



Algoritmos e Lógica de Programação

Aula 08

Strings

Prof. Dr. Dilermando Piva Jr

1º Semestre - DSM



Estrutura de dados STRING

Para começar...

- *Strings* são cadeias de caracteres que armazenam dados textuais e, portanto, podem armazenar informações para as mais diversas finalidades.
- O conteúdo de uma *string* pode representar um fato em si, ou uma informação. Por exemplo, se uma *string* armazena um valor igual a 120, isso é um dado, que somente será entendido conhecendo seu contexto.
- Agora, se uma String armazena a frase: “Neste mês vendemos 120 motores, e isso foi muito bom, pois significou um aumento de 30% nas vendas desse produto, em relação ao mesmo período do ano passado.”. Temos, então, uma informação armazenada na **string**.

Para começar...

- Podemos, a partir desses exemplos simples, imaginar a importância desse tipo de variável *Strings*, na construção de algoritmos ou programas de computadores.
- A partir do estabelecimento de relações entre dados, contidos em *strings*, podemos gerar informação, que por sua vez poderá, a partir de outros relacionamentos, gerar conhecimento.
- A maioria dos mecanismos de busca, que conhecemos na Internet, funcionam manipulando extensas cadeias de caracteres, contidas em milhares de bases de dados nessa rede.
- Essas cadeias de caracteres, contidas em textos curtos ou longos, são denominadas *strings*.

Strings

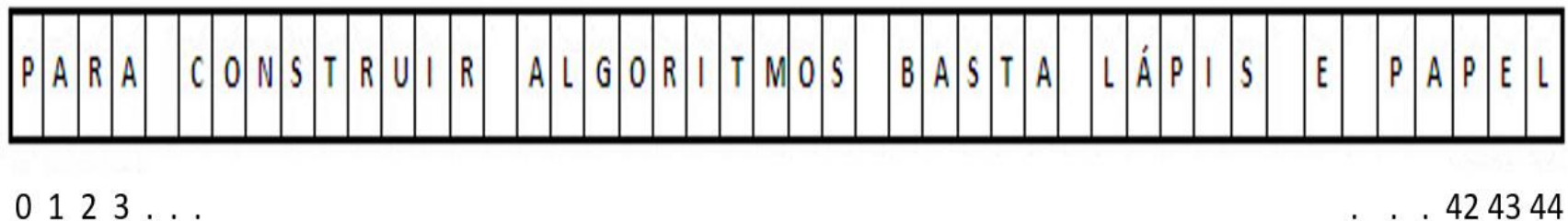
- Vamos aprender a manipular variáveis do tipo *string*, atribuindo valores a mesma, e recuperando seu conteúdo, usando linguagem algorítmica ou de programação de computador.
- Dependendo da linguagem de programação, *string* pode ser um tipo de dado primitivo, uma classe, ou mesmo um tipo criado pelo programador.
- A *string* referencia como “cadeia de caracteres” e, portanto, poder ser visualizada como uma lista linear ou vetor.
- Cada elemento da string é um caractere, e o agrupamento deles irá representar um dado ou uma informação.

Tipo Strings

- Em *strings*, os caracteres são armazenados da esquerda para a direita.

Exemplo de uma *string* (uma frase) contendo 45 posições:

String ou Cadeia de Caracteres



Tipo String

TIPO STRING (CHAR ou character)

ATENÇÃO: 1 ou mais caracteres = STRING

ATENÇÃO 2: `s = 'a'` ou `s = "a"` -> sem quebra de linha

`s = '''a'''` ou `s = """a"""` -> com quebra de linha

Tipo String

```
letra='a'  
palavra='pyPRO'
```

```
type(letra)  
type(palavra)
```

```
frase="Seja um profissional Python!"  
print(frase)  
print(frase[1])
```

Variável String

Matriz unidimensional (vetor) do tipo char (objeto)

- cada caractere de um string pode ser acessado individualmente
- vetor de tamanho $n \rightarrow$ posição varia de 0 a $n-1$

Ex:

```
s1 = "Exemplo1"  
s2 = "123"  
print(s1[0])  
print(s2[2])
```

Saída:

E

3

Lendo Strings

- `input()` → a função `input` retorna um objeto do tipo `string` (sempre). Para obter outro tipo de objeto, temos que fazer a conversão.

