Aula 05 – Operações com Strings

Professor Rodrigo Maciel

Variáveis do tipo String

- Representam informação textual;
- Uma string é uma sequência de caracteres simples;
- Na linguagem Python, as strings são utilizadas com aspas simples ('... ') ou aspas duplas ("...").

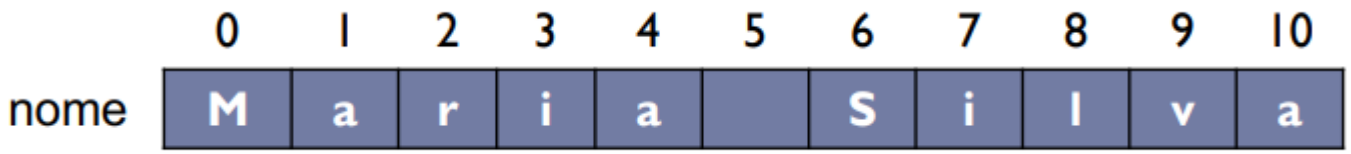
```
nome = "Maria Silva"
nacionalidade = "brasileira"
nome_mae = "Ana Santos Silva"
nome_pai = "Jonas Nunes Silva"
```

```
>>> print("Olá Mundo")
Olá Mundo
>>>
```

Acessando caracteres de uma String

- Caracteres podem ser acessados pela sua posição dentro da String;
- Primeira posição é a posição ZERO.

```
>>> nome = "Maria Silva"
>>> print(nome[0])
M
>>> print(nome[6])
S
```



Alteração de um caractere em Strings

- O conteúdo de uma determinada posição de uma string não pode ser alterado – são sequências imutáveis

```
>>> nome = "Maria Silva"
>>> nome[3] = "t"
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'str' object does not support
item assignment
```

Fatiamento das Strings

- O fatiamento é uma ferramenta usada para extrair apenas uma parte dos elementos de uma string;

string[Limite_Inferior : Limite_Superior]

- Retorna uma string com os elementos das posições do limite inferior até o limite superior - 1.

Fatiamento das strings

```
s = "Python"
s[1:4]      → seleciona os elementos das posições 1,2,3
'yth'

s[2:]       → seleciona os elementos a partir da posição 2
'thon'

s[:4]       → seleciona os elementos até a posição 3
'Pyth'
```

Principais operadores para Strings

- Alguns operadores importantes que podem ser usados em strings:
  - In
  - len
  - +
  - *

Operador in

- substring in string
  - ➢ retorna True ou False

```
>>> nome = "Maria Silva"
>>> "M" in nome
True
>>> "B" in nome
False
>>> "m" in nome
False
>>> "ria" in nome
True
```

Operador len()

- len(string)

  - Retorna a quantidade de caracteres da string.

```
>>> nome = "Maria"
>>> len(nome)
5
>>> nome = "Maria Silva"
>>> len(nome)
11
```

Operador de concatenação – “+”

- string1 + string2
  - ➢ Concatena duas strings

```
>>> nome = "Maria" + "Silva"
>>> nome
MariaSilva

>>> nome = "Maria"
>>> sobrenome = "Silva"
>>> nome_completo = nome + sobrenome
>>> nome_completo
MariaSilva
```

Operador de repetição – “*”

- string * int
  - Repete a string int vezes

```
>>> nome = "Maria"
>>> nome_repetido = nome * 2
>>> nome_repetido
MariaMaria
```

Outros operadores uteis

- Outros operadores uteis para se utilizar com strings:
  - upper
  - lower
  - split
  - partition

Operador upper()

- string.upper()
  - Retorna a string com letras minúsculas substituídas por maiúsculas

```
>>> texto = "Quem parte e reparte, fica com a maior
parte"
>>> texto_up = texto.upper()
>>> texto_up
"QUEM PARTE E REPARTE, FICA COM A MAIOR PARTE"
>>> texto
"Quem parte e reparte, fica com a maior parte"
```

Operador lower()

- string.lower()
  - Retorna a string com letras minúsculas substituídas por maiúsculas

```
>>> texto = "Quem parte e reparte, fica com a maior
parte"
>>> texto.lower()
"quem parte e reparte, fica com a maior parte"
```

Operador split()

- string.split(separador)
  - Separa a string em "pedaços" que aparecem antes e depois do separador.
  - Se o separador não for especificado, é usado espaços em branco, tabs e quebras de linha como separador.
  - Útil para ler várias entradas de uma única vez.

