traceback (most recent call last):
  File "<pyshell#3>", line 1, in <module>
    nota[5]
IndexError: list index out of range
>>> print(nota)
[60, 95, 80, 50, 98]
>>> |
```

▶ Lista vazia, ou seja, sem elementos

▶ **Lista_vazia = []**

▶ Lista com elementos inteiros:

▶ **Lista_inteiros = [2, 4, 6, 8, 10]**

▶ Lista com elementos reais (oat):

▶ **Lista_reais = [9.0, 10.0, 8.5, 7.8]**

▶ Lista com elementos string:

▶ `Lista_frutas = ["abacaxi", "pera", "uva", "abacate"]`

▶ Lista de inteiros com tamanho 5 preenchida por zeros:

▶ `Lista_zerada = [0]*5`

▶ Lista de strings com tamanho 4 preenchida com strings vazias:

▶ `Lista_strings = [""]*4`

```
lista_vazia = []  
print(lista_vazia)  
  
lista_inteiros = [2,4,6,8,10]  
print(lista_inteiros)  
  
lista_reais = [9.0, 10.0, 8.5, 7.8]  
print(lista_reais)  
  
lista_frutas = ['abacaxi', 'pera', 'uva', 'abacate']  
print(lista_frutas)  
  
lista_zerada = [0] * 5  
print(lista_zerada)  
  
lista_strings = [""] * 4  
print(lista_strings)
```

```
>>>  
[]  
[2, 4, 6, 8, 10]  
[9.0, 10.0, 8.5, 7.8]  
['abacaxi', 'pera', 'uva', 'abacate']  
[0, 0, 0, 0, 0]  
['', '', '', '']  
///
```

▶ O acesso aos elementos de uma lista ocorre por meio de índices.

▶ O tamanho de uma lista é devolvido pela função **len()**

▶ **Lenght = Comprimento**


```
>>> []  
>>> [2, 4, 6, 8, 10]  
>>> [9.0, 10.0, 8.5, 7.8]  
>>> ['abacaxi', 'pera', 'uva', 'abacate']  
>>> [0, 0, 0, 0, 0]  
>>> ['', '', '', '']  
>>>
```

```
>>> comprimento = len(lista_inteiros)  
>>> comprimento  
5  
>>> |
```


FDS



FUTRIQUEI

 **AGORA:** Sandy e Lucas Lima, que se separaram em setembro, resolveram reatar o casamento, que já dura mais de 25 anos.



Acessando os Elementos da lista

```
lista_inteiros = [2,4,6,8,10]  
lista_frutas = ['abacaxi', 'pera', 'uva', 'abacate']
```

```
#PRIMEIRO ELEMENTO DA LISTA  
print(lista_inteiros[0])  
print(lista_frutas[0])  
#SEGUNDO ELEMENTO DA LISTA  
print(lista_inteiros[1])  
print(lista_frutas[1])  
#ÚLTIMO ELEMENTO DA LISTA  
print(lista_inteiros[-1])  
print(lista_frutas[-1])
```

```
>>>  
2  
abacaxi  
4  
pera  
10  
abacate  
>>>
```

```
lista_inteiros = [2,4,6,8,10]
```

```
i = 0
```

```
while (i < len(lista_inteiros)):  
    print(lista_inteiros[i])  
    i = i + 1
```

```
lista_inteiros = [2,4,6,8,10]
```

```
for i in range(0, len(lista_inteiros),1):  
    print(lista_inteiros[i])
```

```
>>>
```

```
2
```

```
4
```

```
6
```

```
8
```

```
10
```

```
>>> |
```

```
lista_inteiros = [2,4,6,8,10]
```

```
for i in range(len(lista_inteiros)):  
    print(lista_inteiros[i])
```

```
lista_inteiros = [2,4,6,8,10]
```

```
for i in lista_inteiros:  
    print(i)
```

```
>>>
```

```
2
```

```
4
```

```
6
```

```
8
```

```
10
```

```
>>>
```

Atribuição em listas em Python

➤ A atribuição permite criar uma nova lista ou modificar um elemento existente de uma lista.

```
lista = [3,5,7,9]  
print(lista) # saída: [3,5,7,9]
```

```
lista = [2]  
print(lista) # saída: [2]
```

```
lista = []  
print(lista) # saída: []
```


Atribuição em listas em Python

▶ Sintaxe:

▶ variável lista[índice] = elemento.

```
lista = [3,5,7,9]  
print(lista) # saída: [3,5,7,9]
```

```
lista[1] = 10  
print (lista) # saída: [3,10,7,9]
```

```
lista[0] = 8  
print (lista) # saída: [8,10,7,9]
```

```
lista[4] = 1 # ERRO: não existe índice 4
```

► Uma lista vazia é diferente de uma lista com elementos vazios.

