INSTITUTO FEDERAL
Catarinense
Campus Camboriú

AULA XI - STRINGS

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.
  - Verificação parcial de strings
  - Contagem
  - Pesquisa de strings

Uma String é uma sequência de caracteres

Podemos acessar um caractere de cada vez com o operador de colchete:

```
1   fruta = "banana"
2   letra = fruta[1]
```

A segunda instrução seleciona o caractere número 1 de `fruta` e o atribui a variável `letra`
A expressão entre colchetes chama-se **índice**. O índice aponta qual caractere da sequência queremos imprimir.

```
>>> letra
'a'
```

O índice é uma referência do começo da string e a referência da primeira letra é zero

```
1   fruta = "banana"
2   letra = fruta[0]
```

Visualização do armazenamento de uma String

b	a	n	a	n	a
0	1	2	3	4	5

Uma String é uma sequência de caracteres

O índice é uma referência do começo da string e a referência da primeira letra é **zero**.

```
1   fruta = "banana"
2   letra = fruta[0]
```

```
>>> letra
'b'
```

Então **b** é a ("zerésima") letra de `'banana'`, **a** é a (primeira) letra e **n** é a (segunda) letra.

Trabalhando com strings

Podemos acessar strings como listas, mas strings são imutáveis em Python:

```
1   s = "Alô mundo!"
2   print(s[0])
```

```
>>> print(s[0])
'A'

>>> s[0] = "a"
Traceback (most recent call last):
  File "<stdin>, line 1, in <module>
TypeError: 'str' object does not support item assignment
```

Se quisermos trabalhar caractere a caractere com uma string , alterando seu valor, teremos de primeiro transformá-la em uma lista.

Trabalhando com strings

A função **list** transforma cada caractere da string em um elemento da lista retornada. Já a função **join** faz a operação inversa, transformando os elementos da lista e, string.

```
>>> L = list("Alô mundo")
>>> L[0] = "a"
>>> print(L)
['a', 'l', 'ô', '', 'm', 'u', 'n', 'd', 'o']

>>> S = "".join(L)
>>> print(S)
alô mundo
```

Verificação parcial de strings

Podemos verificar se uma string começa ou termina com alguns caracteres. Para isso utilizamos os métodos `startswith` e `endswith`. Esses métodos verificam apenas os respectivos caracteres indicados e retornam **True** ou **False**.

```
>>> nome = "João da Silva"
>>> nome.startswith("João")
True

>>> nome.startswith("joão")
False

>>> nome.endswith("Silva")
True
```

Note que no segundo exemplo o método considera letras maiúsculas e minúsculas **diferentes**.

Verificação parcial de strings

Podemos converter a string para maiúsculas ou minúsculas antes de realizar a comparação utilizando os métodos `upper` e `lower` respectivamente.

```
>>> s = "O Rato roeu a Roupa do Rei de Roma"
>>> s.lower()
'o rato roeu a roupa do rei de roma'

>>> s.upper()
'O RATO ROEU A ROUPA DO REI DE ROMA'

>>> s.lower().startswith("o rato")
True

>>> s.upper().startswith("O RATO")
True
```

Verificação parcial de strings

Outra forma de verificar se uma palavra pertence a uma string é utilizando o operador **in**:

```
>>> s = "Maria Amélia"
>>> "Amélia" in s
True

>>> "Maria" in s
True

>>> "a A" in s
True

>>> "amélia" in s
False
```

Verificação parcial de strings

Podemos testar se um string **não** está contida em outra, utilizando **`not in`**:

```
>>> s = "Todos os caminhos levam a roma"
>>> "levam" not in s
False

>>> "Caminhos" not in s
True

>>> "AS" not in s
True
```

Verificação parcial de strings

Podemos podemos combinar `lower` e `upper` com **`in` e `not in`**:

```
>>> s = "João comprou um carro"
>>> "joão" in s.lower()
True

>>> "CARRO" in s.upper()
True

>>> "comprou" not in s.lower()
False
```

Contagem

Se precisarmos contar as ocorrências de uma letra ou palavra em uma string podemos utilizar o método `count`:

```
>>> t = "um tigre, dois tigres, três tigres"
>>> t.count("tigre")
3

>>> t.count("tigres")
2

>>> t.count("t")
4

>>> t.count("z")
0
```

Pesquisa de strings

Para pesquisarmos se uma string está dentro de outra e obter a posição da primeira ocorrência podemos utilizar o método `find`:

```
>>> s = "Alô mundo"
>>> s.find("mun")
4

>>> s.find("ok")
-1
```

Caso a string seja encontrada receberemos um valor maior ou igual a zero, caso contrário **-1**.

Pesquisa de strings

Se o objetivo for pesquisar da **direita** para a **esquerda** utilizamos o método `rfind`:

```
>>> s = "Um dia de sol"
>>> s.rfind("d")
7

>>> s.find("d")
3
```

Pesquisa de strings

Tanto `find` quanto `rfind` suportam duas opções adicionais: *start* e *end.* Se especificarmos o início a pesquisa começará a partir dessa posição. Se especificarmos o fim a pesquisa utilizará esta posição como último caractere a considerar. Por exemplo:

```
>>> s = “um tigre, dois tigres, três tigres”
>>> s.find(“tigres”)
15

>>> s.find(“tigres”, 7) #início=7
15

>>> s.find(“tigres”, 30) #início=30
-1

>>> s.find(“tigres”, 0, 10) #início=0 fim=10
-1
```

Pesquisa de strings

Podemos utilizar o valor retornado por `find` e `rfind` para achar todas as ocorrências da string. Por exemplo:

```
1   s = "um tigre, dois tigres, três tigres"
2   p = 0
3   while(p > -1):
4       p = s.find("tigre", p)
5       if p >= 0:
6           print(f"Posição: {p}")
7           p += 1
```

Produz como resultado:

```
Posição: 3
Posição: 15
Posição: 28
```

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