Operador split()

- string.split(separador)

```
>>>x, y = input("Digite dois valores: ").split()
Digite dois valores: 10 20
>>>x
"10"
>>>y
"20"
```

Operador split()

- string.split(separador)

```
>>>dia, mes, ano = input("Digite uma data:
").split("/")
Digite uma data: 10/04/2018
>>>dia
"10"
>>>mes
"04"
>>>ano
"2018"
```

Operador partition()

- string.partition(separador)
  - Separa a string em três pedaços: o que vem antes da primeira ocorrência do separador, o separador e o que vem depois do separador
  - Útil para ler várias entradas de uma única vez, quando não é possível (pela lógica do problema) usar o for para iterar sobre o split da entrada)

Operador partition()

- string.partition(separador)

```
>>>antes, sep, depois = input("Digite valores:
").partition("-")
Digite valores: 10-20-30-40
>>>antes
"10"
>>>sep
"-"
>>>depois
"20-30-40"
```

Operador partition()

- string.partition(separador)

```
>>>antes, sep, depois = input("Digite valores:
").partition(" ")
Digite valores: 10 20 30 40
>>>antes
"10"
>>>sep

>>>depois
"20 30 40"
```

Outros operadores para Strings

Método	Descrição	Exemplo
len()	Retorna o tamanho da string.	teste = “Apostila de Python” len(teste) 18
capitalize()	Retorna a string com a primeira letra maiúscula	a = "python" a.capitalize() 'Python'
count()	Informa quantas vezes um caractere (ou uma sequência de caracteres) aparece na string.	b = "Linguagem Python" b.count("n") 2
startswith()	Verifica se uma string inicia com uma determinada sequência.	c = "Python" c.startswith("Py") True
endswith()	Verifica se uma string termina com uma determinada sequência.	d = "Python" d.endswith("Py") False

Outros operadores para string

Método	Descrição	Exemplo
isalnum()	Verifica se a string possui algum conteúdo alfanumérico (letra ou número).	e = "!@#$%" e.isalnum() False
isalpha()	Verifica se a string possui apenas conteúdo alfabético.	f = "Python" f.isalpha() True
islower()	Verifica se todas as letras de uma string são minúsculas.	g = "pytHon" g.islower() False
isupper()	Verifica se todas as letras de uma string são maiúsculas.	h = "# PYTHON 12" h.isupper() True
lower()	Retorna uma cópia da string trocando todas as letras para minúsculo.	i = "#PYTHON 3" i.lower() '#python 3'

Outros operadores para string

Método	Descrição	Exemplo
upper()	Retorna uma cópia da string trocando todas as letras para maiúsculo.	j = "Python" j.upper() 'PYTHON'
swapcase()	Inverte o conteúdo da string (Minúsculo / Maiúsculo).	k = "Python" k.swapcase() 'pYTHON'
title()	Converte para maiúsculo todas as primeiras letras de cada palavra da string.	l = "apostila de python" l.title() 'Apostila De Python'
split()	Transforma a string em uma lista, utilizando os espaços como referência.	m = "cana de açúcar" m.split() ['cana', 'de', 'açúcar']

Outros operadores para string

Método	Descrição	Exemplo
replace(S1, S2)	Substitui na string o trecho S1 pelo trecho S2.	n = "Apostila teste" n.replace("teste", "Python") 'Apostila Python'
find()	Retorna o índice da primeira ocorrência de um determinado caractere na string. Se o caractere não estiver na string retorna -1.	o = "Python" o.find("h") 3
ljust()	Ajusta a string para um tamanho mínimo, acrescentando espaços à direita se necessário.	p = " Python" p.ljust(15) ' Python '
rjust()	Ajusta a string para um tamanho mínimo, acrescentando espaços à esquerda se necessário.	q = "Python" q.rjust(15) ' Python'
center()	Ajusta a string para um tamanho mínimo, acrescentando espaços à esquerda e à direita, se necessário.	r = "Python" r.center(10) ' Python '
lstrip()	Remove todos os espaços em branco do lado esquerdo da string.	s = " Python " s.lstrip() 'Python '
rstrip()	Remove todos os espaços em branco do lado direito da string.	t = " Python " t.rstrip() ' Python'
strip()	Remove todos os espaços em branco da string.	u = " Python " u.strip() 'Python'

Praticando Python...

- Crie um programa que imprima o comprimento de uma string fornecida pelo usuário.

- Escreva um programa para verificar se a palavra 'laranja' está presente em "Isto é suco de laranja".

- Escreva um programa que solicite uma frase ao usuário e escreva a frase toda em maiúscula e sem espaços em branco.