Aula 03

LIP

Linguagem de Programação

Python

Variáveis Strings

Variáveis do tipo string armazenam **cadeias de caracteres** como nomes e textos em geral.

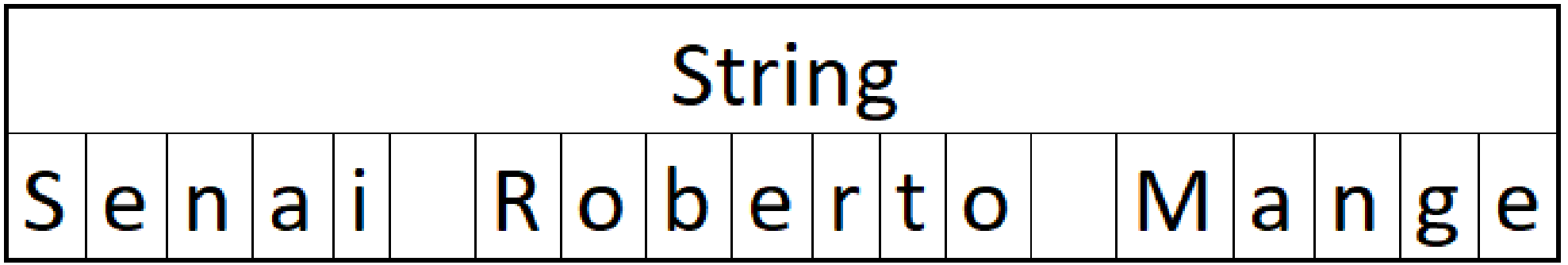
Chamamos cadeia de caracteres uma s**equência** de símbolos como letras, números, sinais de pontuação etc.

Exemplo: **Senai Roberto Mange**. Nesse caso, **Senai** é uma **sequência** com as letras **S**, **e**, **n**, **a** e **i**.

Para simplificar o texto, utilizaremos o nome **string** para mencionar **cadeias de caracteres**.

Python

Variáveis Strings



Função Len

Essa função retorna o número de caracteres na string.

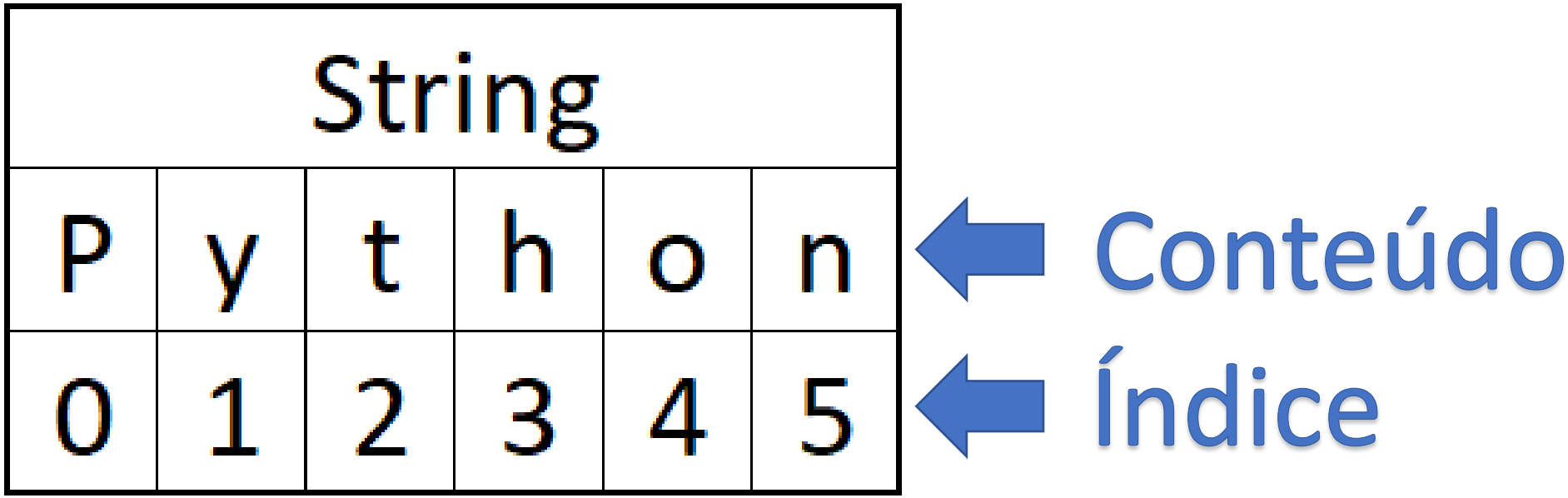
```
>>> print (len("A"))
1
>>> print (len("AB"))
2
```