Ex:

```
nome = input("Digite seu nome: ")  
print(f"Bom dia {nome} !")  
print(type(nome))
```

Saída:

Digite seu nome: **Jose Maria**

Bom dia **Jose Maria !**

<class 'str'>

Tipo String

```
frase2=['S','e','j','a',' ','u','m',' ','p']  
print(frase2[1])
```

Tipo String - slice

```
frase="Seja um profissional Python!"
```

```
#Slices de strings  
print(frase[0:4])
```

Tipo String - slice

```
frase="Seja um profissional Python!"
```

```
#Slice de strings: 3 parâmetros:
```

```
#1 - início
```

```
#2 - limite superior (ele pegará até o n-1)
```

```
#3 - tamanho do passo (se deixar em branco, passo igual a 1)
```

```
print(frase[0:15:1])
```

```
print(frase[0:15:2])
```

```
print(frase[15:0:-1])
```

```
print(frase[15::-1])
```

```
print(frase[::-1])
```

String é um interável

```
s1 = 'Piva'  
s2 = 'Jr.'
```

- Pode se usar operações como:

- slicing (`[]`, `[:]`)

```
print(s2[0])
```

J

```
print(s1[1:3])
```

iv

- concatenação (+)

```
print(s1+' '+s2)
```

Piva Jr.

- repetição (*)

```
print(s1*3)
```

PivaPivaPiva

- *membership* (in)

```
for i in s1:
```

```
    print(i)
```

P
i
v
a

Funções de manipulação de strings

<code>str (num)</code>	Converte um número em String
<code>len (str)</code>	Retorna o tamanho de uma String

```
n = 123456789
numero = str(n)
print(type(n))
print(type(numero))

print(len(numero))
```

```
<class 'int'>
<class 'str'>

9
```

Métodos de strings

<code>str.count (s)</code>	Retorna a quantidade de conjuntos <code>s</code> presentes na string
<code>str.isalpha ()</code>	Retorna <code>False</code> se a string contiver algum caracter que não seja letras
<code>str.isdigit ()</code>	Retorna <code>False</code> se a string contiver algum caracter que não seja número
<code>str.lower ()</code>	Retorna a string transformada em minúsculos
<code>str.upper ()</code>	Retorna a string transformada em maiúsculos
<code>str.replace (old, new)</code>	Substitui uma porção da string por outro conteúdo
<code>str.strip ()</code>	Retira espaços em branco no começo e no fim da string
<code>str.title ()</code>	Retorna a string capitalizada (iniciais em maiúscula)
<code>str.split (delimiter)</code>	Separa uma string conforme um delimitador. É o inverso do <code>join()</code>
<code>str.join (sequence)</code>	Junta cada item da string com um delimitador especificado. É o inverso do <code>split()</code> .

Tipo String - métodos

```
dir(frase)
print(frase.split())
print(frase.split()[2])
print(frase.upper())
print(frase.lower())
print(frase.swapcase())
```

```
frase2 = '    Texto    '
print(frase2)
print(frase2.strip())
```

VAMOS PARA A PRÁTICA ?!!!



EXERCÍCIO 1

Elabore um algoritmo para ler/receber, separadamente, o primeiro nome, o nome do meio e o sobrenome de uma pessoa. Em seguida, mostre o nome completo, correspondente.

EXERCÍCIO 2

Faça um algoritmo que solicite uma data no formato de uma string – dd/mm/aaaa.

Mostre essa data no formato AAAAMMDD

EXERCÍCIO 3

Elabore um algoritmo para determinar quantas vogais existem dentro de uma determinada frase (que deve ser recebida do usuário).

EXERCÍCIO 4

Faça um algoritmo para determinar quantas palavras existem em uma determinada frase Obs: tanto a palavra, quanto a frase, devem ser informadas pelo usuário.

EXERCÍCIO 5

Faça um algoritmo para determinar se um determinada palavra, digitada pelo usuário, é um palíndroma.

Palíndromo: lido da direita para a esquerda, ou vice versa, representam a mesma coisa.

Ex: AMA

EXERCÍCIO 6

Faça um algoritmo para ler nove caracteres numéricos em uma string. Mostre o conteúdo dessa string colando pontos e vírgula, respectivamente nas posições inteiras e decimais.

Exemplo:

Digitado> 987654321

Mostrado> 9.876.543,21