```
[] != [""] != [0]
```

```
[] # possui nenhum elemento e o tamanho é 0.
```

```
[""] # possui o elemento string vazia e o tamanho é 1.
```

```
[0] # possui o elemento zero e o tamanho é 1.
```

- ▶ Em Python, a estrutura de lista é dinâmica, ou seja, permite adicionar e remover elementos em uma lista existente.
- ▶ Ao adicionar ou remover um elemento, o tamanho da lista também é modificado.
- ▶ Podemos adicionar um elemento ou vários elementos.

➤ Usando o operador + (concatenação)

```
lista = [2,4,6,8,10]
print(lista)
lista = lista + [5.0]
print(lista)
```

```
>>>
[2, 4, 6, 8, 10]
[2, 4, 6, 8, 10, 5.0]
>>> =====
```

```
lista = [2,4,6,8,10]
print(lista)
lista += [5.0]
print(lista)
```

➤ Usando o método append

```
lista = [2,4,6,8,10]  
print(lista)  
lista.append(99.0)  
print(lista)
```

```
>>>  
[2, 4, 6, 8, 10]  
[2, 4, 6, 8, 10, 5.0]  
>>> =====
```

➤ O append adiciona elementos e o extend expande listas

➤ O append permite adicionar listas dentro de listas, enquanto o extend é apenas para concatenar listas

```
lista = [2,4,6,8,10]
print(lista)
lista.extend([99.0, 10.1, 50])
print(lista)
```

```
>>>
```

```
[2, 4, 6, 8, 10]
```

```
[2, 4, 6, 8, 10, 99.0, 10.1, 50]
```

```
>>>
```

➤ O append adiciona elementos e o extend expande listas

➤ O append permite adicionar listas dentro de listas, enquanto o extend é apenas para concatenar listas.

➤ `lista = [7, 9, 11]`

➤ `lista.append([13, 15])`

➤ `print (lista)` #saida: `[7, 9, 11, [13, 15]]`

➤ `print (lista[3])` #saida: `[13, 15]`

▶ Entrada de dados em uma lista de tamanho fixo

```
nomes = [""] * 4 # define uma lista de tamanho 4
print(nomes)
```

```
print ("Digite 4 nomes:")
for i in range(4):
    nomes[i] = input()
print(nomes)
```

```
>>>
['', '', '', '']
Digite 4 nomes:
Joao
Maria
Jose
Rogério
['Joao', 'Maria', 'Jose', 'Rogério']
>>>
```

▶ Entrada de dados em uma lista de tamanho fixo

```
notas = [] * 4 # define uma lista de tamanho variável  
print(notas)
```

```
nota = float(input("Digite as notas (Digite -1 para Sair):"))
```

```
while (nota >=0):  
    notas.append(nota)  
    nota = float(input("Digite as notas (Digite -1 para Sair):"))  
print(notas)
```

```
>>>  
[]  
Digite as notas (Digite -1 para Sair):90  
Digite as notas (Digite -1 para Sair):80  
Digite as notas (Digite -1 para Sair):70  
Digite as notas (Digite -1 para Sair):60  
Digite as notas (Digite -1 para Sair):50  
Digite as notas (Digite -1 para Sair):-1  
[90.0, 80.0, 70.0, 60.0, 50.0]  
>>> |
```

- ▶ Uma operação comum em listas é verificar a presença de um determinado elemento.
- ▶ Em muitas situações também é necessário saber a posição do elemento.
- ▶ A forma mais comum é percorrer a lista comparando cada elemento com o valor procurado.
- ▶ Em Python, o operador **in** verifica se um elemento está contido na lista e o operador **not in** verifica se um elemento não está contido na lista.

```
lista = [6.0, 8.5, 9.0, 4.2]
valorProcurado = 8.5

for i in range(len(lista)):
    if valorProcurado == lista[i]:
        print("Valor Procurado no Indice: ", i) # Retorna índice do valor (i)
    else:
        print("Diferente!") # Se não existir o valor
```

```
>>>
Diferente!
Valor Procurado no Indice:  1
Diferente!
Diferente!
>>>
```

```
lista = [6.0, 8.5, 9.0, 4.2]  
valorProcurado = 8.5
```

```
if valorProcurado in lista:  
    print("Valor Procurado encontra-se na Lista ")
```

```
>>>  
Valor Procurado encontra-se na Lista  
>>>
```

- ▶ Em uma lista podemos remover elementos.
- ▶ O Python possui o operador **del** para remover um elemento em uma posição específica da lista.
- ▶ O índice da posição deve existir, senão o operador devolve erro ao remover o elemento.
- ▶ Após a remoção de um elemento da lista, o tamanho da lista é diminuído e o índice dos elementos subsequentes são diminuídos em uma unidade.

Removendo Elementos da lista

```
lista = [6.0, 8.5, 9.0, 4.2]  
print(lista)
```

```
del lista[2]  
print(lista)
```

```
>>>  
[6.0, 8.5, 9.0, 4.2]  
[6.0, 8.5, 4.2]  
>>>
```


1. Faça um programa que possua um vetor denominado A que armazene 6 números inteiros. O programa deve executar os seguintes passos:

- (a) Atribua os seguintes valores a esse vetor: 1, 0, 5, -2, -5, 7.
- (b) Armazene em uma variável inteira (simples) a soma entre os valores das posições A[0], A[1] e A[5] do vetor e mostre na tela esta soma.
- (c) Modifique o vetor na posição 4, atribuindo a esta posição o valor 100.
- (d) Mostre na tela cada valor do vetor A, um em cada linha.

2. Crie um programa que lê 6 valores inteiros e, em seguida, mostre na tela os valores lidos.

3. Ler um conjunto de números reais, armazenando-o em vetor e calcular o quadrado das componentes deste vetor, armazenando o resultado em outro vetor. Os conjuntos têm 10 elementos cada. Imprimir todos os conjuntos.

4. Escreva um programa que leia 10 números inteiros e os armazene em um vetor. Imprima o vetor, o maior elemento e a posição que ele se encontra.
5. Crie um programa que lê 6 valores inteiros e, em seguida, mostre na tela os valores lidos na ordem inversa.
6. Crie um programa que lê 6 valores inteiros pares e, em seguida, mostre na tela os valores lidos na ordem inversa.
7. Faça um programa para ler a nota da prova de 15 alunos e armazene num vetor, calcule e imprima a média geral.

