INSTITUTO FEDERAL
Catarinense
Campus Camboriú

AULA XII – STRINGS (Continuação)

Professora: Lidiane Visintin

lidiane.visintin@ifc.edu.br

Professor: Rafael de Moura Speroni

rafael.speroni@ifc.edu.br

Objetivos da Aula:

- Compreender o uso de strings e os métodos para manipulação de strings.
  - Quebra ou separação de strings
  - Substituição de strings
  - Remoção de espaços em branco
  - Validação por tipo de conteúdo

Quebra ou separação de strings

Podemos **quebrar** uma string a partir de um caractere passado como parâmetro através do método `split`:

```
>>> s = "um tigre, dois tigres, três tigres"

>>> s.split(",")
['um tigre', 'dois tigres', 'três tigres']

>>> s.split(" ")
['um', 'tigre,', 'dois', 'tigres,', 'três', 'tigres']

>>> s.split()
['um', 'tigre,', 'dois', 'tigres,', 'três', 'tigres']
```

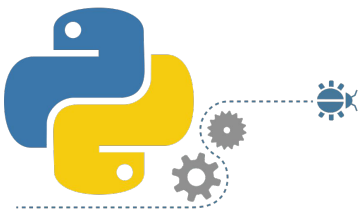
Observe que o caractere que utilizamos para dividir a string **não** é retornado na lista, ele é utilizado e descartado.

Quebra ou separação de strings

Se nosso objetivo for separar uma string com várias linhas de texto, utilizamos o método `splitness`:

```
>>> m = "Um linha\noutra linha\ne mais uma linha"

>>> m.splitlines()
['Uma linha', 'outra linha', 'e mais uma linha']
```



Substituição de strings

Para substituirmos trechos de uma string por outros, utilizamos o método `replace`. O primeiro parâmetro é a string a substituir; e o segundo parâmetro o conteúdo que a será inserido no lugar.

```
>>> s = "um tigre, dois tigres, três tigres"

>>> s.replace("tigre", "gato")
'um gato, dois gatos, três gatos'

>>> s.replace("tigre", "")
'um, dois s, três s'
```

- Se passarmos uma string vazia no **segundo** parâmetro o primeiro será **apagado**.
- Se o **primeiro** parâmetro for uma string vazia, o segundo será **inserido** antes de cada caractere da string.

Substituição de strings

O método `replace` possui um **terceiro parâmetro opcional** que limita quantas vezes queremos realizar a repetição.

```
>>> s = "um tigre, dois tigres, três tigres"

>>> s.replace("tigre", "gato", 1)
'um gato, dois tigres, três tigres'

>>> s.replace("tigre", "gato", 2)
'um gato, dois gatos, três tigres'
```

Remoção de espaços em branco

O método `strip` é utilizados para **remover espaços em branco** do início ou do fim da string. Já os métodos `lstring` e `rstring` removem os caracteres em branco à **esquerda** ou à **direita**, respectivamente:

```
>>> t = “      Olá     ”

>>> t.strip()
‘Olá’

>>> t.lstrip()
“Olá     ”

>>>t.rstrip()
“     Olá”
```

Remoção de espaços em branco

Se passarmos um parâmetro paro método `strip` este será utilizado como caractere a **remover:**

```
>>> t = “....Olá....”

>>> t.strip(“.”)
‘Olá’

>>> t.lstrip(“.”)
‘Olá....’

>>>t.rstrip(“.”)
‘....Olá’
```

Validação por tipo de conteúdo

Strings em Python podem ter seu conteúdo **analisado** e **verificado** através de métodos específicos. Esses métodos verificam se todos os caracteres são **letras**, **números** ou **uma combinação** deles. Vejamos:

```
>>> s = “125”
>>> p = “alô mundo”

>>> s.isalnum()
True

>>> p.isalnum()
False

>>>s.isalpha()
False

>>>p.isalpha()
False
```

Validação por tipo de conteúdo

O método **`isalnum`** retorna verdadeiro se a string não estiver vazia, e se todos os seus caracteres são letras e/ou números. Se a string contiver outro tipo de caractere como espaços, pontuação ou caracteres de controle, retorna **False**.

```
>>> s.isalnum()
True
```

Já **`isalpha`** retorna verdadeiro apenas se todos os caracteres são letras incluindo vogais acentuadas. Se a string estiver vazia ou possuir algum outro tipo de caractere incluindo espaço, retorna **False**.

```
>>> p.isalpha()
False
```

Validação por tipo de conteúdo

O método **`isdigit`** verifica se o valor consiste em **números**, retornando **True** se a string não estiver vazia e contiver apenas números. Se a string contiver espaços, pontos, vírgulas ou sinais (+ ou -), retorna falso. Vejamos:

```
>>> "771".isdigit()
True

>>> "10.4".isdigit()
False

>>>"+5".isdigit()
False

>>>"-10".isdigit()
False
```

Validação por tipo de conteúdo

Podemos verificar também se todos os caracteres de uma string são letras **maiúsculas** ou **minúsculas** usando os métodos `isupper` e `islower`, respectivamente:

```
>>> s = "ABC"
>>> p = "abc"
>>> e = "aBc"

>>> s.isupper()
True

>>> p.isupper()
False

>>> p.islower()
True

>>> e.islower()
False
```

Resumo

Estudamos o uso de strings e alguns métodos de manipulação.

- Quebra ou separação de strings
  - `split`
  - `splitliness`
- Substituição de strings
  - `replace` (com dois ou três argumentos)
- Remoção de espaços em branco
  - `strip` (com ou sem argumentos)
  - `ltrip`
  - `rstrip`
- Validação por tipo de conteúdo
  - `isalnum`
  - `isalpha`
  - `isdigit`
  - `isupper`
  - `islower`

Referências

Referências Básicas

FORBELLONE, André Luiz Villar; EBERSPÄCHER, Henri Frederico. Lógica de programação: a construção de algoritmos e estruturas de dados. 3. ed. Pearson Prentice Hall. 2005

MANZANO, José Augusto N. G; OLIVEIRA, Jayr Figueiredo de.. Algoritmos: lógica para desenvolvimento de programação de computadores.. 27. ed.. Érica. 2014

Referências Complementares

DOWNEY, Allen B. **Pense em Python**. 2ª Ed. Novatec. 2016

MENEZES, Nilo Ney de Coutinho. **Introdução a programação com Python**. 3ª Ed. Novatec. 2019

CORMEN, Thomas H et al. **Algoritmos: teoria e prática**. 2. ed. Elsevier, Campus,. 2002

Referências na Internet

https://docs.python.org/3/

https://www.w3schools.com/python/default.asp