Função Len

```
>>> print (len(""))
0
>>> print (len("Senai Roberto Mange"))
19
```

String																		
S	e	n	a	i		R	o	b	e	r	t	o		M	a	n	g	e
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19

Manipulação de Strings



Manipulação de Strings

```
>>> s = “Python“
>>> print(s[0])
p
>>> print(s[1])
y
>>> print(s[5])
n
```

String					
P	y	t	h	o	n
0	1	2	3	4	5

Manipulação de Strings

```
>>> s = “Python“

>>> print(s[6])
```

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

IndexError: string index out of range

String					
P	y	t	h	o	n
0	1	2	3	4	5

Python

Operações com Strings

As variáveis de tipo string suportam operações como **fatiamento**, **concatenação** e **composição**.

Por **fatiamento**, podemos entender a capacidade de utilizar apenas **uma parte de uma string**, ou **uma fatia**.

A **concatenação** nada mais é que poder **juntar duas** ou **mais strings** em uma **nova string maior**.

A **composição** é muito utilizada em mensagens que enviamos à tela e consiste em **utilizar strings como modelos** onde podemos inserir outras strings.

Concatenação

O conteúdo de variáveis string podem ser **somados**, ou melhor, **concatenados.**

Para **concatenar** duas strings, utilizamos o operador de **adição (+).**

Assim, **"AB" + "C"** é igual a **"ABC".** Um caso especial de concatenação é a repetição de uma string várias vezes. Para isso, utilizamos o operador de multiplicação (*): "A" * 3 é igual a "AAA".

Concatenação

```
>>> p = “PYTHON“
```

```
>>> print (p + “N")
```

PYTHONN

```
>>> print (p+ “N” * 4)
```

PYTHONNNN

Concatenação

```
>>> p = “PYTHON“
```

```
>>> print (“P” + "-"*10 + “P")
```

P----------P

```
>>> print (p+“C4 = "+p*4)
```

SENAIk4 = SENAISENAISENAISENAI

Composição

“Senai Campinas tem X anos", onde **X** é uma **variável numérica.**

Usando a composição de strings do Python, podemos escrever:

“Senai Campinas tem %d anos“ %X

Composição

“Senai Campinas tem %d anos“ %X

O símbolo de **%** foi utilizado para indicar a composição da string anterior com o conteúdo da variável **X**.

O **%d** dentro da primeira string é o que chamamos de **marcador de posição**. O marcador indica que naquela posição estaremos colocando um valor inteiro, daí o **%d**.

Composição

Tipos de marcadores:

Marcador	Tipo
%d	Números inteiros
%s	Strings
%f	Números decimais

Composição com Números Inteiros

```
>>> idade = 22
```

```
>>> print("[%d]" % idade)
[22]
>>> print("[%03d]" % idade)
[022]
```

Marcador	Tipo
%d	Números inteiros
%s	Strings
%f	Números decimais

Composição com Números Inteiros

```
>>> idade = 22
```

```
>>> print("[%3d]" % idade)
[ 22]
>>> print("[%-3d]" % idade)
[22 ]
```

Marcador	Tipo
%d	Números inteiros
%s	Strings
%f	Números decimais

Composição com Números Decimais

```
>>> print("%f" % 5)
5.000000

>>> print("%.2f" % 5)
5.00

>>> print("%10.5f" % 5)
  5.00000
```

Marcador	Tipo
%d	Números inteiros
%s	Strings
%f	Números decimais

Composição de String

```
>>> nome = “Pedro“
>>> idade = 25
>>> grana = 25.35
```

Marcador	Tipo
%d	Números inteiros
%s	Strings
%f	Números decimais

```
>>> print("%s tem %d anos e R$%f no bolso." % (nome, idade, grana))
```

Pedro tem 25 anos e R$25.350000 no bolso.

Composição de String

```
>>> nome = “Pedro“
>>> idade = 25
>>> grana = 25.35
```

Marcador	Tipo
%d	Números inteiros
%s	Strings
%f	Números decimais

```
>>> print("%12s tem %3d anos e R$%5.2f no bolso." % (nome, idade, grana))
```

Pedro tem 25 anos e R$25.35 no bolso

Composição de String

```
>>> nome = “Pedro“
>>> idade = 25
>>> grana = 25.35
```

Marcador	Tipo
%d	Números inteiros
%s	Strings
%f	Números decimais

```
>>> print("%12s tem %03d anos e R$%5.2f no bolso." % (nome, idade, grana))
```

Pedro tem 025 anos e R$25.35 no bolso

Composição de String

```
>>> nome = “Pedro“
>>> idade = 25
>>> grana = 25.35
```

Marcador	Tipo
%d	Números inteiros
%s	Strings
%f	Números decimais

```
>>> print("%-12s tem %-3d anos e R$-5.2%f no bolso." % (nome, idade, grana))
```

Pedro tem 25 anos e R$25.35 no bolso.

Fatiamento

Uma operação muito interessante que Python fornece para manipulação de Listas, strings e tuplas é o fatiamento (slicing).

Fatiamento significa extrair apenas uma parte (subconjunto) da Lista, String ou Tupla.

Fatiamento

```
>>> s="ABCDEFGHI“

>>> print (s[0:2])
```

AB

```
>>> print (s[1:2])
```

B

```
>>> print (s[2:4])
```

CD

String								
A	B	C	D	E	F	G	H	I
0	1	2	3	4	5	6	7	8

Fatiamento

```
>>> print (s[0:5])
```

ABCDE

```
>>> print (s[1:8])
```

BCDEFGH

String								
A	B	C	D	E	F	G	H	I
0	1	2	3	4	5	6	7	8

Fatiamento

Podemos também **omitir o número da esquerda ou o da direita** para representar do início ou até o final.

Assim, **[:2] indica do início até o segundo caractere** (sem incluí-lo), **e [1:] indica do caractere de posição 1 até o final da string.**

Observe que, nesse caso, **nem precisamos saber** quantos caracteres a string contém.

Fatiamento

Se **omitirmos o início e o fim da fatia**, estaremos fazendo apenas uma **cópia de todos os caracteres da string** para uma nova string.

Podemos também utilizar **valores negativos** para indicar posições a partir da direita.

Assim **-1** é o **último** caractere; **-2**, o **penúltimo**; e assim por diante.

Fatiamento

```
>>> print (s[:2])
```

AB

```
>>> print (s[1:])
```

BCDEFGHI

```
>>> print (s[0:-2])
```

ABCDEFG

```
>>> print (s[:])
```

ABCDEFGHI

String								
A	B	C	D	E	F	G	H	I
0	1	2	3	4	5	6	7	8

Fatiamento

```
>>> print (s[-1:])
```

I

```
>>> print (s[-5:7])
```

EFG

```
>>> print (s[-2:-1])
```

H

String								
A	B	C	D	E	F	G	H	I
0	1	2	3	4	5	6	7	8

Sequencias e Tempos

Um programa é executado **linha por linha** pelo computador, executando as operações descritas no programa **uma após a outra**.

Quando trabalhamos com **variáveis**, devemos nos lembrar de que **o conteúdo de uma variável pode mudar com o tempo**.

Isso porque a cada vez que alteramos o valor de uma variável, o valor anterior é substituído pelo novo.

Exemplo

dívida = 0 1

compra = 100 2

dívida = dívida + compra 3

compra = 200 4

dívida = dívida + compra 5

compra = 300 6

dívida = dívida + compra 7

compra = 0 8

print(dívida)

Rastreamento e Testes de Mesa

```
dívida = 0 1
compra = 100 2
dívida = dívida + compra 3
compra = 200 4
dívida = dívida + compra 5
compra = 300 6
dívida = dívida + compra 7
compra = 0 8
print(dívida)
```

Passos	dívida	compra
1	0	0
2	0	100
3	100	100
4	100	200
5	300	200
6	300	300
7	600	0

Introdução

Chamamos de **entrada de dados** o momento em que seu programa **recebe dados** ou **valores** por um **dispositivo de entrada de dados** (como o **teclado** do computador) ou de um **arquivo em disco**.

A função **input** é utilizada para **solicitar dados do usuário**.

Ela recebe um parâmetro, que é a mensagem a ser exibida, e retorna o valor digitado pelo usuário.

Exemplos

```
x = input("Digite um número: ")

print(x)


nome = input("Digite seu nome:")

print("Você digitou %s" % nome)

print("Olá, %s!" % nome)
```

Exemplos

```
x = input("Digite um número: ")

print(x)


nome = input("Digite seu nome:")

print("Você digitou %s" % nome)

print("Olá, %s!" % nome)
```

Conversão de Entrada de Dados

A função **input** sempre retorna valores do tipo **string**, ou seja, **não importa se digitamos apenas números**, o resultado sempre é string.

Para resolver este problema, vamos utilizar a função **int** para converter o valor retornado em um **número inteiro**, e a função **float** para convertê-lo em número **decimal ou de ponto flutuante.**

Conversão de Entrada de Dados

```
anos = int(input("Anos de serviço: "))

valor_por_ano = float(input("Valor por ano: "))

bônus = anos * valor_por_ano

print("Bônus de R$ %5.2f" % bônus)
```

Python

Atividade 03

https://forms.gle/42GAh8LwfLDdBqveA